Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 2623–2633

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition using Image Processing Techniques

Ravi Kiran Varma P[a]*, Srikanth Ganta[a], Hari Krishna B[b], Praveen SVSRK[c]

[a]*MVGR College of Engineering, Vizianagaram, AP, India*
[b]*CMR Engineering College, Hyderabad, Telangana, India*
[c]*Commvault, Bangalore, Karnataka, India*

Abstract

Number Plate recognition, also called License Plate realization or recognition using image processing methods is a potential research area in smart cities and Internet of Things. Many of the existing automated number plate recognition systems work only in a controlled environment where images are captured from a straight angle with good illumination, clarity and standard fonts. Another drawback of existing works is that, they are based on UK number plates and may not suite for Indian number plates. This paper presents a novel image processing system for Indian number plate detection and recognition that can deal with, noisy, low illuminated, cross angled, non-standard font number plates. This work employs several image processing techniques such as, morphological transformation, Gaussian smoothing, and Gaussian thresholding in the pre-processing stage. Next, for number plate segmentation, contours are applied by border following and contours are filtered based on character dimensions and spatial localization. Finally, after the region of interest filtering and de-skewing, K-nearest neighbor algorithm is used for character recognition. The proposed methods demonstrated promising results.

Keywords: Image processing, Number Plate Detection, License plate detection, Vehicle number plate recognition, character recognition.

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
E-mail address: ravikiranvarmap@mvgrce.edu.in

1. Introduction

The focus of the proposed work to apply novel image segmentation and other processing techniques in the context of registration number plate identification and realization. Most of the current works target standard Number plate formats such as the UK, Argentina or Russia. Nicolas et al. introduce a novel approach called Intelligent Template Matching that is focused on Argentinian Number plates [1]. In a vast country like India, most of the plates are not standard, so a solution is needed that targets multiple formats and fonts. Also, there is no standard data set available for Indian Plates, so images and videos of Cars with Number plates were manually taken, considering cases like low, non-uniform illumination, distant plates, blurry plates, varied fonts etc. Since all the data were collected in India and most of the Number plates are composed of four letters and six numbers; The assumption is that the Number plates are in the following format: AP 31 BA 1432, where the first two characters signify the State of the Vehicle's Registration. Firstly, Number Plate region (the Region-of-Interest) is required to be located and extracted amongst the larger captured image of the vehicle. Later, the task is to recognize the numbers and characters present in the region of interest. By localizing the search to the region of interest a minimum of 90% percent of the full image can be avoided because the number plate region consists of only 10% of the full image of the vehicle under consideration. Another advantage of number plate identification is to optimize the time and space complexities.

One of the earlier work on automated number plate recognition (ANPR) is done by Lotufo et al. [2].. Nijhuis et al. [3] combined neural networks and fuzzy logic in recognition of car number plate for the case of the Dutch number plates. Fuzzy clustering was used for segmentation and neural networks for feature extraction and classification. An automated license plate recognition for images with deformation is addressed by Kim et al. [4]. They used genetic algorithms for image segmentation and achieved an accuracy of 92.8%. Another former research on Korean number plate recognition is done by Lee et al. [5]. In their work, firstly the region of interest is identified based on color classification using neural networks. Hue, Lightness and Saturation (HLS) values of the pixels are used in the classification. Length and breadth ratio of the plate is used in identifying the plate region. Histogram of the color of the character is used for character extraction and template matching algorithm for identification.

Wavelet transformation based character extraction is proposed by Hsieh et al. [6] for a distorted background case. However, only plate detection is considered in this work with an accuracy of 92.4%. Sindh standard number plate recognition is done by Quadri and Asif [7] wherein the number plate region is cornered with the help of yellow color identification, later using smearing algorithm the plate is segmented, then the Optical Character Recognition (OCR) is used to identify the characters. However the type of OCR algorithm used is not mentioned and the accuracy rate is also not given. Surekha et al. [8] performed number plate detection but did not report any character recognition. They have used morphological processing, canny edge detection and neural networks in their work. An accuracy of 97% was reported. They did not consider several environmental complexities of the images in their approach.

Kaur [9] worked with Indian number plate detection and recognition, where the author used morphological operations for number plate region extraction, connected component analysis and boundary box analysis for character extraction, and template loading & matching for character recognition. An accuracy of 96.67% was reported for character recognition. The advantage of this approach is detecting lower contrast, noisy, and blurred images. However skewed number plates are not addressed in this approach. Shi et al. [10] performed Chinese number plate recognition for various illuminations of images. An accuracy of 90% number recognition was achieved. However, noisy, blurred and skewed images are not considered in the test cases. Chang et al. [11] proposed an efficient number plate detection and recognition system. Color edge mapping and Hue Saturation Intensity (HIS) based color transformation methods including fuzzy aggregation are used for number plate extraction. OCR techniques are used in character recognition. An accuracy of 93.7% was reported with an advantage of performance in noisy environments, different illuminations, and even damaged number plates. However, the Indian number plates are not tested. Tejas et al. [12] proposed Indian number plate detection and recognition using techniques like Sobel edge detection, bounding box segmentation, and neural networks for recognition. However, bad environmental conditions were not tested in their method. A systematic review of several number plate recognition techniques can be found in [13]. Salau et al. [14] achieved an impressive accuracy of 99.8% for number plate region detection, however, character recognition is not considered.

In this work firstly, several image processing techniques viz., morphological transformation, Gaussian smoothing, Gaussian thresholding are used in the pre-processing stage. Next, for number plate segmentation, contours are applied by border following and contours are filtered based on character dimensions and spatial localization. Finally, after the region of interest filtering and de-skewing, K-nearest neighbor algorithm is used for character recognition.

The major contributions of this work is to design an Indian vehicle number plate detection and recognition using image processing system to deal with the following challenges:

- Dealing with varying illuminated images.
- Dealing with bright and dark objects.
- Dealing with noisy images.
- Dealing with non-standard number plates.
- Dealing with cross-angled or skewed number plates.
- Dealing with partially worn our number plates.

2. Methodology

The proposed methodology consisting of three major phases viz., pre-processing, detection, and recognition are shown in Figure 1.

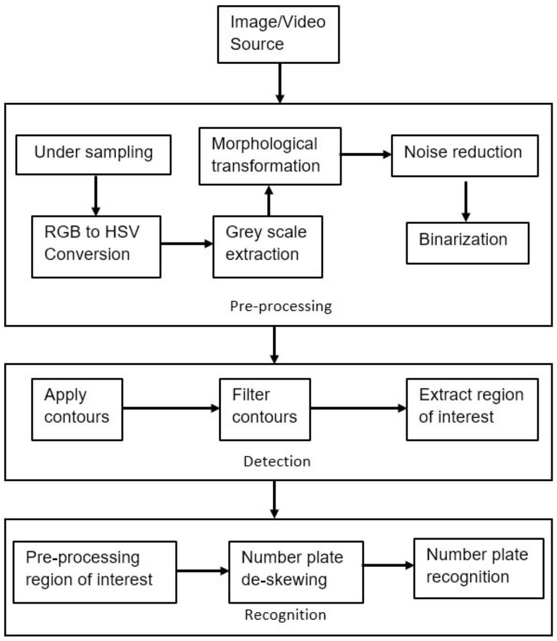

Fig 1. The proposed number plate recognition system

2.1. Phase 1: Pre-processing

The input can either be an image or a video. Video is considered as a series of Images/frames, before starting with Number plate detection, the Image source must be made suitable for further processing. Figure 2. (a) is the sample input image used for showcasing the process. The following is the order in which Image processing techniques are applied:

2.1.1. Image Under-Sampling

The Number plate detection and recognition algorithm are supposed to work at a steady and consistent frame rate. Unsurprisingly, for high-resolution images, image processing algorithms tend to work slow. It is in fact unnecessary to consider images with such a high resolution. This stage reduces the resolution if it crosses a predefined threshold.

In the case of videos, every 20th frame is under-sampled and sent for further steps. The input in this stage is a Food truck with a Number Plate. The Number plate is non-uniformly illuminated.

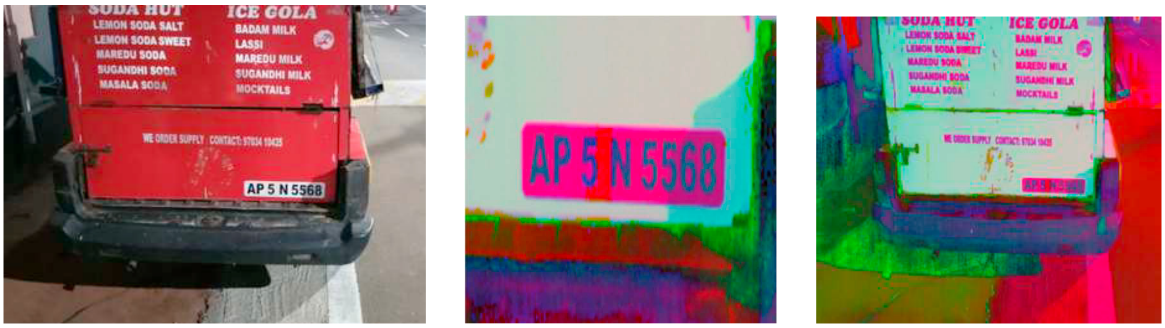

Fig. 2. (a) An input image; (b) HSV converted and cropped image; (c) HSV Converted Image

2.1.2. RGB to HSV Conversion

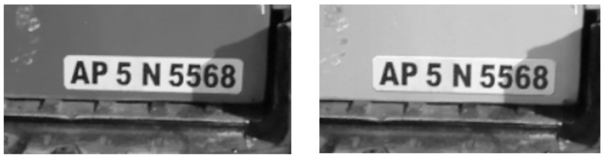

Fig. 3. (a) Grey scale images of RGB; (b) HSV channels

In OpenCV, the following function converts input image in RGB channels to HSV channels and returns the image after conversion Here, cv2 is the image, which is internally stored as a numpy array and cv2.COLOR_BGR2HSV is a flag for RGB to HSV conversion. Comprehensive documentation for changing color spaces is discussed in this tutorial [15]. HSV channels have the added advantage of separating color description from the luminance. This ensures working of the algorithm for images with a wide range of illumination. Figure 2 (b) and Figure 2 (c) show the output of HSV conversion.

2.1.3. Grayscale extraction

The luminosity or the Value channel of the converted HSV image is extracted to obtain the grayscale image. The obtained grayscale image undoubtedly is quite different from its RGB grayscale counterpart. The grayscale images of the sample under consideration is shown in Figure 3.

2.1.4. Morphological transformations

Top-hat and Black-hat filters are part of Morphological transformations that are some operations that can be performed on binary images. The Top-hat operation is used to enhance bright objects of interest in a relatively dark background, while the black-hat operation (also known as bottom-hat) is used to do the opposite, i.e., enhance dark objects of interest in a relatively bright background. Mathematically, the difference between the opening of the image and the image is the Top-Hat, and the difference between the image and the closing of the image is the Black-Hat. In this work, top-hat results are added to the original image and black-hat results are subtracted from it [16].

2.1.5. Gaussian Smoothing

Gaussian filtering, or Gaussian smoothing uses a linear Gaussian function. The objective of Gaussian filtering is to reduce noise and detail. This will serve well for further image processing steps. Applying Gaussian filter to an image has the added advantage of preventing aliasing artifacts. In Gaussian filtering, every point of the input array is convolved using the Gaussian equation as shown in equation 1. The output array is obtained by the summation of all such points. Mathematically, a one-dimensional Gaussian function can be expressed as:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \qquad (1)$$

However, for an Image, a two-dimensional version of this function is used, which is just the product of two one-dimensional functions. Mathematically, it can be expressed as:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \qquad (2)$$

where

x & y are the distances from the origin in the horizontal axis and vertical axis respectively and the standard deviation of the Gaussian distribution is denoted by σ.

Firstly, a sequence of one-dimensional Gaussian matrices are applied horizontally and the process is repeated vertically to achieve a two-dimensional matrix. That results in a reduction of computational complexity when compared to its two-dimensional counterpart [17]. In OpenCV, Gaussian smoothing can be applied using the following function:

$$cv2.GaussianBlur(image, (5{,}5), 0)$$

Here, the second argument (5, 5) is the size of the Gaussian kernel, the bigger the size, the more is the intensity of smoothing. The third argument 0 is the standard deviation for the Gaussian kernel [18].

2.1.6. Inverted Adaptive Gaussian Thresholding

The purpose of thresholding, as part of image segmentation, is to create a binary image from a grayscale image. This process is popularly known as Image binarization. The simplest thresholding method uses a global threshold to binarize the image. However, this method may not be suitable in conditions where illumination is non-uniform. A window of predefined size is taken and a weighted sum of neighborhood pixels is found to perform the adaptive thresholding. The 'Inversion' is a mathematical negation performed to suit our practical needs. Mathematically, each pixel in the output threshold image can be calculated as

$$output\_image(x,y) = \begin{cases} 0, & input\_image(x,y) > T(x,y) \\ 255, & otherwise \end{cases} \qquad (3)$$

where, T (x, y) is a threshold function that calculates threshold individually for each pixel.

OpenCV provides a handy function to perform Adaptive Gaussian thresholding.

```
cv2.adaptiveThreshold(image, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C ,
    cv2.THRESH_BINARY_INV ,
    BLOCK_SIZE, WEIGHT)
```

Here, BLOCK_SIZE is the size of the threshold window and WEIGHT is used to calculate the weighted sum of the values in the neighborhood [19].

2.2 Phase 2: Number plate detection

At the end of the previous stage of image pre-processing, Inverted Adaptive Gaussian Thresholding, returns a binarized image, with values of either 0 or 255. The binarized image serves as input to the further stages in both detection and recognition phases.

2.2.1. Applying contours

Contour Tracing, also called as Border following is the algorithm used for generating Contours. A contour is a link of equal intensity points along the boundary. In OpenCV, finding contours is like finding a white object from the black background, therefore during the Adaptive Gaussian Thresholding stage, Inversion operation had to be applied. Figure 4 (a) shows the result of applying contours to a binarized image.

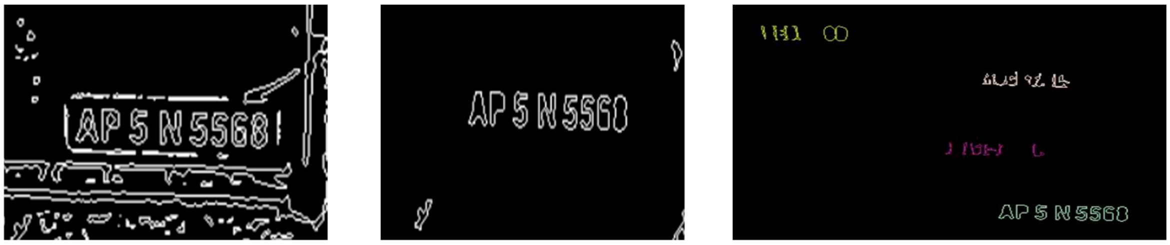

Fig. 4. (a) Applying contours; (b) Filtering contours; (c) Grouping contours

2.2.2. Filtering and Grouping contours

For small regions, especially sharp edges and noise outliers, contours are applied. A human eye can easily figure out that such contours are unnecessary, but this must be incorporated into a program. Initially, Bounding boxes were applied to each contour. Then, for each contour, the following factors were considered such as minimum contour area, minimum contour width and height, minimum and maximum possible aspect ratios. This resulted in the filtering of most of the unnecessary contours, propelling us near to our objective – to detect a Number plate. The second stage of filtering compares each contour to every other contour on parameters such as distance between contours, the delta angle between them, delta change in their area, delta width and delta height. If a group of contours satisfies all these conditions, they are grouped into one. It is possible that two or more such groups may be obtained. Figure 4 (b) and (c) shows the output of filtering and grouping.

2.2.3. Plate angle correction

In this stage, a bounding box is applied to each Number plate. If any of the plates suffer from angle distortion, an affine transformation is performed. An affine transformation, a mapping between two spaces, is used to conserve points and line. Post the transformation the parallel line retains their parallelism. The ratio of lengths between the points dwelling on a straight line is retained. However, the angles within the lines and lengths within the points are not preserved. In OpenCV the plate angle correction can be achieved by the function getRotationMatrix2D [20]:

$$cv2.getRotationMatrix2D(center, angle, scale)$$

Where, *center* is the axis of rotation, *angle* is rotation angle in degrees, and *scale* represents the scale factor. This function returns the following affine matrix:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot x - \beta \cdot y \\ -\beta & \alpha & \beta \cdot x + (1-\alpha)\cdot y \end{bmatrix}$$

Where,
α = scale * cos(angle), β = scale *, x is the x-coordinate of centre of the plate
y is the y-coordinate of centre of the plate.

This matrix is passed as input to warpAffine [21] as follows:

$$cv2.warpAffine(image, rotationmatrix, width, height)$$

Where, width and height are the dimensions of the number plate. For the sake of comparison, figure 5 (a) and (b) shows the image before and after angle correction respectively.

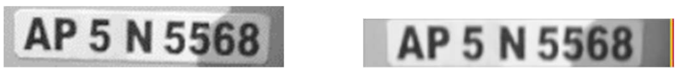

Fig. 5. (a) Number plate before correction; (b) Number plate after correction

2.2.4. Remove overlapping characters/contours

It is possible that two or more contours may completely overlap with each other, as in the case with the number 'zero'. The inner contour, if detected in the contour process, may lie completely inside its outer contour. Due to this phenomenon, both contours may get recognized as separate characters during the recognition process. Therefore, this step is introduced to ensure the removal of such overlapping characters.

2.3 Phase 3: Character recognition

2.3.1. Characters transformation and Prediction

Once overlapping characters are removed, each contour, which represents a character in the number plate, is resized into a 20 x 30 image. This operation is performed to ensure consistency with the input format of the learning model, where each font character that is fed during the training process is a resized 20 x 30 image. After the resized image is fed to the model, the character is predicted. This is done for each contour to generate a string of characters. The remaining contour groups generated in Fig 6, which do not represent the Number plate, fail in the prediction stage. The group with the most number of predicted characters is recognized as the number plate.

2.3.2. Training the model

For training the model, K Nearest Neighbours (KNN) algorithm was used. Many other models such as Decision Tree, Gradient Boosting were tested, but KNeighbours performed better. To extract the best possible hyper parameters for the model, Randomized search was used. Randomized search is an optimized version of parameter sweep or grid search, wherein a rigorous search is made out of manually formed space of subset of hyper-parameters that belongs to the learning algorithm. Performance metrics are used in the guidance of grid search such as, cross-validation of the training-set or evaluation of the validation-set. The parameter space explored by the grid search and the randomized search is the same. The parameter setting is rather similar, however the run-time in case of randomized search is by far lesser.

In the KNN classification, the query point is labeled with a class that has maximum neighborhood support that is the nearest neighbor's simple majority vote. For the implementation of this model, scikit-learn was used. Scikit-learn provides KNeighborsClassifier, where k is an integer [22]. The advantage of a bigger k value is that the noise is suppressed but the boundaries of classes will become less distinguishable. The best k-value depends on the data. KNeighboursClassifier [23] also accepts 'weight' as a parameter. Weight parameter is used in prediction. Weight can be either 'uniform' or 'distance'. In the case of 'uniform' weight, equal weights are assigned to all the points of a neighborhood. In case of 'distance' weight, the nearer points are assigned heavier weights and the farther points are assigned lighter weights. The randomized search takes as input, the model and a list of hyper parameters that are tried during training. For this project, different values were tried for the parameters 'K' and 'weight', and it was found out that the algorithm performed better during the recognition process when K = 4, weight = distance.

For training the model, various fonts that resemble the conventional Number Plate fonts were used. Before saving the character, it is transformed to a standard size of 20 x 30 pixels. This is to ensure consistency in input to the model. Figure 6 (a) shows the fonts used and Figure 6 (b) shows the extracted images for the character 'P' as an example.

3. Results and discussion

The experiments are conducted on a Windows 10 based machine with 8GB RAM, and having an i5 processor operating with 2.4 GHz frequency. The python OpenCV library is used for implementing the image processing tools.

Testing of the system has been done with both images and videos. All the fore-mentioned cases, such as irregularly illuminated plates, plates with stylized fonts, plates in close-up, plates far away, angularly skewed plates were considered as a part of System test cases. Images with different environmental conditions are taken for testing. Figure 7 (a) shows an image for testing the case of irregular and small number plates. Figure 7 (b) shows a case of partially worn-out and a standard number plate.

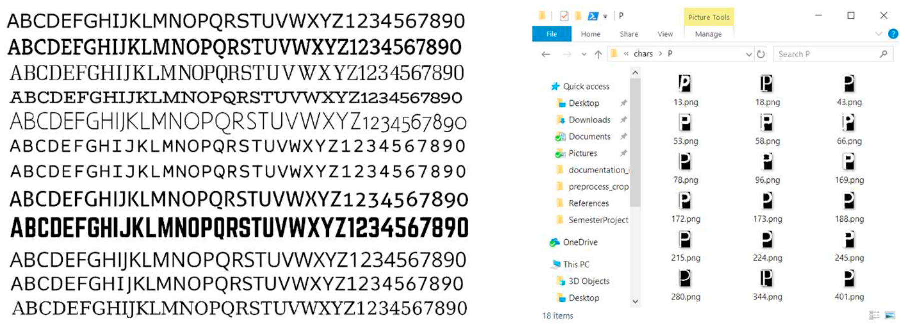

Fig. 6. (a) Fonts used for training; (b) Extracted images for a the letter 'P'

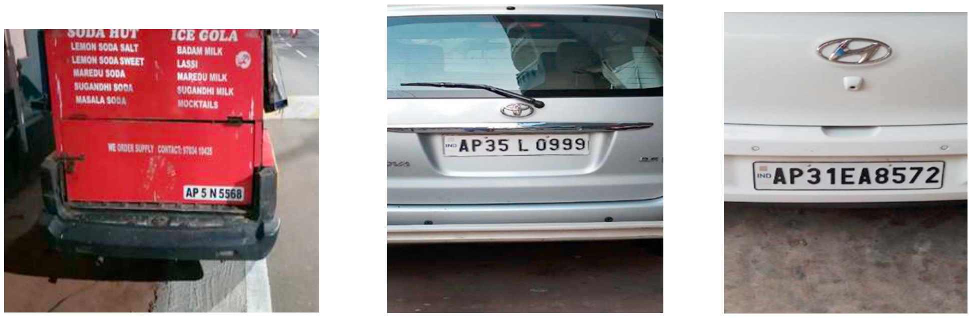

Fig. 7. (a) Irregular illumination and small number plate; (b) A partially worn out number plate; (c) standard number plate

Figure 8 (a) shows a test case considered for non-standard stylish number plate. Figure 8 (b) shows a test case image for the distorted and angled number plate. Figure 9 is an image to test a blurry number plate. Figure 10 shows the outputs of the number plate detection and recognition for all the test cases. The system has successfully detected 98% of the number plates from the provided dataset of 101 plates including Indian and foreign plates, and it successfully recognizes over 96.2% of the characters from plates. Table 1 shows the accuracy of number plate detection and recognition obtained by the methodology proposed in this work. The results are also compared with similar works who worked on Indian number plate detection and recognition. Figure 11 shows the graphical comparison of accuracies of the number plate detection system with other similar works.

3.1 Number Plate Detection

Image Processing based Number Plate Detection is simpler and faster compared to machine learning-based and other complex techniques, as it requires just simple mathematical operations. However, to incorporate a wide range of biases and real-world cases, it will take a lot of work. In such cases, machine learning-based models tend to perform better. Data is a crucial part of any machine learning-based algorithm. In the case of this project, not much data was available. However, it has been proven that in situations where data is a deficit, Image processing techniques perform extremely better than their machine learning-based counterparts. As previously mentioned, it is crucial to have a reliable input source for good detection and recognition accuracy, especially a still camera with good shutter speed and illumination. Although the system happens to perform well for blurred cases, it is not necessarily a generalized expression.

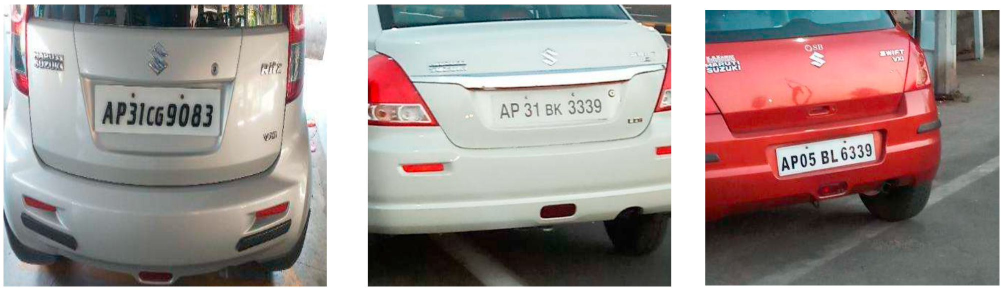

Fig. 8. (a) A number plate with non-standard stylish font; (b) number plate with distorted angle; (c) number plate with distorted angle

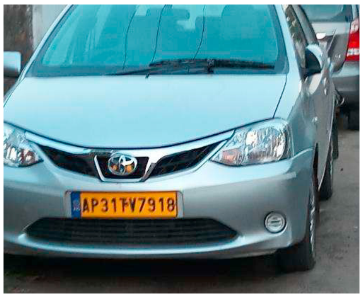

Fig. 9. A blurry number plate

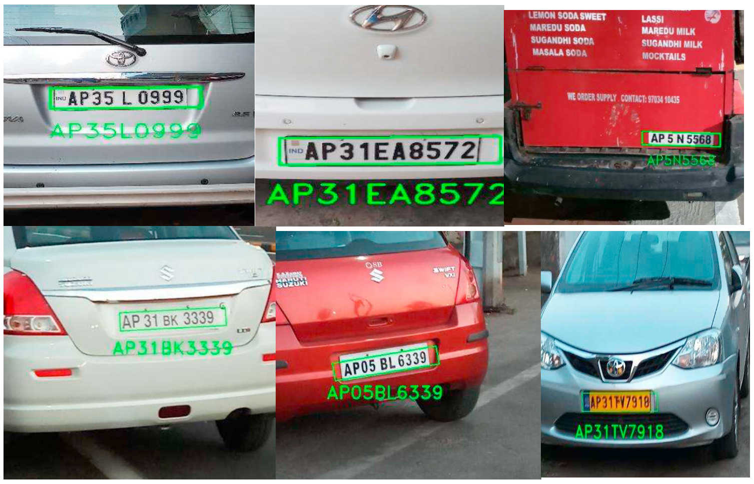

Fig. 10. Number plates detected and recognized

3.2 Number Plate Recognition

Recognition of characters of various fonts was successful to some extent. However, it is important to note that a good balance must be achieved when considering various fonts. Too many fonts can lead to overfitting of the model, resulting in bad generalization and biased prediction. Too fewer fonts will underfit the model, resulting in poor prediction. Also, sophisticated models such as Gradient Boosting tend to overfit the data as not much data was available; This resulted in poor predictions. So, simpler models such as KNeighbours appeared to perform well.

Table 1. Accuracy of the proposed system.

Work	Component	Number of samples	Accuracy
This work	Number plate Detection	101 (Images)	**98.02%**
This work	Character Recognition	768 (Characters)	**96.22%**
[24]	Number plate Detection	250	85%
[24]	Character Recognition	214	80%
[12]	Number plate Detection	95	97.89%
[12]	Character Recognition	-	96.84%

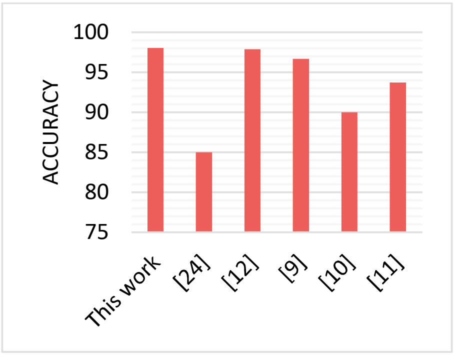

Fig. 11. Comparison of accuracies

4. Conclusion

This work deals with Number Plate Detection and Number Plate Recognition, concerning Indian vehicle number plates or license plates. The major contributions of this work include, consideration of challenging situations like varying illumination, blurred, skewed, noisy images, non-standard and partially worn out number plates. In this work firstly, several image processing techniques viz., morphological transformation, Gaussian smoothing, Gaussian thresholding are used in the pre-processing stage. Next, for number plate segmentation, contours are applied by border following and contours are filtered based on character dimensions and spatial localization. Finally, after the region of interest filtering and de-skewing, K-nearest neighbor algorithm is used for character recognition. For future work, we would like to incorporate a Convolutional Neural Network that combines both detection and recognition into a single framework. CNN's are proven to work well with images, provided that a huge amount of data is available. This is just to avoid overfitting the data and perform better in generalizing unseen data.

References

[1] N. F. Gazcón, C. I. Chesñevar and S. M. Castroa. (2012) "Automatic vehicle identification for Argentinean license plates using intelligent template matching." *Pattern Recognition Letters,* **33 (9)**: 1066-1074.

[2] R. A. Lotufo, A. D. Morgan and A. S. Johnson. (1990) "Automatic number-plate recognition," *IEE Colloquium on Image Analysis for*

Transport Applications, London.

[3] J. A. G. Nijhuis, M. H. Ter Brugge, K. A. Helmholt, J. P. W. Pluim, L. Spaanenburg, R. S. Venema and M. A. Westenberg. (1995) “Car license plate recognition with neural networks and fuzzy logic,” *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth.

[4] Sang Kyoon Kim, D. W. Kim and Hang Joon Kim. (1996) “A recognition of vehicle license plate using a genetic algorithm based segmentation,” *Proceedings of 3rd IEEE International Conference on Image Processing*, Lausanne.

[5] Eun Ryung Lee, Pyeoung Kee Kim and Hang Joon Kim. (1994) “Automatic recognition of a car license plate using color image processing,” *Proceedings of 1st International Conference on Image Processing*, Austin.

[6] Ching-Tang Hsieh, Yu-Shan Juan and Kuo-Ming Hung. (2005) “Multiple License Plate Detection for Complex Background,” *19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Taipei.

[7] Muhammad Tahir Qadri and Muhammad Asif. (2009) “Automatic Number Plate Recognition System for Vehicle Identification Using Optical Character Recognition,” *2009 International Conference on Education Technology and Computer*, Singapore.

[8] P Surekha, Pavan Gurudath, R Prithvi and VG Ritesh Ananth. (2018) “Automatic license plate recognition using image processing and neural networks,” *ICTACT Journal on Image and Video Processing (IJIVP),* **8 (4)**: 1786-1792.

[9] S. Kaur, (2016) “An Automatic Number Plate Recognition System under Image Processing,” *International Journal of Intelligent Systems and Applications,* **8 (3)**:14-25.

[10] Xifan Shi, Weizhong Zhao and Yonghang Shen, (2005) “Automatic License Plate Recognition System Based on Color Image Processing,” *Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2005.Lecture Notes in Computer Science, vol 3483*, Singapore.

[11] Shyang-Lih Chang, Li-Shien Chen, Yun-Chung Chung and Sei-Wan Chen, (2004) “Automatic License Plate Recognition,” *IEEE Transactions on Intelligent Transportation Systems,* **5 (1)**: 42-53.

[12] K Tejas, K Ashok Reddy, D Pradeep Reddy, K P Bharath, R Karthik and M. R. Kumar, (2018) “Efficient License Plate Recognition System with Smarter Interpretation Through IoT,” *Bansal J., Das K., Nagar A., Deep K., Ojha A. (eds) Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing,* 817: 207-220.

[13] Md Yeasir Arafat, Anis Salwa Mohd Khairuddin, Uswah Khairuddin and Raveendran Paramesran, (2019) “Systematic review on vehicular license plate recognition framework in intelligent transport systems,” *IET Intelligent Transport Systems*.

[14] Ayodeji Olalekan Salaua, Thomas Kokumo Yesufua and Babatunde Sunday Ogundare, (2019) “Vehicle plate number localization using a modified GrabCut algorithm,” *Journal of King Saud University - Computer and Information Sciences,* In Press.

[15] “Changing Colour Spaces,” [Online]. Available: https://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html.

[16] Yu Liu, Beibei Dong, Benzhen Guo, Jingjing Yang and Wei Peng, (2015) “Combination of Cloud Computing and Internet of Things (IoT) in Medical Monitoring Systems,” *International Journal of Hybrid Information Technology,* **10 (2)**:366-376.

[17] “Gaussian Blur - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Gaussian_blur.

[18] “Image Filtering - OpenCV,” [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#gaussianblur.

[19] “Image Thresholding - OpenCV,” [Online]. Available: https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html.

[20] “getRotationMatrix2D - Geometric Image Transformations - OpenCV,” [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#getrotationmatrix2d.

[21] “warpAffine - Geometric Image Transformations - OpenCV,” [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#warpaffine.

[22] “sklearn.neighbors.KNeighborsClassifier - scikit-learn docs,” [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[23] “Nearest Neighbours - scikit-learn docs,” [Online]. Available: http://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification.

[24] M M Shidore, and S P Narote. (2011) “Number Plate Recognition for Indian Vehicles” International Journal of Computer Science and Network Security **11 (2):** 143-146