Optimizing Image Processing

J. Patel

1. ECE 4822 Temple University, Philadelphia, Pennsylvania, USA
tur49364@temple.edu

***Abstract*—** This paper present a GPU-accelerated system for computing the distribution of log-magnitude spectra over a large database of images. This task requires iterating over SVS images using 256x256 windows on a 128x128 frame grid, applying a 7x7 Gaussian smoother, computing the 2D FFT, and forming 16-bin histograms of the log-magnitude spectrum. I implemented this pipeline in Python using PyTorch with CUDA and integrate with the provided image tools for multithreading window reading. My design uses per-channel Gaussian filters using conv2d, GPU accelerated 2D FFTs, and RBG spectral histograms.

I. Introduction

Large scale medical image analysis presents a unique computational challenge due to the size, dimensionality, and complexity of imagery. The database used in this project contains SVS images, each of which may exceed tens of thousands of pixels in width and height. The goal of the final project is to design the fastest possible GPU accelerated system that computes the distribution of spectral fractures across this dataset. For each image, the task requires iterating through the slide using a 128x128 frame grid, extracting corresponding 256x256 pixel RGB windows, applying a 7x7 Gaussian smoothing filter, computing a 2D FFT, and forming a 16-bin histogram of the log-magnitude spectrum. The final output is the mean and standard deviation of these histrograms across all windows and all images.

While the operations required are standard in digital processing, their application on this scale introduction computational difficulty. A single file may contain thousands of candidate frames, and the dataset may exceed millions of windows. A normal CPU implementation becomes expensive, especially when extended to the entire data set. These constraints make GPU acceleration essential to achieving fast results.

This work presents a fully GPU accelerated pipeline in Python using PyTorch and the NEDC imaging tools provided. The system is designed to satisfy the requirements of the assignment while optimizing throughput. These include a uniform frame sampling system, high performance signal processing using GPU based convolution and FFT operations. These components allow the system to process throusand of images efficiently.

II. Method

The input data consists of breast pathology slides stored in SVS format. Each SVS file is a very large color image, often on the order of tens of thousands of pixels. The goal of this project is to compute the global distribution of spectral features across the dataset. For each image I must:

(1) Iterate over the image using a grid of 128x128 frame positions

(2) At each frame position, extracting a 256x256 RBG window

(3) Apply a 7x7 Gaussian smoother to each color channel

(4) Compute the 2D FFT of the smoother window and convert the complex spectrum to its magnitude

(5) Transform the magnitude spectrum to a log scale

(6) Compute a 16 bin histogram

(7) Aggregate the histograms across all frames and images to obtain a global mean and standard deviation

To avoid implantation of the SVS I/O system, the program uses the NEDC tools. The Python interface 'nedc_image_tools.Nil()' is used to open each SVS file and get its dimensions. Frame positions are generated on a 128x128 grid over the full image. For each subset of frame coordinates the programs calls the provided image tools 'read_data_multithread()'. The function returns a list of 256x256 RGB windows corresponding to the requested frame positions. The interface performs multithreaded I/O, which is critical given the large size of SVS files.

Processing every 128x128 frame across extremely large images can lead to a massive number of windows per image, which is impractical due to the time it would take. To make the computation possible while still sampling the image uniformly, I implement a frame sampling strategy controlled by a parameter 'max_reasonable_frames'. For am image of width 'W' and height 'H', and frame size 128x128, the total number of possible frames is:

$$N_{total} = \frac{W}{128} * \frac{H}{128}$$

If $N_{total} \leq$ max _reasonable_frames, all frames are used. Otherwise, the method down samples the grid uniformly in both horizontal and vertical directions. The ratio of target frames over the total number of frames is the sample ratio. I use a step function to choose the frames out of the grid cells to yield as close to the maximum frames possible. This allows for the spectral statistics to be uniformly represented.

All heavy numerical operations are executed on a GPU using PyTorch. The device is selected at runtime.

The required 7x7 Gaussian smoother is implemented

as a fixed 2D filter constructed from first principles. For kernel size $K = 7$, and standard deviation $\sigma = 1.0$, the code generates a grid of coordinates:

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The kernel is then normalized to sum to 1. The kernel is applied to each color channel independently using 'torch.nn.functional.conv2d'. For each smoothed channel, the system computed a 2D discrete Fourier transform using PyTorch's FFT module. It is then recentered so that the DC component is placed at the center of the frequency plane, and the magnitude is then computed.

The raw magnitude spectrum can span several orders of magnitude. To compress this, I applied a logarithmic transformation. Using the minimum and maximum log magnitude values for each window as the histogram range ensures all bins are being used.

The final goal is to compute the average and standard deviation of the histogram across all frames and all images. To accomplish this, the implementation uses a running sum, a running sum of squares, and a count of the total number of windows processed. At the end of processing all images, the mean and variance for each of the 16 bins is computed as:

$$\mu = \frac{S}{n}, \qquad \sigma^2 = \frac{Q}{n} - \mu^2$$

This approach allows the program to scale thousands of images while using constant memory with respect to the number of windows. To further manage memory on the GPU, windows are processed in batches. The batch size is dynamically chosen based on the number of frames in the image, balancing GPU utilization against memory limits. After each batch and chunk, the program releases unused GPU and CPU memory. This memory management is important for long running jobs.

III. Analysis

To evaluate the performance and behavior of the proposed GPU accelerated spectral analysis, I tested the code over a subset of 10 SVS images. Across these images the system processed 10,000 total frames, each consisting of a 256x256 RGB window extracted from the 128x128 frame grid. The results reveal several important characteristics regarding the content of the dataset and efficiency of the code.

Table 1 (below) summarizes the mean and standard deviation of the 16-bin histogram aggregated across all frames. The bins correspond to the increasing ranges of spectral energy, where the lower indices represent low frequency components and higher indices represent higher frequencies.

Bin	Mean	Std Dev
1	25.41	48.72
2	43.47	73.07
3	70.62	80.91
4	121.10	82.28
5	212.59	90.89
6	346.97	114.36
7	533.12	193.55
8	331.46	190.72
9	239.65	112.67
10	250.89	93.07
11	75.99	120.53
12	29.68	139.84
13	14.99	73.55
14	4.92	2.23
15	0.14	0.78
16	3.00	0.00

The histogram exhibits a characteristic peak in mid frequency ranges, where spectral magnitudes are highest. This aligns with the expected structure of histopathology images which contain:

(1) Low frequency background regions around fatty tissue
(2) Mid frequency around cellular structures
(3) Limited high frequency content after Gaussian smoothing

The high standard deviations in bins 6-8 indicate substantial variability in mid frequency textures. These bins capture spatial details. Conversely, bins representing high frequencies produce small means and variances. This is consistent with 7x7 Gaussian smoother which suppresses fine details before the FFT. The small variance in Bin 16 indicates that the bin is dominated by noise.

The global summary provides more insight.

$$\mu = 144.00, \sigma = 153.57$$

The large standard deviation compared to the mean indicated that the histogram is non uniform, with mid frequency bins combining the distribution.

The 10-image subset is completed in 484.393 seconds. This time shows that the GPU created significant speedup where the Gaussian convolution and FFT dominated compute time.

IV. Summary

The analysis of over 10 images demonstrates that the spectral content of the histopathology images is dominated by mid-frequency components. The 7x7 Gaussian smoother effectively suppresses high frequency noise and the GPU based system achieve solid throughput with low memory overhead, validating the design choices of chunking and batching.

V. Acknowledgments

The author thanks Dr. Jacob Scharcanski, Dr. Picone, and the Neural Engineering Data Consortium (NEDC) providing computing resources, image processing tools,

and technical guidance. Additional thanks to Dr. Picone, instructor for ECE 4822.

REFERENCES

[1] NEDC Image Tools Documentation. https://isip.piconepress.com/projects/nedc/tools/image_tools/

[2] Gaussian Blur Overview – Wikipedia. https://en.wikipedia.org/wiki/Gaussian_blur.

[3] PyTorch Documentation – torch.fft and torch.nn.functional. https://pytorch.org/docs/stable/

[4] Multidimensional FFT Background – Wikipedia.

https://en.wikipedia.org/wiki/Multidimensional_Fourier_transform

[5] TUH DPATH Breast Database Description.

https://isip.piconepress.com/projects/nedc/data/tuh_dpath_breast/