Published in final edited form as:

Proc SPIE. 2012 February 23; 8313: 83134C-. doi:10.1117/12.911272.

Graphics Processing Unit (GPU) implementation of image processing algorithms to improve system performance of the Control, Acquisition, Processing, and Image Display System (CAPIDS) of the Micro-Angiographic Fluoroscope (MAF)

S.N. Swetadri Vasan^{1,2}, Ciprian N. Ionita², A.H. Titus^{1,2}, A.N. Cartwright^{1,2}, D.R. Bednarek², and S. Rudin^{1,2}

¹Department of Electrical Engineering, University at Buffalo

²Toshiba Stroke Research Center, University at Buffalo

Abstract

We present the image processing upgrades implemented on a Graphics Processing Unit (GPU) in the Control, Acquisition, Processing, and Image Display System (CAPIDS) for the custom Micro-Angiographic Fluoroscope (MAF) detector. Most of the image processing currently implemented in the CAPIDS system is pixel independent; that is, the operation on each pixel is the same and the operation on one does not depend upon the result from the operation on the other, allowing the entire image to be processed in parallel. GPU hardware was developed for this kind of massive parallel processing implementation. Thus for an algorithm which has a high amount of parallelism, a GPU implementation is much faster than a CPU implementation. The image processing algorithm upgrades implemented on the CAPIDS system include flat field correction, temporal filtering, image subtraction, roadmap mask generation and display window and leveling. A comparison between the previous and the upgraded version of CAPIDS has been presented, to demonstrate how the improvement is achieved. By performing the image processing on a GPU, significant improvements (with respect to timing or frame rate) have been achieved, including stable operation of the system at 30 fps during a fluoroscopy run, a DSA run, a roadmap procedure and automatic image windowing and leveling during each frame.

1. INTRODUCTION

The CAPIDS system [1][2] was developed for the custom built MAF detector [3][4]. The system has three basic functionalities. First is image acquisition control. The images captured from the MAF camera are acquired by a frame grabber board (National Instruments - PCIe - 1430) via a Camera Link communication channel. The second functionality is image processing. The acquired images are post-processed on an 8-core CPU. The image processing involves flat field correction, temporal filtering, image subtraction (as part of digital subtraction angiography (DSA)), and roadmap mask generation (as a part of the road map procedure). The memory management, thread management, CPU core allocation for parallelization of the processes is handled by LabVIEW. The final functionality is the data storage. All the images acquired are stored for further viewing and analysis.

With the current generation multicore-CPU operating at faster speeds, it is possible to process 1Mega pixel images at high frame rates. However, with the introduction of CUDA

(Compute Unified Device Architecture) computing by NVIDIA Corp. (Santa Clara, CA) [5], it has become possible to implement complex image processing algorithms on a GPU which has a hardware platform for massive parallel computing, thus achieving a reduction in processing time. In the new version of CAPIDS introduced here, all the image processing techniques are implemented on a dedicated GPU present on the video card NVIDIA Model GTX 285.

2. METHODS

Image processing for which the operation on one pixel is the same and independent of the operation on other pixels and for which the operation does not depend on the result of the operation of another pixel can be performed on the entire image (1M pixels) in parallel. For an 8 core CPU the maximum number of operations that can happen in parallel is 8. However GPU hardware is developed for this kind of massive parallel implementation. For Example, the NVIDIA GTX 285 GPU currently used in this work has 30 multiprocessors each of which support 1024 concurrent execution threads. Combined, these multiprocessors allow the GPU to manage over 30,000 concurrent individual instruction executions. Thus for an algorithm which has a high amount of parallelism, a GPU implementation is much faster than a CPU implementation.

The following image processing functions can be performed on each pixel independently and thus were implemented on the GPU:

1. Flat field correction.

$$P_{out}(x, y) = \frac{P_{input}(x, y) - darkfield(x, y)}{flatfield(x, y) - darkfield(x, y)} \times (average\ number)$$

where $P_{input}(x, y)$ denotes the pixel in location (x,y).

2. Temporal Recursive Filtering

$$P_{out}(x, y) = \left(\left(1 - \frac{1}{K} \right) \times P_{previous\ output}(x, y) \right) + \left(\frac{1}{K} \times P_{input}(x, y) \right)$$

where K is the weight of the filter which can be changed by the user. A switch is also provided to turn the filter on and off.

3. Image Subtraction: The image is formed by detection of x-rays that have been attenuated exponentially in the body. Hence a logarithmic subtraction is used.

$$P_{out}(x, y) = \frac{P_{input}(x, y)}{mask(x, y)} \times (average\ number)$$

4. Image comparison for roadmap mask generation: In this process before injecting contrast into the vessel an average of 3 images is taken as the initial reference. Once contrast is injected into the vessel, the incoming images are compared with the previous image. If the current pixel intensity value is less than the previous pixel intensity, the current pixel intensity is retained otherwise the previous pixel intensity is retained. This is because the contrast absorbs more x-rays leading to a dark image in the blood vessel region. After acquiring about 10–15 frames the final processed image is taken as the map for the roadmap procedure.

5. Automatic window and leveling for each frame: The histogram of the processed image is read using a virtual instrument (VI) in LabVIEW. The mean and standard deviation of an area in an image are used to calculate the window and level. This is sent to a CUDA dll (Dynamic-link library) which maps the image to 0 to 255 and then displays it to the end user.

3. IMPLEMENTATION

An algorithm can be divided into two components: a sequential part and a parallel part. The scalar part can be implemented on the CPU, whereas the vector part can be implemented on the GPU thus decreasing the processing time and increasing the system performance. Fig. 1 gives the basic data flow involved in GPU programming [6]. The data to be processed is copied from the CPU accessible memory (RAM) to a GPU accessible memory (present on the graphics board). Then the data is processed on the GPU. The processed data is then copied back to CPU accessible memory from the GPU.

A GPU-CUDA kernel can be treated as a function which is executed in parallel on a GPU. The thread organization for CUDA is shown in Fig 2 [7]. A thread can be seen as an execution unit, and each thread executes the same kernel, thus achieving parallelism. For the CAPIDS upgrade, operation on a pixel is treated as a kernel. Each kernel (pixel) is assigned a thread, and the threads (pixels) are executed in parallel.

The image processing algorithms were written in CUDA C language and compiled into a dll file (Dynamic-link library) using the Visual Studio 2008 compiler (Microsoft Corporation, Redmond, Washington). This dll file was then integrated into the CAPIDS program. Every time an image is acquired, it is passed to the CUDA dll which copies the image from the CPU memory to the GPU memory, processes the image and then copies the image back to the CPU memory. Since this process happens in a loop, all the required memories are allocated outside the loop. This also helps in decreasing the processing time since memory allocation in CUDA takes about one to a few hundred milliseconds depending upon the size of the memory [7]. Once the loop is ended all the initialized memories are freed. Instead of passing an image as a whole, the pointer pointing to the first pixel of the image is passed to the dll.

Fig 3 shows the program flow in CAPIDS system during a typical fluoroscopy run. When the run is started all the memories used by both the GPU and CPU are initialized. Then the system enters the acquisition loop. In the loop the system first listens for a trigger signal for a period of time. If there is a trigger signal within that period of time, then it indicates that there is a valid image at the input and the frame grabber acquires this image. If there is no trigger signal, then an acquisition time out has occurred and the last processed image is displayed. After the image is acquired the image is sent to the GPU for processing. While the image is getting processed on the GPU, the previously processed image is being saved on the hard drive. Once the image is processed on the GPU it is then transferred to the CPU to be displayed on the monitor.

3.1 Comparison with the previous version of CAPIDS

Among the 3 functionalities of CAPIDS, data storage is the slowest, as the data is stored on a hard drive. In order to achieve fast frame rates, in the previous version, the CAPIDS system is implemented as a pipelined architecture.

In the previous version of CAPIDS the image processing was done on an 8-core CPU. For a 1024×1024 image with 16 bit pixel, the average time taken by one image processing subroutine was measured to be around 4–5 ms. So in order to achieve fast frame rates such as 30 fps along with image storage, all the subroutines were implemented in a pipelined

architecture. Fig. 4 gives a basic flow of the pipeline during a normal fluoroscopy run. From the figure it can be seen that when the 3rd frame (I_3) is acquired (during loop iteration time t_3), flat field correction is performed on frame 2 (obtained during t_2), temporal filtering is performed on frame 1 (obtained during t_1) and frame 0 (obtained during t_0) is saved. Thus there is a lag of 3 frames from the time the image is acquired till the time it is stored, however the display is real time to the interventionalist.

With the image processing subroutines implemented on GPU, the overall image processing time including flat field correction, temporal filtering and image subtraction was measured to be around 4–5ms. However since timing improvements have not been made in the data storage part, the system is still implemented as a pipelined architecture, but with all the image processing subroutines combined into one block of the pipeline (Fig. 5).

4. RESULTS

With the upgrades implemented in the CAPIDS systems, having all the image processing steps active, we have been able to achieve a stable operation at 30 fps during a fluoroscopy run, DSA run, and roadmap procedure. Although in the previous version, a 30 fps fluoroscopy run was achievable, the system was not stable, resulting in the skipping of some frames. In the previous version of CAPIDS, automatic window and leveling was active for every 3rd frame, but in this version it is done in every frame. Also in the previous version of CAPIDS, the fastest frame rate for a roadmap mask generation was 15 fps, but in the current version it can go up to 20 fps; however, the fluoroscopy mode during the roadmap procedure can go up to 30 fps.

Fig. 6 represents a screen shot of the front panel of the updated CAPIDS program. The front panel shows the acquired and processed image. The image shows a neurostent (Wingspan from Boston Scientific, Natick, MA) placed underneath the aneurysm region (the ROI indicated in green is an arbitrary shape selected by the user at the time of study). The temporal filter can be turned on and off with a switch, and also the weight can be changed interactively. The window and level for the display can be set to automatic or manual without any interruption to the run in progress.

5. CONCLUSIONS

In this paper we have presented the image processing upgrades implemented in the CAPIDS system. By performing the image processing on a GPU, significant improvements (with respect to timing or frame rate) have been achieved. Furthermore this also provides a platform to implement complex processing algorithms in the future, such as image sharpening and spatially different real-time temporal filtering which involves the use of different temporal filter weights in different regions of the image to achieve a compromise between noise reduction and temporal resolution [8].

Acknowledgments

This work was supported in part by NIH Grants R01-EB008425, R01-EB002873 and an equipment grant from Toshiba Medical Systems Corporation.

References

1. Keleshis, CM.; Ionita, CN.; Yadava, GK.; Patel, V.; Bednarek, DR.; Hoffmann, KR.; Verevkin, A.; Rudin, S. LabVIEW Graphical User Interface for a New High Sensitivity, High Resolution Micro-Angio-Fluoroscopic and ROI-CBCT System. SPIE vol. 6913, 69134A, March 2008. Proceedings from Medical Imaging 2008: Physics of Medical Imaging; San Diego, CA. p. 6913-196.

2. Wang, W.; Ionita, CN.; Keleshis, C.; Kuhls-Gilcris, A.; Jain, A.; Bednarek, D.; Rudin, S. Progress in the Development of a new Angiography Suite including the High Resolution Micro-Angiographic Fluoroscope (MAF), a Control, Acquisition, Processing, and Image Display System (CAPIDS), and a New Detector Changer Integrated into a Commercial C-Arm Angiography Unit to Enable Clinical Use. *Proc SPIE*; 2010 Mar 23; 2010. p. 76225I
3. Jain A, Bednarek DR, Ionita C, Rudin S. A theoretical and experimental evaluation of the micro-angiographic fluoroscope: A high-resolution region-of-interest x-ray imager. *Med Phys*. 2011; 38 (7):4112–4126. [PubMed: 21859012]
4. Binning MJ, Orion D, Yashar P, Webb S, Ionita CN, Jain A, Rudin S, Hopkins LN, Siddiqui AH, Levy EI. Use of the Micro-Angiographic Fluoroscope for coiling of intracranial aneurysms. *Neurosurgery*. Nov; 2011 69(5):1131–1138. post acceptance epub June 18, 2011. 10.1227/NEU.0b013e3182299814 [PubMed: 21694658]
5. [last accessed Jan 6 2012.] http://www.nvidia.com/object/cuda_home_new.html
6. [last accessed Jan 6 2012.] [http://en.wikipedia.org/wiki/File:CUDA_processing_flow_\(En\).PNG](http://en.wikipedia.org/wiki/File:CUDA_processing_flow_(En).PNG)
7. [last accessed July 11 211.] http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.3.pdf
8. Vasan, SN.; Swetadri; Sharma, P.; Ionita, CN.; Titus, AH.; Cartwright, AN.; Bednarek, DR.; Rudin, S. Spatially different, real-time temporal filtering and dose reduction for dynamic image guidance during neurovascular interventions. *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*; Aug. 30 2011–Sept. 3 2011; p. 6192-6195.

Figure 1.
Processing Flow on CUDA

Figure 2.

Thread Hierarchy structure in CUDA. A thread can be treated as the smallest execution unit. A group of threads form a block. A group of blocks form a grid. Thus an image can be treated as one grid, consisting of blocks, which in turn consists of threads and each thread can be treated as a pixel.

Figure 3.
Program flow in CAPIDS during a typical fluoroscopy run

Figure 4.
Pipelined architecture of previous version of CAPIDS.

Figure 5.
Pipelined architecture in the current version of CAPIDS.

Figure 6.
Screen shot of the Front Panel of the updated CAPIDS program