Project 1

Programming Project:

Parallel Implementation of Color Conversion (Thresholding) on GPU

Instructions:

In this programming assignment, students are asked to write a program and a report on the program they developed, explaining their approach and results. The project involves taking a gray scale image and converting it to a black and white image using a technique known as intensity thresholding.

Please read the instructions here carefully and start working on your project as soon as possible to give yourself time to debug and complete the program.

1- Overview:

In image processing, the process of taking an input image and labeling its pixels into a binary assignment is called thresholding. This is an important process which takes place as a pre-processing stage to set up an image, or a sequence of images, for further processing and computation.

One of the major video processing techniques used for background subtraction and foreground tracking (also known as background modeling) relies on thresholding as an essential pre-processing stage.

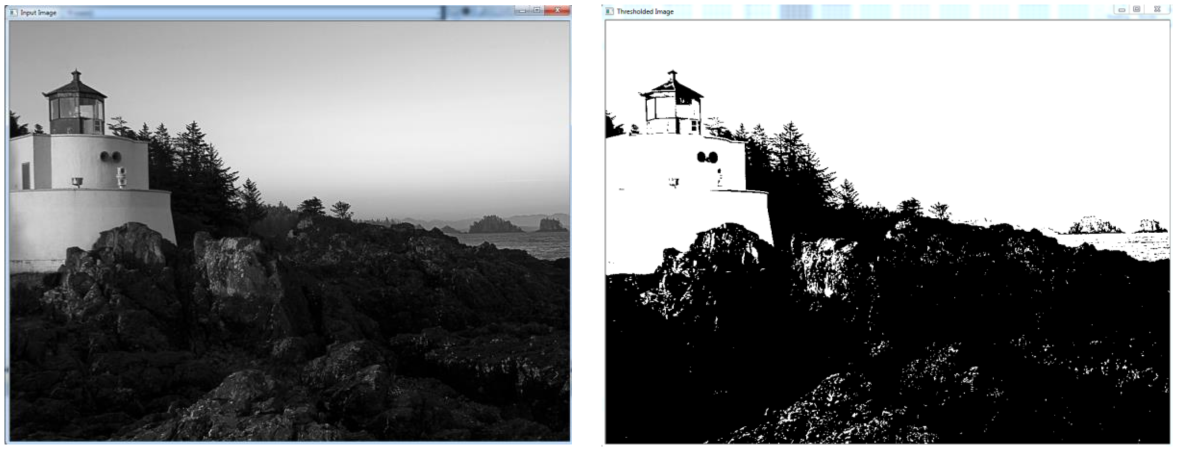
Obviously the thresholding model used in state-of-the-art background subtraction techniques could be a complicated process. But in essence, we have a process which takes each pixel from an input image (or in the case of background modeling a sequence of frames) and assigns to it a binary value (usually from the range <0 or 1> or <0 or 255>). The thresholding process takes each pixel value and labels them as black/white by comparing the value of the pixel with a threshold.

Suppose that we have an input image denoted as *I*. For each pixel at location (*i*,*j*), with *i* representing the *i*th row and *j* representing the *j*th column, we calculate the output image (*O*), using thresholding, by evaluating the following equation:

$$O(i,j) = \begin{cases} 0 & I(i,j) < th \\ 255 & I(i,j) \geq th \end{cases}$$

Where, *th* is a predetermined threshold value – in the range [0,1], or [0, 255].

Figure 1. shows an original gray-scale image (a), and the result of thresholding applied to the original image (b).



(a) – Original Image (b) – Thresholded Image

Figure 1. Image thresholding applied to a gray scale image (a), and the black and white result (b). A threshold of th=64 is used for this image.

2- Tasks:

a. Project Setup:

To set up your project, go to the Microsoft Visual Studio and create a new project. Base your new project on CUDA 4.x+ template. Then add a .cpp file to the project and from the project properties add your OpenCV libraries, include headers, and binaries to your project.

Remove the unnecessary items from the kernel.cu file. You will write your code into the kerne.cu file. NOTE: There is a shell template for your kernel.cu file in the attachment for your reference.

b. Write a serial version of the program

In the first part of the project you are asked to write a serialized code that takes as input an image (in gray-scale) and call a function to perform the thresholding process on the input image.

c. Accelerate your code with parallelizing the program using CUDA

Write a kernel in CUDA that takes as input two pointers on the device to the input data (which comes from the input gray scale image data) and the output data to perform the thresholding. Note that each thread will have an index and will be processing one pixel from the input data to generate one pixel for the output data.

You should particularly note that both the input and output image are on the CPU (host). Make sure that you define two pointers (one for the input and one for output) on your GPU (device). NOTE: the size of both GPU pointer must match the size of their CPU counterparts. Transfer the data from the input image from CPU to the GPU input pointer before invoking the kernel. After the Kernel runs to completion, use cudaDeviceSynchronize();. Then transfer the data from the output pointer on the GPU to the image data structure of the output image on the CPU.

3- Results:

a. Serial Results

Use the images available in the attachments of this project description from the blackboard to run your CPU side code. For each image use the imshow() statement to show the input image and the result of your serial CPU side code.

b. GPU Results

Use the images available in the attachments of this project description from the blackboard to run your GPU side code. For each image use the imshow() statement to show the input image and the result of your parallel GPU side code.

4- Deliverables

a. Code

Submit your kernel.cu code as an attachment to your submission. Please make sure you document your code.

b. Report

Write a report on your project. The report should be about 3-5 pages long and should contain your own description of both your CPU and GPU side code. The report should contain the following items:

- Write a short section on the project summary.
- Your implementation of both the CPU and GPU code. This section should include any parameters you used on the CPU side code as well as your GPU side code, such as; kernel size, block and grid sizes, thresholds, etc.
- Any issues encountered while working on this project, and your attempts to address the issues.
- Results of your program both on the CPU and the GPU.
- Your conclusions about the implementation and the techniques you used and how you see them suitable for a real-life application.