# Parallel image processing with CUDA

Justin Newman, Andrew Fleming, Thomas (TJ) Davies

[a]*James Madison University Department of Computer Science, Harrisonburg, 22801, VA, United States*

**Abstract**

Our mission is to create an application that allows users to apply a variety of image processing techniques to their pictures. The program will offer a list of actions such as desaturation, gaussian blur, rotation, background removal, pixel sorting, that can be applied to the image.

*Keywords:* Linear Algebra, Parallelism, C, OpenCV, CUDA

## 1. Introduction

Our plan is to develop a program that processes an image using a specified filter or action. The program will take the path of an input image and a filter type flag that indicates the desired action to be performed on the image. The user will select one of the actions, and after processing is complete, the newly created image will exported to the current directory. Here we provide and example of our usage functon.

```
Usage: Parallel Image Processing <option(s)> image-file :
 Options are:
    -d     desaturate <threshold>
    -g     gaussian blur <threshold>
    -r     rotate
    -b     background removal <threshold>
    -s     sorting <threshold>
```

Figure 1: Example output

To improve the processing speed, we plan to divide the image into equal sections, depending on the resources available. This block approach will allow us to treat each section as a separate project and use multiple GPU cores through CUDA to process the image faster. Our original idea involved matrix multiplication, which can be extended to image processing since an image is a 2D array of pixels that undergo a series of operations to produce the final result. The result will be stored in a separate image buffer, which can be used to create a new output image at the end of the process.

We plan to use vector addition with CUDA, as discussed in section 6.8 of the textbook, to carry out the image processing. Each row of pixels will be assigned to a worker thread, which will perform the arithmetic operations. If necessary, we may use bitonic sorting to handle the pixel sorting effect.

## 2. Relevance

This project takes advantage of CUDA's speed for performing matrix operations. Images can be represented as matrices, where each element of the matrix represents a pixel in the image. Matrix operations can then be used to perform various image processing tasks, such as filtering, convolution, and transformation.

By using these topics covered in CS470 to perform matrix operations for image processing, we can take advantage of the parallel processing capabilities of the GPU. This will allow us to perform matrix operations on large matrices quickly and efficiently, which can significantly improve the processing speed for image processing tasks.

## 3. General image processing techniques

Implementing parallelized desaturation and blur using CUDA C should be straight forward since most of the code we have from the lab. Implementing a rotate feature in CUDA C would involve modifying the rotation matrix calculation to take advantage of the parallel processing capabilities of the GPU.

CUDA's managed memory will be used for the storage of the user's image. We will write a kernel for each process, and the image will be divided into small blocks that can be processed in parallel by different threads on the GPU. Each thread applies the filter kernel to its specific portion of the image block, which will produce a processed block. The processed blocks will be combined into the final image using a merging algorithm. By utilizing the managed memory and working with the images in blocks we effective parallelization.

## 4. Advanced image processing techniques

We plan to use OpenCV to accelerate the use of machine perception more effectively on the user provided images. We will use the grab cut iterative algorithm in order to identify the objects of the image in the foreground and background. Based on color distributions of both the foreground and the background,

the algorithm assigns each pixel in the image to either the foreground of the background. Using the new mask that represents the foreground of the image, we can continue the standard image filtration process that the other filters use.

### 5. Possible Roadblocks

We anticipate some trouble getting OpenCV working. None of us have used it before and we are not sure what kinds of problems might arise using it. Additionally, background removal is a complicated task. Our final implementation of this may not be the best that it can.

### 6. Draft Deliverable

By the mid-project draft we will have the serial code written for every function except the background removal. We also will have the desaturation, Gaussian blur, and rotate kernels implemented. Significant progress includes a testing suite for desaturation, Gaussian blur, and rotation which will validate both our serial and parallelized implementations. Excellent progress could include a serial version of background removal and kernels written for either background removal or pixel sorting. It also might include a fully automated ”one-button” version of our test suite for all functions.

### 7. Final Project Deliverable

We plan to submit our code as well as some example images to use. This code will use at least 5 kernels, one for each image processing technique. As shown in class, image processing techniques benefit greatly from parallelization because of the amount of data being performed on the same task. We expect all of our kernels to demonstrate an over 10x speedup, and we should have no trouble getting over 20x speed up. The pixel sorting algorithms will require more thought when parallelizing because each thread needs to know where the other threads are placing their sorted pixels to avoid a data race. This code will also demonstrate a “one-touch” testing framework and a fully-correct write-up detailing the analysis of every kernel. Lastly we plan on including a section detailing future work. Because we have included everything mentioned above this project demonstrates an “A” level of work

### Acknowledgements

Special thanks to Dr. Lam for introducing us to CUDA and its benefits. Learning about this powerful tool has been an eye-opening experience that has greatly enhanced our understanding of parallel programming. His guidance and expertise has been instrumental in helping us understand the intricacies of this technology. Through his clear and concise explanations, we have gained a deep appreciation for the efficiency and scalability that OpenMP can provide. We are thrilled to continue exploring the possibilities of OpenMP and CUDA then applying this knowledge to our future projects.

### References

https://w3.cs.jmu.edu/lam2mo/cs470/cluster.html

https://www.sciencedirect.com/book/9780128046050/an-introduction-to-parallel-programming

https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

https://github.com/opencv/opencv
https://github.com/DavidMcLaughlin208/PixelSorting