DUE DATE: Fri Nov 17 5pm (upload to polylearn)

Write a program in CUDA to Perform Image Blurring

1. **What is Image Blurring**

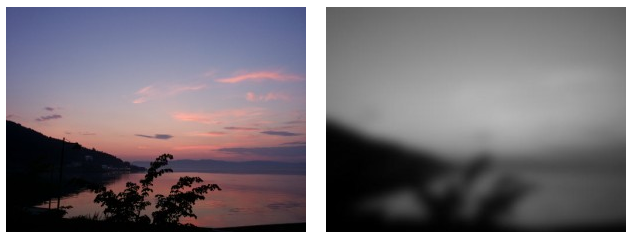


Figure 1 a) Original Image b) Blurred Image

Blurring is smoothing out and image as shown in Fig 1. Why would you want to do that? It may be for stylistic purposes (and that maybe the best reason), but it also serves as very important step in image/object recognition. For example, for Face Detection when you apply an edge detector algorithm to your original image you end with more edges than you need due to background noise. To get better, faster and more accurate edges is very common as a first step to apply a Gaussian blur to the image (Fig 2). Same principle of course apply to Medical Imaging, Scientific Visualization, etc

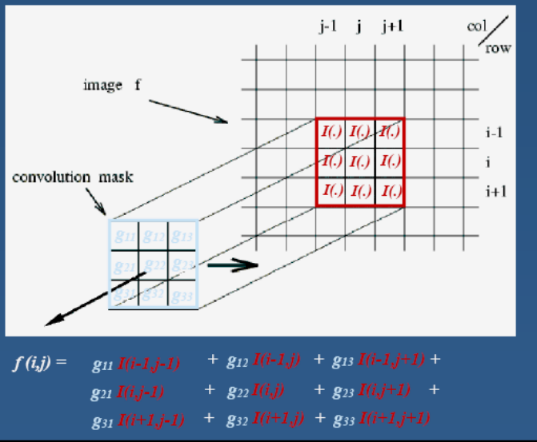
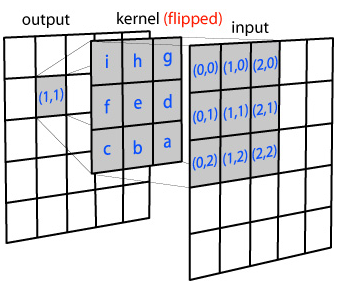
|  |  |  |
| --- | --- | --- |
| Original Image |  | Edge Detector without Blurring |
| Original Image | Image Blur | Edge Detector |

Fig 2 Edge Detector

1. **How to Obtain Image Blur**

Image Blurring is applied as image filtering in which the new image pixels values are calculated as a linear combination of the neighboring pixels. This Operation is called **convolution**, and the linear function applied is called a convolutional **kernel.**

If I call the convolutional Kernel *g* 3 by 3 matrix, the input Image is call *I*; and the output is *f*. where *f* is a convolution of :

What happens at the edges?

Need to do something with the edge pixels. Methods:

clip filter (black). Pad with 0's

wrap around . Pad with the edge mirrored

copy edge. Pad by just copying the edge

reflect across edge

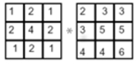
**3. The Gaussian filter is separable**

The above implementation in Fig 3 is easy to understand but is not very efficient.

The Gaussian convolutional kernel is separable can be expressed as the product of two functions, one a function of x and another as a function of y. This is important for a fast implementation since it will allow us to first do rows and then to columns of the image, which saves operations needed to be performed.

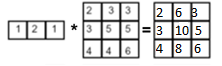
Example:

2D Convolution =

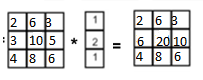
= 1\*2+2\*3+1\*3+2\*3+4\*5+2\*5+1\*4+2\*4+1\*6=65

The convolution can be calculated in 2 steps:

* + - 1. 1D convolution rows



* + - 1. 1D convolution of columns



The convolution of center pixel=2+6+3+6+20+10+4+8+6= 65

void conv(const double input[], size\_t inputLen,

const double filter[], size\_t filerLen, double result[])

{

for (size\_t n = 0; n < inputLen; n++)

start=n-(filterLen/2); Pvalue=0;

for (size\_t j = 0; j < filterLen); j++){

if (start+j>=0 && start+j<inputLen)

Pvalue += input[start+j]\*filter[j];

}

result[n]=Pvalue;

}

The code in C for a 1D convolution ( O(n))

Advantages of a separable convolutional filter:

Can convolve all rows, then all columns (better for Hardware implementations)

Computational complexity? Linear versus quadratic mask size

This is the 5\*5 gaussian convolutional kernel you should use for this assignment

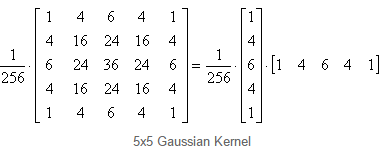


Figure 4 Gaussian Blur 5x5 filter

**Your Job**:

Implement Image Blurring in CUDA.

We are provide you with code to read and write an image into a matrix .

You should use the Convolutional kernel from figure 4 and implement a separable convolution

You will need to blur the Image 30 times

You will need to do the blur on the color image not on the gray

What to turn in, write a report that contains the following:

1. Names of all group members
2. Test the code with some images , we will provide some default ones

Show the original and blurred image

1. Compare execution times for the GPU implementation against the graphics section(correctness, you should get same pixel values)
2. Appendix with all your code
3. Extra credit do convolution in 2D and compare with the two 1D implementation
4. Extra Extra do 1D or 2D in shared memory