Lab 1 Report

Author: *Pablo Ruiz*

Due Date: *10/11/2022 5:00 PM*

Objective:

Computer Vision **Lab 1**
EECE 5841 10/08/2022

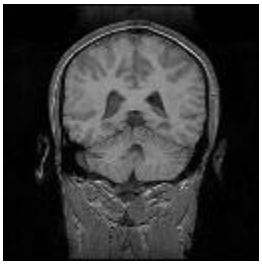
The objective of this lab is to learn and showcase how to read, manipulate and write images using the c/c++ programming language although these concepts apply to all programming languages. The first program provided, "*Program_1.cpp*", simply opens the provided image "*mri.pgm*" and subsequently saves it as a new image called "*mri2.pgm*". The second program provided, "*Program_2.cpp*", opens the provided image "*Porche.pgm*" and removes the lower right quadrant before saving the new image as "*Porche2.pgm*". The third program"*Program_3.cpp*" which is a modified version of the second program. Its purpose is to open the "*Porche.pgm*" image and remove every other pixel in both the horizontal and vertical directions, the new resulting image is then saved as "*Porche3.pgm*".

Algorithms / Functions:

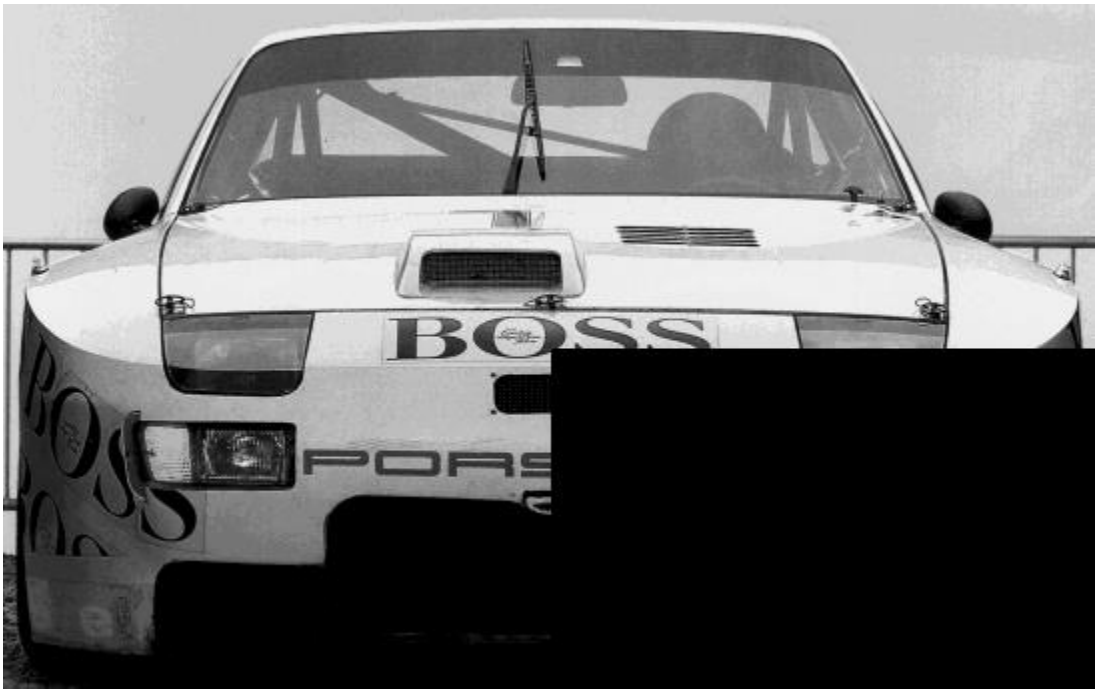
- fopen
  - Opens a file in either read or write modes.
- fscanf
  - Reads a set number of words/numbers from the file.
- fread
  - Reads the entire contents of the file into a buffer.
- fwrite
  - Writes the entire contents of a buffer into a file.
- Fclose
  - Closes the handle to a file which was previously opened using fopen.

Result:

- Program_1 Output:



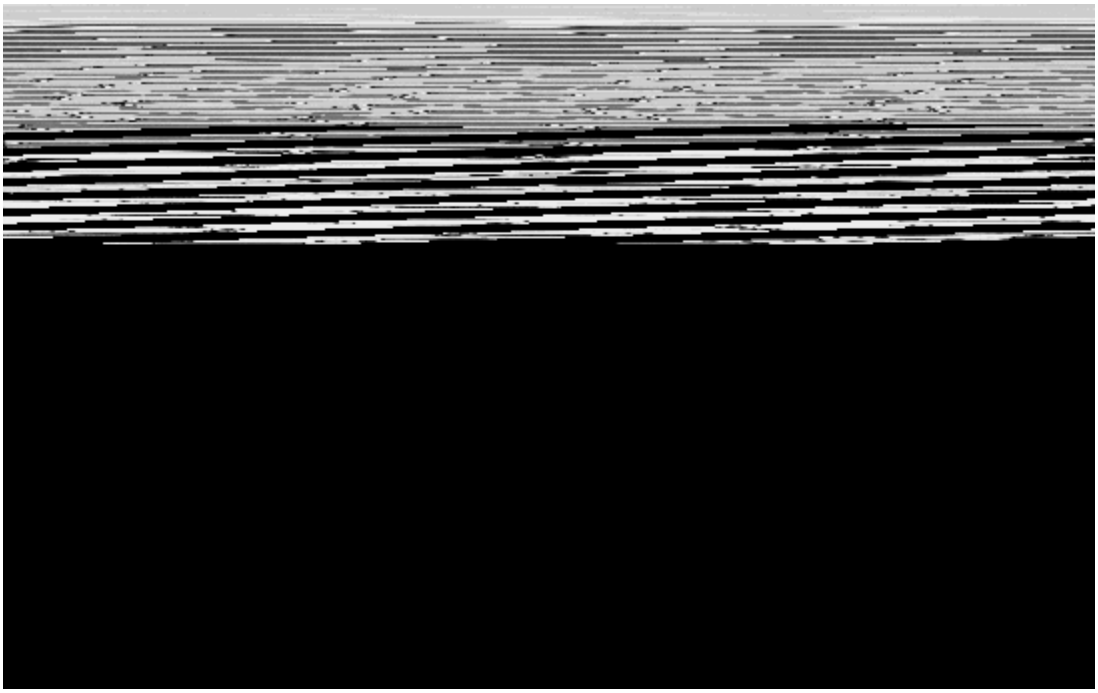
- Program_2 Output:



- Program_3 Output:

Observations:

The provided example program "*Program_2.cpp*" was originally returning a segmentation fault error with the "*mri.pgm*" image as an input and producing the following erroneous output with the "*Porche.pgm*" image as input.



Erroneous Output from original Program_2.cpp with Porche.pgm as input

This behavior was due to the fact that the program had been designed for a different image which was square and had a 256x256 resolution. This size was hard coded into the for loop which generated the output image and thus would produce a try to access out of range array members with mri.pgm because of its smaller size of 128x128, and would not work with Porche.pgm which has a larger resolution of 546x342 and thus produced all of the empty black space in the above erroneous output. Furthermore, there was a bug in the code that was masked by the fact that the previous image was square. The processMatrix 2D array's dimensions were declared incorrectly, originally the declaration was "*unsigned char processMatrix[sizeX][sizeY]*" when it should have been "*unsigned char processMatrix[sizeY][sizeX]*" this is what caused the noise seen in the first half of the above image.

After fixing the bugs present in Program_2, the changes made for Program_3 were minimal. The dimensions of the output image were changed to be half of the vertical and horizontal dimensions for the input image since a large part of the pixels are being discarded. This also means that the information written to the beginning of the output pgm file also has to change to reflect the new dimension of the image. Finally, the doubly nested for loop responsible for creating the output image was modified so that if the row number represented by "i" was uneven it would simply continue to the next iteration and skip the row entirely. Additionally, in the nested for loop when "j" is an uneven value, representing an uneven column number, that pixel value is discarded and not written to the output image.

Conclusions:

This lab showcased how to manipulate images through code, one pixel at a time. It emphasized the necessity to keep track of array indexes and memory allocation, otherwise the output can easily become unrecognizable as shown above. It also indirectly highlighted the importance of understanding image encoding format standards which are what allows a program to read and interpret what is essentially a text file and display it as an image.

Source Code:

Program 1:

```
/*
Computer Vision
Assignment 1 Program 1
Author: Pablo Ruiz
*/
//Sample Program to Read/Write Image
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
unsigned sizeX; //image width
unsigned sizeY; //image height
unsigned char *image; //image array
unsigned levels;
char pgmType[3];

/* Read Image */
FILE *iFile = fopen("mri.pgm","r");
if(iFile==0) return 1;
if(4!=fscanf(iFile, "%s %d %d %d ", pgmType, &sizeX, &sizeY, &levels)) return 1;
image=(unsigned char *) malloc(sizeX*sizeY);
fread(image,sizeof(unsigned char),sizeX*sizeY,iFile);
fclose(iFile);

/*write image to file*/
iFile = fopen("mri2.pgm","w");
if(iFile==0) return 1; //error handling
fprintf(iFile, "%s\n%d %d\n%d\n", pgmType, sizeX,sizeY,255);//write header
fwrite(image, sizeof(unsigned char), sizeX*sizeY, iFile); //write binary image
fclose(iFile);
```

```
return 0;
}
```

Program 2:

```
/*
Computer Vision
Assignment 1 Program 2
Author: Pablo Ruiz
*/

//Takes an image and removes the 4th quadrant
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[]){

unsigned sizeX; //image width
unsigned sizeY; //image height
unsigned char *image; //image array
unsigned char *imageTemp; //temporary image array
unsigned int k; //iterator for storing image data from a matrix
unsigned levels;
char pgmType[3];

/* Read Image */
FILE *iFile = fopen("Porche.pgm","r");
if(iFile==0) return 1;
if(4!=fscanf(iFile, "%s %d %d %d ", pgmType, &sizeX, &sizeY, &levels)) return 1;
image=(unsigned char *) malloc(sizeX*sizeY);
fread(image,sizeof(unsigned char),sizeX*sizeY,iFile);
fclose(iFile);

//The new image
imageTemp = (unsigned char *) malloc(sizeX*sizeY);

//Convert image array into a matrix
unsigned char processMatrix [sizeY][sizeX];
```

```
//This line copies a 1D array into an 2D array
memcpy(processMatrix, image, sizeX*sizeY*sizeof(unsigned char));

k = 0;
//Copy only quadrants 1, 2, and 3. Set 4th quadrant to black
for(unsigned int i=0; i<sizeY; i++){
for(unsigned int j=0; j<sizeX; j++){
if (!(i > sizeY/2 && j > sizeX/2)){
imageTemp[k] = processMatrix[i][j];
//imageTemp[k] = image[k];
}
else{
imageTemp[k] = 0;
}
k++;
}
}

/*write image to file*/
iFile = fopen("Porche2.pgm","w");
if(iFile==0) return 1; //error handling
fprintf(iFile, "%s\n%d %d\n%d\n", pgmType,sizeX,sizeY,255);//write header
fwrite(imageTemp, sizeof(unsigned char), sizeX*sizeY, iFile); //write binary image
fclose(iFile);

return 0;
}
```

Program 3:

```
/*
Computer Vision
Assignment 1 Program 3
Author: Pablo Ruiz
*/

//Takes an image and removes the uneven pixels in the horizontal and vertical directions
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main(int argc, char *argv[]){

unsigned sizeX; //image width
unsigned sizeY; //image height
unsigned char *image; //image array
unsigned char *imageTemp; //temporary image array
unsigned int k; //iterator for storing image data from a matrix
unsigned levels;
char pgmType[3];

/* Read Image */
FILE *iFile = fopen("Porche.pgm","r");
if(iFile==0) return 1;
if(4!=fscanf(iFile, "%s %d %d %d ", pgmType, &sizeX, &sizeY, &levels)) return 1;
image=(unsigned char *) malloc(sizeX*sizeY);
fread(image,sizeof(unsigned char),sizeX*sizeY,iFile);
fclose(iFile);

//The new image
imageTemp = (unsigned char *) malloc((sizeX/2)*(sizeY/2));

//Convert image array into a matrix
unsigned char processMatrix [sizeY][sizeX];
//This line copies a 1D array into an 2D array
memcpy(processMatrix, image, sizeX*sizeY*sizeof(unsigned char));

k = 0;
//Copy only quadrants 1, 2, and 3. Set 4th quadrant to black
for(unsigned int i=0; i<sizeY; i++){

//If the row is uneven skip it
if (i%2 != 0) continue;

for(unsigned int j=0; j<sizeX; j++){
//If the column is even write to output image otherwise skip
if (j%2 == 0){
imageTemp[k] = processMatrix[i][j];
++k;
}

}
}

/*write image to file*/
```

```
iFile = fopen("Porche3.pgm","w");
if(iFile==0) return 1; //error handling
fprintf(iFile, "%s\n%d %d\n%d\n", pgmType,sizeX/2,sizeY/2,255);//write header
fwrite(imageTemp, sizeof(unsigned char), (sizeX/2)*(sizeY/2), iFile); //write binary image
fclose(iFile);

return 0;
}
```