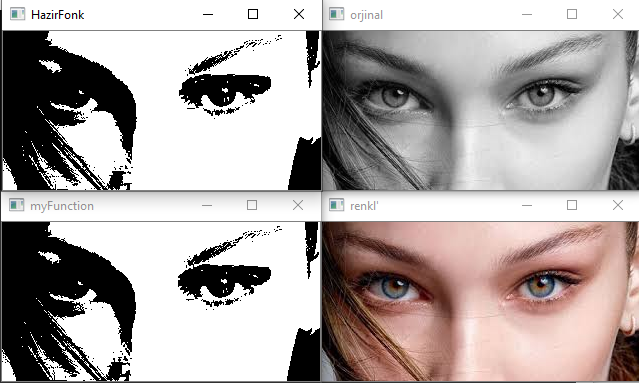
**EE 4064 Introduction to Machine Vision**

**HOMEWORK #3**

1)Write a raw C code under Visual Studio (with OpenCV added to it) to calculate the Otsu􏰀s threshold value for a given grayscale image. Apply the threshold to the image and display your result as output.

Output:



Code:

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

int dizi[255];

void mustafaOtsu(Mat img, int threshValuee,int dizi[]) {

int vMax=0;

for (int i = 0; i < img.cols; i++) {

for (int j = 0; j < img.rows; j++) {

dizi[img.at<uchar>(j, i)]++;

}

}

for (int threshValue = 0; threshValue < 256; threshValue++) {

float ort1 = 0; float ort2 = 0; float weight1 = 0; float weight2 = 0;

for (int i = 0; i <= threshValue; i++) {

ort1 = ort1 + dizi[i] \* i;

ort2 = ort2 + dizi[i];

}

float mean1 = ort1 / ort2;

weight1 = ort2 / img.cols;

weight1 = weight1 / img.rows;

ort1 = 0; ort2 = 0;

for (int i = threshValue + 1; i < 256; i++) {

ort1 = ort1 + dizi[i] \* i;

ort2 = ort2 + dizi[i];

}

float mean2 = ort1 / ort2;

weight2 = ort2 / img.cols;

weight2 = weight2 / img.rows;

cout << "mean1:" << mean1 << " ";

cout << "mean2:" << mean2 << endl;

cout << "weight1:" << weight1 << " ";

cout << "weight2:" << weight2 << endl;

ort1 = 0;

ort2 = 0;

float ort3 = 0;

for (int i = 0; i <= threshValue; i++) {

ort1 = dizi[i] \* pow(abs(i - mean1), 2);

ort2 = ort2 + ort1;

ort3 = ort3 + dizi[i];

}

float sigma1 = ort2 / ort3;

ort1 = 0;

ort2 = 0;

ort3 = 0;

for (int i = threshValue + 1; i < 256; i++) {

ort1 = dizi[i] \* pow(abs(i - mean2), 2);

ort2 = ort2 + ort1;

ort3 = ort3 + dizi[i];

}

float sigma2 = ort2 / ort3;

cout << "sigma1:" << sigma1 << " ";

cout << "sigma2:" << sigma2 << endl;

float Vb = weight1 \* weight2 \* pow(abs(mean1 - mean2), 2);

float Vw = weight1 \* sigma1 + weight2 \* sigma2;

cout << "Vb:" << Vb << endl;

cout << "Vw:" << Vw << endl;

if (Vb > vMax) {

vMax = Vb;

threshValuee = threshValue;

}

}

for (int i = 0; i < img.cols; i++) {

for (int j = 0; j < img.rows; j++) {

if (img.at<uchar>(j, i) <= threshValuee) {

img.at<uchar>(j, i) = 0;

}

else

img.at<uchar>(j, i) = 255;

}

}

imshow("myFunction", img);

}

Mat img3;

int vb = 0;

int main(){

Mat img = imread("eyes.jpg", 1);

imshow("renkl'", img);

cvtColor(img, img, COLOR\_BGR2GRAY);

imshow("orjinal", img);

Mat img2(img.cols,img.rows,CV\_8UC1,Scalar(0));

//imshow("beforethresh", img);

threshold(img, img3, 200, 255, CV\_THRESH\_OTSU);

mustafaOtsu(img, 200, dizi);

imshow("HazirFonk", img3);

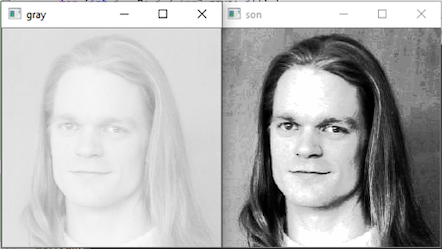
waitKey(0);

return 0;

}

**2)**Write a raw C code under Visual Studio (with OpenCV added to it) to implement the histogram equalization method for a given grayscale image. Apply histogram equalization to any image you pick and provide the result.

Solution:



Code:

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

double pdf[255];

double cdfL[255];

double pmf[255];

double cdf[255];

Mat img2;

float n;

double sayi = 0;

Mat img3;

int d = 0;

void equ(Mat img2, float sayi) {

int n = img2.rows \* img2.cols;

int min;

int max;

/////////////pdf

for (int i = 0; i < img2.cols; i++) {

for (int j = 0; j < img2.rows; j++) {

pdf[img2.at<uchar>(j, i)]++;

}

}

/////////////Prob mass

for (int i = 0; i < 256; i++) {

if (pdf[i] > 0) {

pmf[i] = pdf[i] / n;

}

else pmf[i] = 0;

}

///////////cumulatife

for (int i = 0; i < 256; i++) {

if (i == 0)cdf[0] = 0;

else {

cdf[i] = cdf[(i - 1)] + pmf[i];

}

}

for (int i = 0; i < 256; i++) {

if (cdf[i] > 0 && d == 0) {

min = i;

d = 1;

}

if (pdf[i] > 0) {

max = i;

}

}

int o = max - min;

cout << "min-max:" << o << endl;

///////// cdf \*(level -1)

for (int j = 0; j < 256; j++) {

if (j > min && j < max) {

cdfL[j] = floor(cdf[j] \* 254);

}

}

////////map

for (int i = 0; i < img2.cols; i++) {

for (int j = 0; j < img2.rows; j++) {

for (int k = 1; k < 256; k++) {

if (img2.at<uchar>(j, i) == k)

img2.at<uchar>(j, i) = cdfL[k];

}

}

}

imshow("son", img2);

}

int main(){

Mat img = imread("low.png", 1);

cvtColor(img, img2, COLOR\_BGR2GRAY);

imshow("gray", img2);

equ(img2,sayi);

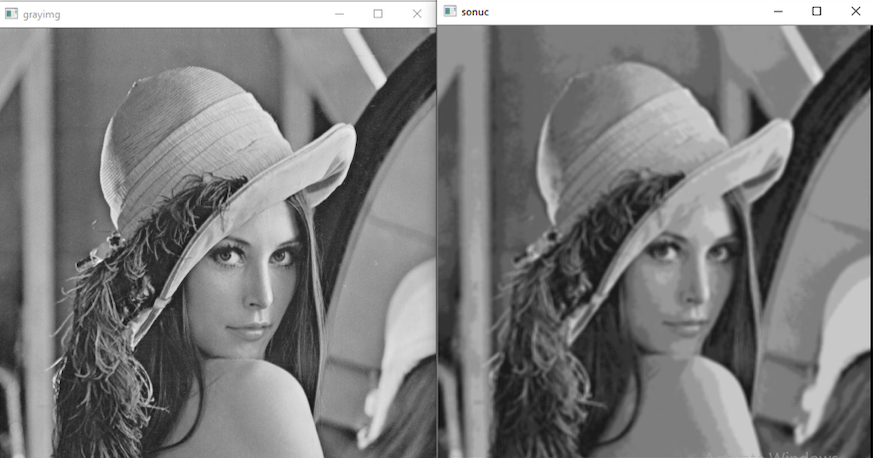
waitKey(0);

return 0;

}

**3)**Write a raw C code under Visual Studio (with OpenCV added to it) to implement the convolution operation for 2D grayscale images. Pick an image of your choice and convolve it with a 5x5 matrix consisting of all 1/25 values.

Solution:



code:

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

Mat gray;

Mat img2;

float filter[5][5] = {

{1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0},

{1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0},

{1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0},

{1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0},

{1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0,1.0 / 25.0}

};

int boyut = 5;

double toplam = 0;

int main(){

Mat img = imread("lena.png", 1);

cvtColor(img, gray, COLOR\_BGR2GRAY);

imshow("grayimg", gray);

Mat Grayresim(gray.cols, gray.cols, CV\_8UC1, Scalar(0));

for (int y = 0; y < img.rows -boyut; y++) {

for (int x = 0 ; x < img.cols -boyut; x++){

for (int t = 0; t < boyut; t++) {

for (int f = 0; f < boyut; f++) {

toplam = toplam + gray.at<uchar>(y + t, x + f)\* filter[f][t];

}

}

Grayresim.at<uchar>(y, x) = toplam;

toplam = 0;

}

}

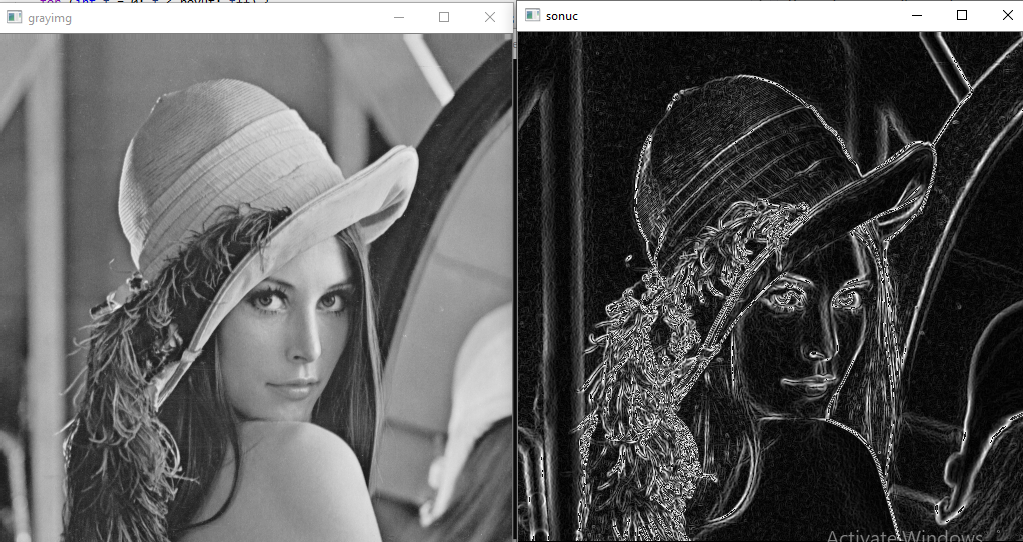
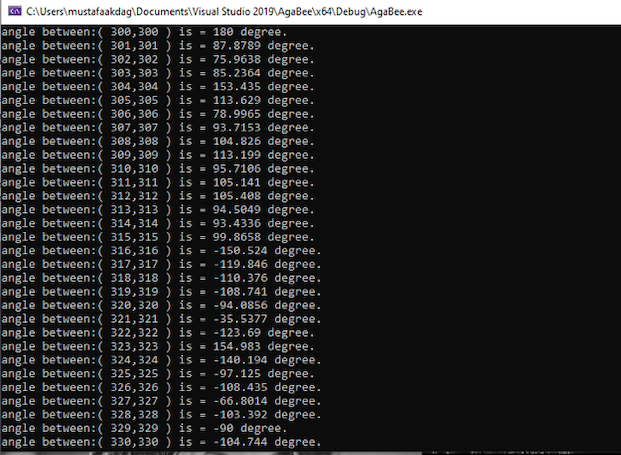
imshow("sonuc", Grayresim);

waitKey(0);

return 0;

}

**4)**Use the generated C code in Question 3 to calculate the gradient magnitude and angle values via Sobel filters.



Code:

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui\_c.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace cv;

using namespace std;

#define PI 3.14159265

double maxi =0;

double pdf[256];

Mat gray;

Mat img2;

float angle;

int filter[3][3] = {

{-1,0,1},

{-2,0,2},

{-1,0,1}

};

int filter2[3][3] = {

{-1,-2,-1},

{0,0,0},

{1,2,1}

};

double magX=0;

double magY=1;

int boyut = 3;

double toplam = 0;

double toplam1 = 0;

double toplam2 = 0;

void functionn(Mat Grayresim,Mat img) {

for (int y = 0; y < img.rows - boyut; y++) {

for (int x = 0; x < img.cols - boyut; x++) {

for (int t = 0; t < boyut; t++) {

for (int f = 0; f < boyut; f++) {

toplam1 = toplam1 + gray.at<uchar>(y + t, x + f) \* filter[f][t];

toplam2 = toplam2 + gray.at<uchar>(y + t, x + f) \* filter2[f][t];

}

}

magX = sqrt(toplam1 \* toplam1 + toplam2 \* toplam2);

if (x == y) {

angle = atan2(toplam2, toplam1) \* 180 / PI;

cout << "angle between:( " << y << "," << x << " ) is = " << angle << " degree." << endl;

}

Grayresim.at<uchar>(y, x) = magX;

toplam1 = 0;

toplam2 = 0;

}

}

for (int i = 0; i < img2.cols; i++) {

for (int j = 0; j < img2.rows; j++) {

pdf[Grayresim.at<uchar>(j, i)]++;

}

}

for (int i = 1; i < 256; i++) {

if (pdf[i] > pdf[i - 1]) {

maxi = i;

}

}

if (magX > maxi)magX = 255;

if (magX <= maxi)magX = 0;

}

int main(){

Mat img = imread("lena.png", 1);

cvtColor(img, gray, COLOR\_BGR2GRAY);

imshow("grayimg", gray);

Mat Grayresim(gray.cols, gray.cols, CV\_8UC1, Scalar(0));

functionn(Grayresim, img);

imshow("sonuc", Grayresim);

waitKey(0);

return 0;

}