CS620: Final Project Report

Elana Lapins

Due: December 18th, 2022

License Plate Recognition Program

Abstract

License Plate recognition plays an important role in society and automates a redundant task for humans. In turn, this program can save everyone time, effort, and resources. This final project aims to create a license plate identification program that can identify and recognize license plate information in different images. License Plate detection is widely used, and selected methods from the literature were used to write this MATLAB program. The main steps of this program include image pre-processing, area filtering, character detection, and character recognition. These steps ultimately produce a text result in the command window that matches the license plate in the image. Results from this program suggest success but also areas for improvement. Out of 10 license plate images tested, the program was able to accurately determine about 7.5 out of the ten license plate images correctly. A pre-registration method may need to be implemented to improve results for images at an angle, as well as further pre-processing or filtering.

1. Introduction

1.1 Background Information

The goal of license plate recognition is to automatically extract information about the plate from a stand-alone image. This type of program can be used in law enforcement to capture cars that are speeding, running lights, or involved with a crime. Ideally, video surveillance captures may be run through such a program to track down information about the individual involved - or at least the car's owner.

To accomplish this, many different image processing and machine learning-based steps must be put in place. Figure 1 below demonstrates the various stages and techniques that may be used. The first step involves pre-processing the image data input. This may include conversion to grayscale, image thresholding, and extraction of contours to determine any characters in the image via edge detection methods. Areas of possible license plate matches will be further analyzed by investigating smaller areas surrounding these places in the image and their corresponding contours. Following this, a character recognition function will be implemented to determine which possible plate is the true match. The final license plate will be recognized, and the plate information will be extracted and printed for the user.

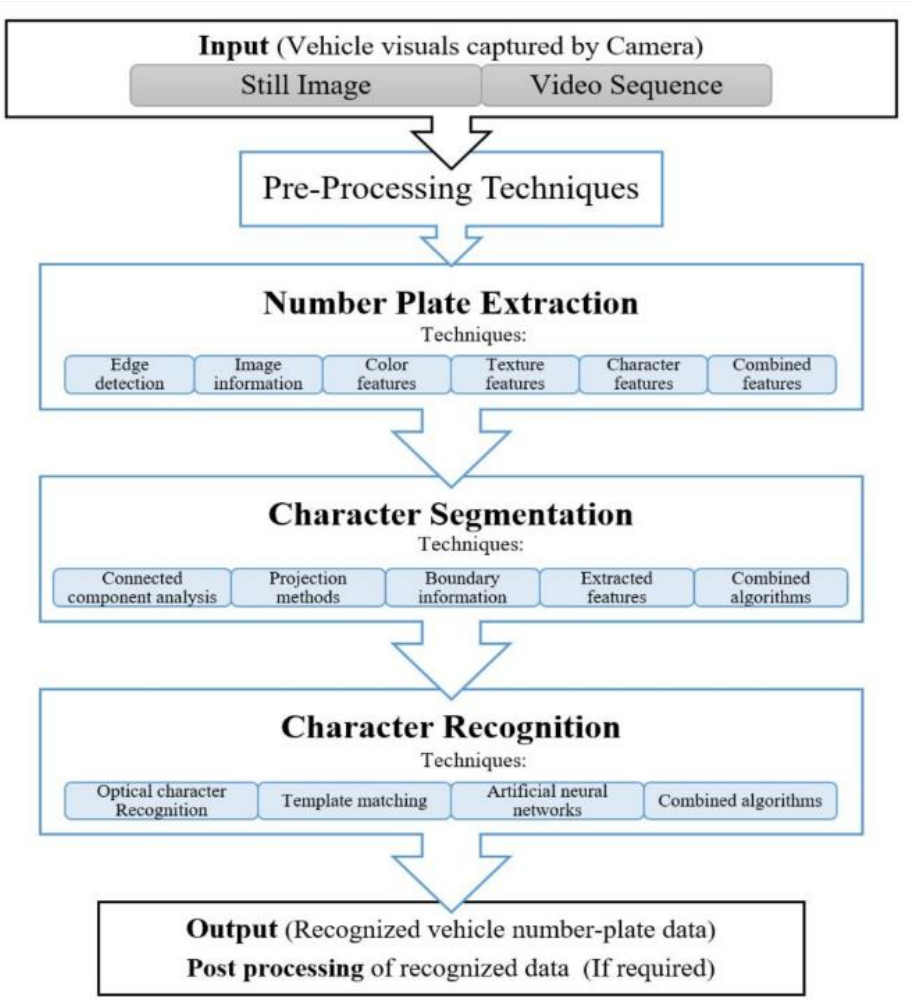

Fig.1 License Plate Recognition Diagram Outline from Gilly & Raimond 2013 [1]

The program presented in this paper is a much simpler version of what Figure 1 illustrates and is intended to be an initial version of a more extensive license plate detection system. This code uses similar pre-processing techniques, segments using boundary information, and applies template matching via correlation coefficient and training data.

Automatic detection and recognition of alphanumeric symbols may be beneficial for society. Many types of these programs are already in place and can save time, effort, and resources by automatically reading image data and its contents. Image processing in this area will constantly evolve, especially with the increased involvement of AI/Deep Learning, and can continually be improved.

1.2 Project Aims

This project aims to create a license plate detection program in MATLAB that can accurately detect and recognize alphanumeric characters in images of license plates.

2. Methods

2.1 Training Image Data

The training image data was obtained from an online source on Mathworks [2]. This data contains training images of alphanumeric symbols that are used to match characters in the license plate image data. Training datasets are very critical in achieving the best results in deep learning. While this training set only contains one image for each alphanumeric character, it could easily be improved by including multiple training images for each character. The training characters are shown in Appendix A. Each character image is set to a corresponding index which is how the recognition portion of the program works.

2.2 License Plate Images

Images used to test this program were obtained from google images and a dataset from *Kaggle* [3]. The dataset includes ten license plate images run through the program to test the recognition accuracy.

2.3 MATLAB License Plate Recognition Program

2.3.1 Pre-Processing

After loading the image data, the image was pre-processed to aid the primary processing process. Immediately after loading the data with the MATLAB *imread()* function, the images were re-sized to a 300x500 matrix. This is important, so all the image data is equally processed and ensures that thresholding via area later in the program will not be skewed depending on the image.

Fig.2 Original License Plate Image

In this case, all the images were loaded as RGB image data, so the first step was to convert these to grayscale. This was achieved by using the MATLAB function *rgb2gray()*. Following this, Otsu's thresholding method was used to convert the grayscale image to binary [9]. This method searches for a threshold value (t) that minimizes the following equation [4]:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

w_0 and w_1 are the weights of this equation. They scale the separation between the two classes. σ_0, σ_1 are the variances of these two classes that are separated by a certain threshold (t).

Both the grayscale and thresholded binary images are shown in Figure 3.

Fig.3 Grayscale and Binarized Image data. The binary image was created using Otsu's method.

2.3.2 Area Filtering

Following pre-processing, the background and noise were removed from the image through the use of the MATLAB function *bwareaopen()*. This function removes any recognized objects with an area less than a given value. All objects less than 100 pixels in area were removed from the image to reduce noise. The before and after are shown in Figure 4.

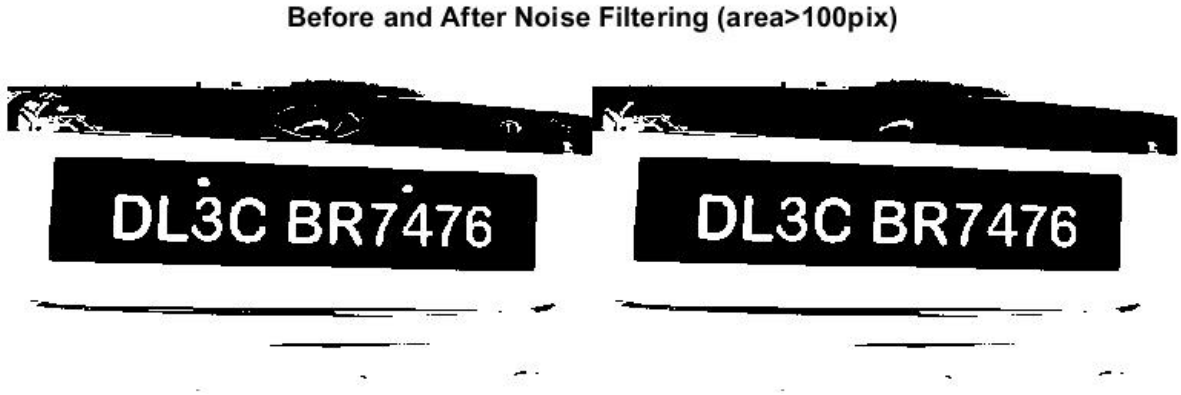

Fig.4 (Left) Binary image. (Right) After area/noise filtering using a value of 100 pixels.

To identify the background, a similar method was used except used a value of 3200 pixels for the area. The results from this background removal are shown in Figure 5.

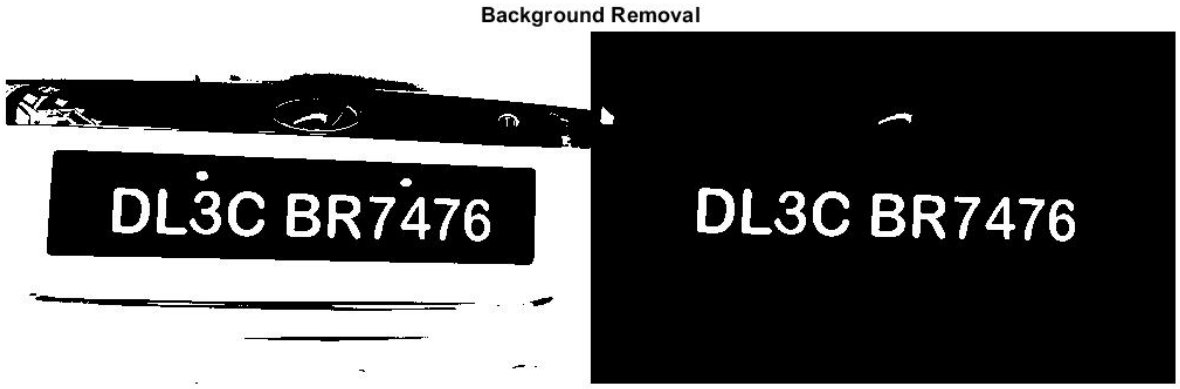

Fig.5 Background removal results. (Left) Original Image. (Right) Image without background.

One more area filter was applied to remove any smaller noise from the image without the background. All objects in the image with less than 250 pixels in the area were removed to produce a cleaner version of the image. Figure 6 represents the results from this filter.

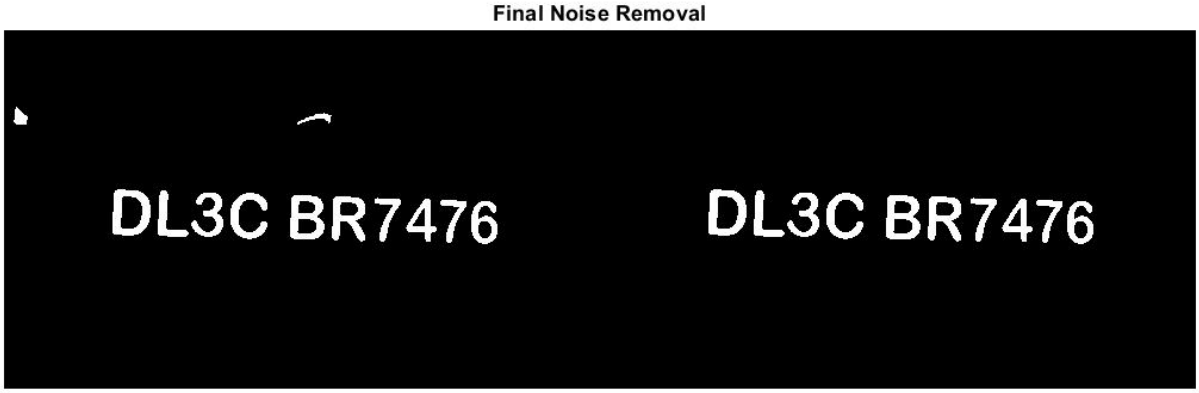

Fig.6 (Left) Image without background. (Right) Filtered image without a background.

The image on the right is the final image used for character detection and recognition.

2.3.3 Character Detection

Character candidates were determined using the MATLAB *bwlabel()* function. This function labels all objects in the image and provides connectivity information. It outputs labels for each object detected and the number of objects.

Information from this function was then used in *regionprops()* function to generate BoundingBox data for each character candidate. The bounding boxes are visualized in Figure 7 below.

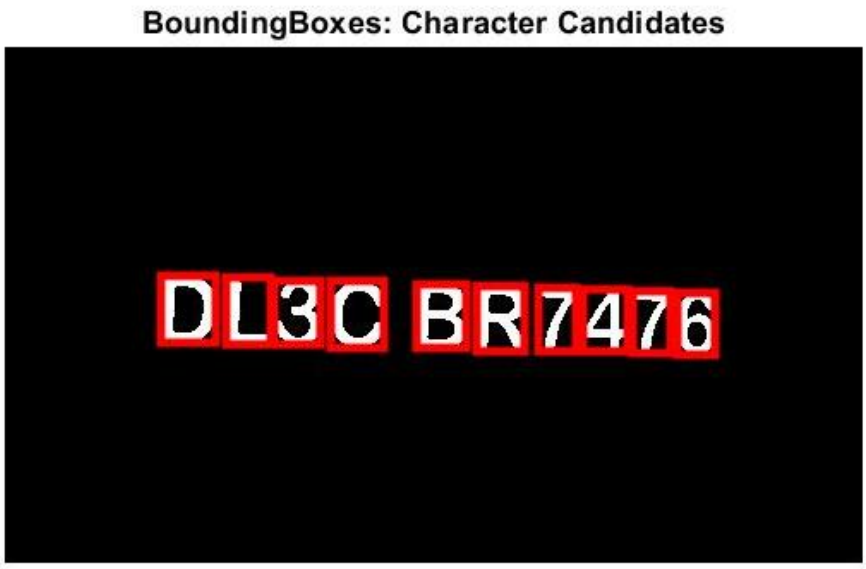

Fig.7 BoundingBoxes visualization around character candidates detected.

3.2.4 Character Recognition

The character candidates enclosed in the bounding boxes, shown above in Figure 7, were then used with the training data to calculate correlation coefficients. MATLAB function *corr2()* was used to calculate a 2-D correlation coefficient for each character and training symbols. The correlation coefficient was calculated using the following equation [5]:

$$r = \frac{\sum_m \sum_n (A_{mn} - \overline{A})(B_{mn} - \overline{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \overline{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \overline{B})^2\right)}}$$

where $\overline{A}$ = mean2(A), and $\overline{B}$ = mean2(B).

A: Character Candidate Image, B: Training Image of Alphanumeric Character

The method used above is often considered a type of template-matching technique. Correlation coefficients closer to 1 indicate a better match, and values closer to 0 mean that it is likely not a match. To determine the best match for each character, all correlation coefficients less than 0.5 confidence were neglected, and the maximum coefficient value was chosen as the best fit for the character candidate. This best fit was then indexed and selected as the final output for that given character. The final results for the entire plate were printed in the command window from left to right.

3. Results

The final results for each plate were displayed in a final figure with the original image. The results for all ten license plate images tested are shown below.

Fig.8 Final Results for ten license plate images tested in the MATLAB program

Out of the ten license plate images tested in the MATLAB program, approximately 7.5 out of 10 were correctly detected and recognized. Looking at individual character recognition accuracy, the program recognized 87.2% of characters in the ten images correctly. There were 86 characters in the ten images, and only 11 were identified incorrectly.

It appears that license plates that are not directly straight across have a more difficult time with processing. This may be fixed with a registration algorithm added to the code. Additionally, some extraneous characters are sometimes detected, so more filtering or pre-processing may be necessary to produce more accurate results.

4. Discussion

4.2 Project Conclusion

The aims of this project were successfully met through this MATLAB license plate detection program. 87.2% of the alphanumeric characters in the ten different images were recognized successfully. About 7.5 out of the ten plates were correctly identified. These results suggest that the program produces good results but that there is room for improvement.

4.3 Future Work

Images with incorrectly recognized characters seemed to be at an angle or not aligned directly across the image. A possible solution to this may include adding a pre-registration (skew-adjustment) step to the pre-processing section [8] or using more training images with characters at different orientations or scales.

The small number of training images used in this work was a significant limitation. A simple fix would be adding more training images for each alphanumeric character to improve the correlation coefficient between character candidates and the training candidates. Real-world applications of this type of algorithm will absolutely require more training data because images taken in nature are far from perfectly straight and will likely be at all sorts of orientations, scales, and resolutions. Adding more image processing to the image data before calculations plays a significant role in the accuracy of the results and can either make or break the program.

The MATLAB image processing toolbox has many different functions for use, but there may be another type of language or library that could produce better or faster results. A higher-level language, such as C++ or Python, may be used with a well-known image-processing library called OpenCV to reproduce or improve this work's results [6].

Another area for expansion in this work is introducing video data rather than image data. This would allow for real-time processing and could be applied in many scenarios in the real-world such as traffic control or surveillance.

4.4 Final Remarks

This MATLAB program successfully determined license plate characters from image data. There are many areas of this work for improvement that could be added with more time. Automatic recognition plays a significant role in society and is only growing more with AI and Deep Learning.

*The main code is attached in this zip folder, but also in the Appendix

5. References

[1] Gilly, D., & Raimond, K. (2013). A survey on license plate recognition systems. International Journal of Computer Applications, 61(6)

[2] Nishant Kumar (2022). License plate recognition (https://www.mathworks.com/matlabcentral/fileexchange/54456-licence-plate-recognition), MATLAB Central File Exchange.

[3] Larxel. (2020, June 01). Car License Plate Detection. https://www.kaggle.com/datasets/andrewmvd/car-plate-detection/code?resource=download

[4] Otsu's method. (2022, November 06). https://en.wikipedia.org/wiki/Otsu%27s_method

[5] MATLAB corr2. (n.d.). https://www.mathworks.com/help/images/ref/corr2.html

[6] OpenCV. (2022, December 14). https://opencv.org/

[7] Lee, E.R., Kim, P.K., & Kim, H. (1994). Automatic recognition of a car license plate using color image processing. *Proceedings of 1st International Conference on Image Processing, 2*, 301-305 vol.2.

[8] A. C. Roy, M. K. Hossen and D. Nag, "License plate detection and character recognition system for commercial vehicles based on morphological approach and template matching," 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), 2016, pp. 1-6, doi: 10.1109/CEEICT.2016.7873098.

[9] K. Yogheedha, A. S. A. Nasir, H. Jaafar and S. M. Mamduh, "Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach," 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), 2018, pp. 1-8, doi: 10.1109/ICASSDA.2018.8477639.

6.Appendix

MATLAB program

```
% CS6210: Final Project - License Plate Detection
% Elana Lapins
clear all; clc;

%Image data Location
direct = dir('plate images');
direct = direct(3:end);

load trainingdata; %obtained this training set online (in Alpha folder)
for i = 1:length(direct)
    filename = direct(i).folder + "\" + direct(i).name;%filename
    fprintf("Image: %s \n",direct(i).name);
    %load image data and resize so all images same size
    img = imread(filename);
    img = imresize(img,[300 500]); %rectangle plate shape
    %imshow(img)

    %1)pre-processing
    gray = rgb2gray(img); %grayscale image
    %imshow(gray)

    t = graythresh(gray);
    %t = adaptthresh(gray,0.8); %adaptive threshold value, 0.8 sensitivity
    bin = ~imbinarize(gray,t); %apply to image (now white and black image)
    %imshow(bin)

    %remove any noise less than 100 pixels in size
    no_noise = bwareaopen(bin,100);
    %imshowpair(bin,no_noise,'montage')

    %locate background
    background = bwareaopen(no_noise,3200);
    %remove background from image
    difference = no_noise - background;
    %imshow(difference)

    %remove smaller noise again
    difference2=bwareaopen(difference,260);
    %imshow(difference2)
    charCandidates = difference2;

    %visualize
%     figure;
%     imshow(background)
%     figure;
%     imshow(difference)
%     figure;
%     imshow(charCandidates)

    %locate objects remaining in image (char candidates)and determine
    %connectivity
    [L,n]=bwlabel(charCandidates);
    props=regionprops(L,'BoundingBox');
%     hold on
%     for ii=1:size(props,1)
%         rectangle('Position',props(ii).BoundingBox,'EdgeColor','r','LineWidth',3)
%     end
%     hold off
```

```
    %pre-allocate data structures
    final_output=[];
%     figure;
    %loop through characters and compare with template images (via corr2)
    for ii=1:n
        [xx,yy] = find(L==ii);%locate charcter position
        char=no_noise(min(xx):max(xx),min(yy):max(yy)); %index character from image
        char=imresize(char,[42,24]); %resize image to match template sizes
%         imshow(char)%display char

        %# of template letters/numbers
        num_temp=size(imgfile,2);
        coefs = zeros(1,num_temp);

        %loop through all letters/numbers and correlate to char
        for j=1:num_temp
            corr=corr2(imgfile{1,j},char);%calculate corr. coefficient
            coefs(1,j) = corr;%save coefficient
        end

        %only keep char's with coefficients > 0.5 confidence
        if max(coefs)>.50
            final_char=find(coefs==max(coefs)); %index the final chars
            output=cell2mat(imgfile(2,final_char));%index template value
            final_output=[final_output output]; %final results
        end
    end
    fprintf('Plate: %s\n',final_output);


    subplot(2,5,i)
    imshow(img)
    tt = "Result: " + final_output;
    title(tt)
    hold on

end

%close all;
```

Training Images