Vehicle Number Plate Identification

Submitted by

Deependra Raliya, 202111023

Devesh Sharma, 202111024

Didhitkumar Patel, 202111025

Prachi Gurav, 202111064

S Pranav, 202111074

Under the guidance of

Dr. Jignesh Patel

Assistant Professor

IIITV-ICD

Indian Institute of Information Technology Vadodara – International Campus Diu, India

December 14, 2023

Contents

1 Abstract 1

2 Introduction 1

2.1 Background 1

2.2 Motivation 1

3 Objective 2

4 Literature Review 2

4.1 Canny Edge Detector 2

4.2 Morphological operations 3

5 Methodology 3

5.1 Image input 3

5.2 Pre-processing 4

5.3 Identification 5

6 Results 6

7 Discussion 7

7.1 Applications 7

7.2 Limitations 7

8 Future work 7

9 Conclusions 8

References 8

1 Abstract

The purpose of this project is to identify a car's license plate from a photograph of the vehicle or while it is stationary. The primary objective is to accurately extract and recognize specific characters from license plates using MATLAB software so that the owner of the car can be identified by subsequent comparison with the licensing database. The project's scope is restricted to number plate character recognition and image processing. By addressing the specific challenges associated with stationary vehicle identification, this project contributes to advancements in automated license plate recognition systems.

2 Introduction

Number plate detection method scans the input image to identify and locate the license plate in the image. Since a plate can exist anywhere in an image with various sizes, it is not possible to check every pixel of the image to locate it[1]. The number plate of the vehicle is automatically recognized and identified by comparing it with a database. It is difficult to detect the boundary of the Number plate from the image of the car in an outdoor scene due to the color of characters and background on the license plate. The process of locating the number plate consists of steps like Edge Detection, Morphological operations, noise filtration, character segmentation, and template matching.

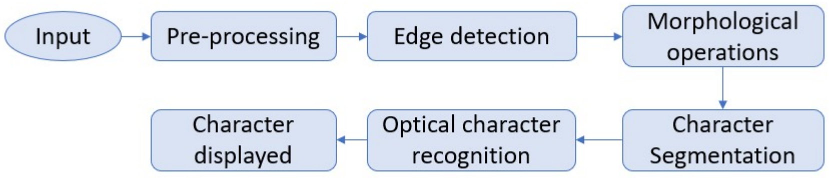


Figure 1: Process Flowchart

2.1 Background

The need for efficient and accurate license plate identification systems has gained prominence in various applications, including law enforcement, parking management, and access control[2]. Traditional methods often rely on manual inspection, which can be time-consuming and prone to mistakes. The ability to create automated systems that can improve the identification process is presented by the development of advanced imaging technologies and image processing techniques.

The project makes use of MATLAB's features to solve problems regarding stationary vehicle scenarios, guaranteeing reliable performance in diverse environmental conditions. By focusing on stationary vehicles, the research aims to overcome specific challenges such as variations in lighting, angles, and environmental factors.

2.2 Motivation

The reason we started this research is to improve the effectiveness and precision of license plate recognition for stationary cars. Automated recognition systems provide promise for reducing law enforcement agencies' burden and improving metropolitan areas' overall security. Identifying individual characters on license plates makes it possible to identify the owner later on, which helps with activities like traffic enforcement, security monitoring, and smooth database integration.

3 Objective

Upon completion of the project, the characters from the number plate should be recognized and displayed. The recognized characters should be correct, and the goal is to keep the process as simple as possible.

4 Literature Review

4.1 Canny Edge Detector

Edge detection is a computer vision technique used to identify boundaries or edges within an image. The goal is to highlight areas where there are significant changes in intensity or color, which often correspond to the boundaries of objects within the image, such as characters on a license plate. The Canny edge detector is a multi-stage algorithm involving Gaussian smoothing, gradient calculation, non-maximum suppression, and edge tracking by hysteresis.It is widely used for its ability to detect edges accurately while minimizing false positives.

Process of Canny edge detection:

1. Gaussian Smoothing: The input image is convolved with a Gaussian filter. This step helps to reduce noise and unwanted details in the image, creating a smoothed version.

2. Gradient Calculation: The gradient of the image is calculated using gradient operators (typically Sobel operators). This step computes the intensity gradients in both the horizontal and vertical directions.

3. Non-Maximum Suppression: The algorithm examines each pixel in the gradient image and suppresses (sets to zero) the values of pixels that are not local maxima along the direction of the gradient. This step helps to thin down the edges and retain only the most prominent ones.

4. Edge Tracking by Hysteresis: Two thresholds, a low threshold and a high threshold, are used to categorize pixels as strong, weak, or non-edge pixels. If a pixel's gradient magnitude is higher than the high threshold, it is considered a strong edge pixel. Pixels with values between the low and high thresholds are classified as weak edges. Weak edges are included in the final edge map only if they are connected to strong edges. This step helps to eliminate weak, spurious edges while retaining strong, continuous edges.

The result is a binary edge map where pixels corresponding to edges are set to white, while non-edge pixels are set to black. This edge map highlights the boundaries of objects in the image, making it easier to identify and segment characters on a license plate[3].

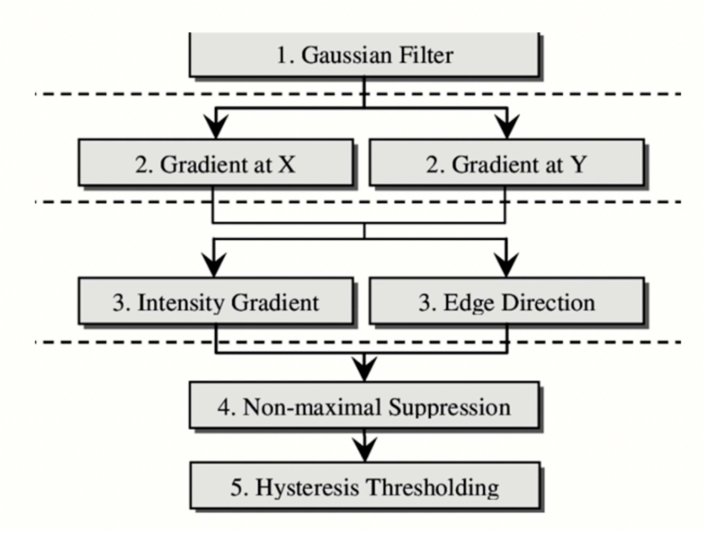


Figure 2: Canny Edge Detection Algorithm

4.2 Morphological operations

1. Erosion: It is a morphological operation that involves the shrinking or thinning of objects in a binary image. The operation is typically applied to binary images, where pixels are either black (background) or white (foreground)[4].

   - Noise Reduction: Erosion can be employed to eliminate small noise or unwanted details in the binary image. For example, if there are small isolated white pixels or speckles that do not contribute to the structure of characters, erosion can help remove them.
   - Character separation: Erosion may be used to separate characters that are close to each other. By carefully choosing the size and shape of the structuring element, erosion can break the connections between characters without excessively shrinking the characters themselves.

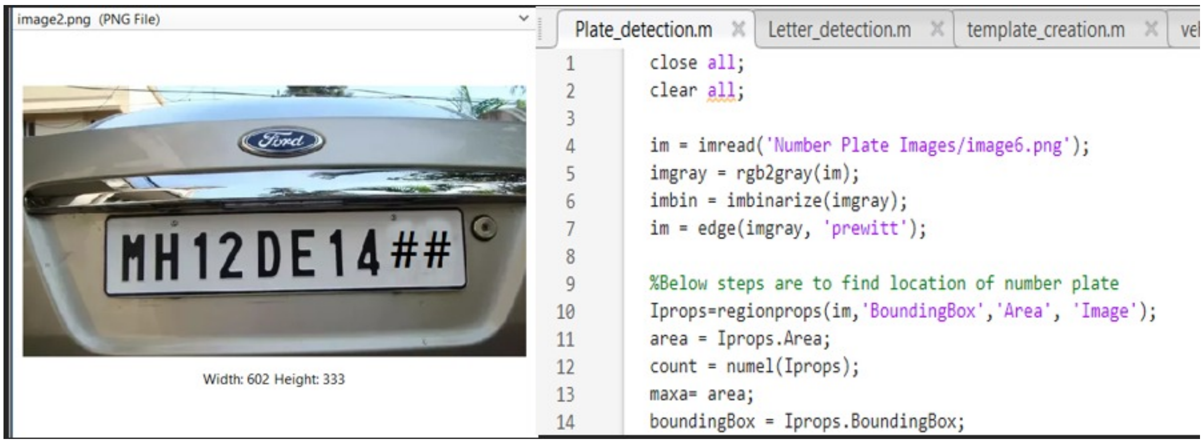
2. Dilation: Dilation involves the expansion or thickening of objects in a binary image. It is useful for tasks such as filling gaps in objects, connecting broken parts, or enlarging features of interest[5]. The result of dilation is a new binary image where the foreground objects have been dilated or expanded based on the structuring element.

   - Connecting Broken Parts: Dilation can help connect broken or fragmented parts of characters, making them more complete and easier to recognize.
   - Enlarging Features: Dilation can be applied to enhance the overall structure of characters, making them more prominent and facilitating subsequent steps such as character segmentation.

5 Methodology

5.1 Image input

For this project, an online image of a license plate was used as an example. To display the image, we must first give it as input into MATLAB. Figure 3 shows the code and the result.



```matlab
close all;
clear all;

im = imread('Number Plate Images/image6.png');
imgray = rgb2gray(im);
imbin = imbinarize(imgray);
im = edge(imgray, 'prewitt');

%Below steps are to find location of number plate
Iprops=regionprops(im,'BoundingBox','Area', 'Image');
area = Iprops.Area;
count = numel(Iprops);
maxa= area;
boundingBox = Iprops.BoundingBox;
```

Figure 3: Code for reading input

5.2 Pre-processing

We employ a variety of techniques in pre-processing to enhance the image. Reduce noise, boost contrast, and transform the picture into a format that is easier to calculate.

1. Converting from RGB to Grayscale: Converting an image to grayscale serves to diminish the impact of luminance while highlighting the image's features, making the darker values more pronounced and distinct.

2. Converting Grayscale to Binary: After transforming the image into black and white, the next step involves converting it into a binary format. This binary conversion is frequently employed to differentiate objects within the image from both the background and other elements. We utilized the imbinarize method, which employs thresholding to transform images into binary by substituting all values surpassing a universally determined threshold with 1s, while assigning all other values as 0s.

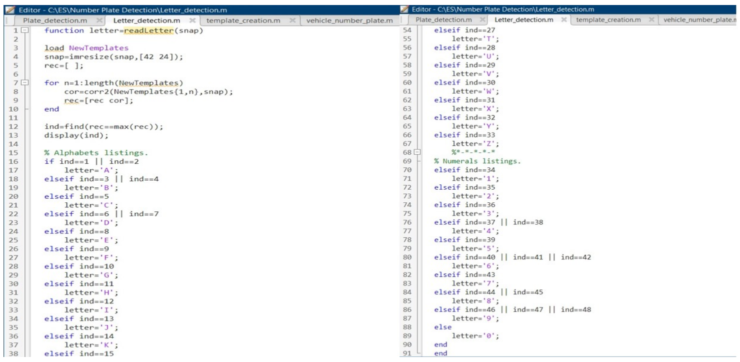
3. Edge detection: The application of Edge Detection is effective in minimizing data within the image while preserving its inherent properties for subsequent processing stages. We employed the Edge Detection method to identify and pinpoint distinct and sharp changes in the image. The Canny operator was selected as the most suitable edge detector for this purpose.

4. Morphological Operations: Morphological operations involve image processing based on shapes, with Erosion and Dilation being fundamental techniques. Dilation, a process of augmenting pixels along the boundaries of objects within the neighborhood of each input pixel, is employed to enhance the thickness of edges, particularly useful in augmenting the edges of number plates.

5. Character Segmentation: Following the dilation phase, the subsequent stage involves segmenting all the characters while retaining their distinctive features. A crucial aspect in character recognition is the Segmentation process. The MATLAB toolbox includes a function known as regionprops(), which measures a set of properties for each labeled region in the label matrix. This function also gauges the properties of the image region through the bounding box. This approach is applied to validate numbers using a template in the template matching algorithm within Optical Character Recognition (OCR)[6].

5.3 Identification

1. Letter Detection: A function has been developed to extract alphanumeric information from the input image. Subsequently, the saved template is loaded, and the function compares the input image with each template image to identify a match.

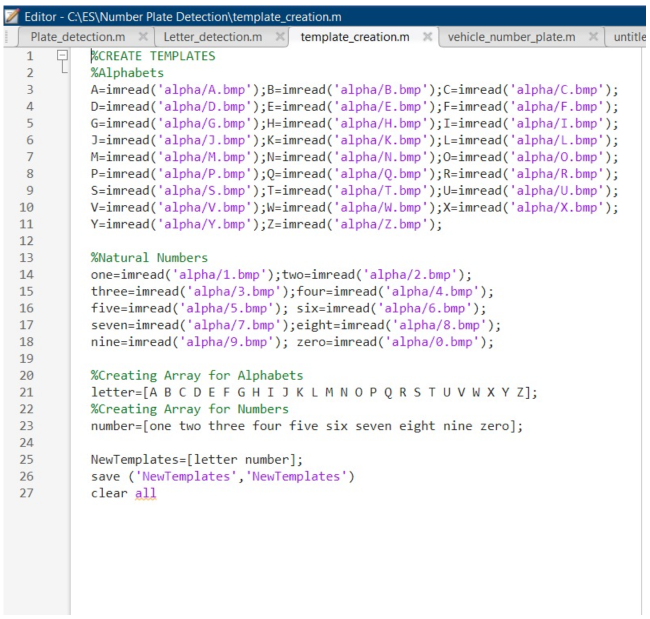


```
function letter=readLetter(snap)

load NewTemplates
snap=imresize(snap,[42 24]);
rec=[ ];

for n=1:length(NewTemplates)
    cor=corr2(NewTemplates{1,n},snap);
    rec=[rec cor];
end

ind=find(rec==max(rec));
display(ind);

% Alphabets listings.
if ind==1 || ind==2
    letter='A';
elseif ind==3 || ind==4
    letter='B';
elseif ind==5
    letter='C';
elseif ind==6 || ind==7
    letter='D';
elseif ind==8
    letter='E';
elseif ind==9
    letter='F';
elseif ind==10
    letter='G';
elseif ind==11
    letter='H';
elseif ind==12
    letter='I';
elseif ind==13
    letter='J';
elseif ind==14
    letter='K';
elseif ind==15
```

```
elseif ind==27
    letter='T';
elseif ind==28
    letter='U';
elseif ind==29
    letter='V';
elseif ind==30
    letter='W';
elseif ind==31
    letter='X';
elseif ind==32
    letter='Y';
elseif ind==33
    letter='Z';
    %*-*-*-*-*
% Numerals listings.
elseif ind==34
    letter='1';
elseif ind==35
    letter='2';
elseif ind==36
    letter='3';
elseif ind==37 || ind==38
    letter='4';
elseif ind==39
    letter='5';
elseif ind==40 || ind==41 || ind==42
    letter='6';
elseif ind==43
    letter='7';
elseif ind==44 || ind==45
    letter='8';
elseif ind==46 || ind==47 || ind==48
    letter='9';
else
    letter='0';
end
end
```

Figure 4: Code for letter detection

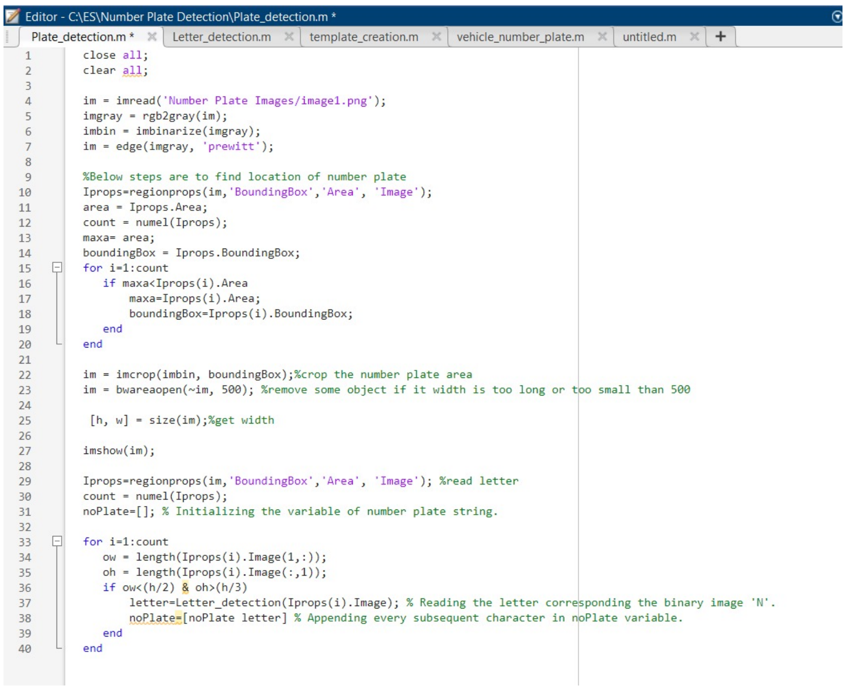
2. Template Matching: The transformed image of the license plate is employed to cross-reference each character with the comprehensive alphanumeric database through template matching. In this process, the template image is systematically positioned across various locations in a larger source image, and a numerical index is calculated to assess the degree of similarity between the template and the image in that specific position. Matching occurs on a pixel-by-pixel basis.



```
%CREATE TEMPLATES
%Alphabets
A=imread('alpha/A.bmp');B=imread('alpha/B.bmp');C=imread('alpha/C.bmp');
D=imread('alpha/D.bmp');E=imread('alpha/E.bmp');F=imread('alpha/F.bmp');
G=imread('alpha/G.bmp');H=imread('alpha/H.bmp');I=imread('alpha/I.bmp');
J=imread('alpha/J.bmp');K=imread('alpha/K.bmp');L=imread('alpha/L.bmp');
M=imread('alpha/M.bmp');N=imread('alpha/N.bmp');O=imread('alpha/O.bmp');
P=imread('alpha/P.bmp');Q=imread('alpha/Q.bmp');R=imread('alpha/R.bmp');
S=imread('alpha/S.bmp');T=imread('alpha/T.bmp');U=imread('alpha/U.bmp');
V=imread('alpha/V.bmp');W=imread('alpha/W.bmp');X=imread('alpha/X.bmp');
Y=imread('alpha/Y.bmp');Z=imread('alpha/Z.bmp');

%Natural Numbers
one=imread('alpha/1.bmp');two=imread('alpha/2.bmp');
three=imread('alpha/3.bmp');four=imread('alpha/4.bmp');
five=imread('alpha/5.bmp'); six=imread('alpha/6.bmp');
seven=imread('alpha/7.bmp');eight=imread('alpha/8.bmp');
nine=imread('alpha/9.bmp'); zero=imread('alpha/0.bmp');

%Creating Array for Alphabets
letter=[A B C D E F G H I J K L M N O P Q R S T U V W X Y Z];
%Creating Array for Numbers
number=[one two three four five six seven eight nine zero];

NewTemplates=[letter number];
save ('NewTemplates','NewTemplates')
clear all
```

Figure 5: Code for template matching

3. Plate detection: A comprehensive code was developed to identify and extract the region containing the license plate, aiming for more efficient and improved image processing.



```
close all;
clear all;

im = imread('Number Plate Images/image1.png');
imgray = rgb2gray(im);
imbin = imbinarize(imgray);
im = edge(imgray, 'prewitt');

%Below steps are to find location of number plate
Iprops=regionprops(im,'BoundingBox','Area', 'Image');
area = Iprops.Area;
count = numel(Iprops);
maxa= area;
boundingBox = Iprops.BoundingBox;
for i=1:count
   if maxa<Iprops(i).Area
       maxa=Iprops(i).Area;
       boundingBox=Iprops(i).BoundingBox;
   end
end

im = imcrop(imbin, boundingBox);%crop the number plate area
im = bwareaopen(~im, 500); %remove some object if it width is too long or too small than 500

 [h, w] = size(im);%get width

imshow(im);

Iprops=regionprops(im,'BoundingBox','Area', 'Image'); %read letter
count = numel(Iprops);
noPlate=[]; % Initializing the variable of number plate string.

for i=1:count
   ow = length(Iprops(i).Image(1,:));
   oh = length(Iprops(i).Image(:,1));
   if ow<(h/2) & oh>(h/3)
       letter=Letter_detection(Iprops(i).Image); % Reading the letter corresponding the binary image 'N'.
       noPlate=[noPlate letter] % Appending every subsequent character in noPlate variable.
   end
end
```

Figure 6: Code for Plate detection

6 Results

Different number plate images were used, and results were analyzed. The results from different images helped us to understand the limitations of the program. Figure 7 illustrates successful working of code, but Figure 8 shows that our program failed to identify number plate for given image due to

1. Automatic cropping of the image
2. Number plate in input image in blurred.

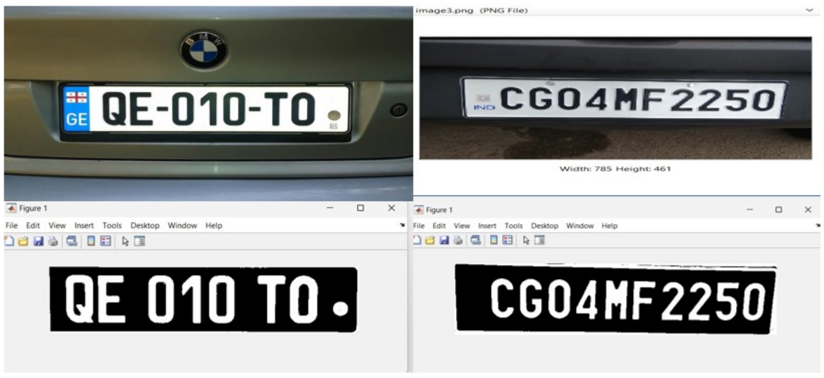


Figure 7: Identified number plate from 2 different images

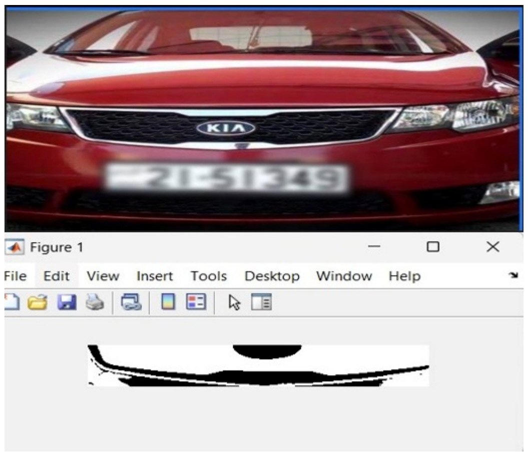


Figure 8: Result of blurred image

7 Discussion

7.1 Applications

1. Parking management: The automatic recognition of license plates made possible by LPR streamlines the parking permit process. For vehicles with authorization, this program speeds up the entry and exit processes.

2. Community Safety: To restrict access to registered vehicles, enhancing community security.

3. Toll Stops: By accurately capturing and identifying license plates, the system verifies the authenticity of each vehicle, thus minimizing the risk of fraudulent activities at toll booths.

4. Border Security: It can be used for registering vehicles entering or leaving a country.

7.2 Limitations

1. The system may face challenges in detecting number plates written in various languages or containing different symbols if they are not pre-existing in the database.

2. A broken number plate cannot be detected.

3. Low resolution or blurry number plates may not get recognized.

4. Inadequate maintenance of number plates may result in character similarities, such as the potential confusion between characters like '5' and '8', or '0' and 'D'.

8 Future work

- Multilingual Support: While the current system focuses on English license plates, future work could extend the recognition capabilities to accommodate different languages and symbols. This would involve expanding the database and optimizing the recognition algorithms to handle diverse character sets.

- Broken plate detection: Implementing a mechanism to identify and handle broken or damaged number plates would further enhance the robustness of the system. This involves developing algorithms capable of recognizing and processing fragmented or partially obscured plates.
- Resolution and Blur Handling: Addressing the limitation related to low-resolution or blurry number plates is essential. Future efforts could involve incorporating advanced image processing techniques or exploring machine learning models to improve recognition accuracy under challenging image quality conditions.

9 Conclusions

To monitor or identify a vehicle by recognizing its number plate, a vehicle number plate recognition system can be very beneficial. The program worked accurate and gave desired output for fixed image style. It has a huge potential for improvement and making it precise.

References

[1] R. Bhat and B. Mehandia, "Recognition of vehicle number plate using matlab," *International journal of innovative research in electrical, electronics, instrumentation and control engineering*, vol. 2, no. 8, pp. 1899–1903, 2014.

[2] B. Tiwari, A. Sharma, M. G. Singh, and B. Rathi, "Automatic vehicle number plate recognition system using matlab," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN*, pp. 2278–2834, 2016.

[3] D. C. MAVUZER, "Canny edge detection algorithm with python," 2022. [Online]. Available: https://medium.com/@ceng.mavuzer/canny-edge-detection-algorithm-with-python-17ac62c61d2e

[4] ağmur Çiğdem Aktaş, "A comprehensive guide to image processing: Part 3," p. 29 pages, 2021. [Online]. Available: https://towardsdatascience.com/image-processing-part-3-dbf103622909#:~:text=Erosion%20removes%20pixels%20on%20object,on%20the%20same%20input%20image.

[5] MathWorks. (Year of the documentation (e.g., 2023)) Matlab documentation: Morphological dilation and erosion. [Online]. Available: https://in.mathworks.com/help/images/morphological-dilation-and-erosion.html

[6] J. A. Mayan, K. A. Deep, M. Kumar, L. Alvin, and S. P. Reddy, "Number plate recognition using template comparison for various fonts in matlab," in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2016, pp. 1–6.