KOCAELİ ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

Görüntü İşleme Final Sınavı

171307039 Merve KARAGÖZ

KOCAELİ 2021

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

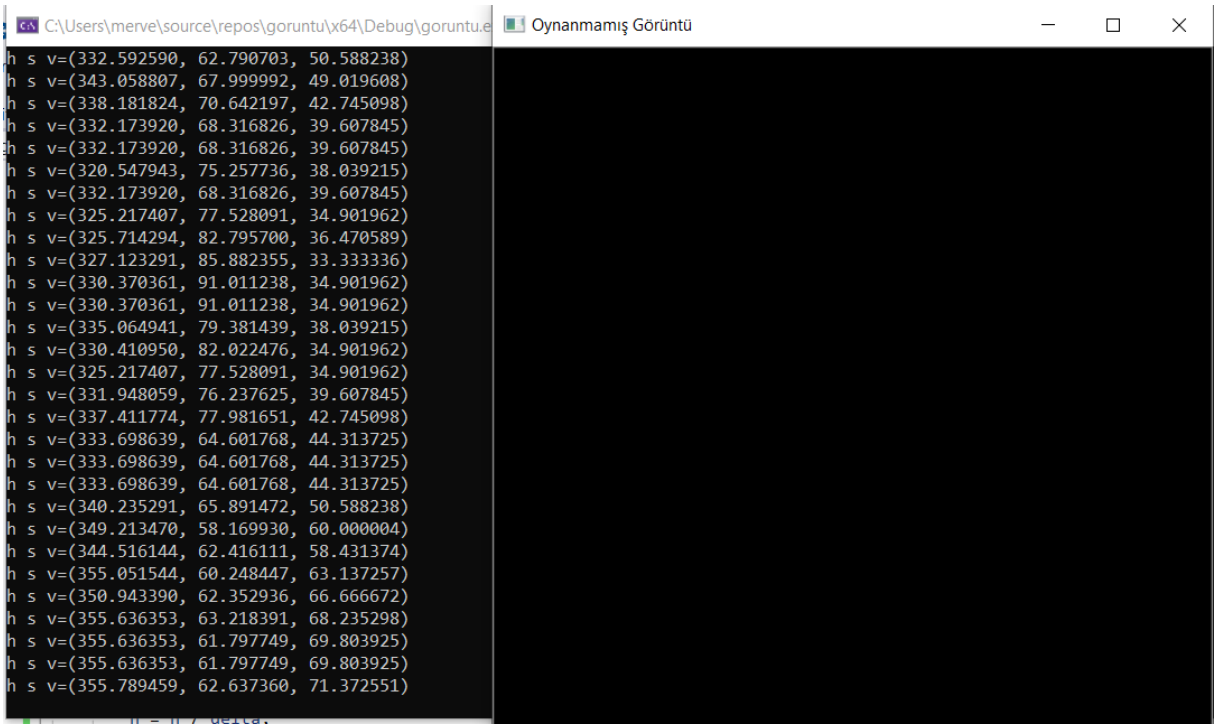
1. KULLANILAN ARAÇLAR

Çözümlenmesi istenilen sorular için kullanılacak araçlar da Microsoft Visual Studio 2019 Community tercih edilmiştir. Çözümlemeler için kullanılan programlama dili C++ dır. Bir konsol uygulamasıyla sorular gerekli çözümlerine ulaştırılmaya çalışılmıştır.

Soru çözümlerinde kullanılmak üzere OpenCv kütüphanesi projeye yüklenmiştir. Bu yükleme için çeşitli kaynaklar incelenmiş ve soru çözümlemelerine başlanmıştır [1].

2. SORU-1

1. sorunun çözümlendirilmesi amacıyla çeşitli araştırmalar yapılmıştır. Öncelikle RGB ve HSV kavramları araştırılmıştır. Soru çözümünde bir sonucu ulaşılamamıştır. Gerekli dönüşüm değerlerinin yapıldığı düşünülse de hatalarla karşılaşılmış ve çözümlendirilememiştir. Şekil 1 'de yer alan program çıktısında okutulan resmin tüm pixel değerleri için RGB değerlerinin HSV değerlerine dönüştürülmüş hali yer almaktadır.



Şekil 1. Soru-1 Ekran Görüntüsü

Şekil 1 'de yer alan ekran görüntüsünü oluşturan kaynak kodlar Şekil 2'de yer almaktadır.

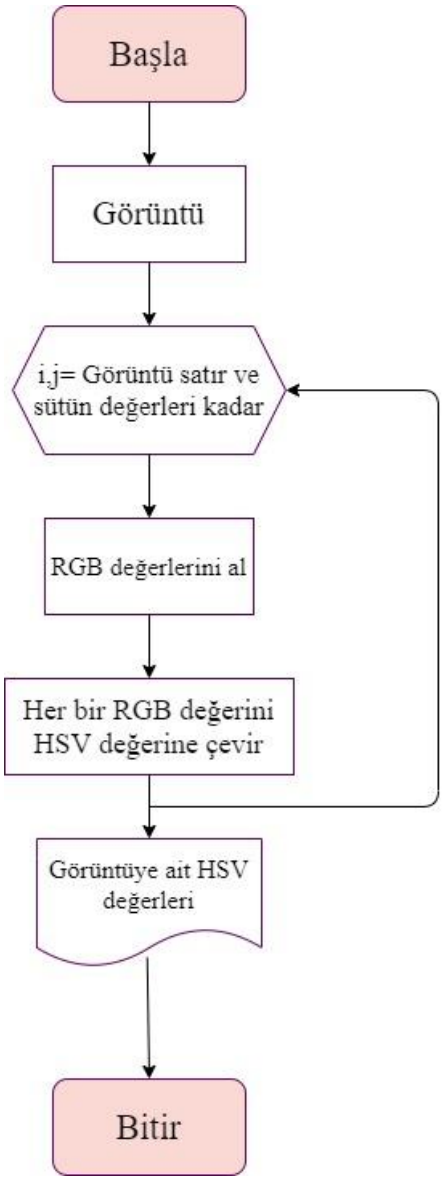
```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/highgui/highgui_c.h>
#include <opencv2/imgproc.hpp>
#include <iostream>
using namespace cv;
using namespace std;

void rgbtohsv(float r,float g,float b) {
    float Cmax, Cmin, delta,h,s,v;

    r /= 255.0;
    g /= 255.0;
    b /= 255.0;
    Cmax = max(r, g, b);
```

```
    Cmin = min(r, g, b);
    delta = Cmax - Cmin;
    if (delta == 0) {
        h = 0;
    }
    else if (Cmax == r) {
        h = g - b;
        h = h / delta;
        h = 60 * h;
        h = fmod(h, 360);
    }
    else if (Cmax == g) {
        h = b - r;
        h = h / delta;
        h = 60 * h;
        h = h + 2;
        h = fmod(h, 360);
    }
    else if (Cmax == b) {
        h = r - g;
        h = h / delta;
        h = 60 * h;
        h = h + 4;
        h = fmod(h, 360);

    }
    if (Cmax == 0) {
        s = 0;

    }
    else {
        s = delta / Cmax;

    }

}
int main(int argc, char const* argv[])
{
    Mat image;
    float wdh, hgh;
    int r;
    int g;
    int b;
    int h, s, v;

        Mat img2 = Mat::zeros(image.rows, image.cols, CV_8UC3);
        for (int i = 0; i < image.rows; i++)
        {
            for (int j = 0; j < image.cols; j++)
            {
                Mat3b image =
imread("C:\\Users\\merve\\OneDrive\\Masaüstü\\goruntuisleme\\Lena.png");
                Vec3b imgcolor = image.at<Vec3b>(Point(i, j));
                 b = imgcolor[0];
                g = imgcolor[1];
                 r = imgcolor[2];
               rgbtohsv(r, g, b);
                Vec3b imgcolor2 = img2.at<Vec3b>(Point(i, j));
                imgcolor2.val[h] = imgcolor.val[r];
                imgcolor2.val[s] = imgcolor.val[g];
                imgcolor2.val[v] = imgcolor.val[b];
            }
```

```
        }

        imshow("a", img2);
        waitKey(0);
        return 0;
}
```

Şekil 2. Soru-1 Kaynak Kodu

Şekil 2 ‘de yer alan kaynak kodlara ait akış şeması Şekil 3 ’de yer almaktadır.



Şekil 3. Soru-1 Akış Şeması

3. SORU-2

2. sorunun çözümlendirilmesi amacıyla çeşitli araştırmalar yapılarak çeşitli kaynaklardan yararlanılmıştır [2]. Öncelikle bir ölçek kısmı yapılmıştır. Bu ölçek kısmından seçilen değerler fotoğraf boyutunda değişim yapabilmek amacıyla yazılmış olan fonksiyona gönderilmiştir. Bu fonksiyonda resmin orijinal boyutlarının RGB değerlerinde ölçek katsayısı kadarınca değişimler yapılarak oluşturulan daha büyük ölçekli resmin görüntülemesi sağlanmıştır. Şekil 4' de görüntünün orijinal hali yer almaktadır. Şekil 5 ve Şekil 6 'da ise görüntünün ölçek değerleri arttırılarak dokümana eklenmiştir. Ölçek değeri 10 olarak ayarlanmıştır.

Şekil 4. Soru-2 Ekran Görüntüsü

Şekil 5. Soru-2 Ekran Görüntüsü

Şekil 6. Soru-2 Ekran Görüntüsü

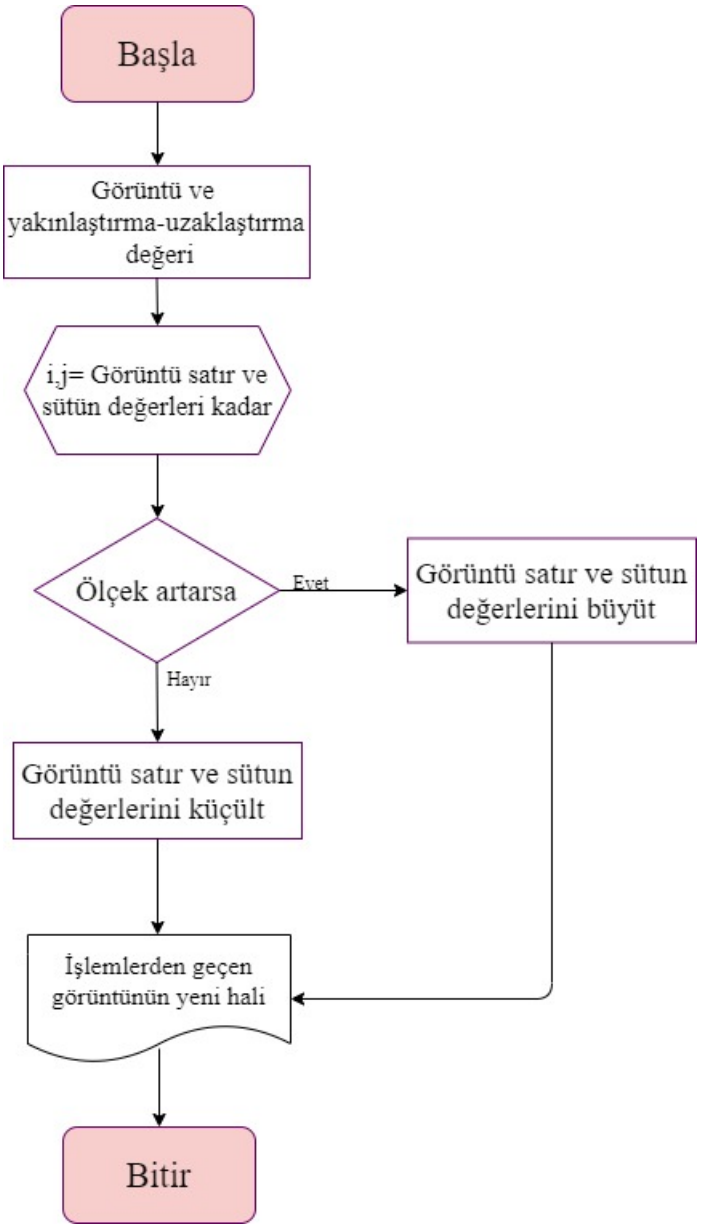
Yukarıda gösterilen ekran görüntülerinin kaynak kodlar Şekil 7'de yer almaktadır.

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include "opencv2/imgproc.hpp"
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>

using namespace cv;
using namespace std;

Mat img2, img;
int olcek_deger = 1;
const int zoom_in = 10;

void zoom_trackbar(int, void*)
{
    img2 = Mat::zeros(img.rows * olcek_deger, img.cols * olcek_deger, CV_8UC3);
    for (int i = 0; i < img.cols; i++)
    {
        for (int j = 0; j < img.rows; j++)
        {

            Vec3b imgcolor = img.at<Vec3b>(Point(i, j));
            for (int p = 0;p < olcek_deger;p++)
            {
                for (int q = 0;q < olcek_deger;q++)
                {
                    Vec3b imgcolor2 = img2.at<Vec3b>(Point(i, j));
                    imgcolor2.val[0] = imgcolor.val[0];
```

```
                    imgcolor2.val[1] = imgcolor.val[1];
                    imgcolor2.val[2] = imgcolor.val[2];

                    img2.at<Vec3b>(Point(i * olcek_deger + p, j * olcek_deger + q))
= imgcolor2;

                }
            }
        }
    }
    imshow("Zoom in-Zoom out", img2);
}
int main()
{
    img = imread("C:\\Users\\merve\\OneDrive\\Masaüstü\\goruntuisleme\\soru2.png");
    namedWindow("Zoom in-Zoom out", CV_WINDOW_AUTOSIZE);
    createTrackbar("Ölcek", "Zoom in-Zoom out", &olcek_deger, zoom_in,
zoom_trackbar);
    zoom_trackbar(olcek_deger, 0);
    namedWindow("Oynanmamış Görüntü", CV_WINDOW_AUTOSIZE);
    imshow("Oynanmamış Görüntü", img);
    waitKey(0);
    return 0;
}
```

Şekil 7. Soru-2 Kaynak Kodu

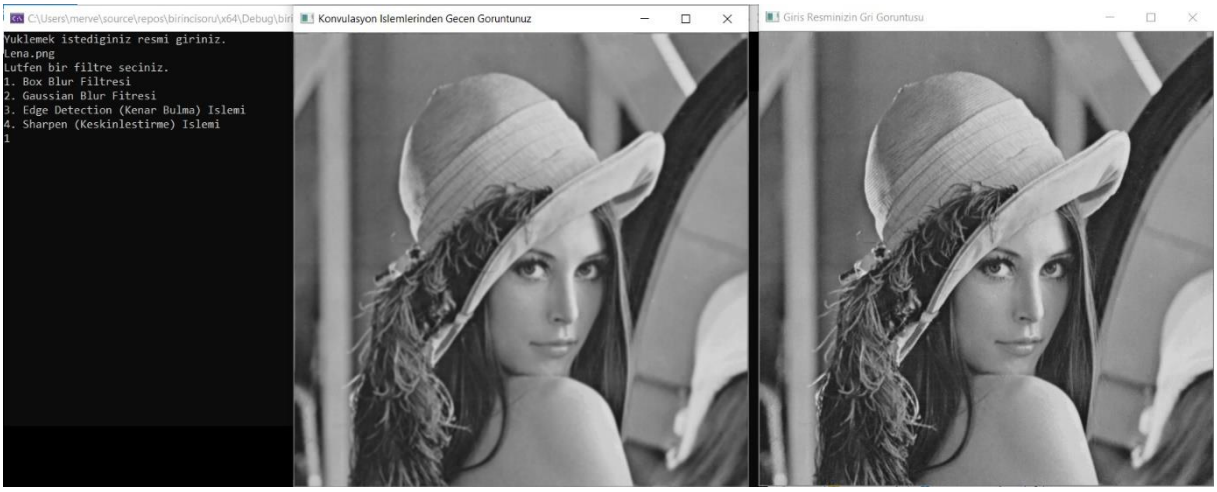
Şekil 7 ‘de yer alan kaynak kodlara ait akış şeması Şekil 8 ’de yer almaktadır.



Şekil 8. Soru-2 Akış Şeması

4. SORU-4

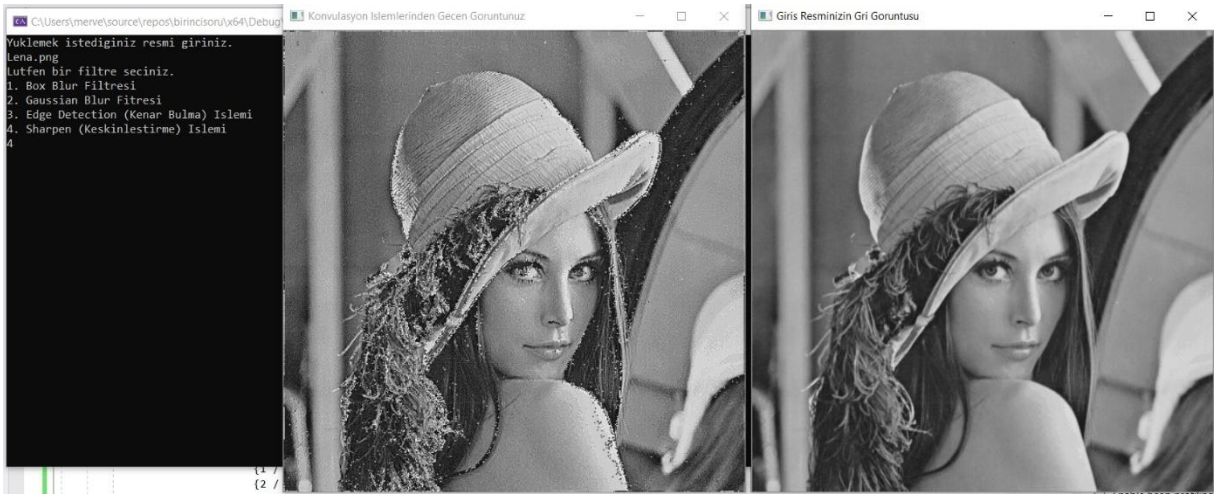
4. sorunun çözümlendirilmesi amacıyla çeşitli araştırmalar yapılarak çeşitli kaynaklardan yararlanılmıştır [3][4]. Öncelikli olarak kullanıcıdan konvolüsyon işleminin gerçekleştirilebilmesi adına bir görüntü girmesi istenmiştir. Bu görüntü belli bir dosya yolu üzerinde olmak zorundadır. Kullanıcıdan dosya yolu alma işlemi başarılı olamamıştır. Daha sonrasında alınan görüntü gri seviyede bir görüntüye dönüştürülerek kullanıcıdan 3x3 ‘lük bir filtre girilmesi istenmiştir. Bu filtre değerleri araştırılarak uygun olan filtre adı üzerinde ayarlamalar yapılmıştır. Şekil 9’ da görülen ekran çıktısı üzerinde 3x3 ‘lük Box Blur filtresi uygulanmıştır. Şekil 10 ’da ise Gaussian Fitresi, Şekil 11’de görüntüden kenar bulma işlemini gerçekleştirebileceğimiz Edge Detection ve son olarak Şekil 12 ‘de görüntüyü keskinleştirmeye yarayacak Sharpen işleminden yararlanılmıştır.



Şekil 9. Soru-4 Ekran Görüntüsü

Şekil 10. Soru-4 Ekran Görüntüsü

Şekil 11. Soru-4 Ekran Görüntüsü



Şekil 12. Soru-5 Ekran Görüntüsü

Soru 4 ‘e ait yukarıda görülen ekran görüntülerinin kaynak kodları Şekil 13 ‘da yer almaktadır.

```
#include "stdlib.h"
#include "math.h"
#include <iostream>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <string>
using namespace std;
using namespace cv;


int reflect(int M, int x)
{
    if (x < 0)
    {
```

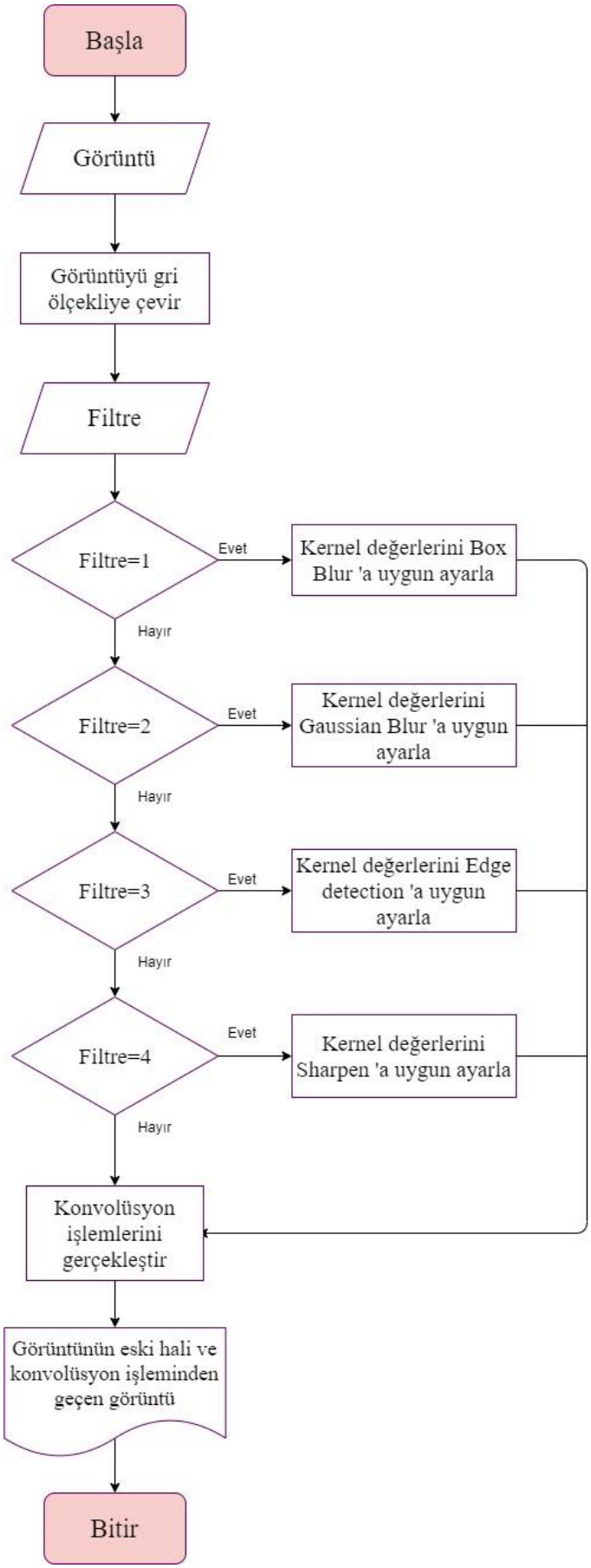
```
        return -x - 1;
    }
    if (x >= M)
    {
        return 2 * M - x - 1;
    }
    return x;
}

int circular(int M, int x)
{
    if (x < 0)
        return x + M;
    if (x >= M)
        return x - M;
    return x;
}


void noBorderProcessing(Mat src, Mat dst, float Kernel[][3])
{

    float sum;
    for (int y = 1; y < src.rows - 1; y++) {
        for (int x = 1; x < src.cols - 1; x++) {
            sum = 0.0;
            for (int k = -1; k <= 1;k++) {
                for (int j = -1; j <= 1; j++) {
                    sum = sum + Kernel[j + 1][k + 1] * src.at<uchar>(y - j, x - k);
                }
            }
            dst.at<uchar>(y, x) = sum;
        }
    }
}

void refletedIndexing(Mat src, Mat dst, float Kernel[][3])
{
    float sum, x1, y1;
    for (int y = 0; y < src.rows; y++) {
        for (int x = 0; x < src.cols; x++) {
            sum = 0.0;
            for (int k = -1;k <= 1; k++) {
                for (int j = -1;j <= 1; j++) {
                    x1 = reflect(src.cols, x - j);
                    y1 = reflect(src.rows, y - k);
                    sum = sum + Kernel[j + 1][k + 1] * src.at<uchar>(y1, x1);
                }
            }
            dst.at<uchar>(y, x) = sum;
        }
    }
}

void circularIndexing(Mat src, Mat dst, float Kernel[][3])
{
    float sum, x1, y1;
    for (int y = 0; y < src.rows; y++) {
        for (int x = 0; x < src.cols; x++) {
            sum = 0.0;
            for (int k = -1;k <= 1; k++) {
                for (int j = -1;j <= 1; j++) {
```

```
                    x1 = circular(src.cols, x - j);
                    y1 = circular(src.rows, y - k);
                    sum = sum + Kernel[j + 1][k + 1] * src.at<uchar>(y1, x1);
                }
            }
            dst.at<uchar>(y, x) = sum;
        }
    }
}

int main(int argc, char* argv[])
{

    Mat src, dst;
    string a;
    int b;

    cout << "Yuklemek istediginiz resmi giriniz." << endl;
    cin >> a;
    src = imread("C:\\Users\\merve\\OneDrive\\Masaüstü\\goruntuisleme\\" + a);

    cvtColor(src, src, COLOR_BGR2GRAY);
    if (!src.data)
    {
        return -1;
    }
    dst = src.clone();
    for (int y = 0; y < src.rows; y++)
        for (int x = 0; x < src.cols; x++)
            dst.at<uchar>(y, x) = 0.0;
    cout << "Lutfen bir filtre seciniz." << endl;
    cout << "1. Box Blur Filtresi" << endl;
    cout << "2. Gaussian Blur Fitresi" << endl;
    cout << "3. Edge Detection (Kenar Bulma) Islemi" << endl;
    cout << "4. Sharpen (Keskinlestirme) Islemi" << endl;
    cin >> b;
    if(b==1){
    float Kernel[3][3] = {
                          {1 / 9.0, 1 / 9.0, 1 / 9.0},
                          {1 / 9.0, 1 / 9.0, 1 / 9.0},
                          {1 / 9.0, 1 / 9.0, 1 / 9.0}
    };
    circularIndexing(src, dst, Kernel);
    }
    if (b ==2) {

        float Kernel[3][3] = {
                              {1 / 16.0, 2 / 16.0, 1 / 16.0},
                              {2 / 16.0, 4 / 16.0, 2 / 16.0},
                              {1 / 16.0, 2 / 16.0, 1 / 16.0}
        };
        circularIndexing(src, dst, Kernel);
    }
    if (b ==3) {
        float Kernel[3][3] = {
                              {-1.0, -1.0, -1.0},
                              {-1.0, 8.0, -1.0},
                              {-1.0, -1.0, -1.0}
        };
        circularIndexing(src, dst, Kernel);
    }
    if (b == 4) {
```

```
        float Kernel[3][3] = {
                                {0.0, -1.0, 0.0},
                                {-1.0, 5.0, -1.0},
                                {0.0, -1.0, 0.0}
        };
        circularIndexing(src, dst, Kernel);
    }

    namedWindow("Konvulasyon Islemlerinden Gecen Goruntunuz");
    imshow("Konvulasyon Islemlerinden Gecen Goruntunuz", dst);

    namedWindow("Giris Resminizin Gri Goruntusu");
    imshow("Giris Resminizin Gri Goruntusu", src);

    waitKey();


    return 0;
}
```

Şekil 13. Soru-4 Kaynak Kodu

Şekil 13 ‘de yer alan kaynak kodlara ait akış şeması Şekil 14 ’de yer almaktadır.



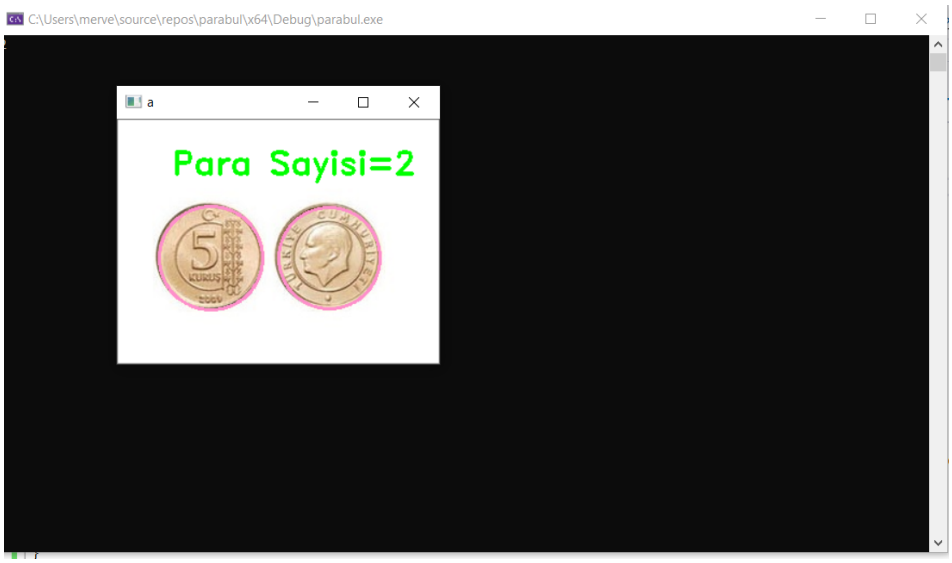
Şekil 14. Soru-4 Akış Şeması

5. SORU-5

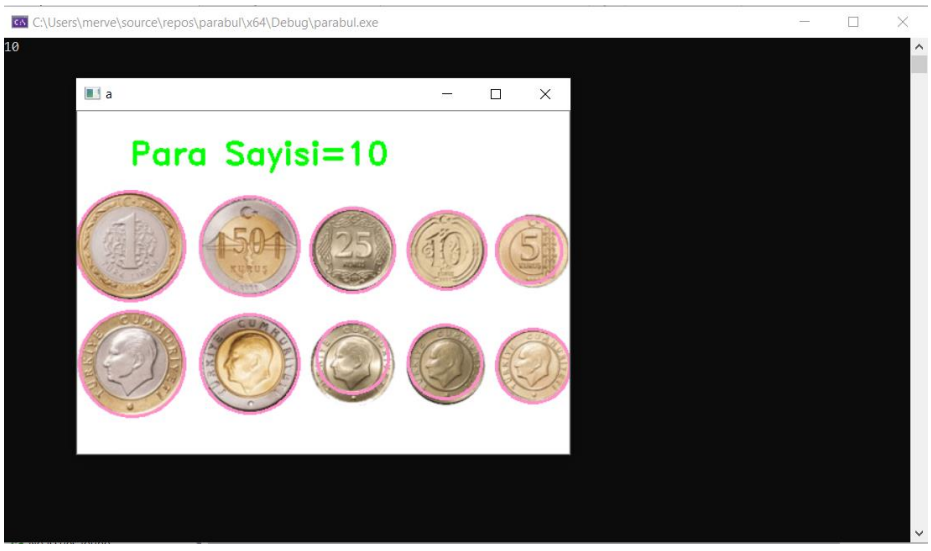
5. sorunun çözümlendirilmesi amacıyla çeşitli araştırmalar yapılmıştır. Öncelikli olarak metal paraların köşelerinin belirginleştirilmesi için çeşitli filtrelerden yardım alınmıştır. Filtrelendirme işleminden sonra metal paraların kenarları istenilen renkte çizdirilmiştir. Bulunan metal paraların sayısı çizilen daire sayılarına orantılı olarak ekrana yazdırılmıştır.

Metal paraların değerleri bulunmak üzere para sayılarının yarıçapının (radius) bulunması ve buna bağlı olarak metal paraların değerleriyle beraber toplam para miktarının bulunması denenmiştir. Ancak resimlere göre yarıçapların boyutları değişkenlik gösterebildiğinden istenilen sonuca ulaşılamamıştır.

Şekil 15 ve Şekil 16 ‘da yukarıda anlatılmış olan soru çözümlemesinin ekran görüntülerine yer verilmiştir.



Şekil 15. Soru-5 Ekran Görüntüsü



Şekil 16. Soru-5 Ekran Görüntüsü

Soru 5 ‘e ait yukarıda görülen ekran görüntülerinin kaynak kodları Şekil 17 ‘de yer almaktadır.

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
#include "opencv2/imgproc.hpp"
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>
#include<opencv2/opencv.hpp>
#include <iostream>

using namespace cv;
using namespace std;
#include <vector>
using namespace cv;

int main()
{

    Mat img_blur, img, gray;
    img = imread("C:\\Users\\merve\\OneDrive\\Resimler\\coin.png");


    medianBlur(img, img_blur, 5);
    cvtColor(img_blur, gray, COLOR_BGR2GRAY);
    vector<Vec3f>  circles;
    HoughCircles(gray, circles, HOUGH_GRADIENT, 1, gray.rows / 8, 80, 75, 0, 0);
    cout << circles.size() << endl;
    for (size_t i = 0; i < circles.size(); i++) {
        Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));
        int radius = cvRound(circles[i][2]);
        cout << radius << endl;
        RNG rng(12345);
        Scalar color = Scalar(rng.uniform(0, 255), rng.uniform(0, 255),
rng.uniform(0, 255));
        circle(img, center, radius, color, 2, 8, 0);
    }
    putText(img, "Para Sayisi=" + to_string(circles.size()), cv::Point(50, 50),
cv::FONT_HERSHEY_DUPLEX, 1, cv::Scalar(0, 255, 0), 2, false);
    imshow("a", img);
    waitKey(0);

    return 0;
}
```

Şekil 17. Soru-5 Kaynak Kodu

Şekil 17 ‘de yer alan kaynak kodlara ait akış şeması Şekil 18 ’de yer almaktadır.

Şekil 18. Soru-5 Akış Diyagramı

REFERANSLAR

[1] https://medium.com/@subwaymatch/opencv-410-with-vs-2019-3d0bc0c81d96

[2] http://opencv-tutorials-hub.blogspot.com/2015/11/image-scaling-image-zooming-tutorial-opencv-hub-how-to-magnify-an-image-using-trackbar-code-opencvzooming-method-replication.html

[3] https://medium.com/analytics-vidhya/image-convolution-from-scratch-d99bf639c32a

[4] https://followtutorials.com/2013/02/calculating-convolution-of-image-with-c_2.html