# Massively Parallel Exact Histogram Equalization

Riley Kirkpatrick and Jeffrey Bush (advisor)
Computer Science
MORAVIAN COLLEGE

## Abstract

Classic method of histogram equalization (HE) is fine for human perception, but results in a loss of information for digital images. When used as a preprocessing step it reduces the quality of the results. Exact methods of histogram equalization, although much slower, do not result in a loss of information and therefore are preferable as a preprocessing step. Thus applications such as real-time medical imaging that require histogram equalization as a part of their image processing currently must use the classical method. This project adapts exact histogram equalization methods to run on graphics processing units (GPUs) to greatly increase their speed. The use of the adapted methods would make general applications of histogram equalization less time consuming and make use of exact histogram equalization methods viable for real-time imaging applications.

## Background

- Histogram specification
  - Transform image so it's histogram matches target histogram
- Histogram equalization
  - Special case where target histogram is uniformly distributed
  - Increases contrast
  - Used as early step in various image processing & computer vision algorithms
- Classical histogram specification
  - Calculate the transform from source to destination histograms using least squares regression
  - Apply the transform to the image to obtain an equalized image
  - Loses information upon transformation of digital images

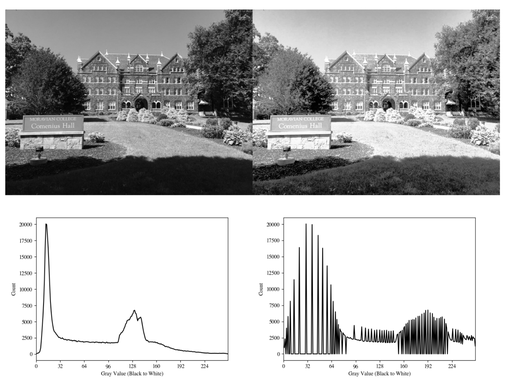


Figure 1: Left: Original, Right: Classically equalized, Below: Images' histograms

- Exact histogram specification
  - First solved for digital images in [3]
  - General approach is to establish strict ordering of pixels by computing extra information for each pixel
  - Pixels are sorting then placed into destination histogram bins in order
  - Many methods exist, mainly different in the establishment of strict ordering

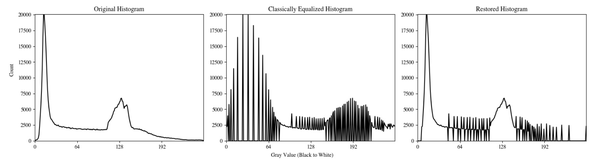


Figure 2: Example of the non-invertible nature of classical histogram specification. Left: Original histogram, Middle: Classically equalized histogram, Right: Restored histogram using classical histogram specification

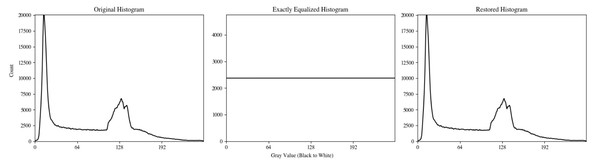


Figure 3: Example of the (mostly) invertible nature of exact histogram specification. Left: Original histogram, Middle: Exactly equalized histogram, Right: Restored histogram using exact histogram specification

## Main

Many operations of the original exact histogram equalization methods are computed using functions from the NumPy library. These can easily be efficiently transferred to the GPU by using CuPy, a numerical computational library that mimics NumPy but is accelerated by CUDA.

**Parallel Classical Histogram Specification**

- Cannot parallelize calculation of the transform well
- Parallelize applying transform by applying per pixel
- Done by calling respective functions from CuPy instead of NumPy

**Parallel Exact Histogram Specification**

- Sorting can be done on GPUs for a significant speedup
- Application of transform uses binary search to find bin for each pixel, increases computation per pixel but allows parallelization per pixel
- Each exact method's strict ordering has a custom parallelization

**LM Method of Exact Histogram Equalization** [3]

- Uses expanding mean filters
- CuPy has correlation function that are well optimized to apply the filters

**VA Method of Exact Histogram Equalization** [4]

- Uses variation method to reconstruct original real-valued version, iterative minimization method, each iteration can be done in parallel but need to sync between iterations
- We use a custom CuPy elementwise kernel that calculates the iterative VA method per pixel and syncs between iterations

## Results

| Image Size | Classic | | LM | | VA | |
|---|---|---|---|---|---|---|
| | GPU | CPU | GPU | CPU | GPU | CPU |
| 256x256 | 81.96 | 23.41 | 110.76 | 8.68 | 102.95 | 6.60 |
| 512x512 | 301.98 | 27.77 | 326.20 | 7.90 | 311.06 | 5.90 |
| 1024x1024 | 933.45 | 28.18 | 641.99 | 6.85 | 586.16 | 4.13 |
| 1920x1080 | 1336.73 | 29.68 | 816.38 | 6.46 | 757.13 | 3.70 |
| 2560x1440 | 1938.93 | 28.47 | 880.33 | 5.67 | 838.77 | 3.35 |
| 2560x1920 | 2053.35 | 26.77 | 882.64 | 5.32 | 830.80 | 3.22 |
| 256x256x128 | 2264.47 | 27.80 | 62.25 | 2.10 | 678.55 | 2.69 |

In MP/s (Megapixels per second) averaged over distinct images of same size. Higher number means faster processing. The images come from github.com/coderforlife/c4l-image-dataset. Hardware: Intel® Xeon® Gold 5122 CPU @ 3.6 GHZ, Nvidia® Titan V GPU.

## Conclusions

GPU implementations greatly increase speed and scalability. Now faster-than-real-time even on very large images. In parallelizing exact histogram equalization methods, they require more computation. In the case of VA, it requires nearly twice as much computation and the syncing of threads adds additional processing. Despite the additional computation, these methods are still much faster than their CPU-based counterparts.

## Future Work

Implementing other exact histogram equalization methods.
Current project addresses issue of speed, want to address other issues of exact histogram specification such as transferability of transforms.

## References

[1] J. A. Bush. *Spatial distribution of subcellular organelles in hippocampal dendrites from high-resolution em images.* Ph.D. dissertation, UC San Diego, 2018.

[2] D. Coltuc and P. Bolon. Strict ordering on discrete images and applications. In *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, volume 3, pages 150–153, Oct 1999.

[3] D. Coltuc, P. Bolon, and J. . Chassery. Exact histogram specification. *IEEE Transactions on Image Processing*, 15(5):1143–1152, May 2006.

[4] M. Nikolova and G. Steidl. Fast ordering algorithm for exact histogram specification. *IEEE Transactions on Image Processing*, 23(12):5274–5283, Dec 2014.