Parallelized Chromaticity-Based Shadow Detection and Removal

Madeleine Armstrong, Coale Cooper, Rituparna Dey

mbarmstrong@email.arizona.edu, cjcoopr@email.arizona.edu, rdey1@email.arizona.edu

Group 2

Brief (less than 200 words) technical abstract of the proposed project	The presence of shadows in an image can negatively impact computer vision applications that depend on accuracy of object recognition. Identification and removal of shadows is thus critical in surveillance applications involving target detection and tracking. As the amount of data involved in image and video manipulation rapidly increases, new levels of computational power and algorithmic efficiency become necessary. Several known methods exist for shadow detection removal based on different features of color and image processing. In this study we propose a chromaticity-based shadow removal algorithm based on an existing serialized Matlab code. By using the CUDA computing platform and API, we aim to achieve speedup by parallelizing the algorithm on a NVIDIA Tesla P100 GPU.
Brief (less than 200 words) discussion of anticipated benefits	Compared to the serialized version of the code, we anticipate a very significant speedup with our parallelized approach. Using the high-performance capability of the Tesla P100 GPU allows the innumerous pixel-wise calculations to be performed simultaneously resulting in faster execution without compromising output quality. Our algorithm will result in high-resolution images without the detrimental effects of illumination and shadows. Applications like the Controlled Environment Plant Production (CEPP) that rely on computer vision sensing are made simpler and more efficient using sophisticated shadow removal algorithms.
A maximum of eight key words that describe the project	shadow, parallelism, GPU, CUDA, image processing, chromaticity, Otsu’s method

1 Identification and significance of the problem or opportunity

1.1 Statement of Problem

Image processing utilizes computer algorithms to efficiently process the large amounts of raw pixel data needed for image/video identification and manipulation. The accuracy of these systems may be impacted by unwanted conditions such as noise or light, for which human vision automatically adjusts or corrects. Shadows are one such condition that may alter or conceal necessary parts of an image. In video surveillance applications where target detection and object tracking are crucial, it is beneficial to digitally remove shadows for better image recognition.

The application of our report will be for plant surveillance and contact sensing using computer vision. This is for use in Controlled Environment Plant Production (CEPP) systems within greenhouse settings. The images that we use for this study come from this application.

Problem Background

The shadow removal algorithm selected for this study is based on "Accelerated Shadow Detection and Removal Method," an IEEE conference paper completed at the University of Arizona [1]. This paper describes a chromaticity-based shadow detection and removal algorithm which consists of five processes: (1) colorspace transformation, (2) thresholding and Otsu's method, (3) convolution, (4) erosion, and (5) result integration.

These five processes will perform multiple sets of calculations using the pixel data of the entire image, with many of these calculations being the same across every pixel. This motivates using the parallel power of the GPU to reduce the computation time. For each of the five processes, there are multiple optimization areas where implementing GPU kernels will take advantage of the repeated calculations. Then combining the accelerated kernels, an end-to-end shadow removal solution will be captured resulting in a significant reduction of processing time.

1.2 Current Level of Technology

As stated above, the approach for this shadow removal algorithm is well studied in [1]. The algorithm has not only been shown to effectively remove shadows, it has also been shown to benefit from GPU acceleration. Secondly, there exists a sequential implementation in Matlab for this project to reference as a starting point and proof of concept.

Current methods in shadow detection can be classified according to feature into four categories: chromaticity, physical, geometry, and texture [3]. This paper will use a chromaticity-based approach, meaning utilizing a measure of color that is independent of intensity. Many chromaticity methods are relatively simple and computationally inexpensive but may produce more noisy results due to the calculations being performed at a pixel level [3].

Chromaticity-based shadow detection and removal algorithms involve color space transformations and pixel-wise computations.

1.3 Limitations of Current Technology

Identifying and adjusting for shadows in an image is computationally expensive, requiring techniques such as pixel-wise transformations and analysis of small "neighborhoods" of pixels. As the technology for image capturing and resolution evolves, the amount of pixel data and computations demand faster and more sophisticated methods of image processing. The parallel structure of GPUs allows blocks of data to be processed simultaneously, greatly increasing the speed of algorithms for high-resolution image and video processing.

1.4 Proposed Solution

In this study we are proposing a shadow removal method, parallelized using a Tesla P100 GPU, and achieve a speedup on an 18 megapixel (MP) resolution image compared to the same method implemented in Matlab.

As with most image processing applications, Graphic Processing Units (GPUs) are an attractive platform for accelerating the capabilities of this application thanks to its high levels of data parallelism. In this work we propose a chromaticity-based shadow detection and removal method and implement on a NVIDIA Tesla P100 GPU.

1.4.1 Background

As our main focus is to realize a low-latency and high-throughput shadow removal, we propose a chromaticity-based algorithm. This choice compliments the utilization of GPUs, as a chromaticity-based algorithm primarily consists of pixel-wise transformations that are suitable for parallelization on GPUs.

1.4.2 Statement of Solution

In this project, we propose a chromaticity-based shadow detection and removal algorithm that involves five intermediate steps (Figure 1) before generating the final RGB shadowless image.

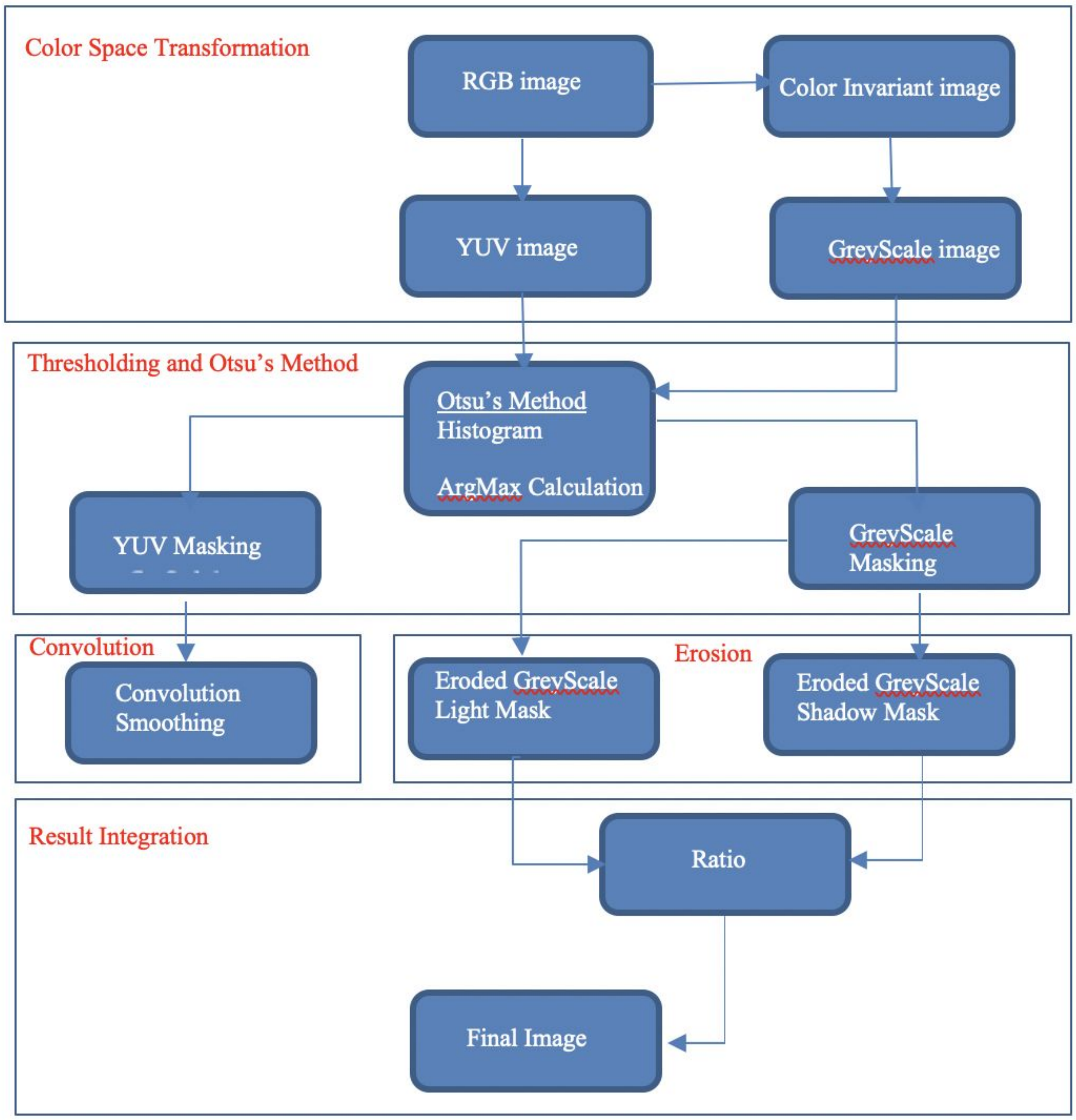

Figure 1: Flowchart of the algorithm steps

1) **Colorspace Transformation**: Standard color images use the RGB color space, which classifies a pixel by its red, green, and blue components. However, other color spaces exist which represent color differently. Two color spaces that are utilized in image processing are HSV (hue, saturation, value) and YUV (luminance and chrominance) [2].

This process takes in the original RGB image and generates YUV and grayscale versions. To generate the grayscale image we first generate a color invariant image from the original RGB input image.

In order to convert the image every pixel must be processed with some conversion operation. The global memory will play an important role in the speed of this transformation because of the amount of memory accesses. For this case coalesced memory reads and writes should be considered to improve the efficiency.

2) **Thresholding and Otsu's Method**: After performing the colorspace transformation, two masks are generated. One mask is generated from the grayscale image and the other mask is generated using the U component of the YUV image.

The masks are generated using Otsu's method , which is an unsupervised method of finding a threshold value to differentiate between the foreground and background of a grayscale image. The input to Otsu's method is a histogram of the pixels in the image, which is to approximate the probability density function of pixel intensities.

The mask generation should show significant improvement when mapped to GPU processing considering the large number of parallel threads involved

3) **Convolution**: In this step we generate the convolution-pixel, which is the summation of the selected image pixel, and its neighboring pixels, multiplied and accumulated with the kernel matrix.

For the convolution operation a tiling approach will be beneficial due to the amount of predictable data accesses used by each computation. Each computation will be mapped to an individual thread where it will perform a single step of the convolution. Each thread will need access to a set of neighboring pixels in order to do its respective calculation. The key here is neighboring pixels. This will allow the implementation to make use of the shared memory space so all threads in a block can share a set of pixel data reducing overall memory latency.

4) **Erosion**: In this step we take the output masks from Otsu's method and pass them through an erosion operation. The goal of erosion is similar to smoothing: produce two closely related masks, a light mask and a dark mask, that have slight overlap in the transition region between light and shadow and can be used to selectively perform operations on the light or dark regions of the image, respectively.

5) **Result Integration**: In this final step, we first create maps of the eroded shadow and multiply light masks by the RGB of the input image. Next we accumulate these maps along with the eroded shadow and lights masks to get the average value of each array. We then use these average values to create our RGB ratio values. Finally, we use the RGB ratio values to map with the input image one last time to remove the shadow and create our output shadow-less image.

1.4.3 How Solution Overcomes Current Technology

As our main focus is to realize a low-latency and high-throughput shadow removal, we propose a chromaticity- based algorithm. This choice compliments the utilization of GPUs, as a chromaticity-based algorithm primarily consists of pixel-wise transformations that are suitable for parallelization on GPUs. The goal here is not focused on creating the 'best' algorithm at removing shadows but to use an existing well understood algorithm to optimize it for speed using sophisticated GPU techniques. This will help meet the ever increasing demand for faster and more sophisticated image processing methods.

1.4.4 Validation and Evaluation Strategy

There will need to be multiple considerations when evaluating the performance of the newly implemented GPU accelerated shadow removal algorithm. First the algorithm needs to maintain the same effectiveness in removing shadows as the baseline Matlab implementation. The team will run the same image through both the baseline and the new GPU accelerated implementations and verify by eye the output is the same. Furthermore, a pixel-by-pixel comparison will be conducted between the two output images to quantify any difference in shadow removal effectiveness.

Second, and the most important evaluation, will be the performance of the GPU implemented shadow removal algorithm. The team will gather a timing analysis of the Matlab serial implementation and each of the respective sub processes of the algorithm. A respective timing analysis of the GPU implementation and subprocess will then be collected for comparison directly to the serial implementation. Results here will be the standard to how effective the GPU implementation is.

In addition, the NVIDIA profiler will be used throughout each kernel's development and integration in order to capture the efficiency of the GPU code. This evaluation could consist of but is not limited to, global and shared memory utilization, compute vs memory utilization and thread divergence metrics.

2 Technical objectives

This solution shall provide an accelerated shadow removal algorithm written in C, using the CUDA API. The new implementation will be derived off a current approach already constructed in Matlab. The Matlab code will act as a baseline reference to produce both image quality and performance improvement metrics obtained by the new GPU implementation The team will end with a report and presentation summarizing the approach and findings of the project.

2.1 Project Milestones and Tasks

2.1.1 Understanding the shadow removal algorithm

The team's first step is to understand all computations that make up the complete shadow removal algorithm. Here the goal is to develop a detailed breakdown of the techniques and characteristics used to implement this algorithm. A current implementation has been provided in Matlab which will be used as a reference point for the team. In addition, using the Matlab code, a timing analysis will be run to establish baseline metrics to be compared against the future GPU implementation.

Tasks:

- Understand color space transformation - **Madeleine**
- Understand erosion - **Madeleine**
- Understand convolution - **Coale**
- Understand result integration - **Ritu**
- Understand Otsu method - **Team**
- Timing analysis - **Coale**

2.1.2 Host-only implementation

Here, the team will implement a host-only version of the shadow removal algorithm. This host only solution will be written in c which will provide the structure of the teams implementation in which GPU kernels will later replace the serial implementations.

Tasks:

- Implement input image reader - **Coale**
- Implement serial color space transformation - **Madeleine**
- Implement serial erosion - **Madeleine**
- Implement serial convolution - **Coale**
- Implement serial result integration - **Ritu**
- Implement serial Otsu method - Team

2.1.3 Implement kernels in CUDA

Each team member will individually be responsible for designing and implementing a kernel or set of kernels to optimize a piece of the shadow removal computation. The member will explore multiple different optimization methods discussed in this class and provide data driven analysis on the results of their implementation(s).

Tasks:

- Implement kernel for color space transformation - **Madeleine**
- Implement kernel for erosion - **Madeleine**
- Implement kernel convolution - **Coale**

- Implement kernel result integration - **Ritu**
- Implement kernel Otsu method - **Team**

2.1.4 Integrate kernels together

Once the individual kernels are completed they will be integrated into the host only implementation to replace the respective serial sections of the code.

Tasks:

- Team effort to coordinate integration

2.1.5 Profile complete GPU algorithm implementation

Following completion of the integration a detailed analysis of the entire shadow removal algorithm will be conducted using the NVIDIA profiling tools. Metrics will be gathered and analyzed for comparison against the baseline. At this point it may be necessary to revisit some of the individual kernels to make some small optimization improvements.

Tasks:

- Team effort to profile integration

2.1.6 Compile report

Finally, a report and presentation will be created summarizing the team's findings.

Tasks:

- Each team member write section on implementation of respective kernel
- Schedule class presentation

2.2 Work Plan Schedule

Table : Task Schedule

Task	Description	Weeks					
		1	2	3	4	5	6
	Understanding the alg.	x					
	Host only implementation		x	x			
	Implement CUDA kernels			x	x	x	
	Integrate kernels					x	
	Profile completed GPU alg.					x	x
	Write final report					x	x

References

[1] A. Akoglu, J. Mack, R. Raettig, E. Richter, B. Unal and S. Valancius, "Accelerated Shadow Detection and Removal Method," *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, Abu Dhabi, United Arab Emirates, November 3-7 2019, pp. 1-8, doi: 10.1109/AICCSA47632.2019.9035242.

[2] P. Kumar, A. Lee and K. Sengupta, "A comparative study of different color spaces for foreground and shadow detection for traffic monitoring system," *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, Singapore, 2002, pp. 100-105, doi: 10.1109/ITSC.2002.1041196.

[3] A. Sanin, C. Sanderson and B. C. Lovell, "Shadow detection: A survey and comparative evaluation of recent methods," *Pattern Recognition*, vol. 45, no. 4, pp. 1684-1695, April 2012.