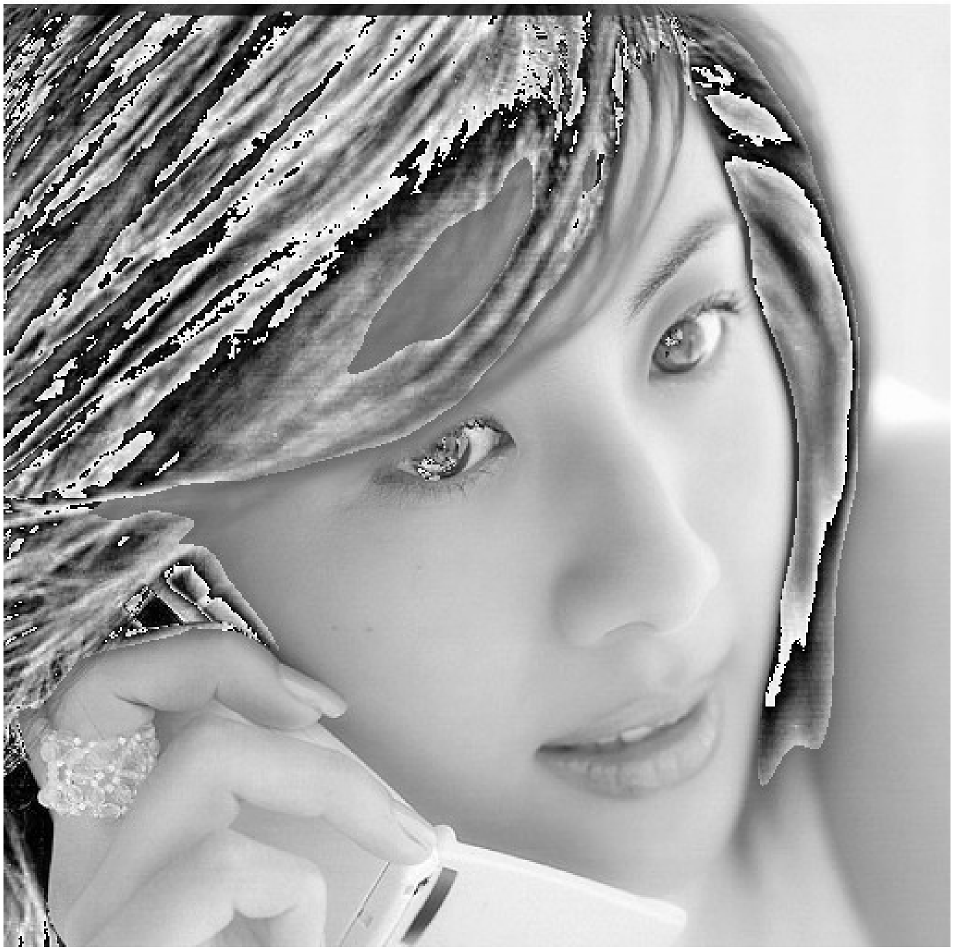
Assignment 3:
Image Enhancement via Client-Server Communication

Paul Prince

Submitted Tues. 2010-04-20

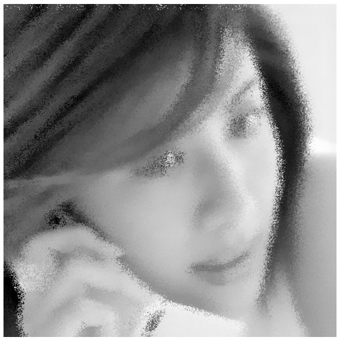
CS350 - Dr. Don Adjeroh, Spring 2010



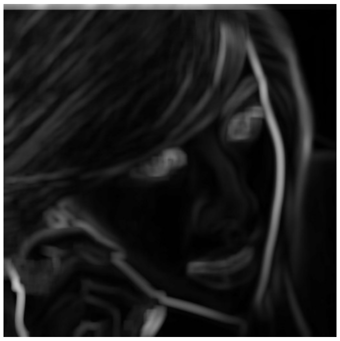
Enhanced Image

Original

Mean



Median



Variance

Introduction

Target Platform

I try to write portable code, however there may be dependencies specific to a Linux operating environment. I also test all submissions on the WVU CSEE shell server. Consult the included files “Makefile” and “run_tests.sh” for assistance building and running the programs.

Image Formats

Currently, the code still supports only PGM greyscale images, i.e. PBM Type 5 images.

If time permits, I may extend it to additional formats before final submission.

Image Processing Algorithms

Window Selection

Same as in Assignment 2:

We attempt to select a square window centered around a given center pixel. If any of the edges of this window fall outside the boundaries of the input image, we crop off the overhanging portions of the window.

Standard Deviation Calculations

Again, same as in Assignment 2, Knuth & Welford's “online” algorithm for calculating population variance in a single pass over the items.

Median Calculations

As yet, unchanged from Assignment 2: use the standard library's *qsort()* routine to find medians naïvely.

This is an aspect of the program I hope to improve before final submission. However, the current method should be numerically accurate, even if inefficient.

Enhanced Value Calculation

Counter to the apparent “conventional wisdom” regarding this assignment, I am currently calculating the enhanced pixel value in the child threads, and not in the parent/main thread.

The constant values have been updated according to Assignment 3.

I will modify this behavior if required.

Global Statistics

I notice on my grade sheet from Assignment 2 that I lost a few of points in the global statistics portion. I am reviewing my code to ensure that any errors are corrected before final submission of Assignment 3.

Changes from Assignment 2

im_shared.c

I intend to produce a multi-process version of the assignment as well, for the bonus credit. As a result, I have moved everything that I think can be shared between the multi-process and multi-threaded versions in *im_shared.c*

Communication Between Threads

Due to the nature of the assignment (and that awesome hint from Dr. Adjeroh after class on 4/15), I have avoided many things that might have been difficult:

- Most thread-shared variables are read-only during the execution of multiple threads.
- Each thread writes only to distinct elements in the output arrays.

As a result, I have not had to implement any locking, etc.. And I don't think I have any deadlocks or race conditions, but I could always be wrong!

Child Thread Entry Point / Main Loop

Execution for the child threads begins in *iterate_input_image()*:

```
/* Iterate over the input image, populating the others along the way. */
void *iterate_input_image(void *arg){
        int i, j, c; /* loop counters */
        double window_mean, window_variance, window_median, window_stddev;

        int child_id = get_child_id();

        c = 0;
        for (i=0; i<rows; i++){
                for (j=0; j<cols; j++) {
                        if (c % num_threads == child_id) {
                                /* Choose the subregion of interest (window) around this pixel. */
                                window_t window = select_window(window_size, i, j, rows, cols);

                                /* Calculate the median of just the window. */
                                windowcalc_median(Image, window, &window_median);

                                /* Calculate the mean and variance of just the window. */
                                windowcalc_mean_and_variance(Image, window, &window_mean, &window_variance);

                                /* Calculate standard deviation of just the window. */
                                window_stddev = sqrt(window_variance);

                                /* Calculate the enhanced value of this pixel. */
                                int window_enhanced = calc_enhanced( Image[i*cols+j], mean, stddev,
                                    window_mean, window_stddev );

                                /* Assignments to the output images. */
                                Mean_Image[i*cols+j]     = window_mean;
                                Median_Image[i*cols+j]   = window_median;
                                Enhanced_Image[i*cols+j] = window_enhanced;
                                Variance_Image[i*cols+j] = window_stddev;
                        }
                }
        }

        return NULL;
}
```