Posterization with CUDA
Ben Kahle, Evan Dorsky, and Madison May

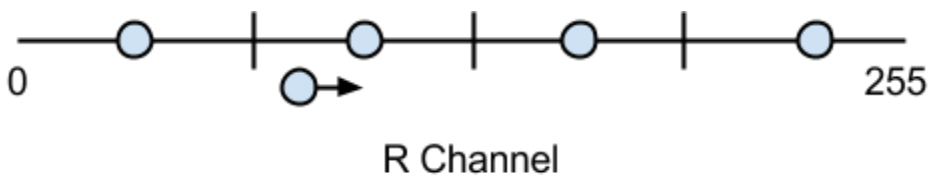
For the final project, our project team set out to learn a bit more about the fundamentals of GPU algorithms, the technical challenges of writing parallel code, and the scenarios that lend themselves well to parallel processing. We each completed the first three weeks of Udacity's Introduction to Parallel Processing course and applied the skills we learned to write a CUDA program for producing "posterized" images. A sample of the results are included at the end of this report.

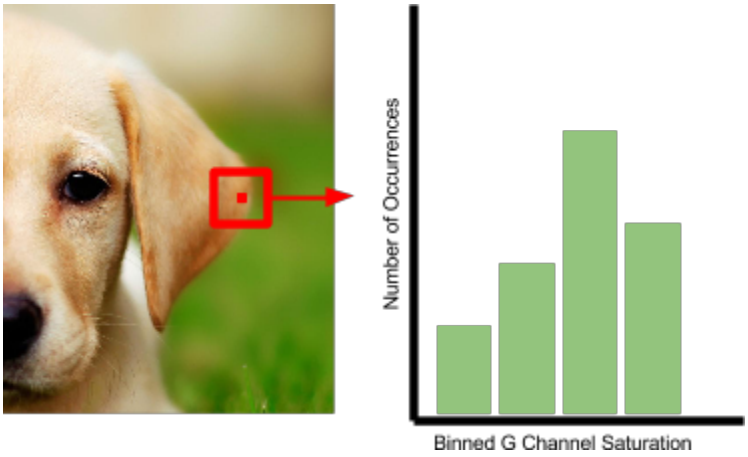
Through the Udacity course and our own project, we gained a basic understanding of GPU programming and the general principles of parallel programming. At the basic level, we learned how GPU programs are structured: a main function that runs on the CPU which sets up the GPU memory environment and initializes the kernel to run on a desired number of blocks of threads on the GPU. Each desired thread runs the kernel once to completion. To manage memory, the CPU allocates memory on the GPU and copies memory to and from the GPU before and after the kernel is run. This management is very similar to heap memory management on the CPU and uses functions including cudaMalloc() and cudaMemcpy().

The algorithmic patterns we learned include "map" (one-to-one), "gather" (many-to-one), "scatter" (one-to-many), and "stencil" (a special case of gather) which apply not just to graphical processing problems, but to any computational task that could be parallelized. We will find our experience with these patterns useful in our future programming careers, whether we work with graphics or not.

As a result of focusing on image manipulation algorithms, this project also required us to learn to access and edit image data in C using the OpenCV library. We learned enough of the OpenCV library to open images, copy the image data, and save a new image with the edited color space.

In order to produce the posterization effect, we broke the CUDA kernel up into two parts. The first of the pair reduces the color space used by the image, dividing RGB colorspace up into sub-cubes and mapping each color to the value at the center of the sub-cube. In other words, we divided the R, G, and B channels of the image into n sections, then mapped each pixel's R, G, and B value to the center of the nearest segment.



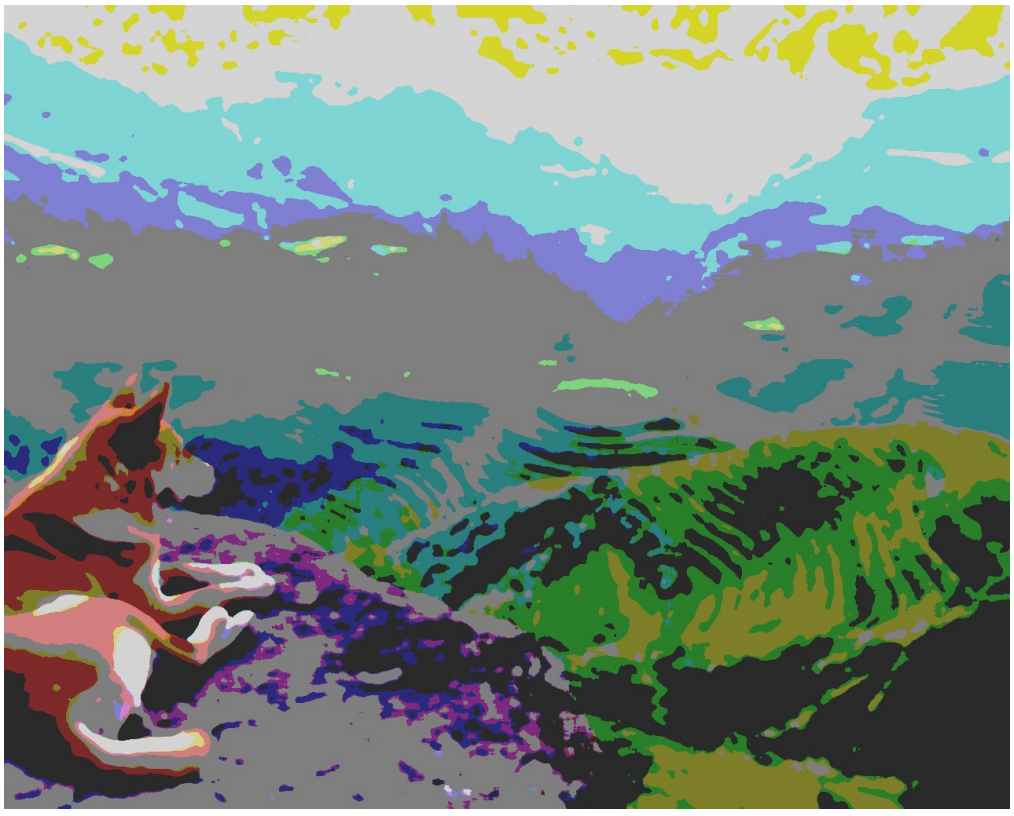


The second kernel maps a given pixel to its “local mode.” It counts the occurrences of the neighboring pixels in an 11*11 square, and sets the output pixel to the most common of its neighbors. The histogram kernel returns an image with lots of “speckles”, as colors in a smooth gradient may get mapped to two very different colors depending on which side of the threshold they fall on. The mode kernel has the effect of smoothing edges, as it modifies these “outlier” pixels to better fit the colors around them. The effect of the mode kernel on the reduced color space image is clear between the above picture (after histogram, before mode) and the below picture (after histogram and mode).

To demonstrate the success of our learning we have produced a number of images showing the effects of reducing the color space, and processing blocks with mode to reduce the edge complexity. These effects together produce a posterization effect as can be seen in the examples below.

For future work which we plan to integrate before our expo presentation, we would like to focus on three main tasks: improve the robustness of our code to allow more real time flexibility through command line arguments, add an additional step in our color space reduction that ensures all final colors are taken directly from the most common colors present in the image, and improve our use of OpenCV in an attempt to create a real time posterization effect using our computer's webcam. Additionally, as the posterization process reduces the color space of an image it also tends to reduce to contrast of the image. If we have time, we would like to explore methods to maintain the original contrast of the image during posterization.

All our code is available on GitHub: https://github.com/madisonmay/posterize




NickchouBG.deviantart.com