Title:

Parking Lot Construction using License Plate Recognition

Objective:

The objective of this project is to create a License Plate Recognition System which can identify the License Plate number of the vehicles and allot the parking lot number according to that.

1. Abstract

There has been a rapid growth in technology leading to the automatic transport system, identification of vehicles, etc. There is a need for this kind of technology as it can become a core part of few essential infrastructures in our daily life like toll payments, parking lot management systems, better security surveillance, etc. We know that all the vehicles worldwide have license plates and these license plates give special identification to vehicles and based on this we create a parking management system.

This project is part of our Artificial Intelligence classroom course, aimed to create a parking lot using License Plate recognition. The input is an image of a car and the output returned is the license plate number of that car.

Our project contains 3 major steps:

1. Plate Recognition
2. Character Segmentation
3. Character Recognition

In this project, we achieved success in performing LPR for parking lot creation.

2. Working Principle

License Plate recognition consists of 3 major steps:

1. Plate Recognition
2. Character Segmentation
3. Character recognition

In general words, first, the license plate is recognized in the input image. Then we perform character segmentation i.e. separating the characters from the plate recognized. After character segmentation, each character is recognized i.e. whether they are A, B, 1, 2, etc.

In all three steps, there is plenty of room to implement image analysis and machine learning schemas. This section provides a brief introduction to the methods used by us at each step.

2.1 Plate Recognition

License Plate Recognition aims for the identification of license plates from the input car image. The general idea is to perform feature extraction from the input image so that we can reduce the amount of data present in the image. We can use several methods for this like region segmentation, fuzzy sets, edge features, etc.
Here we have used two different methods to perform Plate Recognition.

2.1.1 Heuristic Method:

In this method, few assumptions are made to extract the expected cropped section of the image containing the number plate. These assumptions are made keeping in mind the input constraints coming to picture due to the static position of the camera taking pictures of vehicles.

The assumptions made for detecting number plate are:

1. The height of the plate is within 5%-20% of the total image height.
2. The width of the plate is within 15%-60% of the total image width.
3. The area of the region is more than 50 units.
4. Plate height is greater than 20% of plate width.
5. The sum of pixel values is greater than 60% of the pixels present inside the plate.

Steps:

1. Take the image as input.

2. Converts the image to grayscale and then to binary form using an otsu threshold value.
3. Labels all the connected components in the binary image and selected corresponding bounding-box to represent the connected component.
4. Selects the bounding box satisfying the assumption and the list selected bounding boxes is sent as input for character segmentation.

License Plate Detection Step.

Fig: License Plate Recognition

2.1.2 Edge Feature Detection

The basic idea is to identify the number plate and then use this number plate image with an OCR to get the numbers and alphabets in the license plate.

- First, we take the given image containing the license plate of the car and read it as a NumPy array. Now since the license plate number remains unaffected whether the image is colored or black and white, we convert images to grayscale.

Fig: Conversion to grayscale

- The advantage of the image to grayscale is that it has only 1 layer whereas the colored image has 3 layers, hence the processing speed is greatly increased by converting the image to grayscale.
- The next step is to remove noise from the image. Since the region of interest is not affected by this step so it is better to remove the noise especially from the background.

Fig: Removing unwanted noise from the image

- Now we perform the edge detection as the edges are sufficient to identify the number plate and the text (numbers and alphabets) inside it. The whiteness of the number plate nor the blackness (color) of the car is not important for us so we run filters horizontally and vertically to get the image that has only edges.

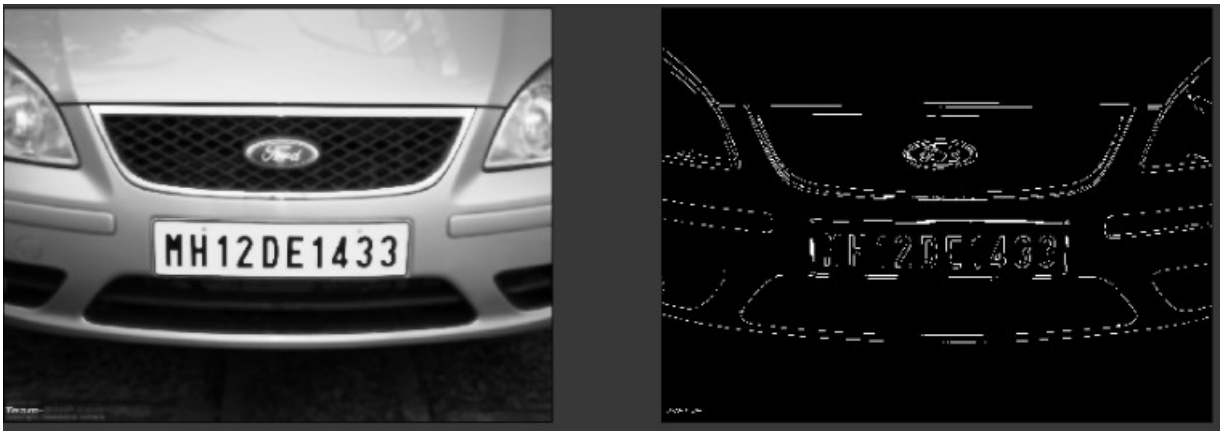

Fig: Car image after performing edge detection

- After getting the edges we perform contour identification. Contours are curves joining continuous points having the same color/intensity. This contour is generated using the edge detected image of the car and then We pass a copy of the image to the contour identification as the function alters the image.
- After identifying all the contours we know that one of the contours is the number plate. We also know that the contour corresponding to the number plate is one of the largest in terms of area and perimeter. Another characteristic of the number plate is that it will always be a polygon of four sides (irrelative of different angles, scale, and location of the number plate).

Fig: Contours of the image

- Hence, we can infer that we can get the number plate by sorting the contours by area in decreasing order, and taking the contours with 4 closed edges and drawing the rectangular bounding box around it. This gives us the number plate.

2.2 Character Segmentation

After the license plate is recognized we need to perform character segmentation. In character segmentation, we identify each character present on the license number plate and later these characters and used for character recognition. For character segmentation, there are various methods available to do this like using contours, projection, transforms, etc. We have done the following:

2.2.1 Heuristic Method (Extended):

The assumptions made for the detection of characters are:

1. The plate contains only 10 characters.
2. The height of each character is within 35%-90% of the total plate height.
3. The width of each character is within 2%-10% of the total plate width.

Steps:

1. Labels all the connected components inside the plate area selected before. It then selects a bounding-box to represent the connected components.
2. These selected boxes are checked against the assumption made for characters.
3. In the end, assumption number 1 is checked and the corresponding segmented character set is returned as output.

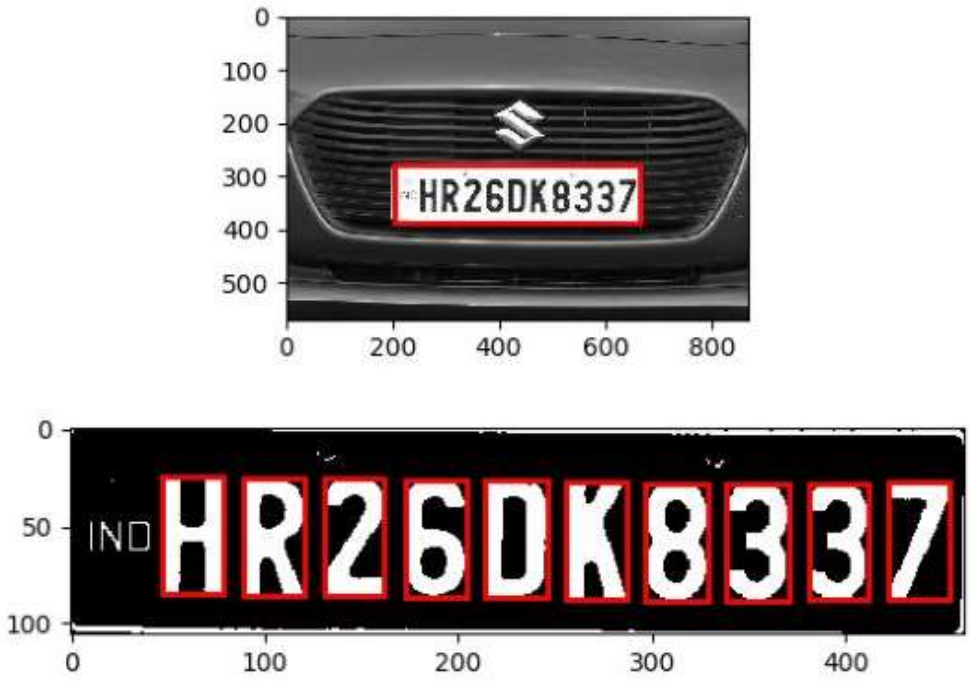

Fig: Character Segmentation of the recognized plate

2.2.2 Using Tesseract (Open Source OCR)

Tesseract is an open-source OCR (optical character recognition) that can work on more than 100 languages and find the text form the input image. It was the first OCR engine to be able to handle black and white texts efficiently.

Working of Tesseract:

- First, it performs connected component analysis in which it stores the outlines of the connected components.
- Next, it creates Blobs by gathering the outlines together.
- These blobs are organized in the form of text lines and then analyzed for a fixed proportion of text. This proportional text is then divided into words.
- Next, the recognition of characters takes place using an adaptive classifier.

All this is wrapped into python library “pytesseract” which is imported when we want to use the Tesseract OCR. Once we have our license plate recognized, extraction of license plate number from the plate image is done using pytesseract image_to_string() function.

2.3 Character Recognition

After we perform character segmentation, we need to recognize each of the segmented characters. In other words, we need to create an OCR (Optical Character Recognition). For this, we can use either a statistical classifier or a computational classifier. Statistical classifiers include Hidden Markov Models and Support Vector Machines. Computational classifier includes Self-organized neural networks and probabilistic neural networks. We used the following:

2.3.1 Support Vector Machine Classifier

The main objective of the support vector machine algorithm is to find out a hyperplane in N-dimensional space (N — the number of features) that uniquely classifies the information points into different predefined groups or segments. To separate the 2 classes of information points, there are many possible hyperplanes that would be chosen. Our objective is to seek out a plane that has the most margin, i.e the most distance between data points of both classes. Maximizing the margin distance provides some reinforcement in order that future data points are often classified with more confidence. There are cases when data points are not always linearly separable. In those cases, SVM uses different kernel modes that measures the closeness (or closeness) of the data points in order to make them linearly separable.

We use a support vector machine classifier to classify each alphanumeric character (characters obtained in the character segmentation step). Support Vector Machine is a linear classifier. To perform a non-linear classification kernel trick is used. Support Vector Machine is used for binary classification to distinguish each alphanumeric character. We get the binary image of each alphanumeric character by using the Otsu threshold value. The training set consists of 20pX20p images of all alphanumeric characters. We are using a sci-kit learn package for implementing our model and cross-validation with four folds (that is 3/4th of the dataset is used for training and 1/4th is used for testing). Once the model is trained it is saved for our later use i.e. to make predictions on the character segmented image of license plate.

3. Results

There are many ways to develop a License Plate Recognition (LPR). Each method requires different algorithms at different stages (stages refers to Plate Recognition, Character Segmentation, and Character Recognition). Each has its own pros and cons.

In this project we have developed LPR using two different methods:
1. In the first method, we have used the above-mentioned edge feature detection for plate recognition and Tesseract for character segmentation and recognition. Since Tesseract is a State of the Art OCR the results obtained using this method was almost perfect. For, all car images in which license plates are readable this method returned the license plate number.

Fig: License Plate Recognized using Edge Feature detection and Tesseract

2. In the second method, we have used the above-mentioned Heuristic Method for plate recognition and character segmentation and SVM classifiers to recognize each segmented character. Since this heuristic method requires car images to fulfill few criteria therefore this method worked on only a selective type of image. But the results obtained from this method even for the images in which the license plate was small were very good.

Final Code Output:

```
Recognized License Plate is:
HR26DK8337
```

Fig: LPR using Heuristic Method and SVM classifier

4. References

- https://www.researchgate.net/publication/260720555_License_Plate_Recognition_A_Brief_Tutorial
- https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa
- https://www.quora.com/What-is-the-best-OCR-software-on-the-market
- https://towardsdatascience.com/support-vector-machine-explained-8d75fe8738fd
- https://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html
- https://docs.opencv.org/3.4/df/d0d/tutorial_find_contours.html
- https://medium.com/@bhadreshpsavani/how-to-use-tesseract-library-for-ocr-in-google-colab-notebook-5da5470e4fe0
- https://www.quora.com/How-does-the-Tesseract-API-for-OCR-work