Automatic License Plate Detection

COMPENG 4TN4

Smit Patel

Tyler Lee

April 6, 2025

Automatic License Plate Recognition (ALPR) is a computer vision application widely used in traffic monitoring, law enforcement, and automated toll collection. This project presents a system for processing and extracting license plate characters from an image containing a vehicle's license plate. We attempt to use conventional image processing techniques in contrast to modern machine learning-based solutions. The system is divided into three stages: license plate localization, image preprocessing, and character segmentation and recognition.

Technical Discussion

The first stage of the system processes the input image to locate the license plate. The image is first converted from RGB to greyscale and binarized. This is done to reduce the number edges in the image, making it easier to identify the plate. Through experimentation, it was found that a fixed threshold of 0.5 performed better than using Otsu's method. The image is then passed through a Canny edge detector to locate the edges. Dilation is performed to connect nearby edges, creating connected regions. The ten largest regions by area are kept, while the rest are removed from the image.

Following this process, the perimeter and area of each region is compiled. A bounding box is also drawn around each region, with its starting location and dimensions saved. The system then calculates various metrics and uses them to filter out regions which do not match the expected characteristics of a license plate. These metrics include the ratios between the perimeter and area of each region with that of its bounding box, its location within the image, and its relative size. After filtering, there should be one region left, corresponding to the license plate. Its bounding box is overlaid on the original image, which is then shown to the user. Finally, the original image is cropped using the bounding box and sent to next stage.

The second stages of the system preprocess the license plate image for character recognition. The third and final stage extracts the characters from the license plate. Otsu's method was used to compute the global threshold for the grayscale image. A dynamic threshold is determined using software. Afterwards, a binary version of the image is created using the threshold and the grayscale image. This returns a binary image where the foreground (license plate characters) is in black and the background is in white. However, since the borders are black, it is necessary to take the complement of the image before clearing the borders.

A square structuring element is then used to erode the image to make the characters larger and more recognizable. The reason erosion is used instead of dilation is because the foreground is in black. Lastly, OCR is used to identify the characters. It treats the image as a single line of text and looks for capital letters as well as numbers.

Discussion of Results

The main objective of being able to identify the characters from an image of a car with the license plate in the image was achieved. This was proven through tests on the various test images used as a validation set for the project. However, there were some drawbacks.

The plate extraction failed on many images due to multiple factors. For example, with images that have poor lighting, the license plate would disappear after thresholding and thus could not be identified. In most cases, the location, orientation, and relative size of the plate would cause it to be filtered out by the system. This puts a limit on the types of images the system will work on.

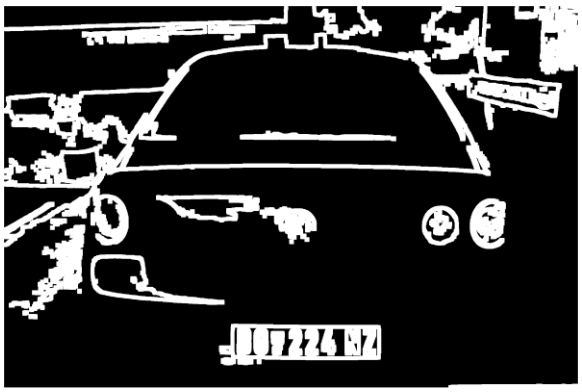
Additionally, as seen in test image 9 (a), there would occasionally be rectangular regions with similar properties to the license plate. In some of these instances, the actual plate would not be identifiable, which led to some images being unusable.

Although most of the images from stage 1 had identifiable license plates, there were some images within it that our algorithm was unable to identify the characters for properly. This was usually because of image resolution causing the Otsu method to return a threshold that didn't optimally binarize the image [ex: test image 8 (c)] or other character regions not part of the license plate characters being recognized [ex: the city name above license plate characters in test plate 7 (c)].

Overall, keeping in mind that we used conventional image processing techniques rather than modern machine learning-based solutions, the project was a success. A high degree of accuracy in determining the license plate characters was achieved and was done through many techniques taught throughout this course.

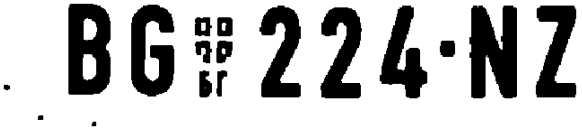
Results

Test image 1:

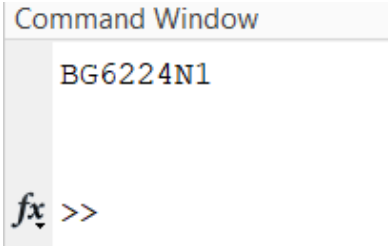


(a)

(b)

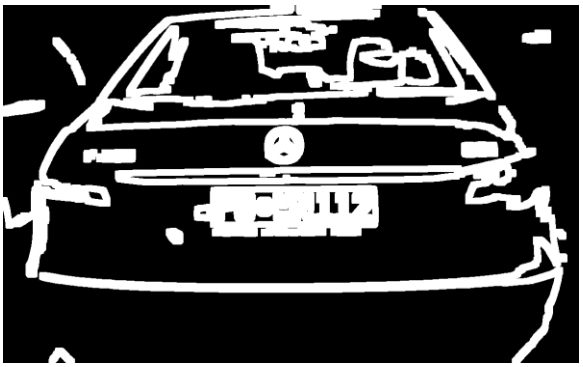


(c)



(d)

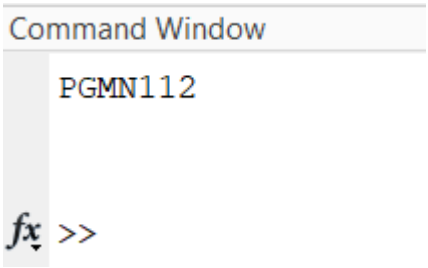
Test image 2:



(a)

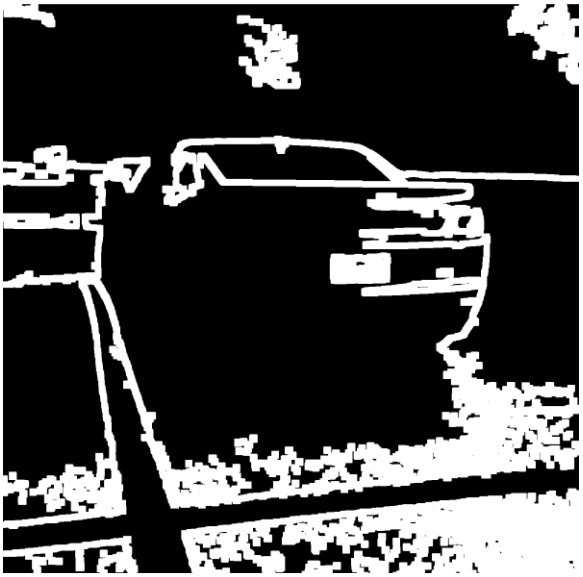
(b)

(c)



(d)

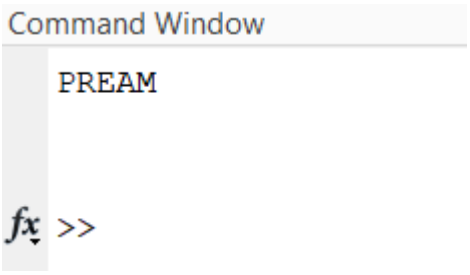
Test image 3:



(a)

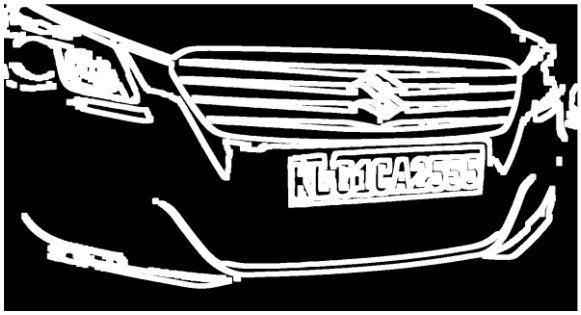
(b)

(c)



(d)

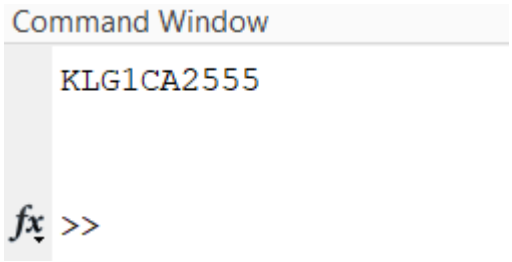
Test image 4:



(a)

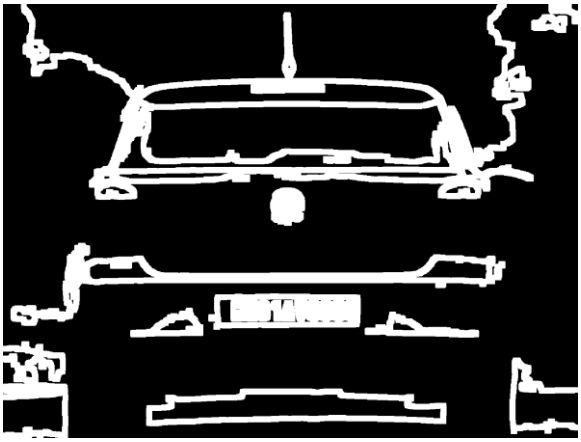
(b)

(c)



(d)

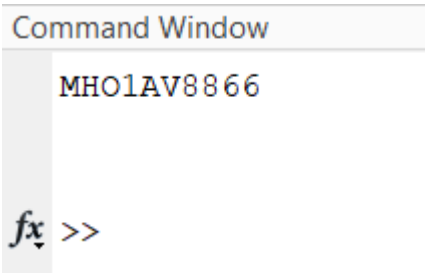
Test image 5:



(a)

(b)

(c)



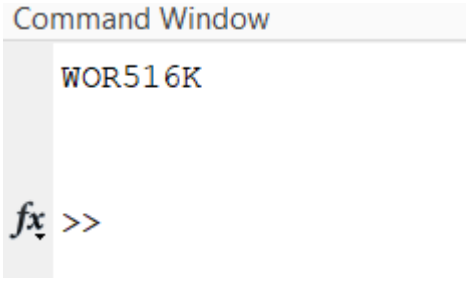
(d)

Test image 6:

(a)

(b)

(c)



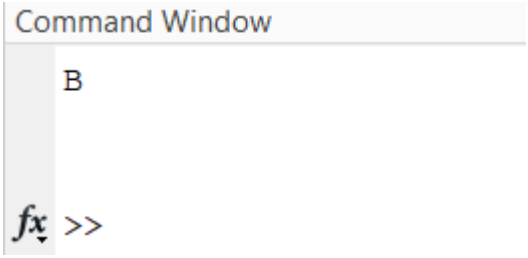
(d)

Test image 7:

(a)

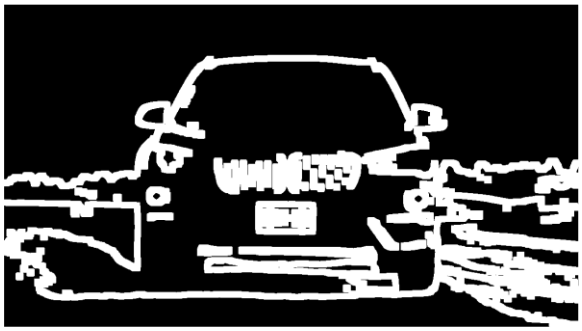
(b)

(c)



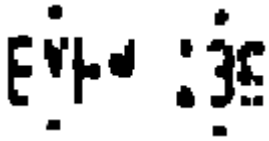
(d)

Test image 8:

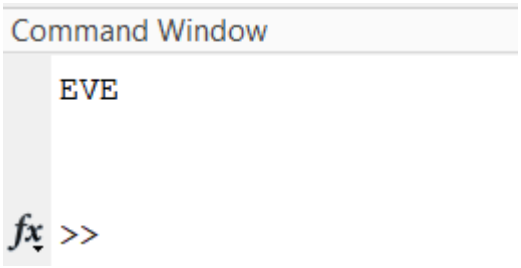


(a)

(b)

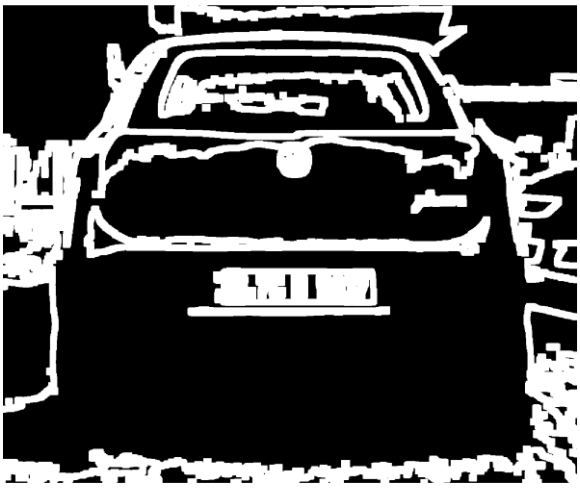


(c)



(d)

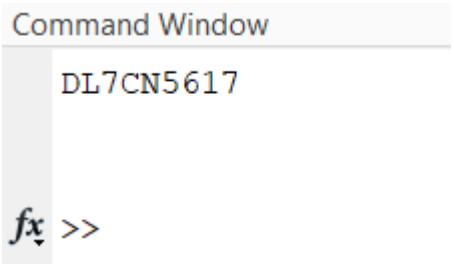
Test image 9:



(a)

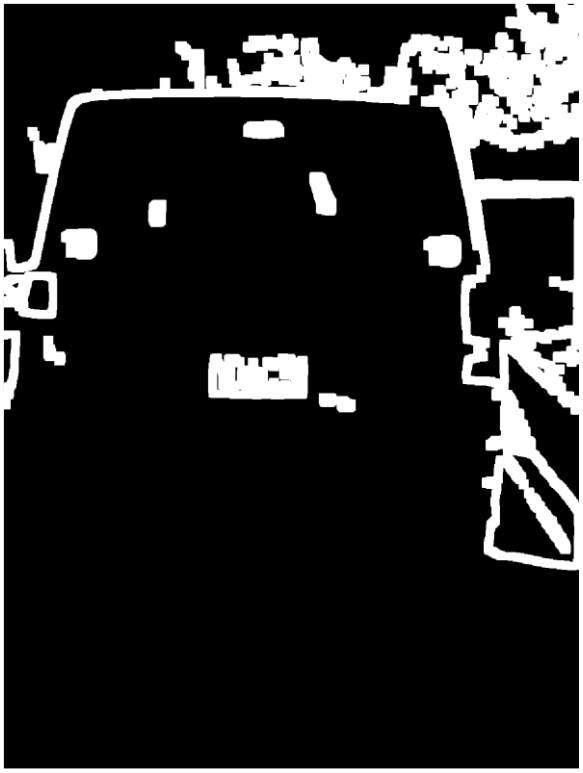
(b)

(c)



(d)

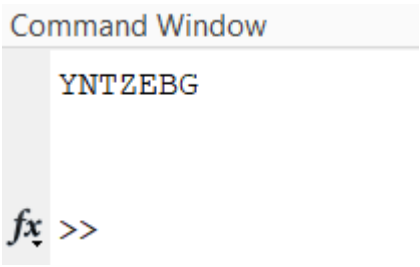
Test image 10:



(a)

(b)

(c)



(d)

Appendix

```
car = imread("Dataset\images\Cars11.png");
[bbox, cropped] = locatePlate(car);
text = readPlate(cropped);

imshow(bbox);
disp(text);

function [bbox_I, cropped_I] = locatePlate(car_image)
    scale_factor = 3; % scale at which the image and structuring element are upsampled

    % Resize the image
    original_I = imresize(car_image, scale_factor);

    % Record image size for later use
    [height, width, depth] = size(original_I);

    % Process the image to extract edges
    I = im2gray(original_I);
    I = imbinarize(I, .5);
    I = edge(I, "canny");

    % Dilate image to create connected regions from edges
    SE = strel('square', 1*scale_factor);
    for i = 1:6
        I = imdilate(I, SE);
    end

    % Keep only the 10 largest regions by area
    I = bwpropfilt(I, "Area", 10, "largest");

    % record the perimeter, area, and bounding box of each region
    stats = regionprops(I, {'Area','BoundingBox','perimeter'});
    stats = struct2table(stats);

```

```
        % Calculate various metrics and construct vectors stating which regions
        % fulfill each metric
        % 1st metric: ratio between perimeters of each region and its bounding box
        stats.Metric1 = 2*sum(stats.BoundingBox(:,3:4),2)./stats.Perimeter;
        idx1 = abs(1 - stats.Metric1) < 0.2;

        % 2nd metric: ratio between areas of each region and its bounding box
        stats.Metric2 = stats.Area./(stats.BoundingBox(:,3).*stats.BoundingBox(:,4));
        idx2 = stats.Metric2 > 0.5;

        % 3rd metric: ratio of bounding box length and width, it should be a rectangle
        stats.Metric3 = stats.BoundingBox(:,3)./stats.BoundingBox(:,4);
        idx3 = stats.Metric3 > 1.5 & stats.Metric3 < 5;

        % 4th metric: the starting point of the bounding box in the x direction must
        % be within the middle fifths of the image width
        stats.Metric4 = stats.BoundingBox(:,1);
        idx4 = stats.Metric4 > width/5 & stats.Metric4 < width*4/5;

        % 5th metric: the starting point of the bounding box in the y direction must
        % be below the top quarter of the image
        stats.Metric5 = stats.BoundingBox(:,2);
        idx5 = stats.Metric5 > height/4;

        % 6th metric: ratio between the region area and the total area of the image
        stats.Metric6 = stats.Area./(width*height);
        idx6 = stats.Metric6 > .005;

        % keep only the region(s) which meet all metrics and remove everything else
        % there should only be one
        idx = idx1 & idx2 & idx3 & idx4 & idx5 & idx6;
        stats(~idx,:) = [];

```

```
    % return the image with the bounding box overlaid
    bbox = stats.BoundingBox;
    bbox_I = insertShape(original_I,"rectangle",bbox,"LineWidth",6);

    % crop and return the bounded region
    cropped_I = imcrop(original_I, bbox);
end

function plate_text = readPlate(plateImage)
    grayImg = rgb2gray(plateImage);  % Convert to grayscale (if not already)

    threshold = graythresh(grayImg);
    binImg = imbinarize(grayImg, threshold); % Binarize the image

    binImg = imcomplement(binImg);
    binImg = imclearborder(binImg);  % Removes components with fewer than 50 pixels
    binImg = imcomplement(binImg);

    se = strel('square', 3);
    openImg = imerode(binImg, se);

    % Perform OCR.
    results = ocr(openImg, 'TextLayout', 'line', 'CharacterSet', 'ABCDEFGHIJKLMNOPQRSTUOVWXYZ0123456789');

    plate_text = results.Text;
end
```