

T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

ENGINEERING FACULTY

COMPUTER ENGINEERING



LICENSE PLATE RECOGNITION SYSTEM

SOFTWARE ENGINEERING APPLICATION

PREPARED

ZEYNEP GİZEM ÇETİNCİ B1605.010034

NURŞAH DEMİRPOLAT B1605.010034

CONSULTANT

DR. ÖĞR. ÜYESİ ADEM ÖZYAVAŞ

CONTENTS

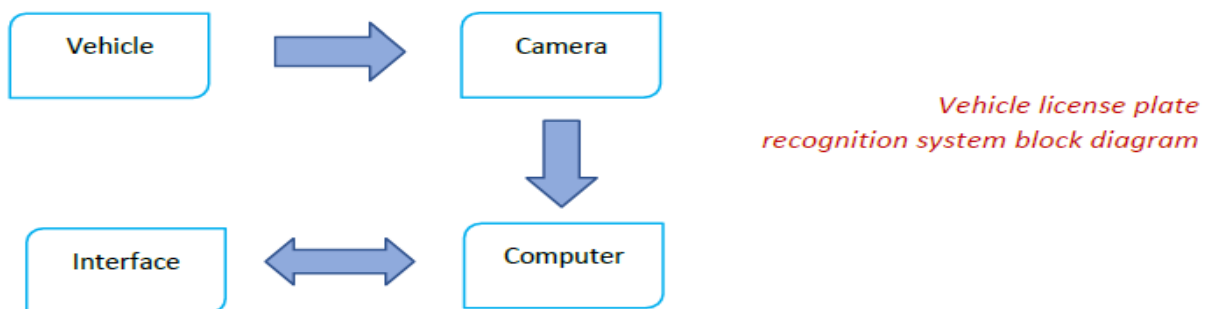
COVER.....	0
CONTENTS.....	1
1.INTRODUCTION.....	2
2.MATERIAL AND METHOD.....	3
2.1. GENERAL ALGORITHM STRUCTURE.....	4
2.2. OPTICAL CHARACTER RECOGNITION(OCR).....	5
2.2.1. CHARACTER RECOGNITION.....	6
2.2.2. FEATURE EXTRACTION.....	6
2.2.3. WHAT IS THE TESSERACT LIBRARY? HOW IT WORKS?.....	8
3. IMAGE PROCESSING.....	9
3.1. ALGORITHMS USED IN IMAGE PRETREATMENT.....	10
3.1.1. GRAYSCALE ALGORITHM.....	10
3.1.1.1. GRAYSCALE ALGORITHM CODE	11
3.1.2. BINARY ALGORITHM.....	12
3.1.2.1. BINARY ALGORITHM CODE.....	12
3.1.3. MEDIAN ALGORITHM.....	13
3.1.3.1. MEDIAN ALGORITHM.....	15
3.1.4. THRESHOLDING ALGORITHM.....	16
3.1.4.1. THRESHOLDING ALGORITHM CODE.....	17
3.1.5. SOBEL EDGE DETECTION ALGORITHM.....	17
3.1.5.1. SOBEL EDGE DETECTION ALGORITHM CODE.....	18
3.1.6. RECTANGULAR ALGORITHM	20
3.1.6.1. RECTANGULAR ALGORITHM CODE.....	20
3.1.7. ALGORITHM TO CROP AND READ THE PLATE IN THE PICTURE	21
3.1.7.1 ALGORITHM TO CROP AND READ THE PLATE IN THE PICTURE CODE.....	21
3.1.7.1. CLIPPING ALGORITHM CODE.....	21
3.1.7.2. READ ALGORITHM CODE.....	22
4. THE STEPS TO UPDOAD THE PROJECT ON GITHUB.....	23
5. CONCLUSION AND SUGGESTIONS.....	25
6. REFERENCE.....	27

1.INTRODUCTION

Nowadays, almost everyone is driving now. The best way to find out who owns the car is to find out who owns the car license plate, and the car license plate recognition system plays an important role. bridge tolls, parking lots... etc. Vehicle license plate recognition systems are needed to determine the location of the vehicle and to observe the illegal behaviors it makes. Embedded electronic systems using radio frequencies, sensors placed under the road to detect the vehicles covered by the system and many other reasons increase the vehicle cost considerably. A project covering the costs of million dollars has been launched. These programs include tracking vehicles, electronic fines, license plate recognition and vehicle recognition. With this, it is aimed to increase safety, efficiency and comfort, to produce economical solutions and to reduce pollution.

Turkey legally registered to the traffic supervisory authorities plates; civil, official, military, diplomatic, etc. There are different types such as and all types of different colors and formats (Traffic Control, 2010). In this study, it is aimed to recognize civilian plates that meet Turkish license plate standards. The general feature of these plates; It consists of black characters on a white background, the numbers in the first two characters indicating the city code, and the characters that come after it consist of random letters and numbers.

As seen in the block diagram, it basically consists of 4 main sections.



Vehicle: Recognition of the license plate is based on the passage of a vehicle by using algorithms written from the camera where the system is installed.

Camera: The image of the vehicle arriving at the place where the system is installed is taken by camera. The video resolution of the camera used should be 1024X768.

Computer: The performance of the computer used should be sufficient for the system to operate fast and the entrance port to the parallel port of the computer. It must be connected.

Interface Software: The interface program written for the system must be installed on the computer. The *.Net Framework* infrastructure must be installed on the computer for this interface to work.

2.MATERIAL AND METHOD

This section describes the paths followed during the creation of the license plate recognition system. In the first part, the hardware requirements required to build the system are mentioned in the other part.

The program developed for the license plate recognition system has intel (R) Core (M) processor 2.90 GHz processor, 8.00 MB RAM, computer with x64 bit based processor system. Developed using C # in Visual Studio 2019 on Microsoft Windows 10 operating system, the program was developed using neural network training and image processing algorithms.

The pictures used by the program were found on the internet. Pictures that are clear and close to the plate were preferred to read the plates.

2.1. GENERAL ALGORITHM STRUCTURE

What happened during the time until the detected vehicle picture is sent to the artificial neural network is shown below.



2.2. OPTICAL CHARACTER RECOGNITION(OCR)

Optical character recognition (also optical character reader, OCR) are mechanical or electronic transformation images from the scanned document, when machine-coded text is written, handwritten or printed text, a document, a scene is not given (a sample landscape for text on sign and ad boards from a television broadcast (in the photo) (eg subtitle text superimposed on an image).

This printed digitization is a common method, which can be used electronically, so that texts are more compressed, used as a form of information input from common printed paper data records - passport documents, invoices, bank statements, computerized receipts, business cards, postal data or any suitable documentation printouts. hiding in a way, the search is shown online and in this way cognitive processing used in machine operations, machine translation, (extract) text-to-speech, key data and text mining. OCR is a research area in pattern recognition, artificial intelligence and computer vision.

He worked on an early version that every character wanted to train with images and one article at a time. Advanced systems that are capable of producing a high degree of recognition for most fonts are now also common and the digital image file format with support for various inputs. Some systems are capable of re-formatting the original page approaching formatted, which includes closely displays, columns and other non-text related components.

2.2.1. CHARACTER RECOGNITION

OCR software, paper etc. black dots on it, letters, symbols, etc. They have a structure that can perceive and read. The first step of OCR is to use a browser to process the physical form of a document. After all the pages have been copied, OCR software converts the document to a black and white version. Dark areas are defined as characters to be recognized, and open areas are defined as backgrounds. Analysis is made for light and dark areas. This is done by keeping each symbol separate and dividing the page into lines. During the analysis, the structural characters (height, width, etc.) of each symbol are evaluated. An algorithm recognizes characters in documents and turns them into a searchable digital format. Various parameters defined for each character are available in the program's own information store. If the detected pixels match one of these parameters, it is possible for the symbol to appear.

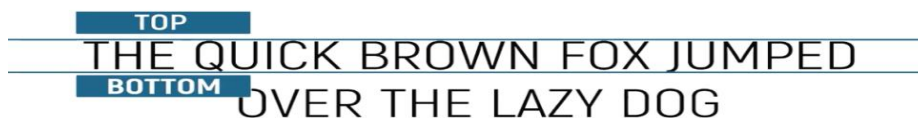
2.2.2. FEATURE EXTRACTION

There are two main methods for extracting features in OCR;

1. In the first method, the feature detection algorithm identifies a character by evaluating its lines and strokes.



2. In the second method, pattern recognition works by defining the entire character.



Pattern recognition on a row of text



Pattern recognition on a single character.

We can recognize a line of text by searching for white pixel lines with black pixels between them. Similarly, we can understand where a character begins and ends. Next, we convert the image of the character into a binary matrix where white pixels are 0s and black pixels are 1s as shown in the following image:

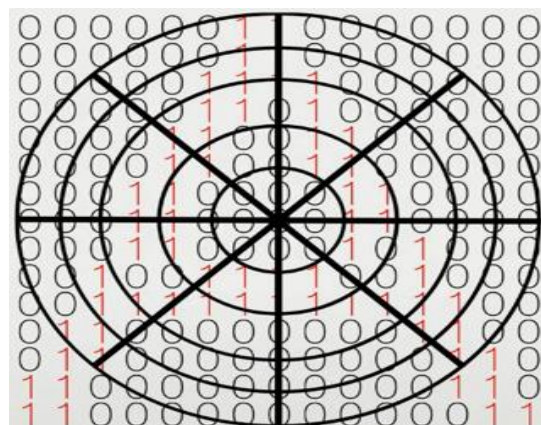


Then, by using the distance formula, we can find the distance from the center of the matrix to the farthest 1.

The distance formula;

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

We then create a circle of that radius and split it up into more granular sections. At this stage, the algorithm compares each subsection to a database of matrices representing characters with different fonts to identify the character it has most in common statistically. It makes it easy to bring printed media into the digital world by doing this for every line and character.



Compare each subsection against the matrix database.

2.2.3. WHAT IS THE TESSERACT LIBRARY? HOW IT WORKS?

Tesseract is one of the most accurate open source OCR engines. Tesseract allows us to convert the given image into the text. Before going to the code we need to download the assembly and tessdata of the Tesseract. We can download the data from GitHub or NuGet. Then add the following package.

```
1. using tessnet2;
2. using System.Drawing;
3. using System.Drawing.Drawing2D;
4. using System.Drawing.Imaging;
5.
6. // now add the following C# line in the code page
7. var image = new Bitmap(@"Z:\NewProject\demo\image.bmp");
8. var ocr = new Tesseract();
9. ocr.Init(@"Z:\NewProject\How to use Tessnet2 library\C#\tessdata", "eng", false);
10. var result = ocr.DoOCR(image, Rectangle.Empty);
11. foreach(tessnet2.Word word in result)
12. {
13.     Console.WriteLine(word.text);
14. }
```

Tesseract — is an optical character recognition engine with open-source code, this is the most popular and qualitative OCR-library.

OCR uses artificial intelligence for text search and its recognition on images. Tesseract is finding templates in pixels, letters, words and sentences. It uses two-step approach that calls adaptive recognition. It requires one data stage for character recognition, then the second stage to fulfil any letters, it wasn't insured in, by letters that can match the word or sentence conte

3. IMAGE PROCESSING

Image Processing is a method developed to digitalize the image and perform some operations, used to obtain specific images or extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image. The input of this method is an image like a video section or a photo. Its output corresponds to the part of the image that is desired or needs attention. Generally, the Image Processing system treats images as two-dimensional signals when applying the Signal Processing methods. Our numerical image structure a [m, n], consisting of 1 and 0, is created using the sampling technique from the a (x, y) function obtained from the 2D world.

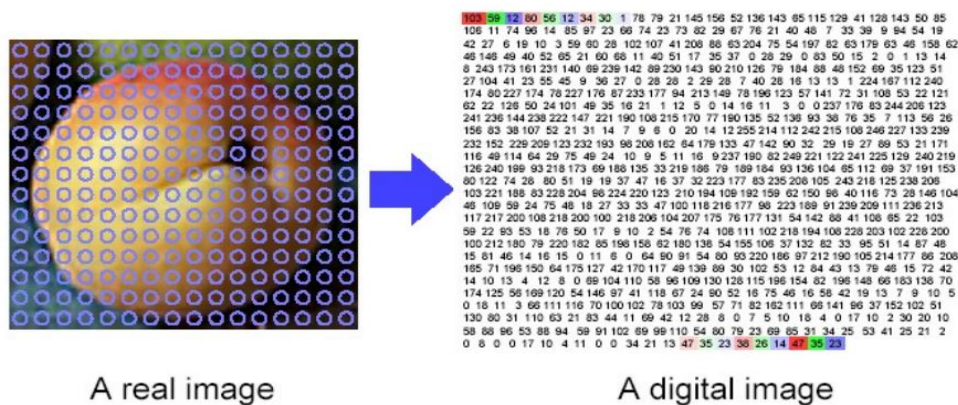


Image processing systems used by companies in various aspects are among the rapidly growing technologies. It also constitutes the basic research area in image processing, engineering and computer science disciplines.

Image processing basically involves the following three steps.

- Image acquisition by optical scanner or digital photos.
- Analyze-use image containing non-human eye spotting patterns such as data compression, image enhancement and satellite photos.
- Output made altered based on image analysis of the results, making them ready for use.

Purpose of image processing

The purpose of image processing is divided into 5 groups. Them:

1. Visualization - Observing hard-to-see objects

2. Image sharpening and restoration - Improving noisy images
3. Image acquisition - Interesting and high resolution image search
4. Pattern Recognition - Identify various objects in an image.
5. Image Recognition - Distinguish objects in an image.

3.1.ALGORITHMS USED IN IMAGE PRETREATMENT

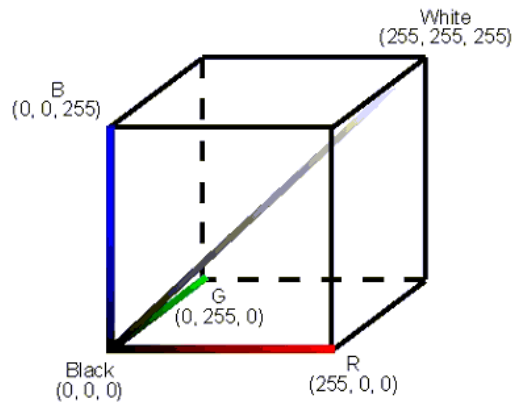
Filtering is the recalculation of each pixel value, thinking as if there is a filter on the picture. Thanks to the filters, the new picture from the input picture is obtained by giving different effects. The filtering process can be obtained with the following formula:

$$f'(x, y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} h(i, j) \times f(x - i, y - j)$$

Here the function h is a filter.

3.1.1. GRAYSCALE ALGORITHM

The fact that there are a lot of unnecessary details and noise that need to be cleaned in the color image makes recognition difficult. Therefore, it is necessary to change the picture to a gray level, simpler picture. RGB values take values between 0-255 and 0- characterizes black color, 255-value white color. Assuming that the black and white colors are at the beginning and end of a straight line, the values on this line will be the values of the shades of gray.



Since $R = G = B$ in each pixel of the gray level image, the easiest method of reducing it to the gray level is to find the average of the R, G, B values of the color image. In this gray image, a gray scale is obtained by averaging the color value in each pixel. Gray scaling; P is given in Equation 1 to indicate an image, i and j in coordinates.

$$\mathbf{I}_{grey}(\mathbf{p}) = \frac{\mathbf{I}_R(\mathbf{p}) + \mathbf{I}_G(\mathbf{p}) + \mathbf{I}_B(\mathbf{p})}{3}$$



3.1.1.1. GRAYSCALE ALGORITHM CODE

```
private Bitmap grayLevel(Bitmap bmp)
{
    for (int i = 0; i < bmp.Height - 1; i++)
    {
        for (int j = 0; j < bmp.Width - 1; j++)
        {
            int value = (bmp.GetPixel(j, i).R + bmp.GetPixel(j, i).G +
                bmp.GetPixel(j, i).B) / 3; Color renk;
            renk = Color.FromArgb(value, value, value); bmp.SetPixel(j, i,
                renk);
        }
    }
    return bmp;
}
```

Another method is to convert it to black and white with the herbaceous algorithm, but we have used the above method because converting to black and white is much simpler with the above method. Pixels that are higher than this threshold value turn white, and pixels that are lower turn black. It is shown in the formula for converting into binary drawing by determining a fixed threshold value;

$$\mathbf{I}_{bin}(\mathbf{p}) = \begin{cases} 1 & \text{if } \mathbf{I}_{grey}(\mathbf{p}) \geq d \\ 0 & \text{otherwise} \end{cases}$$

3.1.2. BINARY ALGORITHM

Binary Algorithm is important in image processing to extract objects from their background into binary image. Binary image is used as input to feature extraction process and have an important role in generating unique feature to distinguish several classes in pattern recognition. This paper proposes an image processing algorithm to obtain a binary image from RGB. The results showed that the binary image of the proposed algorithm contained the desired object.



3.1.2.1. BINARY ALGORITHM CODE

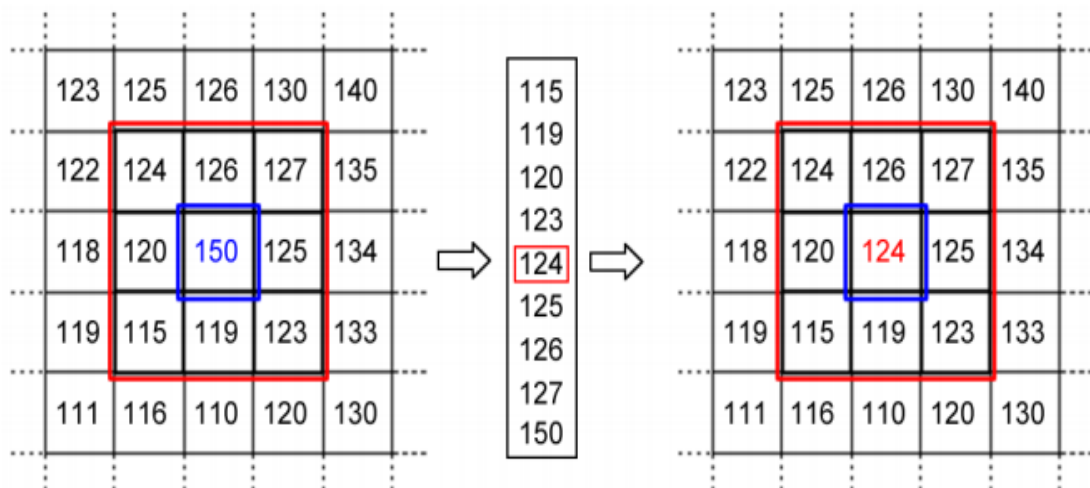
```
private Bitmap Makebinary(Bitmap image)
{
    try {
        grayLevel(image); int temp;
        int level = 127; Color color;
        for (int i = 0; i < image.Height - 1; i++) { for (int j = 0; j < image.Width - 1; j++)
        {
            temp = image.GetPixel(j, i).B; if (temp < level){
                color = Color.FromArgb(0, 0, 0); image.SetPixel(j, i, color);
            }
            else{
                color = Color.FromArgb(255, 255, 255); image.SetPixel(j, i, color);
            }
        }
    }
    return image;
}
```

3.1.3. MEDIAN ALGORITHM

The median filter is normally used to reduce noise in an image, such as the mean filter. However, it does much better than the mean filter at the point that the details on the picture are not lost.

Like the mean filter, the median filter looks at its neighbors nearby to calculate the value of each pixel. Instead of replacing the pixel value with the average of the neighboring pixel values in the median filter (mean filter), it sorts the neighboring pixels and takes the value in the middle of the row. If the examined area (inside the template) has an even number of pixels, the average of the two pixels in the middle is used as the middle value.

If we do our operations according to the middle pixel in the figure below, we can see that the value of this pixel does not represent the pixels around 150, well. When changing the value of this pixel, let's first line up the value of the surrounding pixels. These consist of values (115, 119, 120, 123, 124, 125, 126, 127, 150). In the middle of these values, there are 124 numbers. Accordingly, the number 150 is replaced by the number 124. Here the number 124 becomes the median number (middle value). The template used here is 3x3 pixels in size. Using larger templates produces more smoothing (softening) effects.

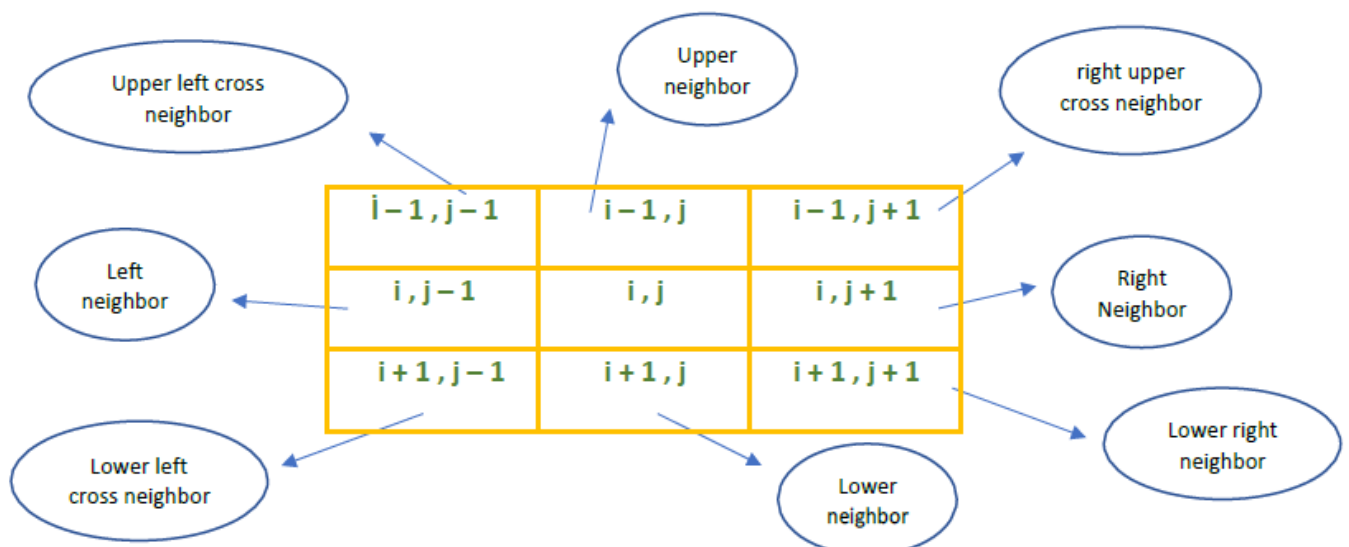


Median filters, The average filter is the simplest filter that can be created with the help of a kernel. For each pixel, the surrounding pixels are averaged. Mean filters are ideal for restoring images with Gaussian noise. However, instead of taking the average here, it is assigned to the pixel, which is the median of the values in the neighborhood. Median filters are ideal for restoring images with impulsive noise.



The image converted to binary image and the median image of that photo are as above.

The median finding algorithm is like this. We try to find the median by crossing the right and left with i and j .



```

rightNeighbor = image.GetPixel(j + 1, i).R;
rightUpperCrossNeighbor = image.GetPixel(j + 1, i - 1).R;
upperNeighbor = image.GetPixel(j, i - 1).R;
upperLeftCrossNeighbor = image.GetPixel(j - 1, i - 1).R;
leftNeighbor = image.GetPixel(j - 1, i).R;
lowerLeftCrossNeighbor = image.GetPixel(j - 1, i + 1).R;
lowerNeighbor = image.GetPixel(j, i + 1).R;
lowerRightNeighbor = image.GetPixel(j + 1, i + 1).R;

```

3.1.3.1. MEDIAN ALGORITHM CODE

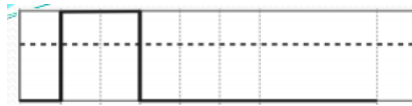
```
private Bitmap medianAlgorithm(Bitmap image)
{
    Bitmap buffer = new Bitmap(image.Width, image.Height);
    Color color;
    for (int i = 0; i < image.Height; i++) { for (int j = 0; j < image.Width; j++)
    {
        if ((i == 0) || (i == image.Height - 1) || (j == 0) || (j == image.Width - 1))
            continue;
        else{
            int ortanca = medianFind(image, j, i);
            color = Color.FromArgb(ortanca, ortanca, ortanca); buffer.SetPixel(j, i, color);
        }
    }
    }
    return buffer;
}

private int medianFind(Bitmap image, int j, int i)
{
    int[] dizi = new int[9];
    Color color;
    int sagkomsu, sagustcaprazkomsu, ustkomsu, solustcapraz, solkomsu, solaltcapraz,
    altkomsu, sagaltcapraz;
    sagkomsu = image.GetPixel(j + 1, i).R; sagustcaprazkomsu = image.GetPixel(j + 1, i -
    1).R;
    ustkomsu = image.GetPixel(j, i - 1).R;
    solustcapraz = image.GetPixel(j + 1, i - 1).R; solkomsu = image.GetPixel(j - 1, i).R;
    solaltcapraz = image.GetPixel(j - 1, i + 1).R; altkomsu = image.GetPixel(j, i + 1).R;
    sagaltcapraz = image.GetPixel(j + 1, i + 1).R;
    dizi[0] = image.GetPixel(j, i).R; dizi[1] = sagkomsu;
    dizi[2] = sagustcaprazkomsu; dizi[3] = ustkomsu;
    dizi[4] = solustcapraz;
    dizi[5] = solkomsu;
    dizi[6] = solaltcapraz;
    dizi[7] = altkomsu;
    dizi[8] = sagaltcapraz;
    for (int x = 0; x < 8; x++)
    {
        for (int y = x + 1; y < 9; y++)
        {
            if (dizi[x] < dizi[y])
                continue;
            else{
                int temp = dizi[y]; dizi[y] = dizi[x]; dizi[x] = temp;
            }
        }
    }
    return dizi[4];
}
```


3.1.4. THRESHOLDING ALGORITHM

This function is often used to create a binary image from a grayscale image. It can also be used with color images. Filters the pixel values of the source image to very large or very small values. It is mostly used to remove noise in images. There are many types of thresholds:

1. Binary Type



$$\text{Hedef } (x,y) = \begin{cases} \text{maks. de\u011fer} & \text{e\u011fer kaynak } (x,y) > E \\ 0 & \text{aksi takdirde} \end{cases}$$

2. Binary Inverted Type



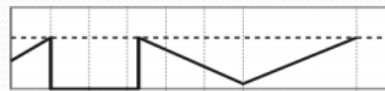
$$\text{Hedef } (x,y) = \begin{cases} 0 & \text{e\u011fer kaynak } (x,y) > E \\ \text{maks. de\u011fer} & \text{aksi takdirde} \end{cases}$$

3. To Zero Type



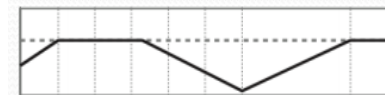
$$\text{Hedef } (x,y) = \begin{cases} \text{e\u015fik de\u011fer} & \text{e\u011fer kaynak } (x,y) > E \\ \text{kaynak}(x,y) & \text{aksi takdirde} \end{cases}$$

4. To Zero Inverted Type,



$$\text{Hedef } (x,y) = \begin{cases} \text{kaynak}(x,y) & \text{e\u011fer kaynak } (x,y) > E \\ 0 & \text{aksi takdirde} \end{cases}$$

5. Truncate Tip



$$\text{Hedef } (x,y) = \begin{cases} 0 & \text{e\u011fer kaynak } (x,y) > E \\ \text{kaynak}(x,y) & \text{aksi takdirde} \end{cases}$$

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity $I_{i,j}$ is less than some fixed constant T or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming completely white. The reason we use this method is that the optical character reader can read better.



3.1.4.1. THRESHOLDING ALGORITHM CODE

```
public Bitmap ResmiEsiklemeYap(Bitmap GirisResmi)
{
    Color OkunanRenk, DonusenRenk; int R = 0, G = 0, B = 0;
    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;
    Bitmap CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);
    int i = 0, j = 0;
    for (int x = 0; x < ResimGenisligi; x++)
    {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
            OkunanRenk = GirisResmi.GetPixel(x, y); if (OkunanRenk.R >= 128)
                R = 255;
            else
                R = 0;
            if (OkunanRenk.G >= 128)
                G = 255;
            else
                G = 0;
            if (OkunanRenk.B >= 128)
                B = 255;
            else
                B = 0;
            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(i, j, DonusenRenk);
            j++;
        }
        i++;
    }
    return CikisResmi;
}
```

3.1.5. SOBELL EDGE DETECTION ALGORITHM

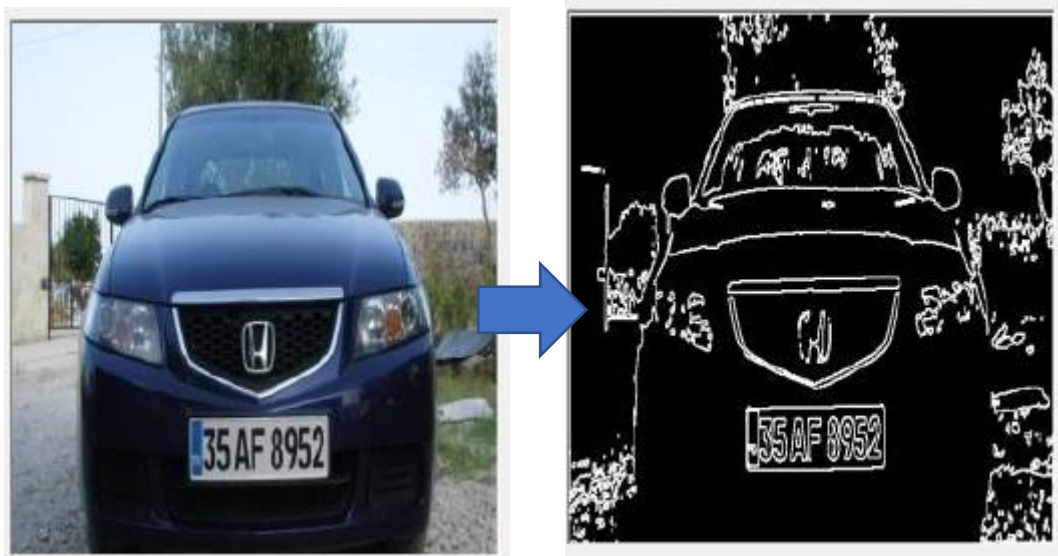
It is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges. It is named after Irwin Sobel and Gary Feldman, colleagues at the Stanford Artificial Intelligence

Laboratory (SAIL). Sobel and Feldman presented the idea of an "Isotropic 3x3 Image Gradient Operator" at a talk at SAIL in 1968. Technically, it is a discrete differentiation

operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel–Feldman operator is either the corresponding gradient vector or the norm of this vector. The Sobel–Feldman operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high- frequency variations in the image.

$$g_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad g_y = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$COLOR = \sqrt{g_x^2 + g_y^2}$$



3.1.5. SOBEL EDGE DETECTION ALGORITHM CODE

```
private Bitmap sobalEdgeDetection(Bitmap image)
{
    Bitmap buffer = new Bitmap(image.Width, image.Height);
    Color renk;
    int valX, valY; int gradient;
    int[,] GX = new int[3, 3];
    int[,] GY = new int[3, 3];
    GX[0, 0] = -1; GX[0, 1] = 0; GX[0, 2] = 1;
    GX[1, 0] = -2; GX[1, 1] = 0; GX[1, 2] = 2;
    GX[2, 0] = -1; GX[2, 1] = 0; GX[2, 2] = 1;
```

```

// For the edge in the vertical direction
GY[0, 0] = -1; GY[0, 1] = -2; GY[0, 2] = -1;
GY[1, 0] = 0; GY[1, 1] = 0; GY[1, 2] = 0;
GY[2, 0] = 1; GY[2, 1] = 2; GY[2, 2] = 1;
for (int i = 0; i < image.Height; i++){ for (int j = 0; j < image.Width; j++)
{
    if (i == 0 || i == image.Height - 1 || j == 0 || j == image.Width - 1)
    {
        renk = Color.FromArgb(255, 255, 255);
        buffer.SetPixel(j, i, renk); valX = 0;
        valY = 0;
    }
    else
    {
        valX = image.GetPixel(j - 1, i - 1).R * GX[0, 0] + image.GetPixel(j, i - 1).R *
        GX[0, 1] + image.GetPixel(j + 1, i - 1).R * GX[0, 2] + image.GetPixel(j - 1, i).R *
        GX[1, 0] + image.GetPixel(j, i).R * GX[1, 1] + image.GetPixel(j + 1, i).R * GX[1,
        2] + image.GetPixel(j - 1, i + 1).R * GX[2, 0] + image.GetPixel(j, i + 1).R * GX[2,
        1] + image.GetPixel(j + 1, i + 1).R * GX[2, 2];
        valY = image.GetPixel(j - 1, i - 1).R * GY[0, 0] + image.GetPixel(j, i - 1).R *
        GY[0, 1] + image.GetPixel(j + 1, i - 1).R * GY[0, 2] + image.GetPixel(j - 1, i).R *
        GY[1, 0] + image.GetPixel(j, i).R * GY[1, 1] + image.GetPixel(j + 1, i).R * GY[1,
        2] + image.GetPixel(j - 1, i + 1).R * GY[2, 0] + image.GetPixel(j, i + 1).R * GY[2,
        1] + image.GetPixel(j + 1, i + 1).R * GY[2, 2];
        gradient = (int)(Math.Abs(valX) + Math.Abs(valY));
        if (gradient < 0) gradient = 0;
        if (gradient > 255) gradient = 255;
        renk = Color.FromArgb(gradient, gradient, gradient); buffer.SetPixel(j, i, renk);
    }
}
}
return buffer;
}

```

3.1.6. RECTANGULAR ALGORITHM

In the rectangular discovery algorithm after the image processing sequence is finished. Thanks to this algorithm, we will be able to easily trim the plate.

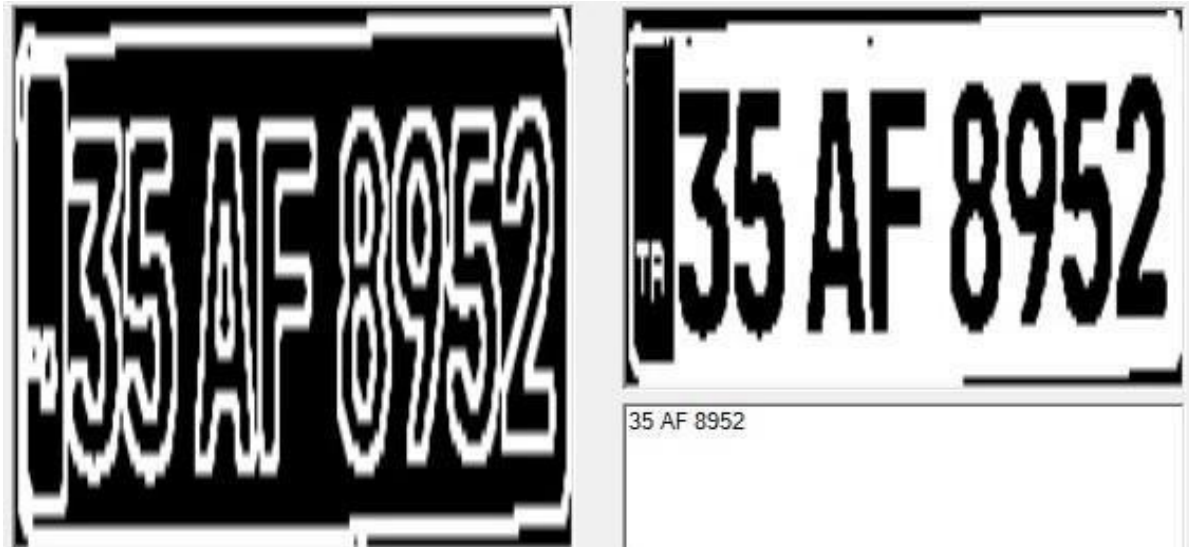


3.1.6.1. RECTANGULAR ALGORITHM CODE

```
private Bitmap sobalEdgeDetection(Bitmap image){
    Bitmap Image = (Bitmap)pictureBox5.Image; BlobCounter blobCounter = new
    BlobCounter(); blobCounter.FilterBlobs = true; blobCounter.MinHeight = 10;
    blobCounter.MinWidth = 10;
    blobCounter.MaxWidth = 350;
    blobCounter.MaxHeight = 350;
    blobCounter.ProcessImage(Image);
    Rectangle[] rects = blobCounter.GetObjectsRectangles();
    Graphics g = Graphics.FromImage(Image);
    if (rects.Length > 1)
    {
        var r2 = rects.OrderByDescending(r => r.Height * r.Width).ToList();
        Rectangle objectRect = r2[1];
        using (Pen pen = new Pen(Color.FromArgb(160, 255, 160), 5))
        {
            g.DrawRectangle(pen, objectRect);
        }
    }
    g.Dispose();
}
```

3.1.7. ALGORITHM TO CROP AND READ THE PLATE IN THE PICTURE

The plate region was found at different angles and at different times of the day, and because of the difference in the resolution of the pictures depending on the intensity, direction of the light and some factors, clarification algorithms were applied to the images.



The areas outside the plate region, which we can call noise on the cropped image of the plate, were destroyed using the Hydrangea Filter. Some regions that cannot be destroyed at the end of the filter are also removed from the image by taking into account the characteristics of the characters such as pixel, alignment and ratio.

We made it possible to read the plate using the Tesseract library. Reading the characters, at this stage, the characters separated in the license plate region are compared with the samples found in a text file, and is found to be equivalent to the value closest to the character. The image above shows an image with the reserved character read.

3.1.7. ALGORITHM TO CROP AND READ THE PLATE IN THE PICTURE CODE

3.1.7.1. CLIPPING ALGORITHM CODE

```
private Bitmap sobalEdgeDetection(Bitmap image)
{
    Bitmap images = new Bitmap(pictureBox4.Image); Bitmap Image =
    sobalEdgeDetection(images); BlobCounter blobCounter = new BlobCounter();
    blobCounter.FilterBlobs = true; blobCounter.MinHeight = 10;
    blobCounter.MinWidth = 10;
    blobCounter.MaxWidth = 350;
    blobCounter.MaxHeight = 350;
    blobCounter.ProcessImage(Image);
    Rectangle[] rects = blobCounter.GetObjectsRectangles();
```

```

if (rects.Length == 0)
{
    System.Windows.Forms.MessageBox.Show("No rectangle found in image ");
}
else if (rects.Length > 1)
{ // get targets rect
    Console.WriteLine("Using largest rectangle found in image ");
    var r2 = rects.OrderByDescending(r => r.Height * r.Width).ToList();
    Image = Image.Clone(r2[1], Image.PixelFormat);
}
else
{
    Console.WriteLine("Huh? on image ");
}
}
pictureBox7.Image = Image;
}

```

3.1.7.2. READ ALGORITHM

```

Bitmap image = new Bitmap(pictureBox2.Image);
Bitmap scr = ResmiEsiklemeYap(image);
Bitmap sc = Select(scr); pictureBox8.Image = sc;
var ocr = new TesseractEngine("./tessdata", "eng");
var page = ocr.Process(sc);
richTextBox1.Text = page.GetText();
private Bitmap Select(Bitmap Image)
{
    BlobCounter blobCounter = new BlobCounter(); blobCounter.FilterBlobs = true;
    blobCounter.ProcessImage(Image);
    Rectangle[] rects = blobCounter.GetObjectsRectangles();
    if (rects.Length == 0)
    {
        System.Windows.Forms.MessageBox.Show("No rectangle found in image ");
    }
    else if (rects.Length > 1)
    {
        // get targets rect
        Console.WriteLine("Using largest rectangle found in image ");
        var r2 = rects.OrderByDescending(r => r.Height * r.Width).ToList();
        Image = Image.Clone(r2[3], Image.PixelFormat);
    }
    else
    {
        Console.WriteLine("Huh? on image ");
    }
}
return Image;
}

```


4. THE STEPS TO UPDOAD THE PROJECT ON GITHUB

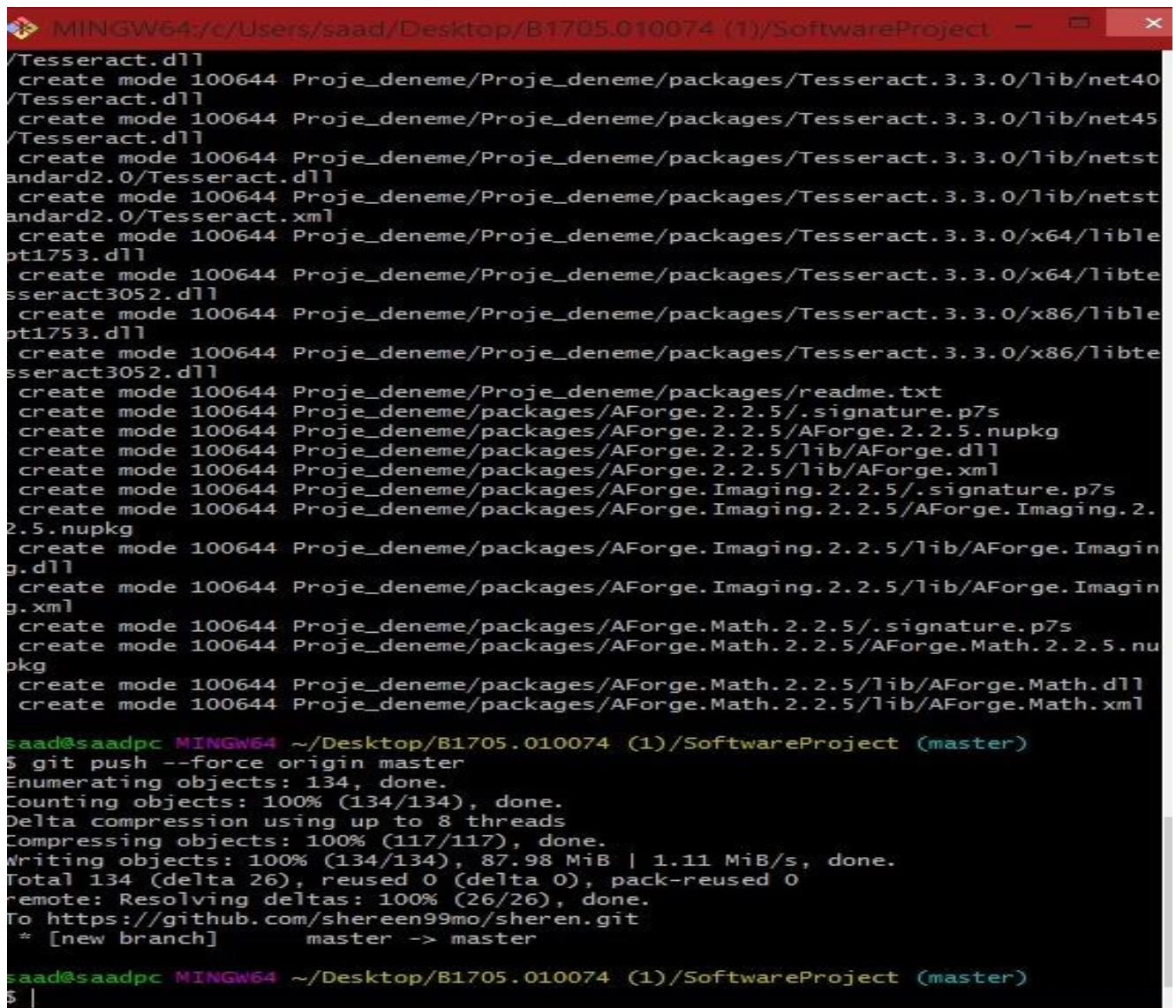
We used these methods when uploading our project to github.

```
git init
git add README.md
git add .
git commit -m "first commit"
git remote add origin https://github.com/username/repoName.git
git push --force origin maste
```

THE LINK:

<https://github.com/shereen99mo/sheren>

THE STEPS BY PICTURES



```
MINIW64:/c/Users/saad/Desktop/B1705.010074 (1)/SoftwareProject
/Tesseract.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/lib/net40
/Tesseract.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/lib/net45
/Tesseract.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/lib/netst
andard2.0/Tesseract.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/lib/netst
andard2.0/Tesseract.xml
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/x64/libte
sseract3052.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/x64/libte
sseract3052.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/x86/libte
sseract3052.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/Tesseract.3.3.0/x86/libte
sseract3052.dll
create mode 100644 Proje_deneme/Proje_deneme/packages/readme.txt
create mode 100644 Proje_deneme/packages/AForge.2.2.5/.signature.p7s
create mode 100644 Proje_deneme/packages/AForge.2.2.5/AForge.2.2.5.nupkg
create mode 100644 Proje_deneme/packages/AForge.2.2.5/lib/AForge.dll
create mode 100644 Proje_deneme/packages/AForge.2.2.5/lib/AForge.xml
create mode 100644 Proje_deneme/packages/AForge.Imaging.2.2.5/.signature.p7s
create mode 100644 Proje_deneme/packages/AForge.Imaging.2.2.5/AForge.Imaging.2.
2.5.nupkg
create mode 100644 Proje_deneme/packages/AForge.Imaging.2.2.5/lib/AForge.Imagin
g.dll
create mode 100644 Proje_deneme/packages/AForge.Imaging.2.2.5/lib/AForge.Imagin
g.xml
create mode 100644 Proje_deneme/packages/AForge.Math.2.2.5/.signature.p7s
create mode 100644 Proje_deneme/packages/AForge.Math.2.2.5/AForge.Math.2.2.5.nu
pkg
create mode 100644 Proje_deneme/packages/AForge.Math.2.2.5/lib/AForge.Math.dll
create mode 100644 Proje_deneme/packages/AForge.Math.2.2.5/lib/AForge.Math.xml
saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ git push --force origin master
Enumerating objects: 134, done.
Counting objects: 100% (134/134), done.
Delta compression using up to 8 threads
Compressing objects: 100% (117/117), done.
Writing objects: 100% (134/134), 87.98 MiB | 1.11 MiB/s, done.
Total 134 (delta 26), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (26/26), done.
To https://github.com/shereen99mo/sheren.git
 * [new branch]      master -> master
saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ |
```



```

MINGW64:/c/Users/saad/Desktop/B1705.010074 (1)/SoftwareProject
saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ got commit -m "software proje"
bash: got: command not found

saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ git remote add origin https://github.com/shereen99mo/sheren.git

saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ git add .

saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ got commit -m "software proje"
bash: got: command not found

saad@saadpc MINGW64 ~/Desktop/B1705.010074 (1)/SoftwareProject (master)
$ git commit -m "software proje"
[master (root-commit) f86b15e] software proje
142 files changed, 487885 insertions(+)
create mode 100644 "AdemProje/Ek A\303\247\304\261klama 2020-05-17 223214.JPG"
create mode 100644 "AdemProje/Ek A\303\247\304\261klama 2020-05-17 224224.JPG"
create mode 100644 "AdemProje/Ek A\303\247\304\261klama 2020-05-17 224509.JPG"
create mode 100644 "AdemProje/Ek A\303\247\304\261klama 2020-05-17 224644.JPG"
create mode 100644 "AdemProje/Ek A\303\247\304\261klama 2020-05-17 225322.JPG"
create mode 100644 LICENSE PLATE RECOGNITION SYSTEM.pdf
create mode 100644 Proje_deneme/.vs/Proje_deneme/v16/.suo
create mode 100644 Proje_deneme/.vs/ProjectSettings.json
create mode 100644 Proje_deneme/.vs/VSSWorkspaceState.json
create mode 100644 Proje_deneme/.vs/slnx.sqlite
create mode 100644 Proje_deneme/Proje_deneme.sln
create mode 100644 Proje_deneme/Proje_deneme/.vs/Proje_deneme/v16/.suo
create mode 100644 Proje_deneme/Proje_deneme/.vs/ProjectSettings.json
create mode 100644 Proje_deneme/Proje_deneme/.vs/slnx.sqlite
create mode 100644 Proje_deneme/Proje_deneme/App.config
create mode 100644 Proje_deneme/Proje_deneme/Form1.Designer.cs
create mode 100644 Proje_deneme/Proje_deneme/Form1.cs
create mode 100644 Proje_deneme/Proje_deneme/Form1.resx
create mode 100644 Proje_deneme/Proje_deneme/Program.cs
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme.csproj
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme.sln
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/App.config
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Form1.Designer.cs
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Form1.cs
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Form1.resx
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Program.cs
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Proje_deneme.csproj
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Properties/AssemblyIn
fo.cs
create mode 100644 Proje_deneme/Proje_deneme/Proje_deneme/Properties/Resources.

```

OUR PROJECT ON GITHUB;

1 commit
1 branch
0 packages
0 releases
1 contributor

Branch: master
New pull request
Find file
Clone or download

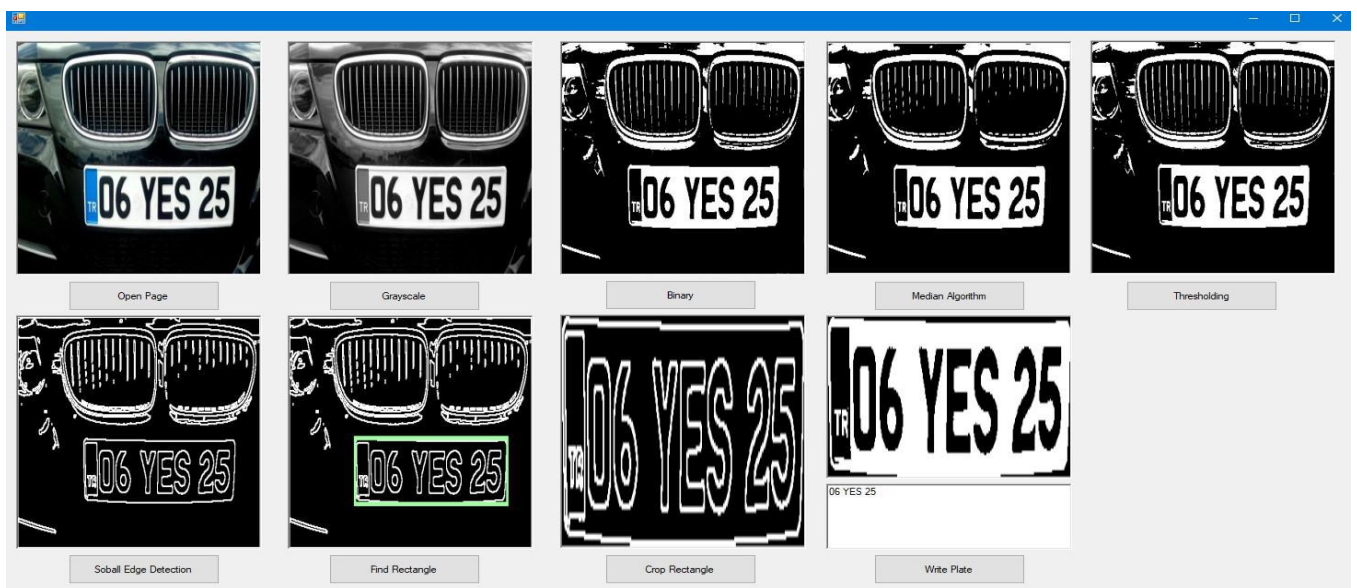
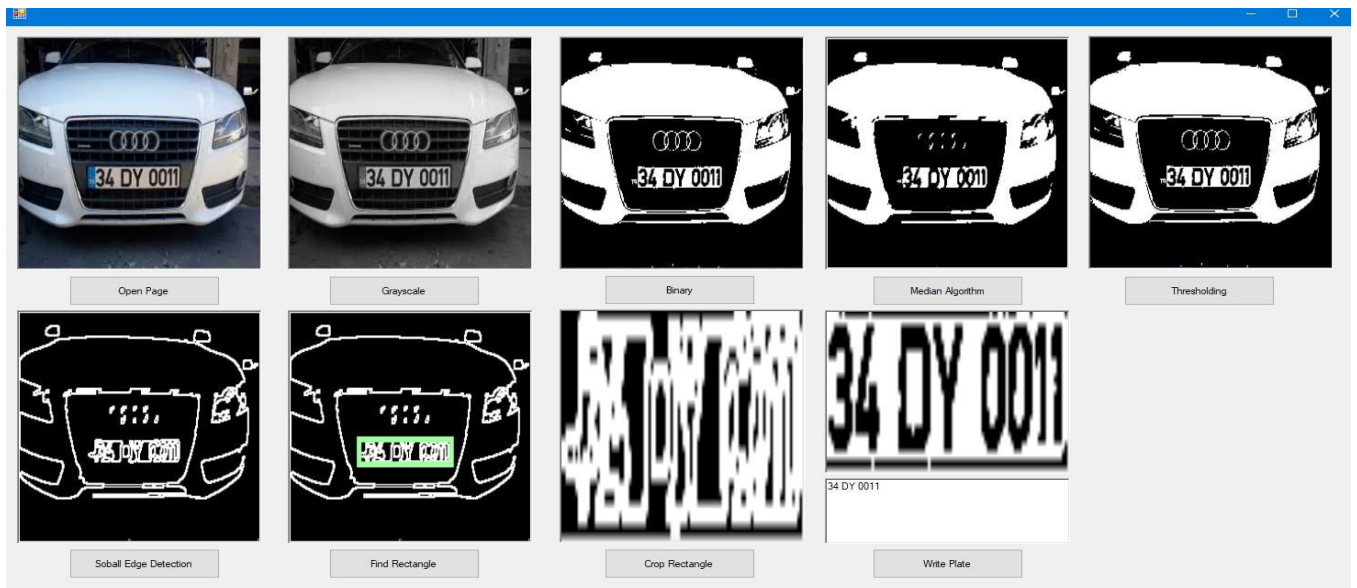
shereen99mo software proje	Latest commit f86b15e 1 hour ago
AdemProje	software proje 1 hour ago
Proje_deneme	software proje 1 hour ago
LICENSE PLATE RECOGNITION SYSTEM.pdf	software proje 1 hour ago

5.CONCLUSION AND SUGGESTIONS

In this study, a license plate recognition system was developed using image processing techniques. Unlike previous studies, all algorithms have been rewritten in the software using C # programming language instead of ready systems. Tesseract is used to read the plates in this software. When the license plate recognition results were examined, it was confirmed by the applications that the plates were improved in reading speed and accuracy rates. The correct reading of the plates with 88.1%, with 98% success, supports this situation.

The study has been kept open to improvement, as it is in a position to perform basic image processing techniques. In subsequent studies, the success rate in character recognition and general success rates can be increased by using artificial intelligence methods in reading characters.





6.REFERENCE

GITHUB: <https://github.com/shereen99mo/sheren>

1. <https://medium.com/@ilkbaharnaz/ocr-optical-character-recognition-optik-karakter-tan%C4%B1ma-268593b6e284>
2. <https://tekrei.gitlab.io/presentations/2003-Goruntu-Isleme-Sunum.pdf>
3. https://tr.qwe.wiki/wiki/Optical_character_recognition
4. <https://www.sciencedirect.com/topics/computer-science/optical-character-recognition>
5. <http://udentify.co/blog/04/2017/goruntu-isleme-nedir/>
6. <http://acikarsiv.aydin.edu.tr/xmlui/bitstream/handle/11547/1950/418502.pdf?sequence=1&isAllowed=y>
7. <http://earsiv.halic.edu.tr/xmlui/bitstream/handle/20.500.12473/1729/233456.pdf?sequence=1>
8. https://www.youtube.com/watch?v=L6o_uG6KIhk
9. <https://www.youtube.com/watch?v=zd0raNsf1Io>
10. <https://www.youtube.com/watch?v=rcLScYitp-U>
11. http://www.ibrahimcayiroglu.com/Dokumanlar/GoruntuIsleme/Goruntu_Isleme_Ders_Notlari-5.Hafta.pdf
12. <https://www.codeproject.com/Articles/139628/Detect-and-Track-Objects-in-Live-Webcam-Video-Base>
13. <https://docplayer.biz.tr/53163025-Goruntu-isleme-3-hafta.html>