Image Segmentation and Recognition of Vehicle Plates using Image processing techniques

Mohit P Sathyaseelan, Department of Electrical & Computer Engineering at the University of Florida, Gainesville

Abstract—**In this paper, we will be talking about the Image processing techniques used in VPR system (Vehicle plate recognition). In real time scenario we see that the images are really blurred and oriented in a different angle. Also, we observe that the light intensity affects the VPR output.**

Our main objective is to perform image segmentation, perform Image processing techniques and finally classification of the images. We will be using an OCR tool to read in the final processed image. Finally, we will compare our results to the current models being used.

Index Terms—***Image processing, VPR, Image segmentation, classification, OCR.***

I. Introduction

Vehicle plate Recognition system are widely being used in the current traffic system. The system works with videos as well as images. Incase of a speeding at a signal or when you cross the speed limit, VPR detects the vehicles plate number and reports it to the end User, which is the Traffic Police. The detection of the Vehicle plate should be precise and accurate; hence we need a good model for a VPR system.

There are many factors which might affect the model's accuracy. The first being the data, a good model performs better if the date is curated properly. The same case applies here, we will be applying image processing techniques to improve the model's accuracy in detecting the vehicle plate accurately.

The model for our project has three broad classes. First being detecting the license plate, Image segmentation and finally classifying and character recognition.

- **Plate Detection:**
  The Image which has to be detected has (x,y) pixels, out of which we only require the plates pixels. Observing this with our current model and data set, the ratio of useful data in our image is very less. Hence, we will be blurring the image using filters to blur the unwanted information from the input image.

.

  Our next goal is to detect the plate. We will be drawing contours in the image. Since all the unwanted information is being blurred in the image, we will get more defined contour of the plate. This provides us with the detected plate

- **Image Segmentation:**
  After the plate detection, we remove the undesired information and work with the plate image. We can easily remove this using crop function

- **Character recognition:**
  Most of the VPR systems have a different pipeline for character recognition. In this project I have used an OCR named Pytesseract to simplify and provide a more accurate reading.

II. Procedure flow

A. Plate detection:

First, we have converted our image from RGB to a gray scale image. This is done so that it simplifies the further image processing techniques used in the project. Next, we are going to remove the undesired data from the image that is the noise [1].
I have used Gaussian as well as Bilateral filter to check which provides a better output.

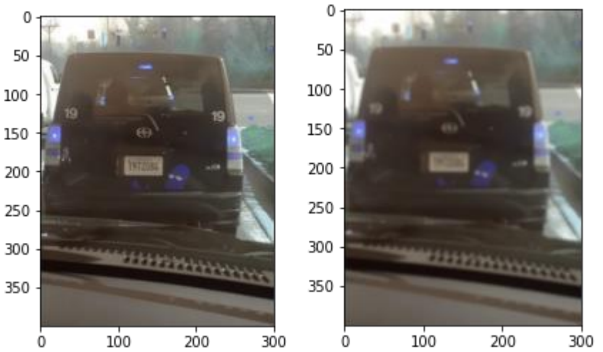


Fig.1 (a) Gaussian blur output (b) Bilateral Filter output

Bilateral filter gave more prominent output compared to the gaussian filter. This is because the bilateral filter is an edge-preserving filter and the prominent features in the image are saved. The filter also takes the weighted average of the nearby pixels, although not over the edges and hence the edge is preserved.

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})\, I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})$$

Where BF is the bilateral filter, Wp is the normalization factor. The filter largely depends on range parameter and spatial parameter.[2] In our project we are using OpenCV bilateral function which takes in diameter, sigma color and sigma space as inputs. Diameter is the diameter of each pixel in the neighborhood during filtering. These parameters are tested according to the accuracy of the model.

The output from bilateral filter helps in edge detection since the output blurs all the pixels in the image except for the edges. We are using OpenCV Canny function for edge detection. Canny edge includes 4 stage process:

- Noise reduction
- Calculating Intensity Gradient of Image
- Suppression of False Edges
- Hysteresis Thresholding.

As we have performed bilateral filtering, the noise is already removed. For the next part of Canny edge detection, the image is filtered with Sobel Kernel.

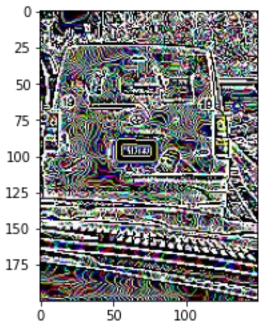


Fig.2 Sobel Kernel Output

The next step is suppression of false edges followed by thresholding. Here we have two parameters for threshold, i.e., maximum, and minimum threshold values. This helps us differentiate from borders of the prominent edges to the smooth pixels. Final output of the canny edge detection shown in fig.3

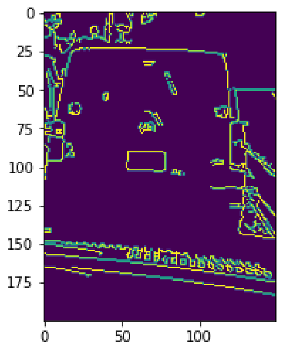


Fig.3 Output from Canny edge detection

Now we can start finding the contours from the output image passed from an edge detector. Here I have used the findContours function which detects all the contours in the image. The main reason we are finding contours is due to the vehicle plates dimensional property. If we look at fig, we see that the vehicle plate has more prominent structure than rest of the contours. Once we grab all the contours, we will loop through the first fifteen contours detected and check if they have rectangle properties to it. The contour is then highlighted and masked.
The main reason why we are masking, is so that segmentation becomes easier.

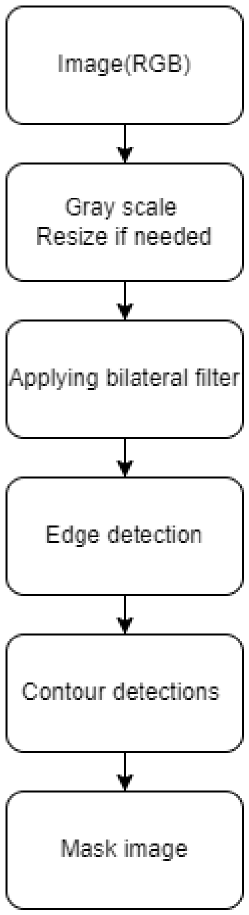


Fig.4 Plate detection pipeline

B. *Image Segmentation*

For image segmentation, we attain the x and y values from the masked image to identify (x, y) coordinates of the image.

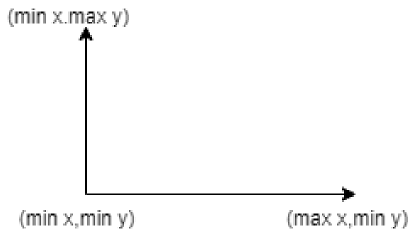


Fig.5 Cropping limits

Once we get the maximum and minimum, x and y coordinates we can crop the vehicle’s plate for Character recognition

C. *Character recognition*

For character recognition I can use a CNN or a pre-existing Optical Character recognition. In this project, I have used PyTesseract which was initially developed HP but currently it is owned by google. [3] Ray Smith who developed tesseract has explained how exactly the character recognition process works.

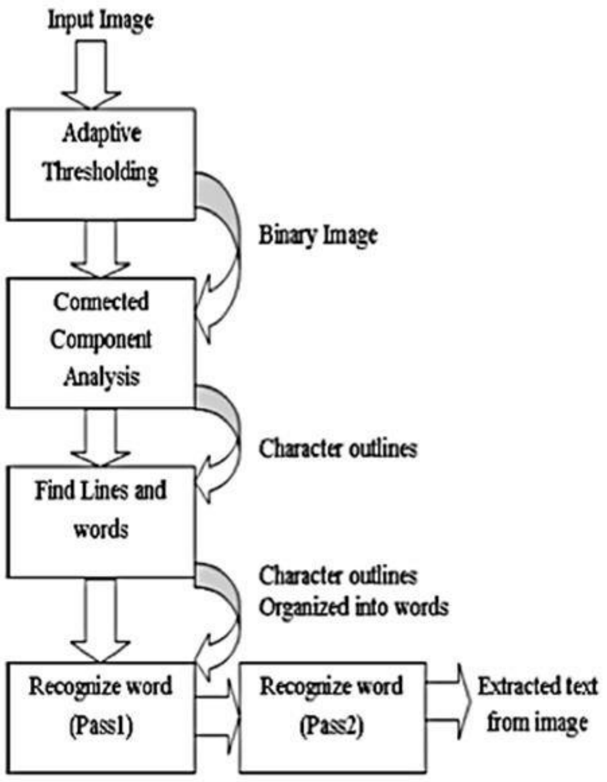


Fig.6 Pipeline of Tesseract [4]

Connected component analysis converts the input image into outlines by nesting. This is further converted into blobs. As the blobs are created, the tesseract engine identifies the text lines. The text lines provide us the spaces between each word. The model tries to identify the spacing patterns and separates the words. These words are further divided, and recognition process starts, which is a two-process stage. The words that are accurate in the first pass, moves to the adaptive classifier. Here the data is trained, and the classifier can recognize the words in the next process. The second pass is generally used since few data is passed on to the first pass as training data. Since this data is not accurately recognized it passes on to the second pass.

The PyTesseract finally does a good job in providing the Vehicle plate number as shown in fig

```
Detected Number is: | YNTZDBE |
```

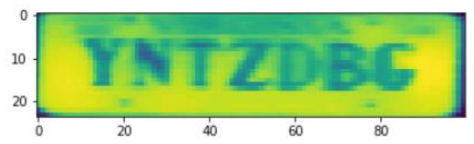
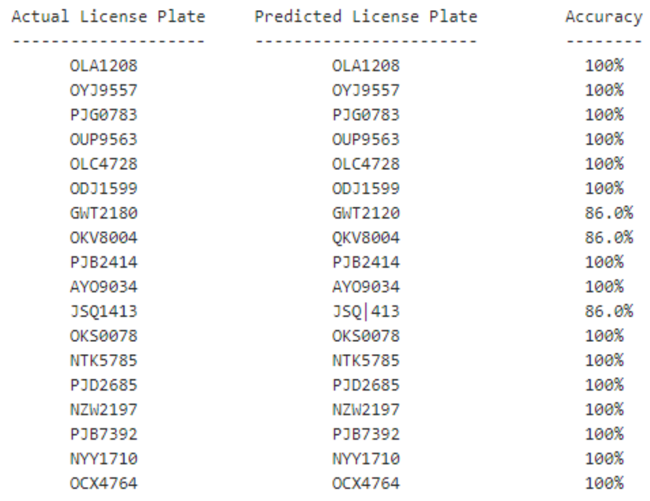


Fig.7 Output from the model

III. Evaluation

In our current model, we can improve the Character recognition by building a CNN model. This can be done for a specific data set. This was another reason why we implemented our model with tesseract. PyTesseract is a wrapper for Google’s Tesseract Engine, it has various languages prestored in the Engine which is useful for our project.

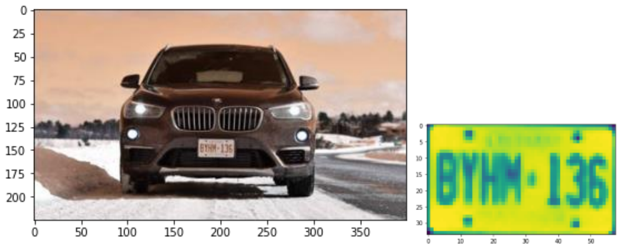
We can develop a CNN for the project but since we are targeting to read any vehicle plate, i.e., vehicles from different countries. This will be hard for the CNN model to be trained, because every country has its own font, color, and style of writing the number plate. Let’s say even if we do train a model with such a huge data set, Getting the output from the model will take more time than the Tesseract OCR. A VPRS model looks for a fast and accurate output.



```
Actual License Plate     Predicted License Plate        Accuracy
--------------------     -----------------------        --------
      OLA1208                  OLA1208                    100%
      OYJ9557                  OYJ9557                    100%
      PJG0783                  PJG0783                    100%
      OUP9563                  OUP9563                    100%
      OLC4728                  OLC4728                    100%
      ODJ1599                  ODJ1599                    100%
      GWT2180                  GWT2120                    86.0%
      OKV8004                  QKV8004                    86.0%
      PJB2414                  PJB2414                    100%
      AYO9034                  AYO9034                    100%
      JSQ1413                  JSQ|413                    86.0%
      OKS0078                  OKS0078                    100%
      NTK5785                  NTK5785                    100%
      PJD2685                  PJD2685                    100%
      NZW2197                  NZW2197                    100%
      PJB7392                  PJB7392                    100%
      NYY1710                  NYY1710                    100%
      OCX4764                  OCX4764                    100%
```

Fig.8 Accuracy of PyTesseract

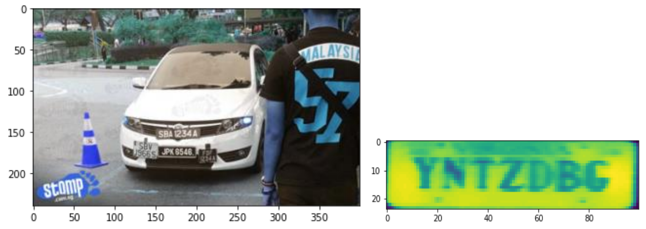
Fig shows accuracy model of Tesseract. One thing to be noted is that the character recognition works better when the images have clear visible image of the plate.



```
Detected Number is: YH: 136
Detected Number is: BIH: 136
```

Fig.9 Low quality Image output

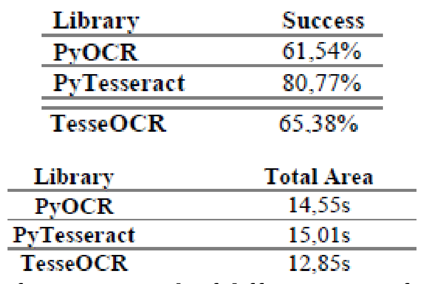
The OCR still does a good job providing 71.4% even though we have a blurred plate. The second output is from the model when we tried to reduce the bilateral parameters, i.e., reducing the gaussian blur. If we look at the input image, we realize that the model performs better for better clarity images.



```
Detected Number is:  YNTZDBE
```

Fig.10 High Clarity output.

Hence the model does provide 100% accuracy with respect to clear image we provide. We can also see that the PyTesseract is ideal for a VPR as it provides a quick and accurate result. [5] in this paper, Mário R. M. Ribeiro, Duarte Jùlio, Vasco Abelha, António Abelha and José Machado have given us a detailed performance bench for PyOCR, PyTesseract and TesseOCR.



Library	Success
PyOCR	61,54%
PyTesseract	80,77%
TesseOCR	65,38%

Library	Total Area
PyOCR	14,55s
PyTesseract	15,01s
TesseOCR	12,85s

Fig.11 Performance Bench of different types of Optical Character Recognition

It's noted that amongst the Optical Character Recognizers, PyTesseract does a goo job of detecting the numbers accurately.

IV. Related Work

The same project can be implemented using template matching, [6] here they have a pre attained template for Arabian characters, which is stored in the database. This paper has similar principle as compared to the proposed system, although as mentioned; Character Recognition for VPRS tend to change from one model to another. From the example provided it looks like the data set being used in [6] is clear and hence a better result is provided. The better the curated data is, the better the model turns to perform

[7] The model given in this research is similar to the one we have proposed, although here they have taken Myanmar plates as data set. They have also built the model for live detection of plates and as well as still images. The still images have provided them a 90% accuracy with tesseract OCR and 100% with still images.

[8] Here Haixiang Li, Xiaohui Chen and Ran Yang have implemented a VPRS using CNN. As mentioned earlier a custom CNN takes time to get trained, although once the model is trained this seems more applicable solution to our model. The only issue is while training the model, we are training it with a data set which is taken from that country. So, testing the same in a different data set, i.e., of a different country the accuracy reduces. Hence if we want to design a VPRS for a particular country or state, CNN is the best choice. Although if we can curate the data accurately tesseract OCR gives the results faster than a CNN.

[9] considers a model using CNN and YOLO. YOLO can detect objects with disturbed backgrounds. This can be a more practical model if the data set are images from a CCTV camera. In conclusion it can also been seen that the SVM model performs 70% better with YOLO.

V. Summary and Conclusions

The final model has been tested with data set 100 out of 433 images from kaggel provided by Larxel [10]. The accuracy changes according to the image's clarity. The accuracy before image processing was around 70%. With appropriate Image processing techniques, we have increased the accuracy to eighty seven percent as mentioned in Fig

S.no.	Name:	Accuracy
1	Images with clarity	87%
2	Disoriented and poor lighted Images	40%

Fig.12 Table depicting accuracy of the model.

The model can be improved further using Machine Learning models using Linear neural networks or Convolutional Neural network.

We can conclude that the various image processing techniques are useful in such projects. Image blurring is an essential task which removes unwanted noise for edge detection, this provides us a better detailed output from our edge detector. Edge detection has been on of the most useful image processing technique over the period, it helps us with image analysis, pattern recognition. Masking is another such image processing technique which helps us remove the undesired data from images.

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), 1998, pp. 839-846, doi: 10.1109/ICCV.1998.710815.

[2] opencv.org cv2.*bilateral Filter*Available: https://pytorch.org/docs/stable/nn.functional.html

[3] Ray Smith Google Inc An Overview of the Tesseract OCR Engine

[4] Audichya, Milind & Saini, Jatinderkumar. (2017). A Study to Recognize Printed Gujarati Characters Using Tesseract OCR. Engineering, Technology and Applied Science Research. 5. 1505-1510. 10.22214/ijraset.2017.9219.

[5] M. R. M. Ribeiro, D. Jùlio, V. Abelha, A. Abelha and J. Machado, "A Comparative Study of Optical Character Recognition in Health Information System," 2019 International Conference in Engineering Applications (ICEA), 2019, pp. 1-5, doi: 10.1109/CEAP.2019.8883448.

[6] M. J. Ahmed, M. Sarfraz, A. Zidouri and W. G. Al-Khatib, "License plate recognition system," 10th IEEE International Conference on Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003, 2003, pp. 898-901 Vol.2, doi: 10.1109/ICECS.2003.1301932.

[7] N. H. Lin, Y. L. Aung and W. K. Khaing, "Automatic Vehicle License Plate Recognition System for Smart Transportation," 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), 2018, pp. 97-103, doi: 10.1109/IOTAIS.2018.8600829.

[8] H. Li, R. Yang and X. Chen, "License plate detection using convolutional neural network," 2017 3rd IEEE International Conference on Computer and Communications (ICCC), 2017, pp. 1736-1740, doi: 10.1109/CompComm.2017.8322837.

[9] C. Lin, Y. Lin and W. Liu, "An efficient license plate recognition system using convolution neural networks," 2018 IEEE International Conference on Applied System Invention (ICASI), 2018, pp. 224-227, doi: 10.1109/ICASI.2018.8394573.

[10] Larxel Car License Plate Detection (2009)