Final Labwork Advanced Programming for HPC

Ta Quang Hieu - M22ICT002

November 2023

1 The problem

This project concerns image contrast enhancement using some histogram equalization techniques. During this project, I will implement various methods on the GPU to solve this problem, using the following techniques :

- HE: classical Histogram Equalization.
- AHE: Adjusted Histogram Equalization.
- WHE: Weighted Histogram Equalization.
- EIHE.
- MMSICHE.
- CLAHE: Contrast Local Adaptive Histogram Equalization.
- ACLAHE: Automatic Contrast Local Adaptive Histogram Equalization.

To study the efficiency of the algorithms, I will use two different metrics:

- AME which is the sum of the absolute value of the difference per pixel. This is a reduction made from the transformation of two images.
- Entropy: it relies on the global histogram of the image.

2 Introduction

During the project week, because I have some issues in my family - as I have emailed you before - so during this week I need to do the project for both parts of this course Dr. Son part, and also yours. This makes me unable to finish all the other parts but I have tried my best with this project. In this project, I have done the implementation of HE, AHE and EIHE algorithm, I also tried to finish WHE but I don't understand it, for MMSICHE, CLAHE and ACLAHE, I haven't been able to read them. For the original image that I use, please take a look at Figure 1.

Figure 1: The original image.

3 The general steps

The general steps of these Histogram Equalization algorithms are listed below:

- Convert the image from RGB to HSV and we will modify the V channel
- Calculate the histogram (and the modified histogram - if not the original HE) for the V channel
- Calculate the CDF from the histogram
- Mapping each pixel from the original image based on the CDF

4 Implementing Histogram Equalization algorithms

4.1 Convert the image from RGB to HSV

After loading the image, I will use a function to convert the image from RGB to HSV in order to use the V channel for calculating the histogram.

This is the function that I used to turn the image from RGB to HSV.

```
from numba.typed import List

#scale image [0..255] to [0..1]
@cuda.jit(nopython=True)
def scale(x):
  x = x/255
  return x

#find max and min
@cuda.jit(nopython=True)
def min_and_max(red, green, blue, src):
  #find max
  if red > green and red > blue:
    max_c = (red, 0)
  elif green > red and green > blue:
    max_c = (green, 1)
  else:
    max_c = (blue, 2)

  #find min
  if red < green and red < blue:
    min_c = (red, 0)
  elif green < red and green < blue:
    min_c = (green, 1)
  else:
    min_c = (blue, 2)
```

```
  delta = max_c[0] - min_c[0]

  return (max_c, min_c, delta)

@cuda.jit
def rgb2hsv(src, dst):
    tidx = cuda.threadIdx.x + cuda.blockIdx.x * cuda.blockDim.x
    g = np.float32(((src[tidx, 0] + src[tidx, 1] + src[tidx, 2]) / 3))

    red = scale(np.float32(src[tidx, 0]))
    green = scale(np.float32(src[tidx, 1]))
    blue = scale(np.float32(src[tidx, 2]))

    # min_c, max_c, delta = min_and_max(red, green, blue, src)
    max_c, min_c, delta = min_and_max(red, green, blue, src)
    H = 0
    S = 0
    V = 0
    if delta == 0:
      H = 0
    elif max_c[1] == 0:
      H = 60*(((green-blue)/delta)%6)
    elif max_c[1] == 1:
      H = 60*(((blue-red)/delta)+2)
    else:
      H = 60*(((red-green)/delta)+4)

    if max_c[0] == 0:
      S = 0
    else:
      S = delta/max_c[0]

    V = max_c[0]

    dst[tidx, 0] = H
    dst[tidx, 1] = S
    dst[tidx, 2] = V
```

Figure 2 shows the result obtained after converting the image from RGB to HSV.

4.2 The classical Histogram Equalization

I applied the basic steps as in section 3, and Figure 3 is the resulting image, and we can see that this image looks very sharp.

Figure 2: The original image in HSV color space.

Figure 3: The classical Histogram Equalization

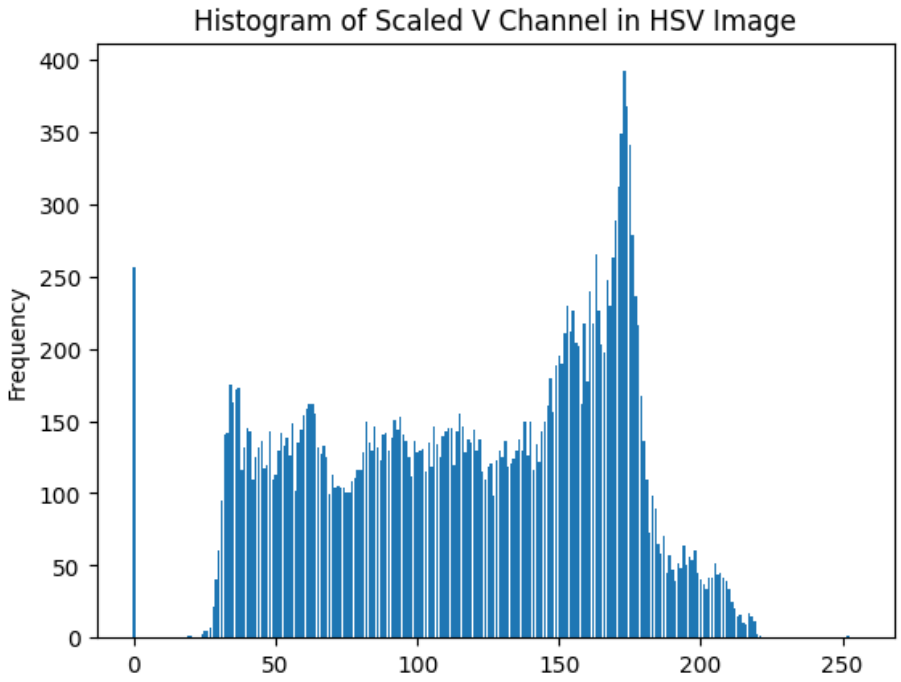

Figure 4: The original histogram

For more information Figure 4 shows the original image histogram and Figure 5 shows the CDF calculated from the histogram:

For this part, I tried to follow the coding style that you had shown us during the lecture week but I was still unable to follow it and needed to re-implement some kernel.

4.3 The Adjusted Histogram Equalization

For this part, I need to modify the original histogram to create a new histogram to make the histogram more uniform. The formula that I use to find the new histogram is shown in Figure 6.

In which:

- hi is the input original histogram as in the previous part
- u is the uniform histogram where all the column in this histogram has equal value
- lamba is the parameter that we need to choose

For more information Figure 7 shows the modified image histogram and Figure 8 shows the CDF calculated from this histogram. We can see that using

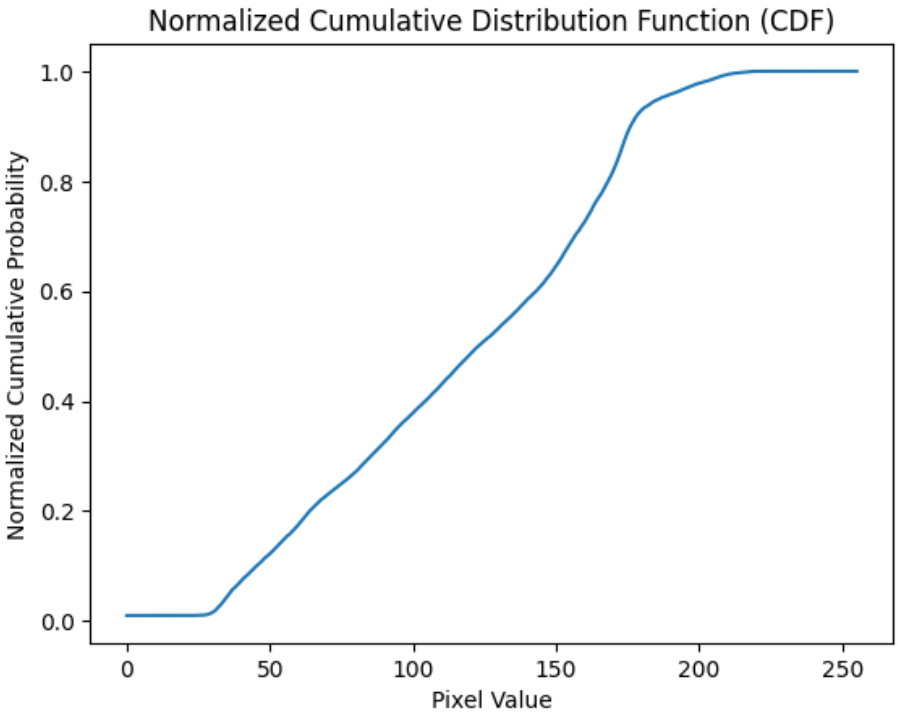

Figure 5: The original CDF

$$\tilde{\mathbf{h}} = \frac{\mathbf{h_i} + \lambda \mathbf{u}}{1 + \lambda} = \left(\frac{1}{1+\lambda}\right)\mathbf{h_i} + \left(\frac{\lambda}{1+\lambda}\right)\mathbf{u}.$$

Figure 6: The formula to find new histogram of AHE

AHE, the cdf now looks really like a straight line.

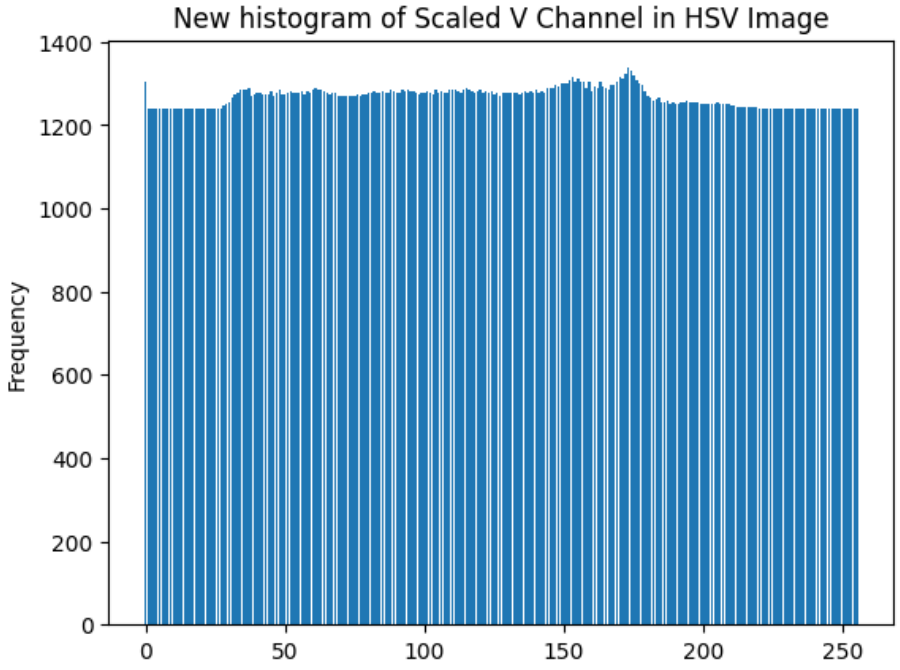

Figure 7: The modified histogram

Figure 9, Figure 10 and Figure 11 show the resulting image with lambda is 1, 3, 5 respectively.

As you can see the image from this algorithm looks more natural compared to the classical one.

4.4 Weighted Histogram Equalization

For this part, I also need to modify the original histogram to create new histogram. The formula that I use to find the new histogram is shown in Figure 12.

In which:

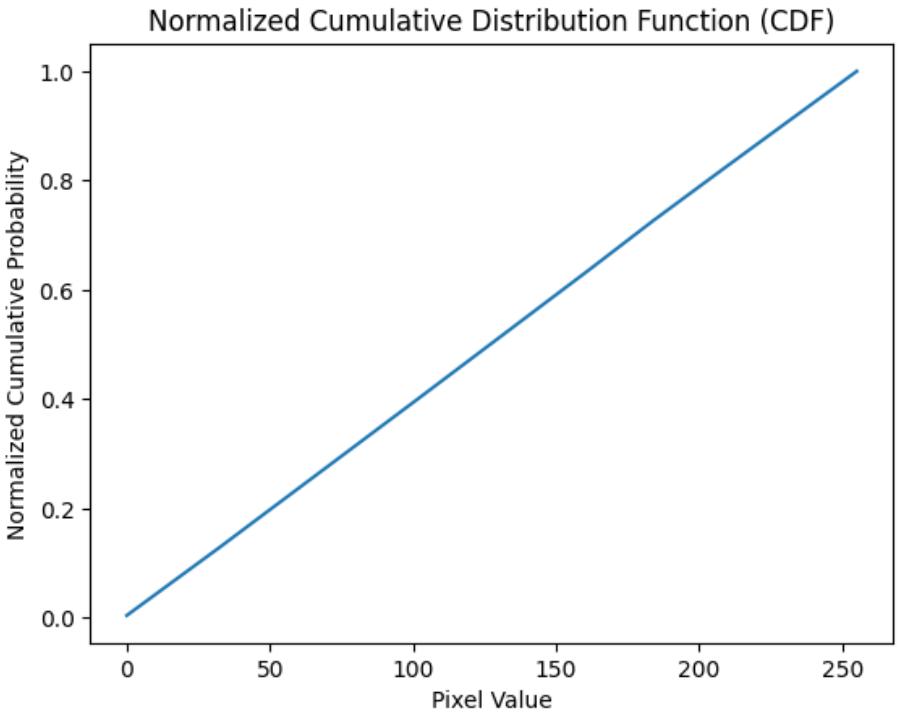

Figure 8: The new CDF

- W is the diagonal error weight matrix
- hi is the input original histogram as in the previous part
- u is the uniform histogram where all the column in this histogram has equal value
- lamba is the parameter that we need to choose

In this part, I have tried to figure out how to take an invert of a matrix using cuda and perform the multiplication but I wasn't successful in doing so, and overall in this part, I was only able to calculate the W matrix but still only in CPU and I also not sure that I is the correct one :(

4.5 EIHE

For this part, I follow the Algorithm 1 as stated in the paper in the bibliography of the project.

Figure 9: The AHE lambda 1

Algorithm 1 ESIHE Algorithm

Require: A grayscale image I

Ensure: A histogram-equalized image J

1: Compute the histogram $h(k)$ of I
2: Compute the value of exposure and threshold parameter x_a
3: Compute the clipping threshold T_c, and clip the histogram $h_c(k)$.
4: Divide the clipped histogram into two sub-histograms using the threshold parameter x_a
5: Apply the histogram equalization on individual sub-histograms.
6: Combine the sub-images into one image J for analysis
7: **Return:** J

Figure 10: The AHE lambda 3

For more information Figure 13 shows the clipping histogram. And you can take a look at the notebook for other figures of this part

Figure 14 is the resulting image, and we can see that this image shows

Figure 11: The AHE lambda 5

some really weird parts and I think that I have made some mistakes in the implementation of it.

$$\tilde{\mathbf{h}} = (\mathbf{W} + \lambda \mathbf{I})^{-1}(\mathbf{W}\mathbf{h_i} + \lambda \mathbf{u}).$$

Figure 12: The formula to find new histogram of WHE

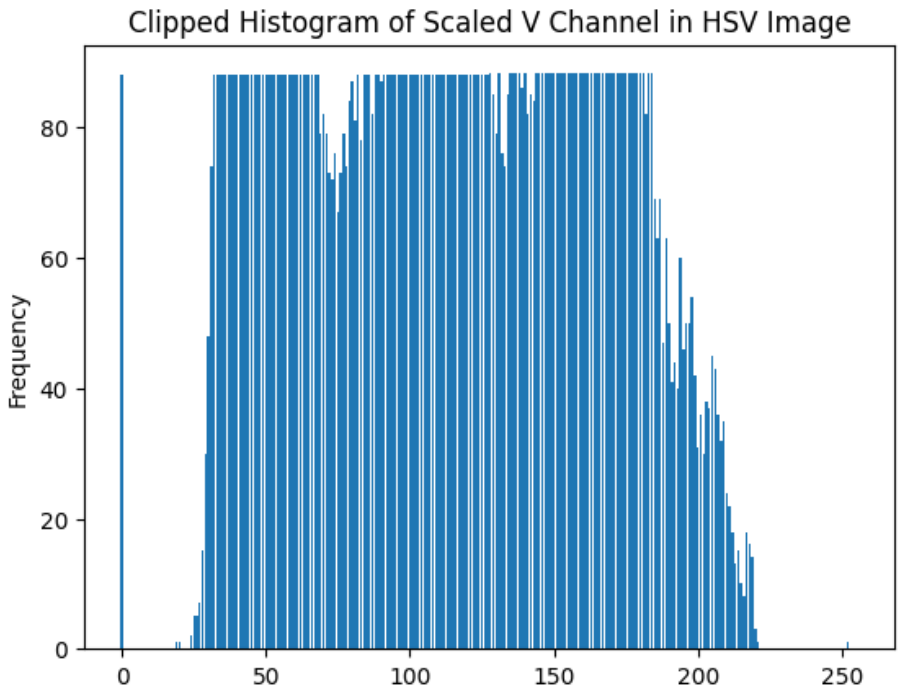

Figure 13: The clipping histogram of EIHE.

5 Comparision

Table 5 shows the result of 2 metrics for these algorithms. The entropy is used to measure the content of an image, where a higher value indicates an image with richer details and the method with the highest entropy is EIHE and it is also the method that gives the most difference to the original image - through AME.

	HE	AHE	EIHE
AME	56000788.0	56020416.0	58478404.0
Entropy	7.4060	7.3808	7.5297

Figure 14: The EIHE result.

Table 5 shows the running time of the algorithms. This result is measured by running the file 5 times for each algorithm and taking the average result of them.

	HE	AHE	EIHE
Running time	2.7003s	4.1533s	7.645s

6 Conclusion

What I have done:

- Implementation of 3 algorithms HE, AHE and EIHE
- I tried to work with WHE but I can only find the W matrix
- All of the calculations are performed by Google Colaboratory

What is still missing and needs improvement:

- The code is not optimized and does not follow the format that you have shown us, I really need to reformat the code but it is too rushed for me :(
- There are still a lot of algorithms that I haven't implemented like WHE and I wasn't able to read about like MMSICHE, CLAHE, and ACLAHE
- The result of EIHE still has some weird parts.

Overall, I have tried my best with this project and I have been able to be more familiar with the concept of the course through this project.