**Practical No 12.B**

**Implementation of Basic Intensity Transformation- Log transformation.**

**Aim: Write a program to implement a Basic Intensity Transformation using Log transformation algorithm.**

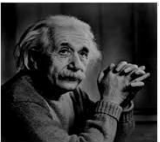
**Theory:**

The log transformations can be defined by this formula

s = c log (r + 1).

Where s and r are the pixel values of the output and the input image and c are a constant. The value 1 is added to each of the pixel value of the input image because if there is a pixel intensity of 0 in the image, then log (0) is equal to infinity. So, 1 is added, to make the minimum value at least 1. During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation. This result in following image enhancement. The value of c in the log transform adjusts the kind of enhancement you are looking for.

**Input Image**



**Log Transform Image**



The inverse log transform is opposite to log transform.

**Conclusion: We have implemented Basic Intensity Transformation using Log transformation algorithm.**

**Code:**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<string.h>

#include<math.h>

#include<graphics.h>

struct pix

{

unsigned char b,g,r;

}pixel; class logtr

{

int L;

char Header[54];

int c;

ifstream in;

ofstream out,out1;

public:

logtr()

{

L=255;

}

void process()

{

char infile[]="cat.bmp";

char outfile[]="output1.bmp";

char imdata[]="logdata.rtf";

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);

c=L/log(1+L); while(!in.eof())

{

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

out1<<"ORIGINAL : " <<(int)pixel.r<<" ,"<<(int)pixel.g<<" , "<<(int)pixel.b<<endl; pixel.r=c\*log(1+pixel.r);

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

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

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

}

in.close();

out.close();

}

};

void main(){

clrscr();

int gd=DETECT,gm; initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

setcolor(BLUE);

logtr l; l.process();

cout<<"Log Transformation Done Successfully";

getch();

}

**Output:**

****

Before Transformation After Transformation