CS350 Assignment 3

Michael Austin

# Problem Definition or Specification

### Image Subsampling

Given an m× m image A, the required task is to reduce the image to a smaller image B of size s× s. Your program should produce a reduced version of an original image. For instance, the main function could partition the input image into a total of ***s2 n***× ***n*** smaller image blocks, and then call a function to compute some image statistics. Distribute the jobs (the image blocks) amongst the *p* child-processes. Wait for the children to return their results. Arrange the returned results to form the reduced image. Write the reduced image to a file.

# Problem Analysis

Define n=m/s. For each n× n block of the image, we compute the average pixel value, and the standard deviation of the values in the block. (Overlapping blocks are not allowed). The average value then becomes the value for the corresponding single pixel position in the smaller s× s image. For a given block size, the standard deviation of the pixel values in each block gives us a rough indication of how well the reduced image will approximate the original image. Write a program that reads in a .ppm or .pgm image, subsamples the pixel values based on a reduction factor that evenly divides the dimensions of the original picture, then takes the mean of each of these s^2 blocks and stores the value in the pixel array for the smaller output image of the same format.

# Algorithm Design

1. The image\_ptr structure is imported from the given ip2Libc.h file and to read the .pgm files.
2. The row and column dimensions are stored in global integer variables during the read process.
3. The mean and standard deviation methods are defined within the perform\_analysis method and are called in the thread function.
4. Initially, a separate partition method was developed to chunk the input image into separate blocks; however, returning 2D arrays became aggravating and it became more expedient to copy and paste the loops from assignment 2 into a separate function and call the analysis portion during thread creation.
5. The pixel array was subsampled based on reduction factor s^2 times and each block was averaged, with this value cast to an unsigned char to become the new image pixel value.
6. Finally, the new pixel array of averaged values was cast to an image\_ptr and output to a writepnm() imported from the given image library file.

Refinement: The reduction factor needed to be converted to an int after read from the command line, so the atoi() was used. I tried to use pipes and have everything occur under one method, but too many parameters needed to be passed in to the thread function and casting to void \* became frustrating. The notes from class about pipes also helped with the counting/ iteration.

Although the image is subsampled, it is somewhat darker.

# Test Plan (Desk Checking or Algorithm Verification)

### Problem 2

I used the CLion Debug Configurations to pass in command line arguments, somewhat automating the process.

# Implementation or Coding

Each code file is contained in PDF form in the code pdf folder within the zipped directory submitted.

# Testing

Command Line Entries:

**Format:** **path/to/input/file path/to/output/file reduction\_factor threads**

### Image Subsample

Original:



After Subsampling



Before:



After:

