ECEN 5593: ADVANCED COMPUTER ARCHITECTURE

FINAL PROJECT

Image Processing Algorithms Implementation and Analysis in CUDA

SANJANA KALYANAPPAGOL
saka2821@colorado.edu

May 12, 2017

SPRING 2017

ABSTRACT:

Many image processing algorithms require dozens of floating point computations per pixel, which can result in slow runtime even for the fastest of CPUs. The slow speed of a CPU is a huge bottleneck to productivity. While increased pixel counts lead to greater area coverage and higher resolution, it also results in higher image processing time. So, the implementation method designed has significantly improve performance of image processing algorithms and real time image and video processing, the only overhead will be the data transfer times.

The project is primarily being focused on basic Image processing algorithms. The objective is to implement Image-Invert Algorithm and Grayscale algorithm for 32 bits per pixel (bpp) color encoded bitmap images and to measure execution times on CPU as compared to GPU. The project also aims to analyze the performance of the algorithms in different processors and present a detailed report regarding the same.

INTRODUCTION:

Using CUDA in a GPU environment, one can spawn one thread per pixel so that each thread is responsible for carrying out computations for one pixel. This design decreases the computation time hugely. In systems where the GPU is a peripheral device, the image is transferred to the GPU over PCI express bus and then the GPU takes over. The output image is sent back to the CPU over the same bus.

The primary difference between CPU and GPU model of execution is that its sequential in the CPU model while the GPU model involves multi-threaded execution. While processing images in a sequential model, the pixels are accessed sequentially and therefore all computations happen sequentially. In a multi-threaded environment, like the GPU environment, it is possible to have one thread access one pixel or just a few pixels and do the computation on that pixel or group. This is useful in image processing applications because these applications generally require certain filter convolution to be implemented on each pixel in which none of them have any correlation. Thus, parallel processing improves the time required to access each pixel and apply the filter on it.

```
//Iterates through every pixel
for(i=0;i<ImageWidth;i++)
{
   for(j=0;j<ImageHeight;j++)
  {
     image_processing_algo(); //Image Processing
Algorithm
   }
```

Fig 1:Pseudo Code for CPU Process

```
//Same task is performed by every thread

int i= blockIdx.y * blockDim.y + threadIdx.y;

int j= blockIdx.x * blockDim.x + threadIdx.x;

int globalId = i*ImageWidth + j;

uint32_t Pixel = InImage[globalId];

OutImage[globalId] = processImage(Pixel)
```

Fig 2: Pseudo code for GPU Process

IMPLEMENTATION:

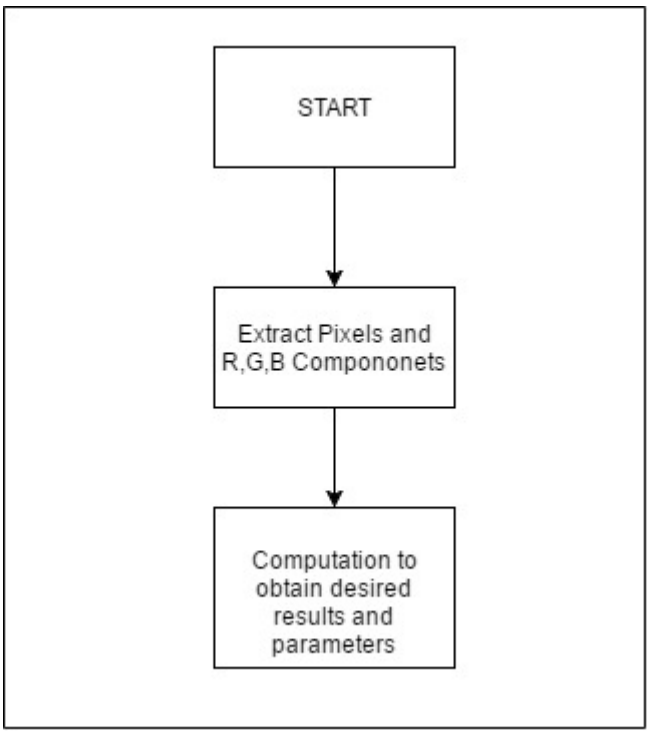

Fig 3: Flowchart of the Code

ALGORITHMS:

Image-Invert Algorithm:

This algorithm replaces a color for its complimentary color and it is also known as the negative effect. In this algorithm, we extract the color components of each pixel and replace them with their complimentary value. A complimentary value is computed by subtracting an intensity level from 255.
The image below shows the complimentary color to replace the original color:

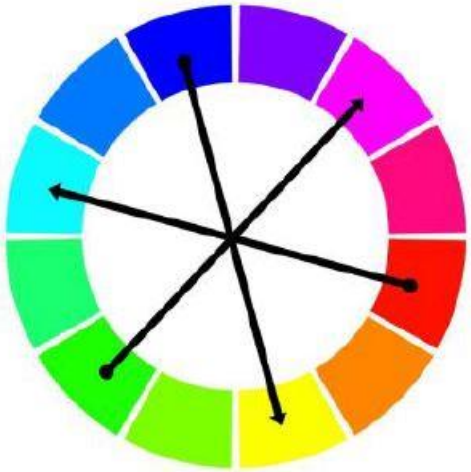

Fig 4: Complimentary Color Wheel

Gray-Scale Algorithm:

In photography and computing, a grayscale or greyscale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Grayscale images have many shades of gray in between. Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum .The filter implementing this algorithm transforms a colored image to grayscale. As mentioned earlier, in a 32-bit coloured image each of these three-color components occupy fixed positions spanning 8 bits. In a gray scale image, all the 3 bytes have the same value. To convert a color image to grayscale, the three-color components are extracted and averaged out. This average value is then assigned to the byte positions of the three-color components. The resulting image is in grayscale.

RESULTS AND ANALYSIS:

The algorithms were implemented for execution both in the host CPU and the NVIDIA GPU device. The algorithm wise timing analysis is as follows.

Image-Invert Algorithm:

Fig 5: Parrot.bmp original image(left) and inverted image(right)

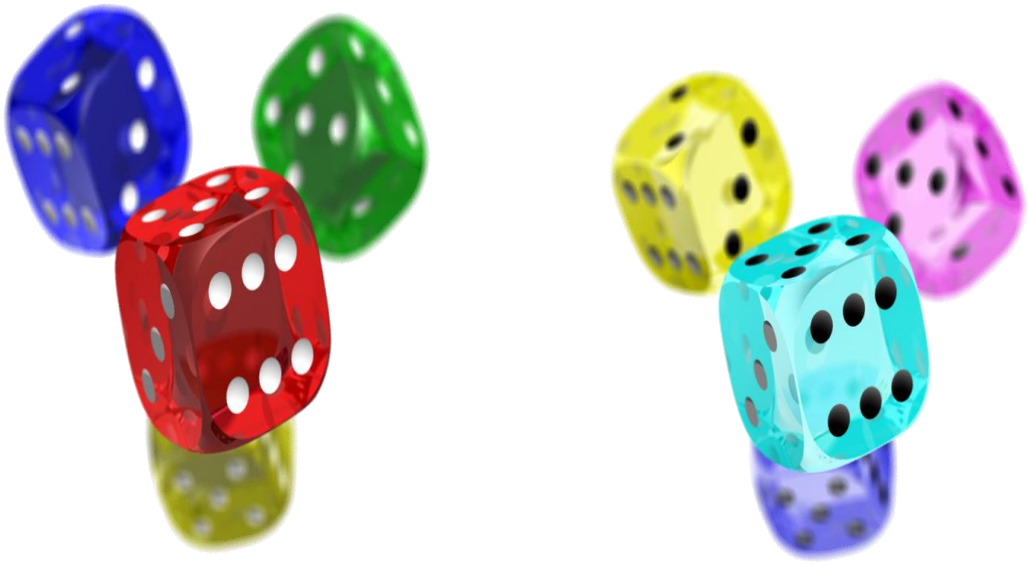

Fig 6: Dice.bmp original image(left) and inverted image(right)

Fig 7: Bouquet.bmp original image(left) and inverted image(right)

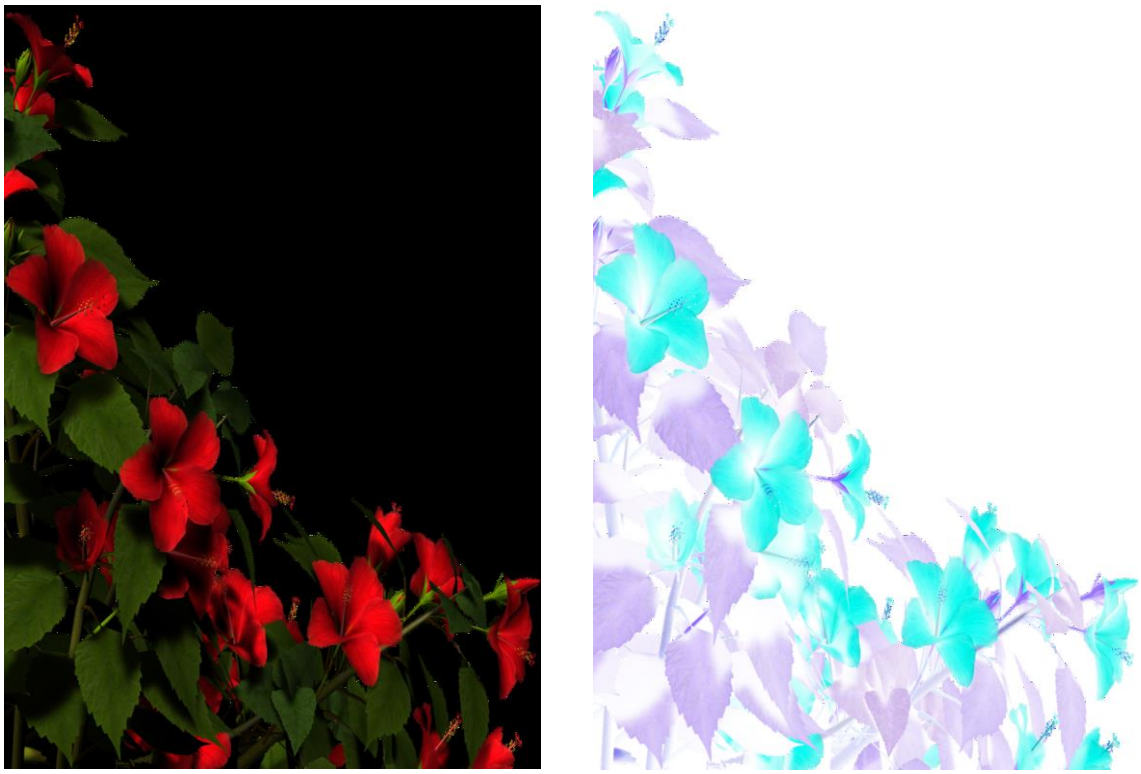

Fig 8: Hibiscus.bmp original image(left) and inverted image(right)

Fig 9: Rose.bmp original image(left) and inverted image(right)

Fig 10: Sunflower.bmp original image(left) and inverted image(right)

Timing Analysis of Inversion Algorithm:

Image	Image Size in Pixels	Total Memory Copy Time (in ms)	CPU Execution Time (in ms)	GPU Execution time (in ms)	Total Execution Time in GPU (in ms)	Speed Up
Parrot.bmp	360x360	0.279	0.836	0.139	0.418	6.01
Rose.bmp	700x700	0.962	3.177	0.351	1.313	9.05
Dice.bmp	800x600	0.955	3.169	0.337	1.292	9.40
Bouquet.bmp	827X720	1.18	3.854	0.378	1.558	10.19
Hibiscus.bmp	894x894	1.39	5.124	0.458	1.848	11.18
Sunflower.bmp	2014x2238	6.142	28.598	1.833	7.975	15.60

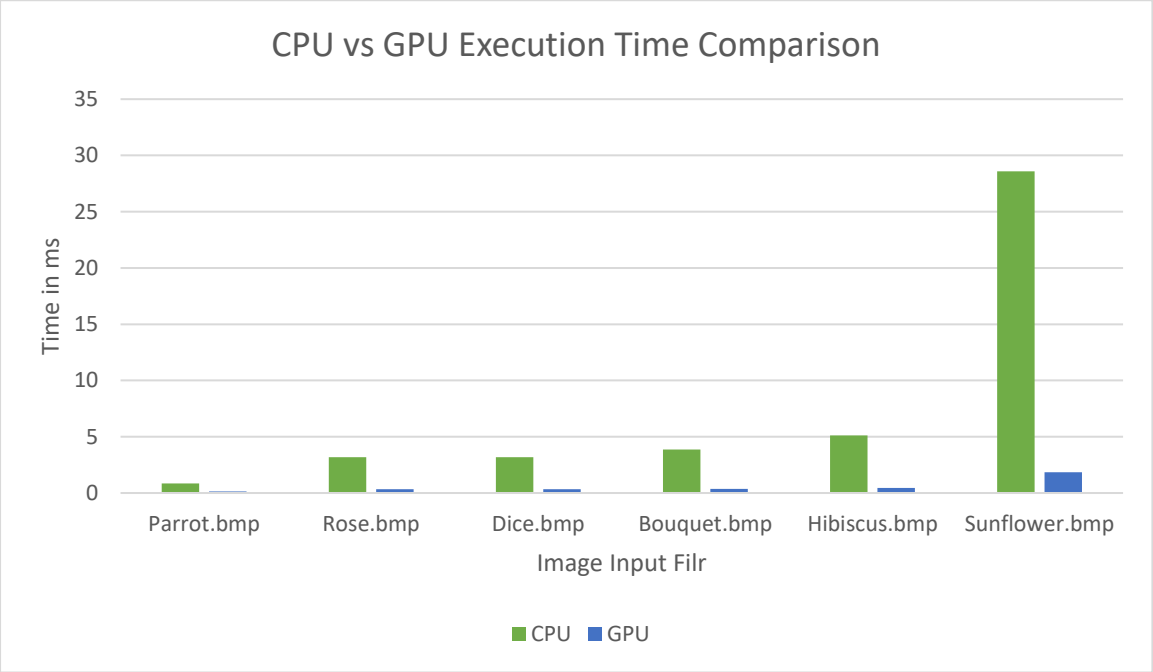

Fig 11: Execution time comparison in Inversion Algorithm

Gray-Scale Algorithm:

Fig 12: Parrot.bmp original image(left) and gray-scale image(right)

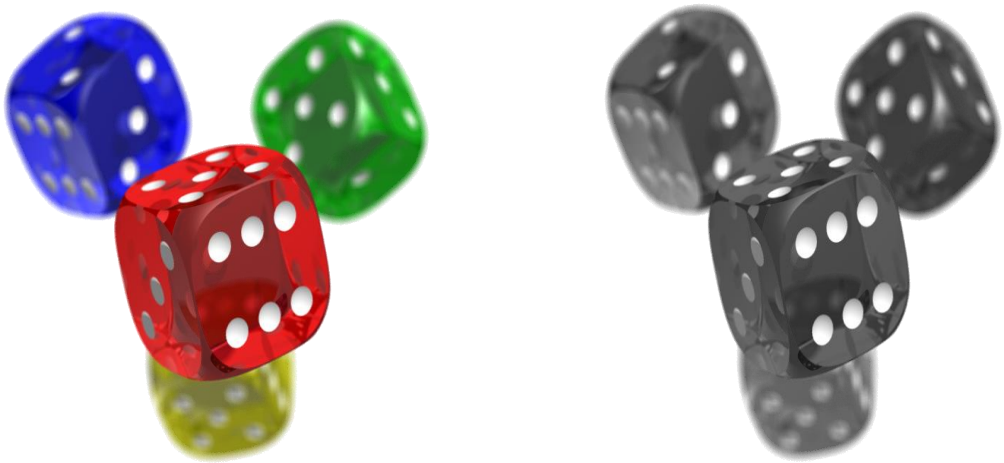

Fig 13: Dice.bmp original image(left) and gray-scale image(right)

Fig 14: Bouquet.bmp original image(left) and gray-scale image(right)

Fig 15: Hibicus.bmp original image(left) and gray-scale image(right)

Fig 16: Rose.bmp original image(left) and gray-scale image(right)

Fig 17: Sunflower.bmp original image(left) and gray-scale image(right)

Timing Analysis in Gray-Scale Algorithm:

Image	Image Size in Pixels	Total Memory Copy Time in ms	CPU Execution Time in ms	GPU Execution time in ms	Total Execution Time in GPU (in ms)	Speed Up
Parrot.bmp	360x360	0.276	1.019	0.142	0.418	7.17
Rose.bmp	700x700	0.89	3.977	0.366	1.256	10.86
Dice.bmp	800x600	0.993	3.881	0.352	1.345	11.03
Bouquet.bmp	827X720	1.172	4.793	0.396	1.568	12.10
Hibiscus.bmp	894x894	1.344	6.308	0.48	1.824	13.14
Sunflower.bmp	2014x2238	6.082	34.955	1.957	8.039	17.86

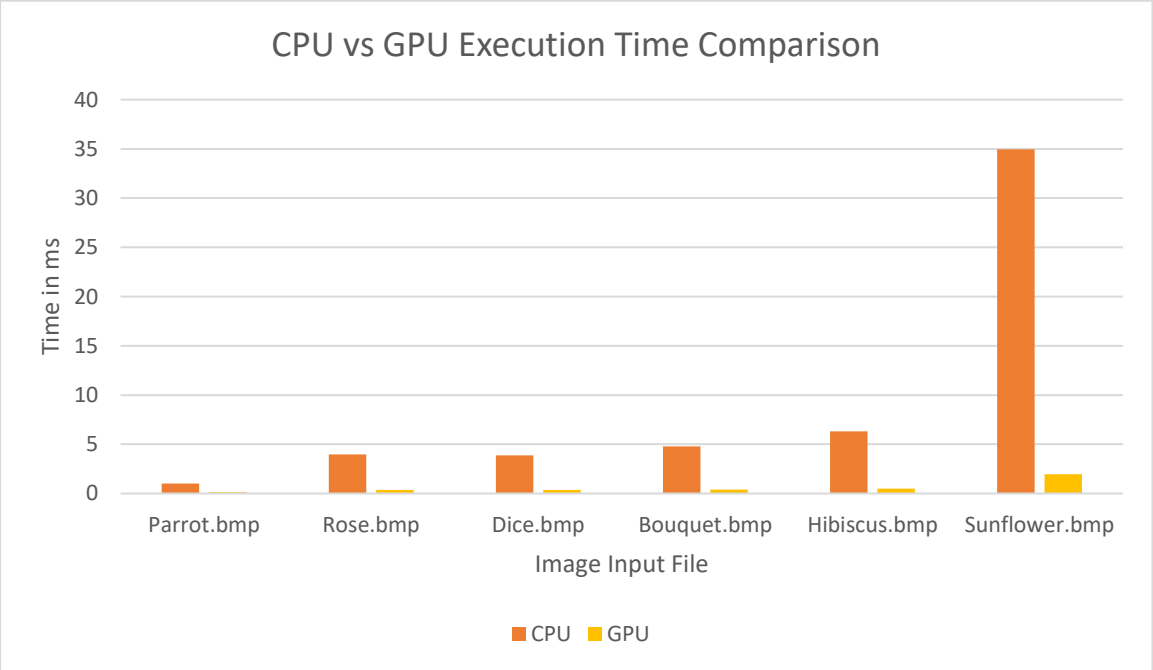

Fig 18: Execution time comparison in Gray-Scale Algorithm

CONCLUSION:

1. CPU execution time significantly increases as the complexity of image processing algorithm increases. Thus, there is a need of parallel processing.
2. Performance improvement in GPU is not significant for small images with respect to the CPU.
3. As per the observation, even with the kernel launch and memory transfer overhead, GPU execution time is around 16 times faster than CPU execution time for a very large image.
4. Memory transfer overhead from CPU to GPU and back is too large to be used for images of smaller size. It is better to do image processing for smaller images in the CPU itself
5. Kernel execution time is significantly less than CPU execution time for images of very large size 20Mb.

FUTURE SCOPE:

1. The algorithms can be implemented on various other sizes to perform the analysis.
2. Images used are 32 bit per pixel images, hence the algorithms can be designed to execute on other resolution images like 24 bit , 8 bit etc.
3. Many other trending image processing algorithms can be implemented and analysed.

REFERENCES:

- http://www.nvidia.com/content/nvision2008/tech_presentations/Game_Developer_Track/NVISION08-Image_Processing_and_Video_with_CUDA.pdf
- https://pngtobmp32.codeplex.com/downloads/get/341737
- http://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-7-colour-inversion-solarisation/
- http://supercomputingblog.com/cuda/advanced-image-processing-with-cuda
- https://en.wikipedia.org/wiki/BMP_file_format
- https://en.wikipedia.org/wiki/Grayscale

Images:

- Dice.bmp: https://upload.wikimedia.org/wikipedia/commons/4/47/PNG_transparency_demonstration_1.png
- Sunflower.bmp http://pngimg.com/download/13389
- Hibiscus.bmp http://moonglowlilly.deviantart.com/art/Hibiscus-Border-387233421
- Rose.bmp http://pngimg.com/download/638
- Bouquet.bmp http://pngimg.com/download/21672
- Parrot.bmp http://pngimg.com/download/723