# Advanced High Performance Computing - Final Project Report

Histogram Equalization Implementation on GPU

Hoang Manh Truong

2023-11-12

## 1 Introduction

The goal of the project is to implement Histogram Equalization (HE) using CUDA. The project is implemented in Python using `numba`. The following HE methods are implemented:

| Method | Is implemented |
|---|---|
| HE | Yes |
| AHE (Adjustable HE) | Yes |
| WHE (Weighted HE) | Yes |
| EIHE/ESIHE (Exposure-based Sub Image HE) | Yes (but result is not very good) |
| MMSICHE | No |
| CLAHE | No |
| ACLAHE | No |

In the next section, each method is explained in detail.

## 2 Implementation and Results

### 2.1 RGB-HSV conversions

Since we are working with colored images, we need to convert the RGB images to HSV images. After that, the HE methods are applied on the $V$ component, or the luminance component. This means that the $V$ component needs to be extracted from the HSV image. After the HE methods are applied, the $V$ component is merged back into the HSV image and the HSV image is converted back to RGB.

The implementation of the RGB-HSV kernels can be found in the following functions in the notebook:

- `rgb_to_hsv`
- `hsv_to_rgb`

### 2.2 Histogram Equalization (HE)

The HE method is implemented in 3 steps:

1. Calculate the histogram of the $V$ component
2. Calculate the cumulative distribution function (CDF) of the histogram
3. Equalize the $V$ component using the CDF

In fact, all other HE methods also follow this process, with some modifications. The implementation of the HE method can be found in the following functions in the notebook:

- `compute_hist`
- `compute_cdf`
- `equalize_hist`

The result for the HE method is shown in Figure 1. The original image shows two different regions, one being the sky and the other being the stone gate that is in the shadow. In other words, the sky region is correctly exposed while the stone gate region is underexposed. After applying the HE method, the stone gate region has now become brighter, and we can see more details in the image that were not clearly visible before.

(a) Original Image (b) HE result

Figure 1: Original image vs. HE result

## 2.3 Adjustable HE (AHE)

The AHE method follows the same process as the HE method, but with some modifications to the histogram computation. In particular, the following functions are used:

- `compute_adjusted_hist`
- `compute_cdf`
- `equalize_hist`

For this method, a parameter $\lambda$ is used to control the level of contrast enhancement applied to the image. Note that when $\lambda$ is equal to zero, the result of this method is identical to the result of the HE method as discussed in Section 2.2. The result for the AHE method is shown in Figure 2. As we can see, there are a small difference between the results when $\lambda > 0$ versus when $\lambda = 0$. Apparently, when $\lambda > 0$, the darker regions in the stone remain quite dark, while in the result for $\lambda = 0$, the image seems to uniformly become brighter.

(a) Original Image (b) $\lambda = 0$ (c) $\lambda = 1$ (d) $\lambda = 2$

Figure 2: Original image vs. AHE results

## 2.4 Weighted HE (WHE)

The WHE method follows the same process as the HE method, but with some modifications to the histogram computation. In particular, the following functions are used:

- `compute_weighted_hist`
- `compute_cdf`
- `equalize_hist`

For this method, a weight map $W$ is used to control the level of contrast enhancement applied to the image. In particular, the weights are controlled such that the darker regions in the image are enhanced more than the brighter regions (see code below). The result for the WHE method is shown in Figure 3. As we can see, compared to the HE and AHE method, the image using the WHE method seems to have a more balanced exposure, with very dark regions staying dark.

```
for i in range(128):
    W[i] = 2
for i in range(128, 256):
    W[i] = 0.5
```

(a) Original Image (b) HE result

Figure 3: Original image vs. HE result

## 2.5 Exposure-based Sub Image HE (EIHE/ESIHE)

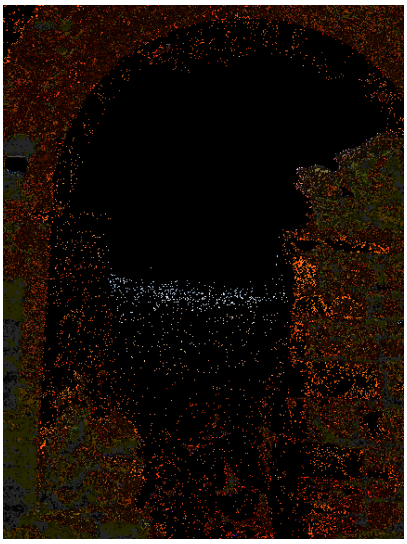
Unfortunately, the EIHE method is implemented in this project, however the result is not very good (as shown in Figure 4). In the image after being applied with EIHE, the sky region becomes very dark, while the stone gate region is able to keep its original color. This might be due to an incorrect implementation of the EIHE method. However, the overal principle of the EIHE method is as follows:

1. Compute the histogram of the $V$ component. For this step, we use `np.histogram` instead of using the GPU.
2. The two parameters $X_a$ and $T_c$ are computed, which are the exposure and threshold parameter and the lipping threshold. $T_c$ is used for clipping the histogram to prevent over-enhancement.
3. The histogram is divided into two sub-histograms using $X_a$.
4. Apply the histogram equalization on individual sub histograms.
5. Combine the sub-images into one image.

## 2.6 Runtime and metrics

The project uses the following metrics to evaluate the performance of the HE methods:

(a) Original Image



(b) HE result

Figure 4: Original image vs. HE result

- AME (Average Mean Error)
- Entropy
- Runtime

The results are shown in the table below:

| Method | AME | Entropy | Runtime (ms) |
|---|---|---|---|
| HE | 72.33 | 6.264 | 1.734 |
| AHE | 70.434 | 6.231 | 1.912 |
| WHE | 72.296 | 6.255 | 1.877 |
| EIHE | 70.409 | 2.685 | 1.791 |