A Parallel Image and Video Processing Application

H. Parker Shelton, Adam Feinstein

Abstract— **We built a go-fast image and video editor.**

I. Introduction

MULTIMEDIA

II. Image Editing

Several common image editing operations were implemented in the program and are discussed below.

A. Grayscale

Because the image was stored internally in RGBA format, the conversion to grayscale requires application of the luminance formula to each pixel:

$$luminance = 0.3 * red + 0.59 * green + 0.11 * blue$$

This is in contrast to an image in YCrCb format, where the color space would simply be thrown away.

B. Brightness

Increasing the brightness of an image can be achieved by scaling each of the RGB values by a factor. This adjusts the luminance of the entire image.

C. Contrast

Contrast is a measure of the distance of each pixel from the average luminosity. In order to increase contrast, a linear combination of the average luminosity and the RGB values are taken:

$$red = (1 - factor) * average\_lum + (factor) * red$$

D. Saturation

In contrast to contrast, increasing saturation requires scaling the RGB values of a pixel away from its luminosity, rather than the global average luminosity. The formula appears the same as for contrast, with $average\_lum$ replaced with lum.

E. Crop

Cropping was implemented to remove unwanted portions or highlight desired portions of an image. The function takes four parameters defining the top-left and bottom-right corners of the desired sub-image.

The authors are with the Departments of Computer Science and Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218. Email: {parker.shelton, afeinst1}@jhu.edu.

F. Rotation

Rotation was implemented using Gaussian sampling of the original image in order to determine pixel color values in the rotated image. The Gaussian used was defined with variance 0.6 and radius 4 pixels.

G. Scale

Scaling was also implemented using Gaussian sampling, also with variance 0.6, but with a radius that linearly depends on the scaling factor to compensate for the enlarged (or shrunken) size of the image.

H. Blur

Blurring was performed using a 3x3 convolution filter that weights the original pixel heavily, but allows for contributions from neighboring pixels:

$$mask[3][3] = \left\{ \left\{ \frac{1}{16}, \frac{2}{16}, \frac{1}{16} \right\}, \left\{ \frac{2}{16}, \frac{4}{16}, \frac{2}{16} \right\}, \left\{ \frac{1}{16}, \frac{2}{16}, \frac{1}{16} \right\} \right\}$$

I. Edge Detection

Edge detection was similarly implemented as a 3x3 matrix convolution, but with a mask that highlights pixels with colors sufficiently different from its neighbors:

$$mask[3][3] = \{\{-1, -1, -1\}, \{-1, 8, -1\}, \{-1, -1, -1\}\}$$

III. Theorem

IV. Performance Metrics

A. Coding Gain

V. Conclusion