|  |
| --- |
| **Implementation of Basic Intensity Transformations** |

**// RGB Color Model**

#include <iostream>

#include <string>

// #include<math>

// #include<conio>

#include<fstream>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

char h[54];

ifstream in;

ofstream out,out1;

int main()

{

char infile[]="c:\\marble.bmp";

int i,j;

i=j=0;

in.open(infile,ios::in|ios::binary);

in.read((char\*)(&h),sizeof(h));

while(!in.eof())

{

in.read((char\*)(&pixel),sizeof(pixel));

cout<<"\n"<<(int)pixel.r <<" "<<(int)pixel.g <<" "<<(int)pixel.b ;

}

in.close();

//getch();

}

**INPUT**

****

**OUTPUT**

****

**//Program to show negative image**

#include <iostream>

#include <string>

// #include<math>

// #include<conio>

#include<fstream>

using namespace std;

int L=255;

typedef unsigned char byte;

byte h[54];

byte pixel;

ofstream out;

ifstream in;

int main()

{

in.open("c:\\car.bmp",ios::in|ios::binary);

out.open("c:\\negcar.bmp",ios::out|ios::binary);

in.read((char\*)(&h),sizeof(h));

out.write((char\*)(&h),sizeof(h));

while(!in.eof())

{ in.read((char\*)(&pixel),sizeof(pixel));

pixel=L-pixel;

out.write((char\*)(&pixel),sizeof(pixel));

}

in.close();

out.close();

return 1;

}

**INPUT**

****

**OUPUT**

****

**//Power Log Transformation**

#include<iostream>

#include<fstream>

#include<math.h>

typedef unsigned char byte;

using namespace std;

int c=5,g;

float gamma;

byte h[54];

byte pixel;

ofstream out;

ifstream in;

int main()

{

cout<<"Enter value of gamma";

cin>>g;

in.open("c:\\Penguins.bmp",ios::in|ios::binary);

out.open("c:\\logp.bmp",ios::out|ios::binary);

in.read((char\*)(&h),sizeof(h));

out.write((char\*)(&h),sizeof(h));

while(!in.eof())

{

in.read((char\*)(&pixel),sizeof(pixel));

pixel=c\*(pixel^g);

out.write((char\*)(&pixel),sizeof(pixel));

}

in.close();

out.close();

}

**INPUT**

****

**OUTPUT**

****

**// Log Transformation**

#include<iostream>

#include<fstream>

#include<string>

#include<math.h>

using namespace std;

struct pix {

unsigned char b,g,r; }pixel;

int L=255;//L is set to highest intensity per byte

char Header[54];

ifstream in;

ofstream out,out1;

int main() {

char infile[]="c:\\car.bmp";

char outfile[]="c:\\carlog.bmp";

char imdata[]="c:\\imdata.dat";

in.open(infile,ios::in|ios::binary);

in.read((char\*)(&Header),sizeof(Header));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&Header),sizeof(Header));

out1.open(imdata,ios::out);

while(!in.eof()) {

in.read((char\*) (&pixel),sizeof(pixel));

out1<<"ORIGINAL : "<<(int)pixel.r<<" , "<<(int)pixel.g<<" , "<<(int)pixel.b<<endl;

pixel.r=L\*log(pixel.r+1);

pixel.g=L\*log(pixel.g+1);

pixel.b=L\*log(pixel.b+1);

out.write((char\*) (&pixel),sizeof(pixel));

out1<<"MODIFIED : "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b<<endl;}

in.close();

out.close();

out1.close();

}

**INPUT**

****

**OUPUT**

****

|  |
| --- |
| **Implementation of Piecewise-Linear Transformation Functions** |

**// Contract Stretching**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

int ut=150, lt=80; // upper and lower threshold values

char Header[54];

ifstream in;

ofstream out,out1;

int main()

{

char infile[]="car.bmp";

char outfile[]="car-contraststreching.bmp";

char imdata[]="imdata.dat";

in.open(infile,ios::in|ios::binary);

in.read(( char\*)(&Header),sizeof(Header));

out.open(outfile,ios::out| ios::binary);

out.write(( char\*)(&Header),sizeof(Header));

out1.open(imdata, ios::out);

while(!in.eof())

{

in.read((char \*)(&pixel),sizeof(pixel));

out1<<"ORIGINAL : " <<(int)pixel.r<<" , "<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

int intensity= (pixel.r+pixel.g+pixel.b)/3; //gray scale

//Image contrast streching

if (intensity <= lt)

pixel.r=pixel.g=pixel.b = 0.5\*intensity;

else if( intensity <=ut)

pixel.r=pixel.g=pixel.b = 2\*(intensity-lt)+0.5\*lt;

else

pixel.r=pixel.g=pixel.b = 0.5\*(intensity-ut)+0.5\*lt + 2\*(ut-lt);

out.write((char \*)(&pixel),sizeof(pixel));

out1<<"MODIFIFED (Contrast Streching) : " <<(int)pixel.r<<" ,"<<(int)pixel.g<<" , "<<(int)pixel.b<<endl;

}

in.close();

out.close();

}

**INPUT**

****

**OUTPUT**

****

**// Grey Level Slicing**

#include<iostream>

#include<fstream>

#include<string>

#include<math.h>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

int L=255; // L is set to higest intensity per byte

char Header[54];

ifstream in;

ofstream out,out1;

int main()

{

char infile[] = "mm.bmp";

char outfile[] = "mm-gray.bmp";

char imdata[]="imdata.dat";

float powfact = 0;

in.open(infile,ios::in|ios::binary);

in.read((char\*)(&Header),sizeof(Header));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&Header),sizeof(Header));

out1.open(imdata,ios::out);

cout<<"Enter power amount :-";

cin>>powfact;

while(!in.eof())

{

in.read((char\*) (&pixel),sizeof(pixel));

out1<<"ORIGINAL : "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b<<endl;

//original to grayscale

pixel.r=pixel.r/3;

pixel.g=pixel.g/3;

pixel.b=pixel.b/3;

out.write((char\*) (&pixel),sizeof(pixel));

out1<<"MODIFIED : "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b<<endl;

}

in.close();

}

**INPUT**

****

**OUTPUT**

****

**// Bit Plane Slicing**

#include<iostream>

#include<fstream>

#include<string>

#include<math.h>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

int L=255;//L is set to highest intensity per byte

//char Header[54];

int bi;

char Header[54],tmp;

ifstream in;

ofstream out,out1;

int main()

{

char infile[]="c:\\car.bmp";

char outfile[]="c:\\bitplanecar.bmp";

char imdata[]="imdata.dat";

in.open(infile,ios::in|ios::binary);

in.read((char\*)(&Header),sizeof(Header));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&Header),sizeof(Header));

out1.open(imdata,ios::out);

do

{

cout<<"Which bit image you want to extract 0=LSB 7=MSB : ";

cin>>bi;

}while(bi<0||bi>7);

while(!in.eof())

{

in.read((char\*)(&pixel),sizeof(pixel));

cout<<"\n ORIGINAL "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b;

out1<<"ORIGINAL : "<<(int)pixel.r<<" , "<<(int)pixel.g<<" , "<<(int)pixel.b<<endl;

tmp=(pixel.b+pixel.g+pixel.r)/3;

//Extact specified bit plane

pixel.b=pixel.g=pixel.r=tmp&(int)pow(2.0,bi);

if(pixel.b!=0)

pixel.b=pixel.g=pixel.r=255;

else

pixel.b=pixel.g=pixel.r=0;

cout<<"\n MODIFIED : "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b<<endl;

out.write((char\*) (&pixel),sizeof(pixel));

out1<<"MODIFIED : "<<(int)pixel.r<<" "<<(int)pixel.g<<" "<<(int)pixel.b<<endl;

}

in.close();

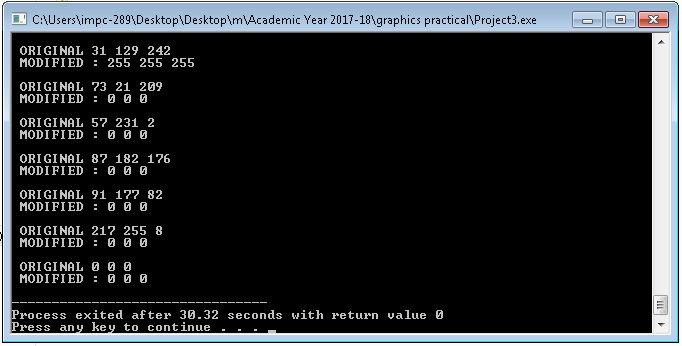
out.close();

out1.close();

}

**OUTPUT**

**Enter value between 0 to 7**



**Input Image**

|  |
| --- |
|  |



**Output Image**

****

|  |
| --- |
|  |

**Image Enhancement Techniques –**

|  |
| --- |
|  |

**// Image Histogram and Histogram Equalization**

**#include<iostream>**

**#include<fstream>**

**#include<string>**

**#include<graphics.h>**

**using namespace std;**

**struct pix**

**{**

**unsigned char b,g,r;**

**}pixel;**

**char Header[54];**

**int hist[256]={0};**

**int rmin=30, rmax=200;**

**int s,smin=0,smax=255;**

**ifstream in;**

**ofstream out,out1;**

**int main()**

**{**

**char infile[]="mm.bmp";**

**char outfile[]="mm-histogram-linear.bmp";**

**char imdata[]="imdata.dat";**

**int intensity;**

**//int gd=DETECT, gm;**

**//initgraph(&gd,&gm,"C:\Program Files\Dev-Cpp\MinGW64\lib\libbgi.a");**

**line(10,10,10,450);**

**line(10,450, 600,450);**

**in.open(infile,ios::in|ios::binary);**

**in.read(( char\*)(&Header),sizeof(Header));**

**out.open(outfile,ios::out| ios::binary);**

**out.write(( char\*)(&Header),sizeof(Header));**

**out1.open(imdata, ios::out);**

**while(!in.eof())**

**{**

**in.read((char \*)(&pixel),sizeof(pixel));**

**//Image 24-bit to gray scale conversion**

**intensity=(pixel.r+pixel.g+pixel.b)/3;**

**pixel.r=pixel.g=pixel.b=intensity;**

**out1<<"ORIGINAL : " <<(int)pixel.r<<" , "<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;**

**for (int k=0;k<255;k++)**

**if (intensity == k)**

**hist[k]+= 1;**

**int s = (smax-smin)/(float)(rmax-rmin) \*(intensity-rmin)+smin;**

**if(s <= rmin)**

**pixel.r=pixel.g=pixel.b=0;**

**else**

**pixel.r=pixel.g=pixel.b=s;**

**out.write((char \*)(&pixel),sizeof(pixel));**

**out1<<"MODIFIFED : " <<(int)pixel.r<<" , "<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;**

**}**

**for(int k=0;k<255;k++)**

**line(20+k,450,20+k,450-hist[k]/2);**

**in.close();**

**out.close();**

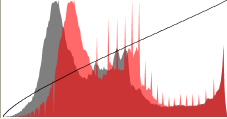
**out1.close();**

**}**

**INPUT**



**OUTPUT**



**// Image Subtraction**

|  |
| --- |
|  |

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

int L=255;

char h[54];

ifstream in,in1;

ofstream out,out1;

int main()

{

char infile1[]="c:\\car.bmp";

char infile2[]="c:\\car1.bmp";

char outfile[]="c:\\carsub.bmp";

char imdata[]="c:\\imdata.dat";

in.open(infile1,ios::in|ios::binary);

in1.open(infile2,ios::in|ios::binary);

in1.read((char\*)(&h),sizeof(h));

in.read((char\*)(&h),sizeof(h));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&h),sizeof(h));

out1.open(imdata,ios::out);

while(!in.eof())

{

pix a1,a2;

//Reading first image

in.read((char\*)(&pixel),sizeof(pixel));

a1.r=pixel.r;

a1.g=pixel.g;

a1.b=pixel.b;

out1<<"Original 1:"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

//Reading second image

in1.read((char\*)(&pixel),sizeof(pixel));

a2.r=pixel.r;

a2.g=pixel.g;

a2.b=pixel.b;

out1<<"Original 2:"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

pixel.r=a1.r-a2.r;

pixel.g=a1.g-a2.g;

pixel.b=a1.b-a2.b;

out.write((char\*)(&pixel),sizeof(pixel));

out1<<"Modified(subtracted):"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

}

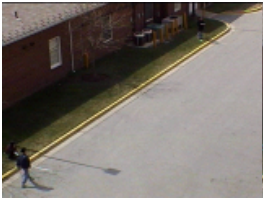
in.close();

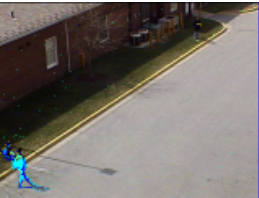
out.close();

return 0;

}

**INPUT**

** Image1**

** Image 2**

**Output**

** Image1-Image2**

**//Image averaging**

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

struct pix

{

unsigned char b,g,r;

}pixel;

int L=255;

char h[54];

ifstream in,in1;

ofstream out,out1;

int main()

{

char infile1[]="c:\\car.bmp";

char infile2[]="c:\\car1.bmp";

char outfile[]="c:\\carsub.bmp";

char imdata[]="c:\\imdata.dat";

in.open(infile1,ios::in|ios::binary);

in1.open(infile2,ios::in|ios::binary);

in1.read((char\*)(&h),sizeof(h));

in.read((char\*)(&h),sizeof(h));

out.open(outfile,ios::out|ios::binary);

out.write((char\*)(&h),sizeof(h));

out1.open(imdata,ios::out);

while(!in.eof())

{

pix a1,a2;

//Reading first image

in.read((char\*)(&pixel),sizeof(pixel));

a1.r=pixel.r;

a1.g=pixel.g;

a1.b=pixel.b;

out1<<"Original 1:"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

//Reading second image

in1.read((char\*)(&pixel),sizeof(pixel));

a2.r=pixel.r;

a2.g=pixel.g;

a2.b=pixel.b;

out1<<"Original 2:"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

pixel.r=(float)(a1.r+a2.r)/2.0;

pixel.g=(float)(a1.g-a2.g)/2.0;

pixel.b=(float)(a1.b-a2.b)/2.0;

out.write((char\*)(&pixel),sizeof(pixel));

out1<<"Modified(subtracted):"<<(int)pixel.r<<" ,"<<(int)pixel.g<<" ,"<<(int)pixel.b<<endl;

}

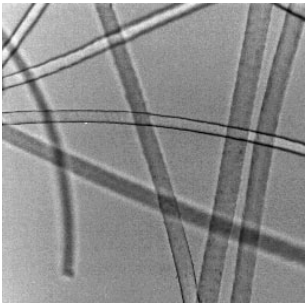
in.close();

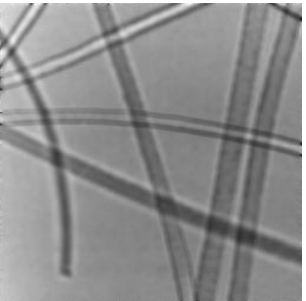
out.close();

return 0;

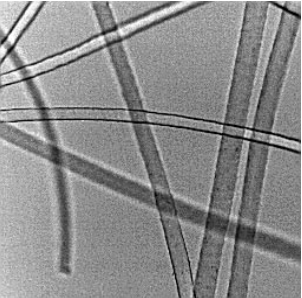
}

**Input**

** Image 1**

** Image 2**

**Output**

****