CUDA Image Color Inversion

By

Mr.Peeranut Duangkaew	5988045
Mr. Chanwit Panleng	5988076
Miss Pitchaya Dachoponchai	5988097
Mr. Chaiyakarn Khanan	5988130

A Report Submitted in finished of

the Requirements for

ITCS443 Parallel and Distributed Systems

Faculty of Information and Communication Technology

Mahidol University

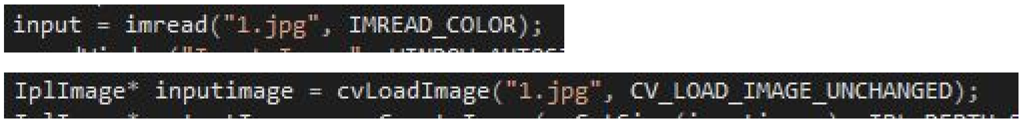
Introduction

Colour in RGB has value is specified with red, green and blue that each of value defines the intensity of the colour as an integer between 0 to 255. In this report, Colour in image inverts the colour that subtracts the value from 255 for each colour component by the CUDA program. Then, image output will show that image subtracts the value of each parameter.

Overview

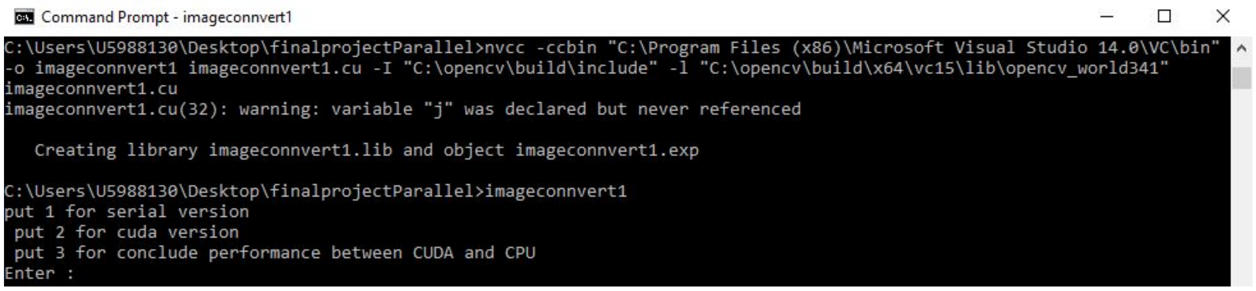
Algorithm which ours group use can explain follow this. First we create two pointer unsigned char for store two image which is input image and output image. After this we send unsigned char of input image to Cuda kernel with dim grid which already round up on the row and columns part. In the kernel we use 255 to minus all channel values in the picture and store in the new unsigned char which is output. After this we send the unsigned char back from GPU device to host. And the last we use the unsigned char to be output image.

Instructions



```
input = imread("1.jpg", IMREAD_COLOR);
IplImage* inputimage = cvLoadImage("1.jpg", CV_LOAD_IMAGE_UNCHANGED);
```

Input file name to code directly. After that on computer which will be sure to compile if it have opencv version 340-341. We use this command for run on command line. Command is nvcc -ccbin **"C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin" -o imageconnvert1 imageconnvert1.cu -I "C:\opencv\build\include" -l "C:\opencv\build\x64\vc15\lib\opencv_world341"** . After compile we should type the file name in command line which is imageconvert1. Follow picture below.

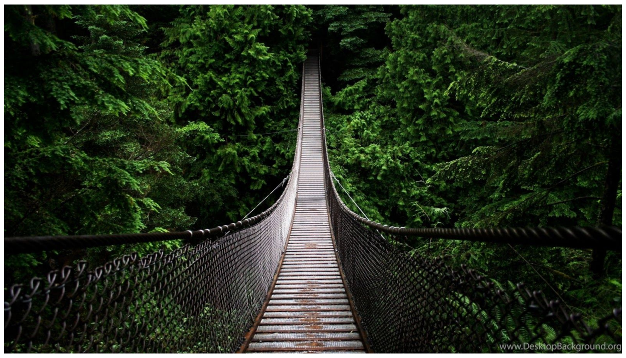


```
C:\Users\U5988130\Desktop\finalprojectParallel>nvcc -ccbin "C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin"
-o imageconnvert1 imageconnvert1.cu -I "C:\opencv\build\include" -l "C:\opencv\build\x64\vc15\lib\opencv_world341"
imageconnvert1.cu
imageconnvert1.cu(32): warning: variable "j" was declared but never referenced

   Creating library imageconnvert1.lib and object imageconnvert1.exp

C:\Users\U5988130\Desktop\finalprojectParallel>imageconnvert1
put 1 for serial version
 put 2 for cuda version
 put 3 for conclude performance between CUDA and CPU
Enter :
```

After this you can type which part you want to use. 1 and 2 will show input and output from CPU or GPU in order, but 3 for see performance of cpu and gpu.

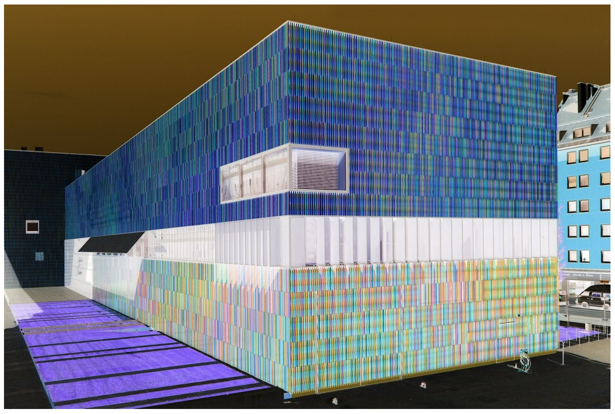
Experiment and results

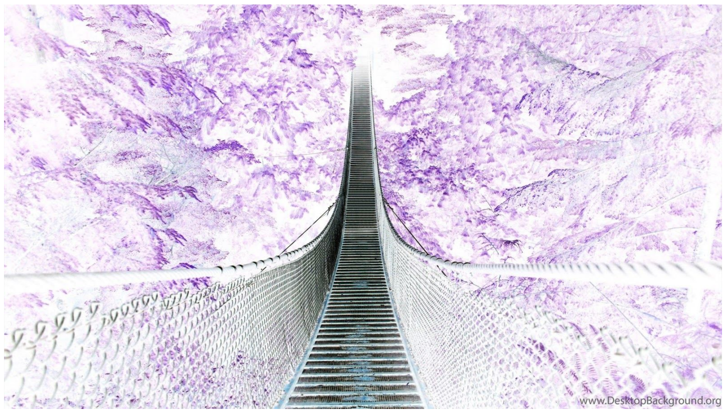
From the code which we already create, we use this picture for input



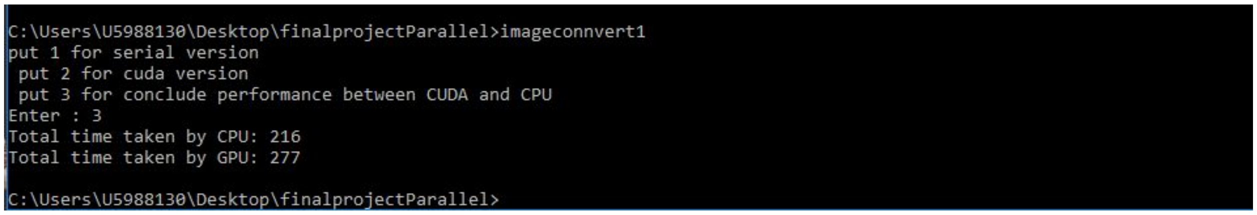
Figure[1]   Figure[2]

From this two picture, we decrease the size before show in this report. But The real size of figure 1 is 3872 x 2592 pixels. And the second figure is 2560 x 1440 pixels. We use this two picture for test the performance of process from cpu and process from Gpu. The output of this two picture show below.



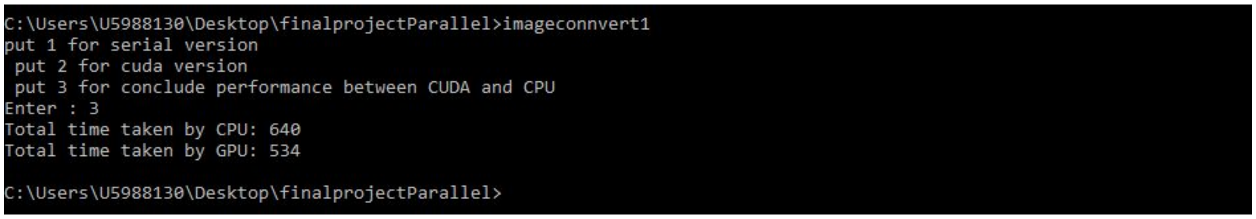


From immediately view we can conclude that it just a output of negative picture. But the performance when process with Cpu and Gpu have some result which show how Gpu improve to process and disadvantage of Gpu.



```
C:\Users\U5988130\Desktop\finalprojectParallel>imageconnvert1
put 1 for serial version
 put 2 for cuda version
 put 3 for conclude performance between CUDA and CPU
Enter : 3
Total time taken by CPU: 216
Total time taken by GPU: 277

C:\Users\U5988130\Desktop\finalprojectParallel>
```

From above picture is result of performance between use Cpu and Gpu to process Figure 2. From the result we can conclude that Cpu can process faster than Gpu



```
C:\Users\U5988130\Desktop\finalprojectParallel>imageconnvert1
put 1 for serial version
 put 2 for cuda version
 put 3 for conclude performance between CUDA and CPU
Enter : 3
Total time taken by CPU: 640
Total time taken by GPU: 534

C:\Users\U5988130\Desktop\finalprojectParallel>
```

But after we test another case which is figure 1. We can see that Gpu faster than Cpu. From research and result which we tested. We can conclude that Gpu will be faster than Cpu when the picture have big size and have lot of pixels. Not only size of image which make Gpu faster than Cpu but also have other thing which should have to make Gpu faster than Cpu. That is detail of picture. If picture aren’t too big but have more color or more detail can make Gpu will process faster than Cpu. So we can conclude that Gpu will process faster than Cpu some picture not only all picture. But if program have more things to do it good for let Gpu process on some part which Gpu can process.

Discussion and conclusion

Processing picture from normal to negative picture on Cpu and Gpu have to manage the value differently. Processing on Gpu will faster than Cpu only some case because of Gpu need to send values and receive values back, so this make Gpu can slower than Cpu if image are too small. Because we use send and receive time not enough to cover. But if image big enough, Gpu will be powerful to use to process on that image. So we conclude that cpu faster than Gpu only small image and low detail.

References

1. https://engineering.purdue.edu/~smidkiff/ece563/NVidiaGPUTeachingToolkit/Mod3/Lecture-3-3-color-to-greyscale-image-processing-example.pdf

2. https://github.com/drpaneas/cuda