----------------------------------------------------------------------------------------------

EE569 Sahana Venkatesh #1

# Date: 5 February 2017

# Name: Sahana Venkatesh

# ID: 6643244225

# email: [sahanave@usc.edu](mailto:sahanave@usc.edu)

OS:macOS Sierra 10.12.2

Problem 1)

A)(i)Mirroring

// This sample code reads in image data from a RAW image file and

// writes it into another file

// NOTE: The code assumes that the image is of size 256 x 256 and is in the

// RAW format. You will need to make corresponding changes to

// accommodate images of different sizes and/or types

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

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

{

// Define file pointer and variables

FILE \*file;

int BytesPerPixel;

int i;

int j;

int k;

int Wid\_Size=256;

int H\_Size=256;

// Check for proper syntax

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "./main dog.raw output\_image.raw [BytesPerPixel=3][Height=300][Width=300]" << endl;

return 0;

}

// Check if image is grayscale or color

if (argc < 4){

BytesPerPixel = 1; // default is grey image

}

else {

BytesPerPixel = atoi(argv[3]);

// Check if size is specified

if (argc >= 5){

Wid\_Size = atoi(argv[4]);//width of the image

H\_Size=atoi(argv[5]);//height of the image

}

}

// Allocate image data array

unsigned char Imagedata[H\_Size][Wid\_Size][BytesPerPixel];

unsigned char outImagedata[H\_Size][Wid\_Size][BytesPerPixel];

// Read image (filename specified by first argument) into image data matrix

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char), H\_Size\*Wid\_Size\*BytesPerPixel, file);

fclose(file);

///////////////////////// INSERT YOUR PROCESSING CODE HERE /////////////////////////

// Write image data (filename specified by second argument) from image data matrix

//\*\*\*\*\*\* //Mirroring//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//1)The idea is the rows of the image matrix remains the same but the columns get flipped when you mirror the image

cout <<"The inputfile has been taken in ."<<endl;

cout<<"Running the code for mirroring";

//CODE FOR MIRRORING

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for (i=0;i<H\_Size;i++)//icounting the rows

{

for (j=0;j<Wid\_Size;j++)//couting all the columns

{

for (k=0;k<BytesPerPixel;k++)

{

outImagedata[i][j][k]=Imagedata[i][(Wid\_Size-1)-j][k];//flipping the imagematrix

}

}

}

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

fwrite(outImagedata, sizeof(unsigned char), H\_Size\*Wid\_Size\*BytesPerPixel, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

return 0;

}

b)resizing

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <cmath>

using namespace std;

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

{

// Define file pointer and variables

FILE \*file;

int BytesPerPixel=3;

int i;

int j;

int k;

double a;

double b;

int Wid\_Size=300;

int H\_Size=300;

int Width=300;

int Height=300;

double ratio\_image;

// Check for proper syntax

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "program\_name dog.raw output\_image.raw [BytesPerPixel = 1] [Initial Size=300][New size=400] " << endl;

return 0;

}

// Check if image is grayscale or color

BytesPerPixel = atoi(argv[3]);

// Check if size is specified

if (argc >= 5){

Wid\_Size = atoi(argv[4]);//width of the original image

H\_Size=atoi(argv[4]);//height of the original image

Width=atoi(argv[5]);//Width of new resiszed image;

Height=atoi(argv[5]);//Width of the new resized image;

}

// Allocate image data array

unsigned char Imagedata[H\_Size][Wid\_Size][BytesPerPixel];//Original image

unsigned char outImagedata[Height][Width][BytesPerPixel];//Output image

// Read image (filename specified by first argument) into image data matrix

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata,sizeof(unsigned char),H\_Size\*Wid\_Size\*BytesPerPixel,file);

fclose(file);

/////////////////////////////////////////////////////////////////////////////////

// Write image data (filename specified by second argument) from image data matrix

for (i=0;i<Height;i++)//counting the rows

{

for (j=0;j<Width;j++)//couting all the columns

{

int p1=0;

int q1=0;

ratio\_image=(1.0\*H\_Size/Height);

double p,q;

p=i\*ratio\_image;

q=j\*ratio\_image;

//ratio\_image=300/200;

//ratio\_image=300/600; //input dimension/output dimension

// Now interpolate

// Interpolate for each colour channel

if(H\_Size>Height)

{

p1=ceil(p);

q1=ceil(q);

outImagedata[i][j][0]=Imagedata[p1][q1][0];

outImagedata[i][j][1]=Imagedata[p1][q1][1];

outImagedata[i][j][2]=Imagedata[p1][q1][2];

}

else{

p1=floor(p);

q1=floor(q);

b=p-p1;

a=q-q1;

outImagedata[i][j][0]=floor((1-b)\*((1-a)\*Imagedata[p1][q1][0]+(a\*Imagedata[p1][q1+1][0]))+b\*((1-a)\*(Imagedata[p1+1][q1][0])+a\*(Imagedata[p1+1][q1+1][0])));

outImagedata[i][j][1]=floor((1-b)\*((1-a)\*Imagedata[p1][q1][1]+(a\*Imagedata[p1][q1+1][1]))+b\*((1-a)\*(Imagedata[p1+1][q1][1])+a\*(Imagedata[p1+1][q1+1][1])));

outImagedata[i][j][2]=floor((1-b)\*((1-a)\*Imagedata[p1][q1][2]+(a\*Imagedata[p1][q1+1][2]))+b\*((1-a)\*(Imagedata[p1+1][q1][2])+a\*(Imagedata[p1+1][q1+1][2])));

}

}

}

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char),Height\*Width\*BytesPerPixel, file);

fclose(file);

cout<<"Resizing the above image to (requested dimensions) the above image would be stored in your directory as output\_image.raw"<<endl;

return 0;

}

3)Image composting

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <cstdint>

#include <cstring>

using namespace std;

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

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Declaring the variabls //

FILE \*file1;

FILE \*file2;

int Wid\_first=400;//width of the first(dog) image

int H\_first=400;//height of the first (dog) image

int Wid\_second=1914;//width of the second(beach) image;

int H\_second=808;

int BytesPerPixel=3;

int j=0;

int k=0;

int i=0;

signed int background[H\_first][Wid\_first];

//height of the second(beach) image

if (argc < 1){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "program\_name input\_image1.raw input\_image2.raw output\_image.raw" << endl;

return 0;

}

cout<<"Assuming the following the width and height of the first image is 300 and width of the second image is 1914 and height is 808"<<endl;

unsigned char Imagedata2[808][1914][BytesPerPixel];

unsigned char Imagedata1[H\_first][Wid\_first][BytesPerPixel];

cout<<"Declaration done ";

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Reading the files //

cout<<"\n Reading the first image";

if (!(file2=fopen(argv[2],"rb"))) {

cout << "Cannot open file: " << argv[2] <<endl;

exit(1);

}

fread(Imagedata2, sizeof(unsigned char), 808\*1914\*3,file2);

fclose(file2);

cout<<"\n Reading the second image"<<endl;

cout<<"The first image is to be placed on the second image with its top-left corner at coordinates (1100, 400) of the second image"<<endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Now merging both the images\*\*\*\*all the non\_background colours dont have blue colour

i=0;

j=0;

if (!(file1=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata1, sizeof(unsigned char), H\_first\*Wid\_first\*BytesPerPixel,file1);

fclose(file1);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Assuming that the background values are same everywhere

int r=0;

int g=0;

int b=0;

int r\_back=0;

int g\_back=0;

int b\_back=0;

r\_back=(int)Imagedata1[0][0][0];

g\_back=(int)Imagedata1[0][0][1];

b\_back=(int)Imagedata1[0][0][2];

for(i=0;i<H\_first;i++)

{

for(j=0;j<Wid\_first;j++)

{

r=(int)Imagedata1[i][j][0];

g=(int)Imagedata1[i][j][1];

b=(int)Imagedata1[i][j][2];

if((r==r\_back)&(g==g\_back)&&(b==b\_back)) {

background[i][j]=(-1);

}

else

{

background[i][j]=1;

}

}

}

for(i=400;i<800;i++)

{

for (j=1100;j<1500;j++)

{

if(background[i-400][j-1100]==1)

{

for (k=0;k<BytesPerPixel;k++)

{

Imagedata2[i][j][k]=Imagedata1[i-400][j-1100][k];

}

}

}

}

if (!(file2=fopen(argv[3],"wb"))) {

cout << "Cannot open file: " << argv[3] << endl;

exit(1);

}

fwrite(Imagedata2,sizeof(unsigned char),H\_second\*Wid\_second\*BytesPerPixel,file2);

fclose(file2);

return 0;

}

B)1.CMYK

/\*

EE-569 HOMEWORK 1 SAHANA VENKATESH 6643244225\*/

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

using namespace std;

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

{

// Define file pointer and variables

FILE \*file;

int BytesPerPixel;

int Width\_Size = 256;

int Height\_Size = 256;

int i,j;

// Check for proper syntax

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "program\_name parrot.raw output\_cyan\_image.raw output\_magenta\_image.raw output\_yellow\_image.raw check.raw" << endl;

return 0;

}

// Check if image is grayscale or color

if (argc < 4){

BytesPerPixel = 1; // default is grey image

}

else {

BytesPerPixel = 3;

// Check if size is specified

if (argc >= 5){

Width\_Size = 789;//Width of the input image

Height\_Size = 600;//Height of the input image

}

}

// Allocate image data array

unsigned char Imagedata[Height\_Size][Width\_Size][BytesPerPixel];

unsigned char Imagedata\_cyan[Height\_Size][Width\_Size];

unsigned char Imagedata\_magenta[Height\_Size][Width\_Size];

unsigned char Imagedata\_yellow[Height\_Size][Width\_Size];

unsigned char check[Height\_Size][Width\_Size];

// Read image (filename specified by first argument) into image data matrix

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char), Height\_Size\*Width\_Size\*BytesPerPixel, file);

fclose(file);

for (i=0; i<Height\_Size; i++) {

for (j=0; j<Width\_Size; j++){

Imagedata\_cyan[i][j]=((255-Imagedata[i][j][0]));

Imagedata\_magenta[i][j]=((255-Imagedata[i][j][1]));

Imagedata\_yellow[i][j]=((255-Imagedata[i][j][2]));

check[i][j]=Imagedata[i][j][0];

}

}

///////////////////////// INSERT YOUR PROCESSING CODE HERE /////////////////////////

// Write image data (filename specified by second argument) from image data matrix

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

cout<<"Saving cyan gray scale"<< endl;

fwrite(Imagedata\_cyan, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

if (!(file=fopen(argv[3],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

cout<<"Saving magenta gray scale"<< endl;

fwrite(Imagedata\_magenta, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

if (!(file=fopen(argv[4],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

cout<<"Saving yellow gray scale"<< endl;

fwrite(Imagedata\_yellow, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

if (!(file=fopen(argv[5],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

cout<<"Saving yellow gray scale"<< endl;

fwrite(check, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

return 0;

}

e)HSL

// This sample code reads in image data from a RAW image file and

// writes it into another file

// NOTE: The code assumes that the image is of size 256 x 256 and is in the

// RAW format. You will need to make corresponding changes to

// accommodate images of different sizes and/or types

/\*Sahana venkatesh hw\_1\_569 dsp;id:6643244225\*/

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <math.h>

#include <algorithm>

using namespace std;

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

{

// Define file pointer and variables

FILE \*file;

int BytesPerPixel;

int Width\_Size = 256;

int Height\_Size = 256;

int i,j;

// Check for proper syntax

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "program\_name cat.raw hue.raw satuaration.raw luminance.raw " << endl;

return 0;

}

// Check if image is grayscale or color

if (argc < 4){

BytesPerPixel = 1; // default is grey image

}

else {

BytesPerPixel = 3;

// Check if size is specified

if (argc >= 5){

//Cat.raw 600x398 Dolphin.raw 640x480//

Width\_Size = 640;//Width of the input image

Height\_Size = 480;//Height of the input image

}

}

// Allocate image data array

unsigned char Imagedata[Height\_Size][Width\_Size][BytesPerPixel];

unsigned char Imagedata\_hue[Height\_Size][Width\_Size];

unsigned char Imagedata\_sat[Height\_Size][Width\_Size];

unsigned char Imagedata\_lum[Height\_Size][Width\_Size];

// Read image (filename specified by first argument) into image data matrix

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char), Height\_Size\*Width\_Size\*BytesPerPixel, file);

fclose(file);

for (i=0; i<Height\_Size; i++)

{

for (j=0; j<Width\_Size;j++)

{

double M,m,C;

double L;

double r,g,b;

r=(Imagedata[i][j][0])/255.0;

g=(Imagedata[i][j][1])/255.0;

b=(Imagedata[i][j][2])/255.0;

M=max(max(r,g),b);

m=min(min(r,g),b);

C=(M-m);

r=r;

g=g;

b=b;

//----------------------------Evaluating the Luminance for each pixel //------------------------------------------------//

L=(M+m)/2;

Imagedata\_lum[i][j]=(unsigned char)255.0\*(L);

//Converting all to 0-255 range for help in displaying the image

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

//Evaluating the Saturation value for each pixel

double S;

if(L==0)

{

S=0;

}

else if (0<L<=0.5)

{

S=C/(2.0\*L);

}

else

{

S=C/(2.0-(2.0\*L));

}

Imagedata\_sat[i][j]=(unsigned char)255.0\*S;

//Converting all to 0-255 range for help in displaying the image

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

//Evaluating the Hue for each pixel

double B,R,G;

double H=0;

if(C==0)

{

H=0;

}

else{

if(M==r) //M=R

{

R=fmod(((g-b)/C),6.0);

H=60.0\*R;

}

if(M==g)//M=G

{

G=((b-r)/C)+2.0;

H=60.0\*G;

}

if(M==b){//M=b

B=((r-g)/C)+4.0;

H=60.0\*B;

//Converting all to 0-255 range for help in displaying the image

}

}

Imagedata\_hue[i][j]=(unsigned char)(255.0\*H/360.0);

//Converting all to 0-255 range for help in displaying the image

}

}

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

cout<<"Saving Hue as a gray scale image"<< endl;

fwrite(Imagedata\_hue, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

if (!(file=fopen(argv[3],"wb"))) {

cout << "Cannot open file: " << argv[3] << endl;

exit(1);

}

cout<<"Saving Saturation as gray scale image"<< endl;

fwrite(Imagedata\_sat, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

if (!(file=fopen(argv[4],"wb"))) {

cout << "Cannot open file: " << argv[4] << endl;

exit(1);

}

cout<<"Saving Luminance as gray scale image"<< endl;

fwrite(Imagedata\_lum, sizeof(unsigned char), Height\_Size\*Width\_Size, file);

fclose(file);

return 0;

}

f)Sepia

/\*

Name:Sahana VENKATESH

ID:6642344225

usc id:sahanave@usc.edu

Date:2/5/17

\*/

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <math.h>

#include <algorithm>

using namespace std;

int Height=808;

int Width=1914;

int BytesPerPixel=3;

unsigned char Imagedata[808][1914][3];

unsigned char Intermediate[808][1914];

void rgb\_to\_grayscale()

{

int i;

int j;

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

Intermediate[i][j]=(unsigned char)floor(0.21\*(Imagedata[i][j][0]))+(0.72\*(Imagedata[i][j][1]))+(0.07\*(Imagedata[i][j][2]));

//I=0.21\*r+0.72\*g+0.07\*b;

}

}

}

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

{

// Check for proper syntax

if (argc < 2){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "./main beach.raw intermidiate\_image.raw output\_image.raw" << endl;

return 0;

}

// Define file pointer and variables

FILE \*file;

int i,j;

// Allocate image data array

// Read image (filename specified by first argument) into image data matrix

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);

fclose(file);

///////////////////////// INSERT YOUR PROCESSING CODE HERE /////////////////////////

// Write image data (filename specified by second argument) from image data matrix

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

//Sepia filter input

rgb\_to\_grayscale();

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

//Sepia filter

int r,g,b;

unsigned char Sepia\_output[Height][Width][BytesPerPixel];

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

r=floor(1.351\*(Intermediate[i][j]));

g=floor((0.349+0.686+0.168)\*(Intermediate[i][j]));

b=floor((0.272+0.534+.131)\*(Intermediate[i][j]));

Sepia\_output[i][j][0]=(unsigned char)max(min(r,255),0);

Sepia\_output[i][j][1]=(unsigned char)max(min(g,255),0);

Sepia\_output[i][j][2]=(unsigned char)max(min(b,255),0);

}

}

if (!(file=fopen(argv[3],"wb"))) {

cout << "Cannot open file: " << argv[3] << endl;

exit(1);

}

fwrite(Sepia\_output, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);

fclose(file);

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(Intermediate, sizeof(unsigned char), Height\*Width, file);

fclose(file);

return 0;

}

g) Blending

// This sample code reads in image data from a RAW image file and

// writes it into another file

// NOTE: The code assumes that the image is of size 256 x 256 and is in the

// RAW format. You will need to make corresponding changes to

// accommodate images of different sizes and/or types

/\*SAHANA VENKATESH 664234425 DSP ASSIGNMENT1 2/5/17 \*/

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

using namespace std;

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

{

// Define file pointer and variables

FILE \*file1,\*file2;

FILE \*file;

int BytesPerPixel=3;

int Height\_top = 900;

int Height\_bottom = 900;

int Width\_top = 890;

int Width\_bottom = 890;

int i,j,k;

// Check for proper syntax

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << "program\_name top\_layer.raw bottom\_layer.raw output\_image.raw" << endl;

return 0;

}

// Allocate image data array

unsigned char Imagedata\_top[Height\_top][Width\_top][BytesPerPixel];

unsigned char Imagedata\_bottom[Height\_bottom][Width\_bottom][BytesPerPixel];

unsigned char outImagedata[Height\_top][Width\_top][BytesPerPixel];

// Read image (filename specified by first argument) into image data matrix

if (!(file1=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata\_top, sizeof(unsigned char), Height\_top\*Width\_top\*BytesPerPixel, file1);

fclose(file1);

if (!(file2=fopen(argv[2],"rb"))) {

cout << "Cannot open file: " << argv[2] <<endl;

exit(1);

}

fread(Imagedata\_bottom, sizeof(unsigned char), Height\_bottom\*Width\_bottom\*BytesPerPixel, file2);

fclose(file2);

double a;

double b;

///////////////////////// INSERT YOUR PROCESSING CODE HERE /////////////////////////

// Write image data (filename specified by second argument) from image data matrix

for(i=0;i<Height\_top;i++)

{

for(j=0;j<Width\_top;j++)

{

for(k=0;k<BytesPerPixel;k++)

{

a=0;

b=0;

a=Imagedata\_bottom[i][j][k]/255.0;

b=Imagedata\_top[i][j][k]/255.0;

outImagedata[i][j][k]=(unsigned char)(floor(255\*(1-((1-a)\*(1-b)))));

}

}

}

if (!(file=fopen(argv[3],"wb"))) {

cout << "Cannot open file: " << argv[3] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char), Height\_top\*Width\_top\*BytesPerPixel, file);

fclose(file);

return 0;

}

2)Problem

a)histogram enhancement method

method A

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

/\*Sahana Venkatesh ;usc id :664324425;sahanve@usc.edu;submission data:2.5.17\*/

using namespace std;

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

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Declaring the variabls //

int j;

int k;

int i;

FILE \*file;

int Width=400;//

int Height=366;//

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << " ./main tulip\_bright.raw output\_image.raw [H\_min\_of first cluster=0][H\_max of first cluster=68] [H\_min\_2nd cluster=190] [H\_max of 2nd cluster=255] " << endl;

return 0;

}

unsigned char Imagedata[Height][Width];

unsigned char outImagedata[Height][Width];

unsigned int intensity[256];

cout<<"Declaration done "<<endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Reading the files //

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char),Width\*Height, file);

fclose(file);

cout<<" Reading the image"<<endl;

unsigned char t=0;

for(i=0;i<256;i++){

intensity[i]=0;

}

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

t=Imagedata[i][j];

intensity[t]=intensity[t]+1;

}

}

signed char u;

if(argc<7)

{

int H\_min;

H\_min=atoi(argv[3]);

int H\_max;

H\_max=atoi(argv[4]);

int G\_min=0;

int G\_max=255;

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

u=floor(((G\_max-G\_min)/(H\_max-H\_min))\*(Imagedata[i][j]-H\_min)+G\_min);

//the above value was calculated by solving the equation y=mx+c and assuming the line passes through (190,0) and(255,255)

if(0<=u<G\_max)

{

outImagedata[i][j]=u;

}

else if(u<0)

outImagedata[i][j]=-u;

else

{

if(u>G\_max)

{

outImagedata[i][j]=G\_max;

}

}

}

}

}

else

{

int H\_min\_1;

H\_min\_1=atoi(argv[3]);

int H\_max\_1;

H\_max\_1=atoi(argv[4]);

int H\_min\_2;

H\_min\_2=atoi(argv[5]);

int H\_max\_2;

H\_max\_2=atoi(argv[6]);

int G\_min\_1,G\_min\_2,G\_max\_1,G\_max\_2;

G\_min\_1=0;

G\_max\_1=H\_max\_1;

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

if(Imagedata[i][j]>H\_min\_2)

{

u=floor(((G\_max\_1-G\_min\_1)/(H\_max\_2-H\_min\_2))\*(Imagedata[i][j]-H\_min\_2)+G\_min\_1);

//the above value was calculated by solving the equation y=mx+c and assuming the line passes through (190,0) and(255,255)

if(0<=u<G\_max\_1)

{

outImagedata[i][j]=u;

}

else if(u<0)

outImagedata[i][j]=-u;

else

{

if(u>G\_max\_1)

{

outImagedata[i][j]=G\_max\_1;

}

}

}

}

}

int G\_max=255;

int G\_min=0;

int H\_max=H\_max\_1;

int H\_min=H\_min\_1;

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

u=floor((G\_max/(H\_max-H\_min))\*(Imagedata[i][j]-H\_min));

//the above value was calculated by solving the equation y=mx+c and assuming the line passes through (190,0) and(255,255)

if(0<=u<256)

{

outImagedata[i][j]=u;

}

else if(u<0)

outImagedata[i][j]=-u;

else

{

outImagedata[i][j]=255;

}

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char), Height\*Width, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}

b)method B:Histogram equalization technique

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

/\*Sahana venkatesh 6643244225 sahanave@usc.edu 2/5/17 \*/

using namespace std;

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

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Declaring the variabls //

FILE \*file;

int BytesPerPixel=3;

int j;

int k;

int i;

int Width=400;//width of the first(dog) image

int Height=366;//height of the first (dog) image

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << " ./main tulip\_bright.raw output\_image.raw " << endl;

return 0;

}

unsigned char Imagedata[Height][Width];

unsigned char outImagedata[Height][Width];

cout<<"Declaration done "<<endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Reading the files //

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char),Width\*Height, file);

fclose(file);

cout<<" Reading the image"<<endl;

//572 pixels per bucket

int total=Height\*Width;

unsigned char pixcel\_value[total];

int count\_no=0;

unsigned char t=0;

int value=0;

t=0;

int location [total][2];

//Arranging the pixels in terms of thier intensity and storing their location

while(value<256)

{ value=int(t);

for(i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

if(Imagedata[i][j]==t)

{

pixcel\_value[count\_no]=Imagedata[i][j];

location[count\_no][0]=i;

location[count\_no][1]=j;

count\_no=count\_no+1;

}

}

}

t=t+1;

value=value+1;

}

cout<<"Arranging the pixels in terms of thier intensity and storing their location"<<endl;

count\_no=0;

t=0;

//Bucket filling method

while(count\_no<(Width\*Height))

{

for(i=0;i<573;i++)

{

if(count\_no<(Width\*Height)){

pixcel\_value[count\_no]=t;

count\_no=count\_no+1;

}

}

t=t+1;

}

cout<<"Performed Bucket filling implementation"<<endl;

//Assining pixles to their modified values

count\_no=0;

int a;

int b;

for(count\_no=0;count\_no<(Height\*Width);count\_no++)

{

a=location[count\_no][0];

b=location[count\_no][1];

outImagedata[a][b]=pixcel\_value[count\_no];

}

cout<<"Assining pixles to their modified values"<<endl;

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

fwrite(outImagedata, sizeof(unsigned char), Height\*Width, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}

c)Method B for three channels

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

using namespace std;

int BytesPerPixel=3;

int Width=940;//width of the first(dog) image

int Height=400;//height of the first (dog) image

unsigned char Imagedata[400][940][3];

unsigned char outImagedata[400][940][3];

void bucketfilling(int k)

{

long int i;

int j;

int total=Height\*Width;

unsigned char pixcel\_value[total];

long int count\_no=0;

unsigned char t=0;

int value=0;

t=0;

int location [total][2];

//Arranging the pixels in terms of thier intensity and storing their location

while(value<256)

{ value=int(t);

for(i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

if(Imagedata[i][j][k]==t)

{

pixcel\_value[count\_no]=Imagedata[i][j][k];

location[count\_no][0]=i;

location[count\_no][1]=j;

count\_no=count\_no+1;

}

}

}

t=t+1;

value=value+1;

}

cout<<"Arranging the pixels in terms of thier intensity and storing their location"<<endl;

count\_no=0;

t=0;

//Bucket filling method

while(count\_no<(Width\*Height))

{

for(i=0;i<1470;i++)

{

if(count\_no<(Width\*Height)){

pixcel\_value[count\_no]=t;

count\_no=count\_no+1;

}

}

t=t+1;

}

cout<<"Performed Bucket filling implementation"<<endl;

//Assining pixles to their modified values

int a;

int b;

int c;

for(c=count\_no;c>0;c--)

{

a=location[c][0];

b=location[c][1];

outImagedata[a][b][k]=pixcel\_value[c];

}

cout<<"Assining pixles to their modified values"<<endl;

}

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

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Declaring the variabls //

FILE \*file;

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << " ./main bedroom.raw output\_bed.raw " << endl;

return 0;

}

cout<<"Declaration done "<<endl;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Reading the files //

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char),Height\*Width\*BytesPerPixel, file);

fclose(file);

cout<<" Reading the image"<<endl;

bucketfilling(0);//red

bucketfilling(1);//green

bucketfilling(2);//bluw

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}

d) Applying Method A to all the three channels

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

/\*Sahana venkatesh 6643244225 sahanave@usc.edu 2/5/17 \*/

using namespace std;

int BytesPerPixel=3;

int Width=940;//width of the first(dog) image

int Height=400;//height of the first (dog) image

unsigned char Imagedata[400][940][3];

unsigned char outImagedata[400][940][3];

void transfer\_function(int k)

{

int i,j;

int intensity[256];

unsigned char t=0;

for(i=0;i<256;i++){

intensity[i]=0;

}

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

t=Imagedata[i][j][k];

intensity[t]=intensity[t]+1;

}

}

int track\_max=0;

int track\_min=0;

for(i=0;i<256;i++)

{

if((intensity[i]!=0)&&(track\_min==0))

track\_min=i;

if(i<254)

{

if(intensity[i+1]==0)

track\_max=i;

}

}

signed char u;

int H\_min;

H\_min=track\_min;

int H\_max;

H\_max=track\_max;

int G\_min=0;

int G\_max=255;

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

u=floor(((G\_max-G\_min)/(H\_max-H\_min))\*(Imagedata[i][j][k]-H\_min)+G\_min);

//the above value was calculated by solving the equation y=mx+c and assuming the line passes through (190,0) and(255,255)

if(0<=u<G\_max)

{

outImagedata[i][j][k]=u;

}

else if(u<0)

outImagedata[i][j][k]=-u;

else

{

if(u>G\_max)

{

outImagedata[i][j][k]=G\_max;

}

}

}

}

}

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

{

int j;

int k;

int i;

FILE \*file;

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char),Height\*Width\*BytesPerPixel, file);

fclose(file);

cout<<" Reading the image"<<endl;

transfer\_function(0);//red

transfer\_function(1);//green

transfer\_function(2);//bluw

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}

histogram matching

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

/\*Sahanavenkatesh 6643244225 sahaave@usc.edu 2/5/17 \*/

using namespace std;

int BytesPerPixel=3;

int Width=960;//width of theimage

int Height=600;//height of the image 550\*413

unsigned char Imagedata[600][960][3];

unsigned char outImagedata[600][960][3];

float cdf[256];

void bucketfilling(int k)

{

long int i;

int j;

int total=Height\*Width;

unsigned char pixcel\_value[total];

long int count\_no=0;

unsigned char t=0;

int value=0;

t=0;

int location [total][2];

int intensity[256];

double cdf\_image[256];

double pdf\_image[256];

//Arranging the pixels in terms of thier intensity and storing their location

for(i=0;i<256;i++){

intensity[i]=0;

}

for (i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

t=Imagedata[i][j][k];

intensity[t]=intensity[t]+1;

}

}

for(i=0;i<256;i++)

{

pdf\_image[i]=intensity[i]/(960\*600);

cout<<pdf\_image[i]<<endl;

}

int temp;

for(i=0;i<256;i++)

{

for(j=i;j>0;j--)

{

temp=pdf\_image[i]+temp;

}

cdf\_image[i]=temp;

temp=0;

cout<<cdf\_image[i]<<endl;

}

int temp2;

int m;

for(i=0;i<Height;i++)

{

for(j=0;j<Height;j++)

{

temp2=Imagedata[i][j][k];

for(m=0;m<256;m++)

{

if(cdf\_image[temp2]<cdf[m])

break;

outImagedata[i][j][k]=m;

}

}

}

}

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

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Declaring the variabls //

int j;

int i;

FILE \*file;

if (argc < 3){

cout << "Syntax Error - Incorrect Parameter Usage:" << endl;

cout << " ./main forest\_1.raw output.raw " << endl;

return 0;

}

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " << argv[1] <<endl;

exit(1);

}

fread(Imagedata, sizeof(unsigned char),Height\*Width\*BytesPerPixel, file);

fclose(file);

cout<<" Reading the image"<<endl;

float pdf[256];

float cdf[256];

for(i=0;i<256;i++)

{

pdf[i]=( 1 /( 20 \* sqrt(2\*3.14) ) ) \* exp( -0.5 \* pow( (i-70.0)/20.0, 2.0 ) );

}

for(i=0;i<256;i++)

{

cdf[i]=0;

}

for(i=0;i<256;i++)

{

for(j=i;j>0;j--)

{

cdf[i]=pdf[j]+cdf[i];

}

}

bucketfilling(0);

bucketfilling(1);

bucketfilling(2);

if (!(file=fopen(argv[2],"wb"))) {

cout << "Cannot open file: " << argv[2] << endl;

exit(1);

}

fwrite(outImagedata, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}

3)Mean AND MEDIAN FILTER

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

#include<vector>

#include<algorithm>

using namespace std;

unsigned char Imagedata\_org[512][512][3];

unsigned char arrange\_red[9];

unsigned char arrange\_blue[9];

unsigned char arrange\_green[9];

void insert\_me(int a[9])

/\*Sahana venkatesh 664324425 sahanve@usc.edu 2/5/15 \*/

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

{

FILE \*file;

int BytesPerPixel=3;

int j;

int k;

int i;

int Width=512;//width of the image

int Height=512;

int new\_Height;

int new\_Width;//height of the image

int window;

int extend;

unsigned char outImagedata[512][512][3];

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " <<endl;

cout<<"Program name pepper\_noisy.raw output\_a.raw windowsize";

exit(1);

}

fread(Imagedata\_org, sizeof(unsigned char),512\*512\*3, file);

fclose(file);

cout<<" Reading the image"<<endl;

//1)Extending the image in accordance with the window size

window=(int)atoi(argv[3]);

extend=floor(window/2);

new\_Height=Height+2\*extend;

new\_Width=Width+2\*extend;

unsigned char Imagedata[new\_Height][new\_Width][3];

unsigned char outImagedata\_mean[Height][Width][3];

unsigned char outImagedata\_median[Height][Width][3];

for(i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

for(k=0;k<BytesPerPixel;k++)

{

Imagedata[i+extend][j+extend][k]=Imagedata\_org[i][j][k];

}

}

}

for(j=0;j<Width;j++)

{

for(k=0;k<BytesPerPixel;k++)

{

int e;

for(e=extend-1;e>-1;e--)

{

Imagedata[e][j+extend][k]=Imagedata\_org[0][j][k];

Imagedata[(Height+extend)+e][j+extend][k]=Imagedata\_org[0][j][k];

}

}

}

for(i=0;i<new\_Height;i++)

{

for(k=0;k<BytesPerPixel;k++)

{

int e;

for(e=extend-1;e>-1;e--)

{

Imagedata[i][e][k]=Imagedata[extend][extend][k];

Imagedata[i][(Width+extend)+e][k]=Imagedata[extend][extend][k];

}

}

}

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

int window\_height=0;

int window\_width=0;

int sum[3]={0,0,0};

int c;

for(i=extend;i<new\_Height-extend;i++)

{

for(j=extend;j<new\_Width-extend;j++)

{

sum[0]=0;

sum[1]=0;

sum[2]=0;

c=0;

for (window\_height=i-extend;window\_height<=i+extend;window\_height++)

{

for (window\_width=j-extend;window\_width<=j+extend;window\_width++){

sum[0]=Imagedata[window\_height][window\_width][0]+sum[0];

arrange\_red[c]=Imagedata[window\_height][window\_width][0];

sum[1]=Imagedata[window\_height][window\_width][1]+sum[1];

arrange\_green[c]=Imagedata[window\_height][window\_width][1];

sum[2]=Imagedata[window\_height][window\_width][2]+sum[2];

arrange\_blue[c]=Imagedata[window\_height][window\_width][2];

c=c+1;

}

}

outImagedata\_mean[i][j][0]=(unsigned char)floor((sum[0]/(window\*window)));

outImagedata\_mean[i][j][1]=(unsigned char)floor((sum[1]/(window\*window)));

outImagedata\_mean[i][j][2]=(unsigned char)floor((sum[2]/(window\*window)));

sort(arrange\_red,arrange\_red+(window\*window));

outImagedata\_median[i][j][0]=arrange\_red[4];

sort(arrange\_blue,arrange\_blue+(window\*window));

outImagedata\_median[i][j][2]=arrange\_blue[4];

sort(arrange\_green,arrange\_green+(window\*window));

outImagedata\_median[i][j][1]=arrange\_green[4];

}

}

//Viewing the images

if (!(file=fopen("median\_output\_first.raw","wb"))) {

cout << "Cannot open file: " << endl;

exit(1);

}

fwrite(outImagedata\_median, sizeof(unsigned char),Height\*Width\*BytesPerPixel,file);//saving the image in row formay

fclose(file);

if (!(file=fopen("mean\_output\_first.raw","wb"))) {

cout << "Cannot open file: " << endl;

exit(1);

}

fwrite(outImagedata\_mean, sizeof(unsigned char),Height\*Width\*BytesPerPixel,file);//saving the image in row formay

fclose(file);

if (!(file=fopen("output.raw","wb"))) {

cout << "Cannot open file: " << endl;

exit(1);

}

fwrite(Imagedata, sizeof(unsigned char),new\_Height\*new\_Width\*BytesPerPixel,file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved."<<endl;

return 0;

}

NLM

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include<math.h>

using namespace std;

unsigned char Imagedata\_org[512][512][3];

unsigned char outImagedata[512][512][3];

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

{

FILE \*file;

int BytesPerPixel=3;

int j;

int k;

int i;

int Width=512;//width of the image

int Height=512;//height of the image

long int c1;//height of the image

int new\_Height;

int new\_Width;//height of the image

int window;

int extend;

int d;

int s\_ht=0;

int s\_wid=0;

int intensity[324][3];

double weight[324];

double distance;

double t;

if (!(file=fopen(argv[1],"rb"))) {

cout << "Cannot open file: " <<endl;

exit(1);

}

fread(Imagedata\_org, sizeof(unsigned char),Height\*Width\*BytesPerPixel, file);

fclose(file);

cout<<" Reading the image"<<endl;

//1)Extending the image in accordance with the window size

window=(int)atoi(argv[3]);

d=pow(window,2);

extend=floor(window/2);

new\_Height=Height+2\*extend;

new\_Width=Width+2\*extend;

unsigned char Imagedata[new\_Height][new\_Width][3];

unsigned char outImagedata\_mean[Height][Width][3];

unsigned char outImagedata\_median[Height][Width][3];

for(i=0;i<Height;i++)

{

for(j=0;j<Width;j++)

{

for(k=0;k<BytesPerPixel;k++)

{

Imagedata[i+extend][j+extend][k]=Imagedata\_org[i][j][k];

}

}

}

for(j=0;j<Width;j++)

{

for(k=0;k<BytesPerPixel;k++)

{

int e;

for(e=extend-1;e>-1;e--)

{

Imagedata[e][j+extend][k]=Imagedata\_org[0][j][k];

Imagedata[(Height+extend)+e][j+extend][k]=Imagedata\_org[0][j][k];

}

}

}

for(i=0;i<new\_Height;i++)

{

for(k=0;k<BytesPerPixel;k++)

{

int e;

for(e=extend-1;e>-1;e--)

{

Imagedata[i][e][k]=Imagedata[extend][extend][k];

Imagedata[i][(Width+extend)+e][k]=Imagedata[extend][extend][k];

}

}

}

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

//Implementing the NLM Filter

int a=0;

int b=0;

int r=0;

long int c=0;

unsigned char Search[window][window][BytesPerPixel];

unsigned char Reference[3];

unsigned char Patch[3];

for(i=0;i<new\_Height-window;i++)

{

for(j=0;j<new\_Width-window;j++)

{

//----------Search window-----------//

for(a=0;a<window;a++)

{

for(b=0;b<window;b++)

{

for(k=0;k<BytesPerPixel;k++)

{

Search[a][b][k]=Imagedata[a+i][b+j][k];

}

}

}

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

//Calculating the mean of the Reference patch of my Search window//

int window\_height=0;

int window\_width=0;

int sum[3]={0,0,0};

for (window\_height=0;window\_height<3;window\_height++)

{

for (window\_width=0;window\_width<3;window\_width++){

sum[0]=Imagedata[window\_height+i+extend][window\_width+j+extend][0]+sum[0];

sum[1]=Imagedata[window\_height+i+extend][window\_width+j+extend][1]+sum[1];

sum[2]=Imagedata[window\_height+i+extend][window\_width+j+extend][2]+sum[2];

}

}

Reference [0]=(unsigned char)floor((sum[0]/(9)));

Reference [1]=(unsigned char)floor((sum[1]/(9)));

Reference [2]=(unsigned char)floor((sum[2]/(9)));

sum[0]=0;

sum[1]=0;

sum[2]=0;

c=0;

for(s\_ht=0;s\_ht<=window-3;s\_ht++)

{

for(s\_wid=0;s\_wid<=window-3;s\_wid++)

{

sum[0]=0;

sum[1]=0;

sum[2]=0;

for (window\_height=0;window\_height<3;window\_height++)

{

for (window\_width=0;window\_width<3;window\_width++)

{

sum[0]=(int)Search[window\_height+s\_ht][window\_width+s\_wid][0]+sum[0];

sum[1]=(int)Search[window\_height+s\_ht][window\_width+s\_wid][1]+sum[1];

sum[2]=(int)Search[window\_height+s\_ht][window\_width+s\_wid][2]+sum[2];

if((window\_width==1)&&(window\_height==1))

{

intensity[c][0]=Search[window\_height+s\_ht][window\_width+s\_wid][0];

intensity[c][1]=Search[window\_height+s\_ht][window\_width+s\_wid][1];

intensity[c][2]=Search[window\_height+s\_ht][window\_width+s\_wid][2];

}

}

}

Patch[0]=(unsigned char)floor((sum[0]/(d)));

Patch[1]=(unsigned char)floor((sum[1]/(d)));

Patch[2]=(unsigned char)floor((sum[2]/(d)));

distance=0;

for(k=0;k<3;k++)

{

t=(double)(Patch[k]-Reference[k]);

distance=pow(t,2)+distance;

}

weight[c]=(double)exp(-distance/100);

c=c+1;

}

}

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

//Calculating the weights//

//Calculating the weights for all teh pixels in the search window

long int b;

long double s=0;

for(b=0;b<c;b++)

{

s=weight[b]+s;

}

for(b=0;b<c;b++)

{

weight[b]=weight[b]/s;

}

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

outImagedata[i+extend][j+extend][0]=0;

outImagedata[i+extend][j+extend][1]=0;

outImagedata[i+extend][j+extend][2]=0;

for(b=0;b<c;b++)

{

outImagedata[i+extend][j+extend][0]=weight[b]\*intensity[b][0]+outImagedata[i+extend][j+extend][0];

outImagedata[i+extend][j+extend][1]=weight[b]\*intensity[b][1]+outImagedata[i+extend][j+extend][1];

outImagedata[i+extend][j+extend][2]=weight[b]\*intensity[b][2]+outImagedata[i+extend][j+extend][2];

}

}

}

if (!(file=fopen("outputnlp.raw","wb")))

{

cout << "Cannot open file: " << endl;

exit(1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

fwrite(outImagedata, sizeof(unsigned char), Height\*Width\*BytesPerPixel, file);//saving the image in row formay

fclose(file);

cout <<"\n The output is saved. "<<endl;

return 0;

}