Lab 6

CMPUT 398

Objective

The lab's objective is to implement a simple image convolution and tiled image convolution using both shared and constant memory. We will have a constant 5x5 convolution mask, but will have arbitrarily sized image (assume the image dimensions are greater than 5x5 for this Lab).

For the simple image convolution, you shouldn’t use any constant or shared memory. Keep it simple. It should look very similar to the pseudo code shown below. However, for the tiled image convolution you need to use both shared and constant memory.

To use the constant memory for the convolution mask, you can first transfer the mask data to the device. Consider the case where the pointer to the device array for the mask is named M. You can use const float * __restrict__ M as one of the parameters during your kernel launch. This informs the compiler that the contents of the mask array are constants and will only be accessed through pointer variable M. This will enable the compiler to place the data into constant memory and allow the SM hardware to aggressively cache the mask data at runtime.

Convolution is used in many fields, such as image processing for image filtering. A standard image convolution formula for a 5x5 convolution filter M with an Image I is:

$$P_{i,j,c} = \sum_{x=-2}^{2} \sum_{y=-2}^{2} I_{i+x,j+y,c} * M_{x,y}$$

where $P_{i,j,c} P_{i,j,c}$ is the output pixel at position i,j in channel c, $I_{i,j,c}$ is the input pixel at i,j in channel c (the number of channels will always be 3 for this MP corresponding to the RGB values), and $M_{x,y}$ is the mask at position x,y.

This lab will be submitted as one zipped file through eclass. Details for submission are at the end of the lab.

Input Data

The input is an interleaved image of height x width x channels. By interleaved, we mean that the the element I[y][x] contains three values representing the RGB channels. This means that to index a particular element's value, you will have to do something like:

    index = (yIndex*width + xIndex)*channels + channelIndex;

For this assignment, the channel index is 0 for R, 1 for G, and 2 for B. So, to access the G value of I[y][x], you should use the linearized expression I[(yIndex*width+xIndex)*channels + 1].

For simplicity, you can assume that channels is always set to 3.

Instructions

Edit the code where the TODOs are specified and perform the following:

- Allocate device memory
- Copy host memory to device
- Initialize thread block and kernel grid dimensions
- Invoke CUDA kernel
- Copy results from device to host
- Deallocate device memory
- Implement the simple 2D convolution kernel with adjustments for channel
- Implement the tiled 2D convolution kernel with adjustments for channels
- Use shared memory to reduce the number of global accesses, handle the boundary conditions in when loading input list elements into the shared memory

Pseudo Code

A sequential pseudo code would look something like this:

```
maskWidth := 5
maskRadius := maskWidth/2 # this is integer division, so the result is 2
for i from 0 to height do
  for j from 0 to width do
    for k from 0 to channels
      accum := 0
      for y from -maskRadius to maskRadius do
        for x from -maskRadius to maskRadius do
          xOffset := j + x
          yOffset := i + y
          if xOffset >= 0 && xOffset < width &&
             yOffset >= 0 && yOffset < height then
            imagePixel := I[(yOffset * width + xOffset) * channels + k]
            maskValue := K[(y+maskRadius)*maskWidth+x+maskRadius]
            accum += imagePixel * maskValue
          end
        end
      end
      # pixels are in the range of 0 to 1
      P[(i * width + j)*channels + k] = clamp(accum, 0, 1)
    end
  end
end
```

where clamp is defined as

```
def clamp(x, lower, upper)
  return min(max(x, lower), upper)
end
```

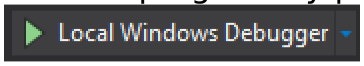
Local Setup Instructions

Steps:

1. Download “Lab6.zip”.
2. Unzip the file.
3. Open the Visual Studios Solution in Visual Studios 2013.
4. Build the project. Note the project has two configurations.
    a. Debug
    b. Submission

But make sure you have the “Submission” configuration selected when you finally submit.

5. Run the program by pressing the following button:



Make sure the “Debug” configuration is selected. Running the program in Visual Studios will run one of the tests located in “Dataset/Test”.

Testing

To run all tests located in “Dataset/Test”, first build the project with the “Submission” configuration selected. Make sure you see the “Submission” folder and the folder contains the executable.

To run the tests, click on “Testing_Script.bat”. This will take a couple of seconds to run and the terminal should close when finished. The output is saved in “Marks.js”, but to view the calculated grade open “Grade.html” in a browser. If you make changes and rerun the tests, then make sure you reload “Grade.html”. You can double check with the timestamp at the top of the page.

You will be given two executable files for reference on simple 2D convolution and tiled convolution (in the director “References”). **Do NOT submit these files.** On each version, your code run time on the **Submission configuration** on **Test 7** should be at most 3 times slower than the reference file of that version to get mark.

Write a report with screenshots show the run time of your code compare to the reference file of each version. **If you don’t submit this report, you will lose 20% of you mark.**

Submission

After you build your project with the “Submission” configuration selected (see above) you will see a new folder called “Submission” in the project directory. If you open the folder you will see your executable. Finally zip the folder “Submission” and upload the zipped folder to eclass.

If you don’t see this folder or are missing the executable, then one of two things happened. First your build could have failed and you should check for any errors in your build log in Visual Studios. Also you might have had the “Debug” configuration selected (see above). Make sure the “Submission” configuration is selected.

Mark Breakdown

It is important that you **do not** add or remove any print statements or wb functions. We use them for part of your grading so if you make changes the marking script could fail.

Part	**Breakdown (%)**
Convolution	50 %
Tiled Convolution	50 %