Character Recognition for Vehicle Number Plate Recognition

Aryan Bansal (18BEC0626), Urjani Chakravarti (18BEC0629), Vikram Baruah (18BEC0967)

Department of Electronics and Communication, Vellore Institute of Technology

***Abstract*— This study focuses on character recognition techniques for vehicle number plate recognition. It employs the KNN algorithm to detect the characters in the vehicle's number plate. It also successfully identifies the state of origin of the vehicle allowing us document and navigate the vehicles optimally. Moreover, the algorithm allows us to mark the time of entry of the vehicle and store it in a systematic manner. The algorithm provides a methodological solution to keep track of the numerous vehicles entering a premise in an orderly fashion. The algorithm can be further improved to account for accuracy and time complexity as well as font variations.**

Keywords*— *Character Recognition, Vehicle Number Plate Recognition, Machine Learning, KNN

I. Introduction

In today's world, there is a growing demand for software systems that recognize characters in computer systems when information is scanned from paper documents. We have a large number of newspapers and books in printed format that cover a wide range of topics with huge demand these days for storing the information available in these paper documents to a computer storage drive and then later reusing this information through a search operation. Scanning the documents and then storing them as images is a simple approach to store information from these paper documents in a computer system, however, reading individual contents and examining the contents of these documents line-by-line and word-by-word makes it tough to reuse this information. The challenge arises because the font qualities of characters in paper documents differ from those of characters in a computer system. As a result, while reading the characters, the computer is unable to recognize them. Document Processing is the concept of keeping the contents of paper documents in a computer storage location and then reading and searching the content. A software system named Character Recognition System is required to perform Document Image Analysis, which converts paper documents into electronic format.

Optical character recognition (OCR) technology is a solution for automating data extraction from printed or written text from a scanned document or image file and then converting the text into a machine-readable form to be used for data processing like editing or searching. OCR is a program that can be used to recognize characters that exist in a certain image. In order to use the program, it needs to be fed with some examples of how each letter looks like. These examples are usually called training data. It can recognize both handwritten and printed text, but its accuracy highly depends on the training data that is given to the program.

The problem of character recognition is one of the most common image recognition problems. Despite the enormous number of various character recognition applications available, the importance of developing new software is not diminished. This is primarily due to the fact that most free software programs have low accuracy and are designed to recognize printed text and the image processing operations required to detect the characters are limited. Based on the principles of Character Recognition, it was decided to develop a system that uses its own character recognition algorithms, built on existing ones to recognize number plates of vehicles as an application.

The number plate recognition system is based on image processing technology. The primary objective is to develop an autonomous system that can recognize a number plate successfully so that the number of fatalities caused by reckless drivers and wrong-doers can be reduced and the chargeable can be detected. In today's world with the increasing number of vehicle day by day it's not possible to manually keep a record of the entire vehicle. With the development of this system, it becomes easy to keep a record and use it whenever required. The system first processes the captured image using image processing techniques to reduce distortion and improve the resolution. Next, the exact location of the number plate is recognized using the segmentation technique. Once the number plate has been segmented, the length of number plate has to be determined and correlated. Finally, the recognized characters are converted to a string and displayed. The system is implemented and simulated on Python 3 and performance is tested on real images. This type of system can be used in densely populated areas, tolling, parking area, etc. This project proposes a method for the detection and identification of vehicle number plate that will help in the detection of number plates of authorized and unauthorized vehicles.

There are various kinds of algorithm which can be used to create an Optical Character Recognition program. This experiment uses the K-Nearest Neighbor approach to train the system. The experiment also calculates the accuracy of said algorithm when implemented into an OCR. The experiment also requires some image processing functions to extract information from the image tested into the system. For that purpose, this experiment requires an additional library called OpenCV (OpenSource Computer Vision Library) for its image processing functions and K-Nearest Neighbor machine learning function.

II. Literature review

One of the earliest studies conducted by J Mantas in 1986, explores the various algorithms and methodologies of character recognition. The study presents an overview of character recognition methodologies that have evolved in this century. For example, one of the earliest technologies of character recognition is that of scanning technologies. These mainly revolve around on-line and handwritten character recognition based on a sample. Using contour and edge detection, characters can be recognized and deciphered.

The paper published by Govindan and Shivaprasad aims to provide an in-depth view about character recognition techniques, various methodologies, the research work in character recognition, and some practically used optical character recognition methods. The paper elaborates the different schemes of extraction used, based on the application of the character recognition. In this study, template matching and correlation techniques as well as feature analysis and matching techniques are discussed.

De Campos et al published a paper in 2009, on camera-based character recognition. The paper highlighted the problem of recognizing characters in images of natural scenes. In particular, they focussed on recognizing characters in situations that would traditionally not be handled well by OCR techniques. For their research, they utilized an annotated database of images containing English and Kannada characters. The database comprised of images of street scenes taken in Bangalore, India using a standard camera. The problem was addressed in an object categorization framework based on a bag-of-visual-words representation. They assessed the performance of various features based on nearest neighbour and SVM classification. De Campos et al found out that the performance of the proposed method, using as few as 15 training images, resulted to be far superior to that of commercial OCR systems. Furthermore, the method benefitted from synthetically generated training data obviating the need for expensive data collection and annotation.

In a study done by Mithe et al in 2013, a new system for OCR was proposed by the group. They scrutinized the existing method of using a scanner and a computer to identify and recognize characters, and came up with the solution of using an android phone with a superior camera instead, which results in saving space, with a tradeoff of slower computational speed. Their system used Tesseract, which is an open-source OCR engine. Their proposed system comprised of the following steps to correctly recognize the text: scanning, segmentation, preprocessing, feature extraction and finally, recognition. Scanning was the process of taking an image of the text that needed to be recognized, and the subsequent grey-scaling required for further processing. Segmentation was the process of locating regions of printed or handwritten text. It is used to isolate words or characters. Preprocessing was used to remove any noise or distortion, which might alter the final results. Features extraction was used to extract the features of the symbol, and in the end, the Tesseract algorithm would recognize the characters.

Another study conducted by ND Cilia, et al, in 2017 stated the various techniques for character recognition. Here, it was mentioned that feature selection was a crucial step in the process. It reduced the computational cost of the classification track. In the framework of handwriting recognition, the large variability of the handwriting of different writers makes the selection of appropriate feature sets even more complex and have been widely investigated. In this study, several classifiers were studied by adopting a feature-ranking-based technique. Different univariate measures were studied and a greedy search approach was selected. In the experiments, one of the most effective and widely used set of features were considered in handwriting recognition to verify whether the approach allows them to obtain good classification results by selecting a reduced set of features. The experimental results were highly favourable and was shown to have high accuracy.

A similar study was conducted in Japan by T Clanuwat, et al, in 2019. This study aimed to recognize characters native to the Japanese language. The study explored Character Recognition using Deep Learning. This study implemented an end-to-end model which jointly recognized an entire page of text by using a residual U-Net architecture which predicted the location and identified all characters given a page of text (without any pre-processing). This allowed the model to handle long range context, large vocabularies, and non-standardized character layouts. Hence, we can conclude that character recognition can be a seamless method for recognizing characters of different fonts and languages.

Another study conducted by TK Hazra, et al, 2017, implemented the simple KNN algorithm to simplify character recognition. This study made use of an efficient method to use a custom image to train the classifier. This OCR extract distinct features from the input image for classifying its contents as characters specifically letters and digits. The input to the system was digital. With the help of the KNN algorithm, this methodology was easily achieved.

In 2021, Goyal et al published a paper which delved into the application of character recognition systems, which is vehicle license plate recognition. Goyal et al argued that vehicle license plate recognition has become essential now, but the proper systems have not yet been designed, which can accurately recognize the character and number on license plates. In this paper, the researchers propose a procedure for successfully recognizing the license plates of cars. They analyzed different methods, and found that template matching with cross correlation was the best way to proceed, as it provided an accuracy of 98.07% for Indian vehicle license plates. Convolution neural network and proficient machine learning methods were also discussed in brief. They also listed the various challenges that arose, such as low file resolution, smear images, overexposure, reflection or shadows on the number plates and dirt on the plate or something cover the characters.

III. METHODS AND DATA SET

A. Reasearch Questions

- To develop an algorithm and code that can recognize printed text in images accurately and efficiently,
- To use the character recognition algorithm and code developed to recognize characters on the license plates of Indian vehicles,
- To recognize and classify the vehicles based according to the states in which they were registered, based on the results obtained from recognizing their number plate.

B. Methodology

The K-nearest neighbor is one of the more popular algorithms in the field of machine learning and data sciences. In this body of research, we opted for the K-nearest neighbor or the KNN algorithm.

To recognize alphabets from A through Z and digits from 0 through 9, we used 5 training images for each alphabet and digit, i.e., 180 training images in total. The training process produced two parallel data structures – a set of images, and a set of numbers indicating which "group" or "classification" each corresponding image is in. The classification is the alphabet or the digit of which the corresponding image belongs to. After the completing the training, we used the trained model to recognize printed characters in images.

The model works by trying to match as many pixels as possible. Suppose, each character's image has dimensions of 10 x 10 pixels, which translates to each image having a total area of 100 pixels. Let us also suppose that we have to classify some character "X", and recognize it. If the character to be recognized (X) has a best match of say, 99 pixels with a given training sample, that sample is considered the nearest neighbour. Each image in the training sample is checked to identify the character "X" correctly. The best match is taken as the character, and then "X" is classified.

At first, we took a dataset of roughly 15-20 images. These images had different resolutions, angles, colours, and redundant objects in the background. The images were selected in such a way as to give vast coverage for the vehicles present in the roads. Moreover, the images were also selected in such a way to challenge us to develop our code to the maximum. Given below is one of the several images we used to test our algorithm. The picture is more or less oriented in a straightforward manner.

Fig. 1. Original Sample Image

Next, we had to convert our image into a greyscale image. Greyscale images require less information per pixel. This helps us to get rid of redundant information. Greyscale intensity is stored as an 8-bit integer giving 256 possible different shades of grey from black to white. In addition, greyscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process colour images. Our greyscale image is converted to the threshold image. In a threshold image, there is a threshold intensity which is the benchmark for the intensity to be considered entirely white or entirely black. All intensities above the threshold intensity is considered to be white. All intensities below the threshold intensity a considered to be black. Given below is a sample for the greyscale and threshold image of the original sample image.

Fig. 2. Grayscale Image

Fig. 3. Threshold Image

Our next step would be contour detection to determine edges. Contour detection is essential to determine the exact location of the number plate. Since number plates are rectangular in shape, detecting contours is a perfect method of detecting the location of the number plate in the image. If we fail to locate the number plate, we cannot obtain the vehicle number. Given below is a set of all contours in the image. As we can see, the number plate is perfectly contoured.

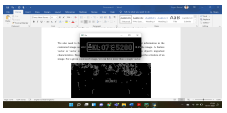

Fig. 4. Contours for the Image

We also need to find the vectors for this contoured image. The redundant information in the contoured image needs to be processed. Hence, we obtain the vectors of the image. A feature vector or vector is just a vector that contains information describing an object's important characteristics. Basically, it is a one-dimensional matrix which is used to describe a feature of an image. For a given contoured image, we can have more than a single vector.

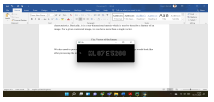

Fig. 5. Vector of the Image

We also need to find vectors of vectors. In order to successfully identify the number plate, we need to group the vectors. On grouping the vectors, we find all the potential vector combinations. For a given image, there can be several

vectors of vectors. In simple words, there can be several possibilities for the number plate to exist.

Fig. 6. Vector of Vectors

Now, we have to iterate through the vectors to find the most probable location of the number plate. First, we get the vectors of the original image. As we can see, the algorithm has identified two potential vectors for the number plate. The first vector is of the original number plate with the redundant information omitted. The second vector is of the background noise, like trees, which will be a 'contender' for the possible vectors. The algorithm then obtains the greyscale and threshold images for these vectors for ease and simplicity

Fig. 7. Potential Vectors of Original Images

Fig. 8. Vector after Grayscale Processing

Fig. 9. Vector of Vectors

Next, we need to find the vectors of possible characters in plates. This is followed by finding the vector of vectors of matching characters. Finally, we remove the inner overlapping shapes and find the longest vector of matching characters in the plate. This is followed by recognizing the characters in the plate. We will now see a step-by-step approach of the previously mentioned process.

Fig. 10. Finding Vectors of Potential Characters

Fig. 11. Longest Vector of Matching Characters in a Plate

Fig. 12. Recognized Characters

Now that the algorithm has successfully recognized the characters, we need to compare each character to a list of characters. We attempt to recognize the characters using the KNN (K-Nearest Neighbour) algorithm. In this algorithm, we use different fonts for numbers through 0 – 9 and letters from A – Z (all characters that may be present on a valid number plate). Let us assume that we are training the algorithm to recognize numbers from 0 to 9, using 5 images for each character. Each character has a width and length of 10 pixels, and hence an area of 100 sq. pixels. The training process generates 2 datasets – the first is a set of numbers indicating which "group" the image belongs to, and the second is a set of images. To recognize an unknown character X, we compare X with the previously mentioned images. We look for the best match. If for example, image4 is the best match, X is classified as '0'.

Fig. 13. Classification Table

Fig. 14. Character Table

The final output of our image processing algorithm is in two formats for our convenience. In the first format, the number plate is printed on the original image while simultaneously highlighting the number plate. In the image given below, we can see that the algorithm has successfully identified the number plate and removed redundant information such as trees, people, other cars, writings on the number plate, etc.

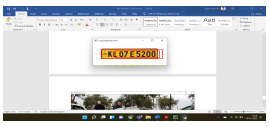

Fig. 15. Output Format 1

The second format of output is in the form of an output on python. The complier we have opted to use is Pycharm. In the image below we can clearly see the output of the algorithm.

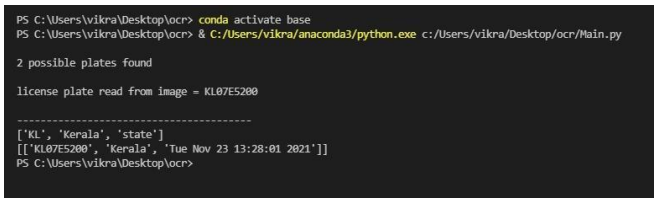

Fig. 16. Output Format 2

In figure 16, we can clearly see that our algorithm is capable of determining the state the vehical has been registered in as well as the time of entry. As a result, this algorithm can help us determine the origin of the vehicle.

For identifying where the vehicle was registered, we used an excel sheet containing a list of the two-letter codes in the first column, the corresponding states and union territories of India in the second column, and the subdivision category in the third column. The subdivision column is used to identify whether the place of registration is a state or a union territory. The excel sheet acts as a database in this case.

After the number plate is correctly identified, the program accesses the excel sheet, and compares the first two characters of the recognized plate with the two-letter codes in the excel sheet's first column. Once the letters on the plate are matched with a code in the database, the state's registration code, name and the subdivision category are displayed as well.

Next, we also stored the time of entry in a systematic manner in order to manage the vehicles in an orderly fashion. This enables us to keep track of the vehicle and allows for better security in the premise. Figure 17 shows the file documenting all important details.

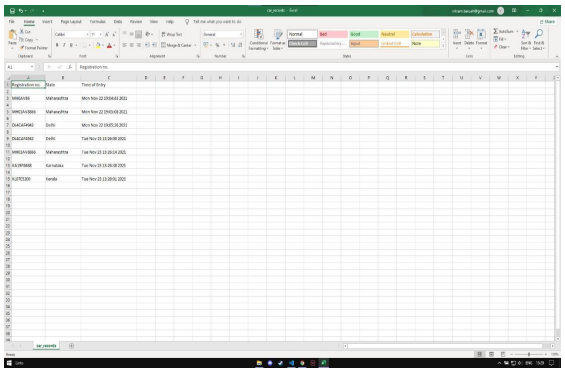

Fig. 17. Output showing time of entry of vehicles

IV. Results and Interpretation

We can see from the results, that the algorithm and the code work well with the images that we have used here. The program is able to correctly identify the number plates of the vehicles, and subsequently classify the state of registration of the vehicle.

The program can be used to identify the license plates on highways, in parking lots, and other locations as well. Its application can be augmented by creating a database containing the numbers of vehicles, which can be used to provide access to vehicles to an enclose area only if their license plate matches a number in the database.

This methodology requires us to train the model before using it, which can be time consuming. However, if a higher number of fonts are used, the model can provide a better result.

The drawback of this model is that it does not produce good results if the fonts in the images are not standardized. For example, in some fonts, the digit "1" looks similar to the alphabet "I" (such as in the font Times New Roman). This proves to be problematic if the training images used do not have a wide variety of fonts. Even if a large number of fonts are used, the model will work best if the characters look as similar as possible in all fonts while training.

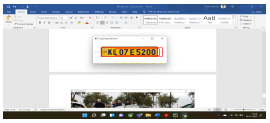

Fig. 18. Number Plate Recognition

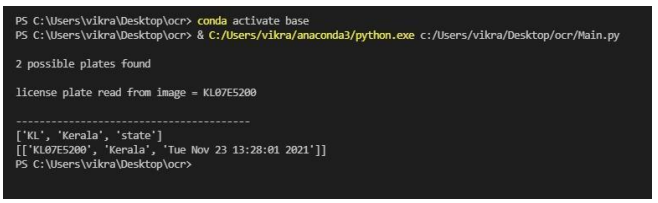

Fig. 19. State recognition

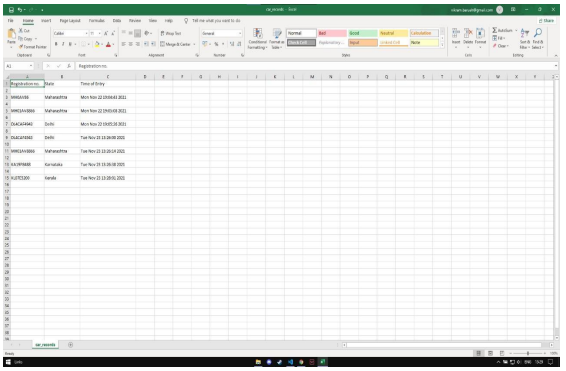

Fig. 20. Time of entry of vehicles

V. Performance Analysis

The project successfully recognizes the vehicle number plates by employing the KNN algorithm. We have utilized a variety of images taken at different angles and having different resolution. The algorithm has been successfully in the case of images of poor resolution and angles. We have also tested the algorithm to identify a variety of fonts, however, we can still include several other fonts like cursive or calligraphy. Moreover, we can also enhance the algorithm to include other languages and roman numerical. Moving on to the accuracy of our algorithm, we have tested over 20 images, each having unique characteristics, and only 2 images showed incorrect character recognition. For

example, it confused the letter ‘I’ for the number ‘1’, and so on. However, these instances can be rectified by training the algorithm to include more fonts. Apart from this, the algorithm has successfully recognized the state of origin of the vehicles along with the time of entry. The algorithm can be improved by making the time complexity more optimal to allow for faster documentation which can be crucial during busy times of the day.

VI. Conclusion

The project successfully recognizes the vehicle number plates by employing the KNN algorithm. The number plates are recognized as a string format. The algorithm also identifies the state of origin of the vehicle. This can be a useful feature as it enables us to monitor where the vehicles are coming from. This gives us many uses, for example, if the vehicle is coming from a state with several COVID 19 cases, we can stop the vehicle from entering to prevent spread of the disease. Apart from this, we can also document the time of entry of the vehicle and store it in a systematic manner. This helps us to keep track of the in-time of vehicles. These features allow us to secure our neighborhood or the premise and allow us to document the cars that enter the premise. There can be several short comings as well, for example, the accuracy of the algorithm can be improved to include more fonts and languages. This can help us to widen our reach and improve our accuracy. The project can be improved to include different languages and a plethora of fonts to make the documentation more accurate. These improvements can make the project extremely function and it can be adopted by universities and offices.

VII. References

1. Mantas, J. (1986). An overview of character recognition methodologies. *Pattern recognition*, *19*(6), 425-430.

2. Govindan, V. K., & Shivaprasad, A. P. (1990). Character recognition—a review. *Pattern recognition*, *23*(7), 671-683.

3. De Campos, T. E., Babu, B. R., & Varma, M. (2009). Character recognition in natural images. *VISAPP (2)*, *7*.

4. Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, *2*(1), 72-75.

5. Cilia, N. D., De Stefano, C., Fontanella, F., & di Freca, A. S. (2019). A ranking-based feature selection approach for handwritten character recognition. *Pattern Recognition Letters*, *121*, 77-86.

6. Clanuwat, T., Lamb, A., & Kitamoto, A. (2019, September). Kuronet: Pre-modern Japanese kuzushiji character recognition with deep learning. In *2019 International Conference on Document Analysis and Recognition (ICDAR)* (pp. 607-614). IEEE.

7. Goyal, S., Dube, S., Mali, N., & Udawant, P. (2021). Vehicle License Plate Detection and Recognition System: A Review. *International Journal of Recent Advances in Multidisciplinary Topics*, *2*(10), 125-128.