

Lab 7

DUE NOV 10 5pm. Work can be done in groups of two people

Write Histogram equalization in Cuda

Histogram equalization is a method in image processing of contrast adjustment using the Image's histogram. Used in different fields from Biomedicine to photography

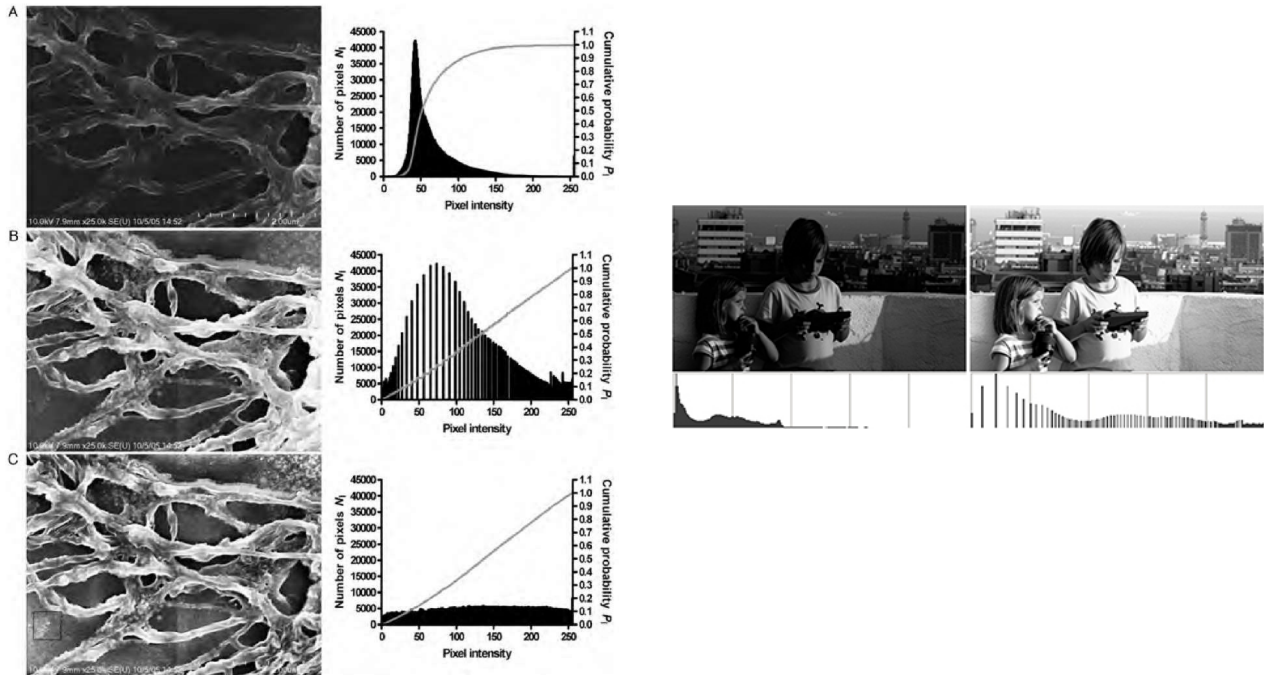


Fig 1 Histogram equalization examples

1. First Step to implementing a Histogram equalization is to create the histogram, an input image can be treated as a 2D Array where the cells values are actually the pixel values ranging from 0-256 creating a histogram then consist in going cell by cell in the array and add the cell value to the corresponding bucket in the histogram.

Since the cell can only have values from 0-256 then the number of buckets needed for the histogram is 256 . For example if a image size is 512x256 rows=512 and col=256 if the pixel value at row=x and col=y is 102 (inputImage[x][y]=102) then the bucket in the histogram for value 102 should increase by one $\text{histogram}[102] += 1$.

The code in C:

```
for (i=0; i<rows i++)  
    for(j=0;j<cols;j++)  
        histogram[InputImage[i][j]]++;
```

To transform the above code to CUDA (or any other parallel programming language) you need to solve the race condition produced by the ++ (read/modiy) operation. Be careful how you do this or you end up slowing down the execution time. The best performance will be obtained when using shared memory and using atomics as little (as fewer instructions involved) as possible. You are definitely allowed, and you should, use atomics

2. Step to normalize the histogram. After the histogram is created we normalize it as follows
a. Based in the histogram calculated I 1. we calculate now the cumulative distribution function
 $\text{cdf}(x) = \text{cdf}(x-1) + \text{cdf}(x-2) + \dots + \text{cdf}(0)$

Original Image Pixel values

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 52 | 80 | 80 | 90 | 92 | 94 | 101 | 110 |
| 55 | 81 | 82 | 91 | 100 | 95 | 100 | 110 |
| 120 | 55 | 85 | 86 | 89 | 90 | 92 | 92 |
| 121 | 180 | 55 | 100 | 101 | 123 | 92 | 92 |
| 122 | 190 | 58 | 58 | 101 | 125 | 61 | 61 |
| 133 | 191 | 192 | 101 | 112 | 134 | 61 | 61 |
| 145 | 191 | 191 | 59 | 123 | 112 | 111 | 112 |
| 150 | 191 | 110 | 59 | 59 | 60 | 111 | 112 |

Example 1st column is the pixel value, 2nd column is the histogram for that pixel, 3rd column is the cumulative distribution function and the 4th column is the equalized pixel value

| Pixel Intensity | Histogram (count) | cdf(v) | h(v) equalized0 using eq.1 |
|-----------------|-------------------|--------------------------|-------------------------------|
| 52 | 1 | 1 | $(1-1)/\text{size} * 255 - 1$ |
| 53 | 0 | $1+0=1$ | 0 |
| 54 | 0 | $1+0+0=1$ | 0 |
| 55 | 3 | $1+0+0+3=4$ | $(4-1)/\text{size} * 255 - 1$ |
| 56 | 0 | $1+0+0+3+0=4$ | 12 |
| 57 | 0 | $1+0+0+3+0+0=4$ | 12 |
| 58 | 2 | $1+0+0+3+0+0+2=6$ | 20 |
| 59 | 3 | $1+0+0+3+0+0+2+3=9$ | 32 |
| 60 | 1 | $1+0+0+3+0+0+2+3+1=10$ | 36 |
| 61 | 4 | $1+0+0+3+0+0+2+3+1+4=14$ | 53 |

You can write the code for this function in the CPU (serial code). For extra credit you can do this in the GPU code is explained in chapter ... in book

b. Once you have the $\text{cdf}(v)$ just apply the following function

eq.1 $h(v) = \text{round}(\text{cdf}(v-1) * 254 / \text{row} * \text{col})$

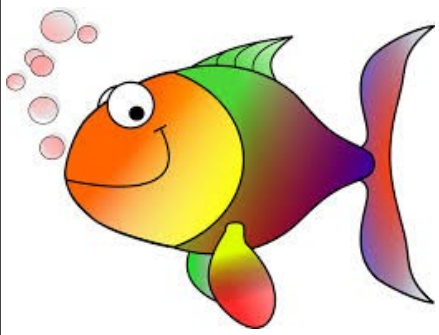
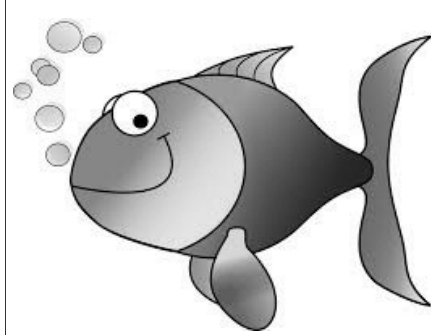
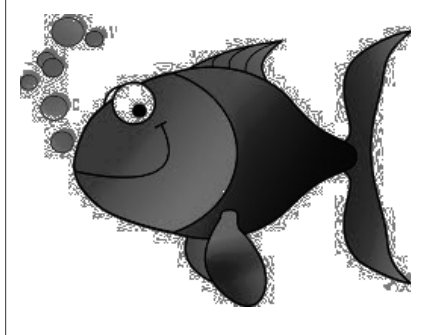
Perform this calculation on the GPU and write the end result to the output file

c. Equalized Image Pixel Values

Output Equalized Image Pixel values

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 0 | | | | | | | |
| 12 | | | | | | | |
| | 12 | | | | | | |
| | | 55 | | | | | |
| | | 20 | 20 | | | 53 | 53 |
| | | | | | | 53 | 53 |
| | | | 32 | | | | |
| | | | 32 | 32 | 36 | | |

Result

| Original Image | GrayImage | Equalized Image |
|--|---|--|
|  |  |  |

So as a summary Histogram equalization consist of the following steps

1. read Image and change it to gray (This is already done in the code that I provided for this lab)
2. Calculate histogram of the gray image created in step 1. This function you MUST write it so it does run on the GPU
3. Calculate the cumulative function. This function you can write on the GPU (for 10 extra credits) or in the CPU

for all elements in histogram

$$\text{cum}(x) = \text{cum}(x-1) + \text{histogram}(x)$$

for all pixel coordinates, x and y , do

4. Normalize the cumulative function obtained in 3 using equation 1. This again can be done on the CPU or in the GPU (but no extra credit since this one is trivial)

5. Modify the original pixel values with the equalized values obtained in 4

so if Original Pixel value was 61 in the Result image all this values will be converted to 53

for all pixels in image

$$g(x, y) = \text{cum}[f(x, y)]$$

Task:

1. Use your previous lab to convert color image to gray as a base code for this project
2. Building a histogram for the grayimage in CUDA. Note: you need to create a new cuda Kernel do not modify the Image to Gray kernel

3. Write cumulative distribution function in C (CPU) (should be no more than 5 lines of code). If you want 10 extra points write this one also in CUDA a basic kernel (good enough for the extra points) can be found in Figure 9.7 book page 209
4. Normalize the cumulative function obtained in 3 . Note: this step can be combined with step 3 if you want
5. Write the result of step 3 back into an image (Using PostProcessing Provided Function)
6. Upload to polylearn a document with the following:
 - a. Kernel for step 2
 - b. functions or Kernels for step 3 and 4
 - c. result Original Image (any picture you like) vs equalized image

Note: I will ask you to demo this code for me in lab or before or after the class , the code MUST work on the lab machines