**CS 350:COMPUTER SYSTEM CONCEPTS Kharon Harrell**

**PROGRAMMING ASSIGNMENT 2**

**Q1. IMAGE SUBSAMPLING**

**Problem Statement**

For this program, we will reduce an image *A* with a size to a new image *B* with a size of . In simple terms, we will be reducing the size of image to the size of the user’s choosing, producing a new, smaller image. The program is simple in theory, but rigorous in practice.

**Problem Analysis**

This program will reduce the size of an image to a smaller size, while preserving the display of the image.

Input(s): initial image

Output(s): reduced image file

Constraint(s): max file size for an image

Formulas: &

**Algorithm Design**

Initial Algorithm

1. Read the input image
2. Partition the original image into smaller subimages
3. Push each block into a function to analyze the image
4. Arrange results into the corresponding image
5. Write reduced image to an output file
6. Calculate the mean, standard deviation, and the average standard deviation for the for the original image

**Test Plan**

Testing this program should only require two test cases; a simple image containing very little complexity, perhaps an image with basic geometry and a regular image.

**Q2. THE CLIENT SIDE (PARENT PROCESS)**

**Problem Statement**

For this program, we will reduce the size of image using a process. The process will partition an image in smaller blocks. In addition, the child processes will imitate an image analysis server.

**Problem Analysis**

This program will reduce the size of an image to a smaller size, while preserving the display of the image.

Input(s): initial image

Output(s): reduced image file, overall mean & standard deviation, individual block averages, and average standard deviation

Constraint(s): max file size for an image

Formulas: &

**Algorithm Design**

Initial Algorithm

1. Read the input image
2. Partition the original image into smaller subimages
3. Distribute the image blocks to *p* child processes
4. Arrange results into the corresponding image
5. Write reduced image to an output file
6. Calculate and print out overall mean & standard deviation, individual block averages, and average standard deviation.

**Test Plan**

Testing this program should only require two test cases; an expected value for the process and an unexpected value.

**Q3. THE Server SIDE (CHILD PROCESSES)**

**Problem Statement**

For this portion of the program, we will utilize the parent process in Q2 to assess an individual block of the image to perform some analysis and return the results of the individual block.

**Problem Analysis**

This program will calculate two image statistics: mean and standard deviation.

Input(s): input image block

Output(s): mean and standard deviation

Constraint(s): max size for a block

Formulas: &

**Algorithm Design**

Initial Algorithm

1. Read the input image block from the client
2. Perform image analysis (mean & standard deviation)
3. Return the results of the image block.

**Test Plan**

Testing this program should only require two test cases; an expected value for the process and an unexpected value.

**Implementation**

//-------------------------------------------------------------------------------------------------------------

//Main program that reduces the size of image using threads.

//Author: Kharon Harrell

//CS350 Section 001

//Last Modified: Dec. 13, 2019

//--------------------------------------------------------------------------------------------------------------

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include "tpool.h"

#include "iplib2New.h"

#include "pthread.h"

//defining stucture for thread job

typedef struct {

image\_ptr imagePtr;

int row;

int col;

int width;

int height;

int rows;

int cols;

float \*mean;

float \*sd;

image\_ptr imagePtr2;

} image\_analysis\_t;

void image\_analysis(image\_analysis\_t \* args);

int main(int argc, char \*\*argv)

{

image\_ptr imagePtr, imagePtr2;

unsigned char \*\* output\_image = NULL; /\* space for output image \*/

char \* output\_filename = NULL;

int factor = 0;

float mean = 0;

float sd = 0;

int rows, cols, type;

int i=0, j=0, value=0, count=0;

int tp;

int n;

/\* check inputs \*/

if (argc != 5)

{

printf("wrong inputs: use %s infile out1 out2 thread\_count \n", argv[0]);

return 0;

}

int nr\_threads = atoi(argv[4]);

//create thread pool

tpool\_t \* pool = tpool\_create(nr\_threads);

/\* first read-in the image \*/

printf("reading input image ... \n");

imagePtr = read\_pnm(argv[1], &rows, &cols, &type);

printf("image read successfully \n");

printf("rows=%d, cols=%d, type=%d \n", rows, cols, type);

/\* printf("rows=%d, cols=%d, type=%d \n", ROWS, COLS, TYPE); \*/

output\_filename = argv[2]; //output file = third arg

factor = atoi(argv[3]); //factor = string to integer of arguemnt 4

n = rows/factor; //n = dimension of reduced image

//imageptr2 = allocates memory for reduced image

imagePtr2 = malloc(n\*n\*sizeof(unsigned char));

//iterates over the entire image and inputs into image analysis

for(int i=0; i<rows; i = i+factor){

for(int j=0; j<cols; j = j+factor){

//allocate memore for structure and assign values

image\_analysis\_t \* args = (image\_analysis\_t \*) malloc(sizeof(image\_analysis\_t));

args->imagePtr = imagePtr;

args->row = i;

args->col = j;

args->width = factor;

args->height = factor;

args->rows = rows;

args->cols = cols;

args->mean = &mean;

args->sd = &sd;

tpool\_add\_work(pool, (void \*)image\_analysis, args);

}

}

tpool\_wait(pool); //wait for all threads to finish

write\_pnm(imagePtr2, output\_filename, rows/factor, cols/factor, PGM); //writes to a file

//image\_analysis(imagePtr, 0, 0, rows, cols, rows, cols, &mean, &sd);

printf("overall mean = %6.2f\n", mean);

printf("overall sd = %6.2f\n", sd);

return 0;

}

//image analysis = takes image, row colum of sub block, height and width of the subblock, rows and cols are dimensions of the original image

void image\_analysis(image\_analysis\_t \* args) {

image\_ptr imagePtr = args->imagePtr;

int row = args->row;

int col = args->col;

int width = args->width;

int height = args->height;

int rows = args->rows;

int cols = args->cols;

float \*mean = args->mean;

float \*sd = args->sd;

image\_ptr imagePtr2 = args->imagePtr2;

int sum =0;

//calculate sum

for(int i = 0; i < height; i++){

for(int j = 0; j < width; j++){

sum += imagePtr[(row + i)\*rows + col + j];

}

}

\*mean = sum/(width\*height);

sum = 0;

//calculate sd

for(int i= 0; i< height; i++){

for(int j=0; j<width;j++){

sum += pow(imagePtr[(row + i)\*rows + col + j]-\*mean, 2);

}

}

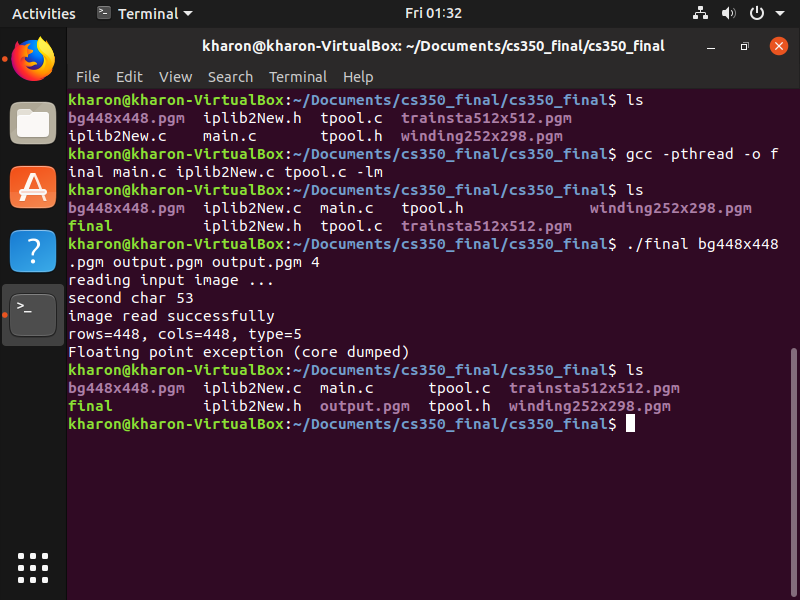
sum = 1.0/(width\*height-1.0)\*sum;

\*sd = sqrt(sum);

imagePtr2[(row/height) \* rows/height + col/width] = \*mean;

}

**Testing**



**Figure 1: Successful Compilation and Execution**

**\* PLEASE FIND OUTPUT ATTACHED IN .zip FILE \***