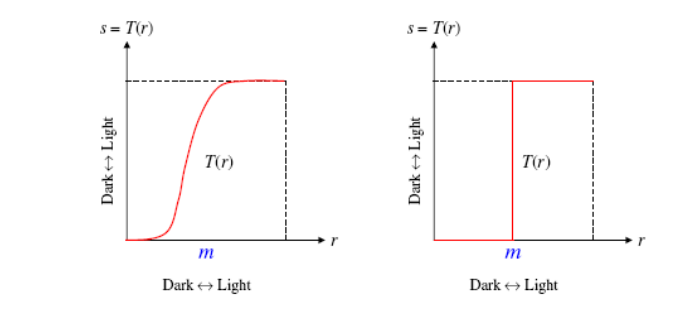
**Practical No 13.A**

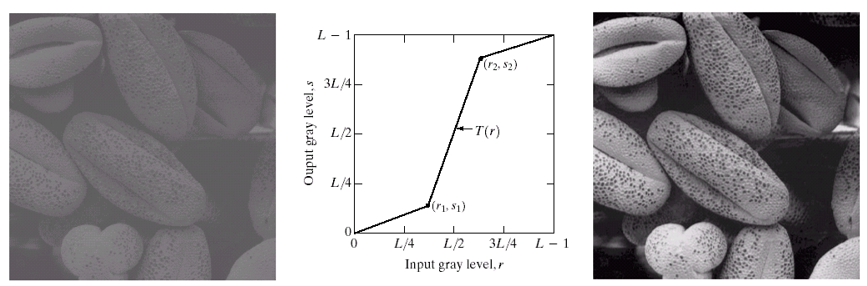
**Implementation of Piecewise-Linear Transformation Functions - Contrast Stretching.**

**Aim: Write a program to implement a Piecewise-Linear Transformation Functions using Contrast Stretching algorithm.**

**Theory:**

One of the simplest piecewise linear functions is a contrast-stretching transformation. Low-contrast images can result from imaging poor illumination, lack of dynamic range in the sensor, or even wrong setting of a lens aperture during image acquisition. The idea Behind contrast stretching is to increase the dynamic range of the levels in the image being processed. Its produces higher contrast than the original by darkening the levels below m in the original image, brightening the levels above m in the original and thresholding: produce a binary image.





**Conclusion: We have implemented Piecewise-Linear Transformation Functions using Contrast Stretching algorithm.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<fstream.h>

#include<string.h>

#include<math.h>

struct pix

{

unsigned char b,g,r;

}pixel;

int L=255;

char Header[54];

ifstream in;

ofstream out,out1;

void main()

{

int gd=DETECT,gm;

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

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

char outfile[]="C:\\TURBOC3\\BIN\\con\_cat.bmp";

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

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.write(imdata,ios::out);

int r1,r2,s1,s2;

cout<<"\n Enter coordinates of r1 & s1: ";

cin>>r1>>s1;

cout<<"\n Enter coordinates of r2 & s2: ";

cin>>r2>>s2;

int v,w,a,b,l,m,n; v=s1;

w=s2; a=r1; b=r2;

l=(float)(s1/r1);

m=(float)(s2-s1)/(r2-r1);

n=(float)((255-s2)/(255-r2));

while(!in.eof()) {

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

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

if(pixel.r<a && pixel.g<a && pixel.b<a) {

pixel.r=1\*pixel.r;

pixel.g=1\*pixel.g;

pixel.b=1\*pixel.b;

}

if(pixel.r<b && pixel.g<b && pixel.b<b)

{

pixel.r=m\*(pixel.r-a)+v;

pixel.g=m\*(pixel.g-a)+v;

pixel.b=m\*(pixel.b-a)+v;

}

else

{

pixel.r=n\*(pixel.r-b)+w;

pixel.g=n\*(pixel.g-b)+w;

pixel.b=n\*(pixel.b-b)+w;

}

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

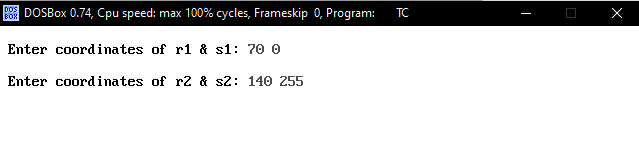
out1<<"mod"<<(int)pixel.r<<","<<(int)pixel.g<<","<<(int)pixel.b<<endl;

}

getch();

}

**Output:**



Original Image New Image