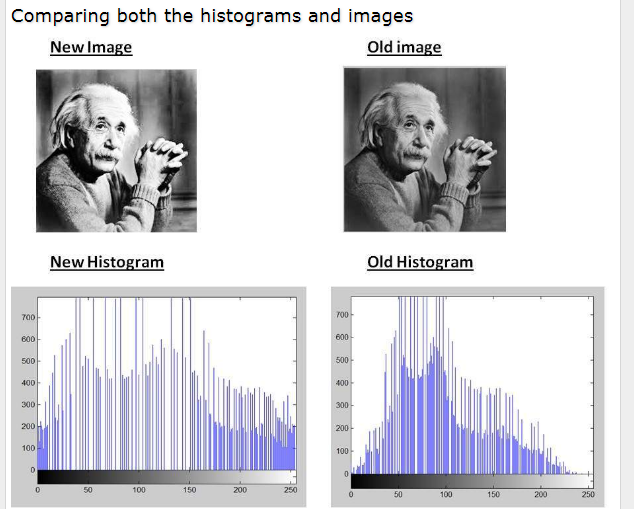
**Practical No 14.A**

**Implementation of histogram equalization - Image histogram & histogram Equalization.**

**Aim: Write a program to implement a histogram equalization using Image histogram & histogram Equalization algorithm.**

**Theory:**

Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered. Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e., stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast. A color histogram of an image represents the number of pixels in each type of color component. Histogram equalization cannot be applied separately to the Red, Green and Blue components of the image as it leads to dramatic changes in the image’s color balance. However, if the image is first converted to another color space, like HSL/HSV color space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image.



**Conclusion: We have implemented histogram equalization using Image histogram & histogram Equalization algorithm.**

**Code:**

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

#include<string.h>

const int size=256;

struct pix {

unsigned char b,g,r;

} p;

class histogram {

int np[size],roff[size],mod[size];

float pdf[size],cdf[size],calc[size],sum;

unsigned char Header[54],tmp;

ifstream in;

public:

void process() {

int i;

char infile[]="C:\\TURBOC3\\BIN\\cat.bmp";

clrscr();

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

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

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

np[i]=0;

mod[i]=0; }

while(!in.eof()) {

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

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

np[tmp]++;

sum++; }

float temp=0;

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

pdf[i]=(float)np[i]/sum;

temp=temp+pdf[i];

cdf[i]=temp;

calc[i]=cdf[i]\*(size-1);

roff[i]=(int)(calc[i]+0.5);

}

int index;

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

{

index=roff[i] ;

mod[index]=mod[index]+np[i];

}

drawhisto();

in.close(); }

void drawhisto()

{

int i=0;

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

{

setcolor(BLUE);

line(10+i,450,10+i,450-np[i]); }

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

setcolor(BLUE);

line(325+i,450,325+i,450-mod[i]);

}

getch();

closegraph();

}

};

void main()

{

int gdriver = DETECT, gmode, errorcode,i;

initgraph(&gdriver, &gmode, "..\\BGI");

setbkcolor(WHITE);

histogram h;

h.process();

closegraph();

}

**Input:**



**Output:**

