**Problem Statement :**

**Perform Distance Transform on the below-mentioned image.**

****

**Convert the image to binary and perform the distance transform. Share the distance matrix with neat documentation. From Scratch implementation with C++ is expected.**

**Convert the image to binary**

**We are trying to convert an image into binary data and vice-versa for a project using C programming. All the other solutions we found on the net are either in C++ or Java. Here is the approach we tried:**

**Convert the image into a text file containing binary data. Each 8 characters corresponds to the character byte when the image is opened using a text editor.**

**Then we try to reconvert the binary data into its respective characters using a C program.**

**Then we open the result using Picasa Photoviewer. We get an invalid image.**

**How do we get back the original image? Here is the code we used to convert the image into a text file:**

**0**

**Down vote**

**Conversion of Image--->binary--->image using C**

**c image image-processing image-conversion**

**We are trying to convert an image into binary data and vice-versa for a project using C programming. All the other solutions we found on the net are either in C++ or Java. Here is the approach we tried:**

**Convert the image into a text file containing binary data. Each 8 characters corresponds to the character byte when the image is opened using a text editor.**

**Then we try to reconvert the binary data into its respective characters using a C program.**

**Then we open the result using Picasa Photoviewer. We get an invalid image.**

**1. How do we get back the original image? Here is the code we used to convert the image into a text file:**

**#include<stdio.h>**

**#include<conio.h>**

**void main()**

**{**

**clrscr();**

**FILE \*fptr;**

**FILE \*txt;**

**int c;**

**fptr=fopen("D:\\aa.bmp","r");**

**txt=fopen("D:\\test1.txt","w");**

**if(fptr==NULL)**

**{**

**printf("NOTHING In FILE");**

**fclose(fptr);**

**}**

**else**

**{**

**printf("success");**

**do**

**{**

**c=fgetc(fptr);**

**for(int i=0;i<=7;i++)**

**{**

**if(c&(1<<(7-i)))**

**{**

**fputc('1',txt);**

**}**

**else**

**{**

**fputc('0',txt);**

**}**

**}**

**// fprintf(txt,"\t");**

**}while(c!=EOF);**

**}**

**fclose(fptr);**

**fclose(txt);**

**printf("writing over");**

**getch();**

**}**

**2. Here is the code to convert the resulting text file to image file full of binary characters, i.e. a text file with only ones and zeroes.**

**#include<stdio.h>**

**#include<conio.h>**

**\\The following function converts the ones and zeroes in the text file into a character.**

**\\For example the text file may have the 8 consecutive characters '1','0','0','0','1','0','0','0'.**

**\\This converts it into the character equivalent of the binary \\value 10001000**

**char bytefromtext(char\* text)**

**{**

**char result=0;**

**for(int i=0;i<8;i++)**

**{**

**if(text[i]=='1')**

**{**

**result |= (1<< (7-i) );**

**}**

**}**

**return result;**

**}**

**void main()**

**{**

**clrscr();**

**FILE \*pfile;**

**FILE \*image;**

**char buf[8];**

**char c;**

**int j=0;**

**image=fopen("D:\\aa2.bmp","w"); //open an empty .bmp file to**

**//write characters from the source image file**

**pfile=fopen("D:\\test1.txt","r");**

**if(pfile==NULL)**

**printf("error");**

**else**

**{**

**c=fgetc(pfile);**

**while(c!=EOF)**

**{**

**buf[j++]=c;**

**if(j==8)**

**{**

**fputc(bytefromtext(buf),image);**

**j=0;**

**}**

**c=fgetc(pfile);**

**}**

**fclose(pfile);**

**fclose(image);**

**}**

**getch();**

**}**

**OUTPUT:**

****

**11111111111111111111111111**

**11111111111110000011111111**

**11111111110000000001111111**

**11111110000000000001111111**

**11111100000001111111111111**

**11111000000001111111111111**

**11110000000001111111111111**

**11110000000001111111111111**

**11100000000011111111111111**

**11100000000011111001111111**

**11100000000001111111111111**

**11110000000001111111111111**

**11110000000000011111001111**

**11110000101001111111101111**

**11111000010000111111111111**

**11111000011000111111001111**

**11111100000000011111111111**

**11111100000000000111111111**

**11111100000000000001111111**

**11111111110000000000001111**

**11111100111110000000011111**

**11111000001111100000111111**

**11111000000111111011111111**

**11110000000001111111111111**

**11110000000000111111111111**

**11100000000000011101111111**

**11000000000000001100111111**

**10000000000000000000011111**

**10000000000000000000011111**

**00000000000000000000001111**

**00000000000000000000000111**

**00000000000000000000000011**

**00000000000000000000000001**

**00000000000000000000000000**

**00000000000000000000000000**

**00000000000000000000000000**

**00000000000000000000000000**

**00000000000000000000000000**

**00000000000000000000000000**

**00000000000000000000000000**

**The distance matrix with neat documentation. From Scratch implementation with C++ is expected.**

**Source code :**

**#include <jpeglib.h>**

**int loadJpg(const char\* Name) {**

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

**int width, height;**

**struct jpeg\_decompress\_struct cinfo;**

**struct jpeg\_error\_mgr jerr;**

**FILE \* infile; /\* source file \*/**

**JSAMPARRAY pJpegBuffer; /\* Output row buffer \*/**

**int row\_stride; /\* physical row width in output buffer \*/**

**if ((infile = fopen(Name, "rb")) == NULL) {**

**fprintf(stderr, "can't open %s\n", Name);**

**return 0;**

**}**

**cinfo.err = jpeg\_std\_error(&jerr);**

**jpeg\_create\_decompress(&cinfo);**

**jpeg\_stdio\_src(&cinfo, infile);**

**(void) jpeg\_read\_header(&cinfo, TRUE);**

**(void) jpeg\_start\_decompress(&cinfo);**

**width = cinfo.output\_width;**

**height = cinfo.output\_height;**

**unsigned char \* pDummy = new unsigned char [width\*height\*4];**

**unsigned char \* pTest = pDummy;**

**if (!pDummy) {**

**printf("NO MEM FOR JPEG CONVERT!\n");**

**return 0;**

**}**

**row\_stride = width \* cinfo.output\_components;**

**pJpegBuffer = (\*cinfo.mem->alloc\_sarray)**

**((j\_common\_ptr) &cinfo, JPOOL\_IMAGE, row\_stride, 1);**

**while (cinfo.output\_scanline < cinfo.output\_height) {**

**(void) jpeg\_read\_scanlines(&cinfo, pJpegBuffer, 1);**

**for (int x = 0; x < width; x++) {**

**a = 0; // alpha value is not supported on jpg**

**r = pJpegBuffer[0][cinfo.output\_components \* x];**

**if (cinfo.output\_components > 2) {**

**g = pJpegBuffer[0][cinfo.output\_components \* x + 1];**

**b = pJpegBuffer[0][cinfo.output\_components \* x + 2];**

**} else {**

**g = r;**

**b = r;**

**}**

**\*(pDummy++) = b;**

**\*(pDummy++) = g;**

**\*(pDummy++) = r;**

**\*(pDummy++) = a;**

**}**

**}**

**fclose(infile);**

**(void) jpeg\_finish\_decompress(&cinfo);**

**jpeg\_destroy\_decompress(&cinfo);**

**BMap = (int\*)pTest;**

**Height = height;**

**Width = width;**

**Depth = 32;**

**}**

**Sample screenshot of coding:**





