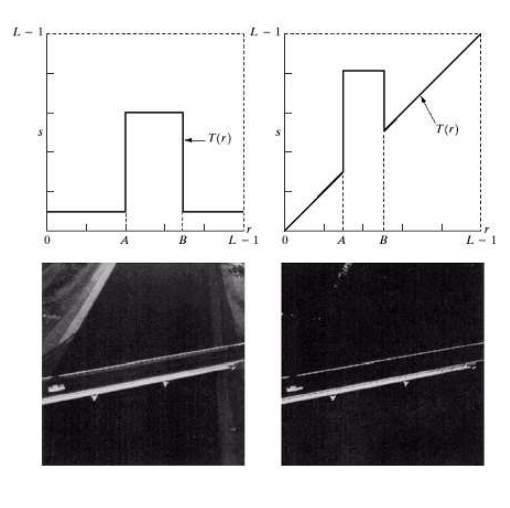
**Practical No 13.B**

**Implementation of Piecewise-Linear Transformation Functions - Grey level Slicing.**

**Aim: Write a program to implement a Piecewise-Linear Transformation Functions using Grey level Slicing algorithm.**

**Theory:**

A specific range of grey levels in an image often is desired. Applications include enhancing features such as masses to display a high value for all grey levels in the range of interest and a low value for all other grey levels. This transformation, produces a binary image. The second approach, based on the transformation brightens the desired range of grey levels but preserves the background and grey- tonlevelalities in the image. Its highlighting a specific range of grey levels in an image by display high value for grey levels in the range of interest, and low value for all other grey levels. The left image highlights range [A, B] and reduces all others to a constant level. The right image highlights range [A, B] but preserves all other levels.



**Conclusion: We have implemented Piecewise-Linear Transformation Functions using Grey level Slicing 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\\greyssb\_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 a,b;

cout<<"Enter the values of a and b: "; cin>>a>>b;

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<b&& pixel.g<b&& pixel.b<a))

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

else

{

pixel.r=pixel.r;

pixel.b=pixel.b;

pixel.g=pixel.g;

}

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

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

}

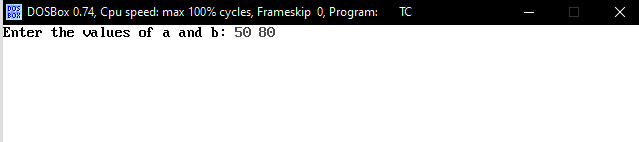
in.close();

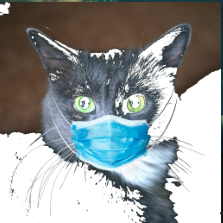
out.close();

getch();

}

**Output:**



Original Image New Image