# Multileveled ALPR using Block-Binary-Pixel-Sum Descriptor and Linear SVC

B. Lavanya[1], G. Lalitha[2]
Associate Professor[1], Research Scholar[2]
Department of Computer Science
University of Madras
Chennai-25, Tamil Nadu, India

***Abstract*—Automatic license plate recognition (ALPR) is essential component of security and surveillance. ALPR mainly aims to detect and prevent the crime and fraud activities; it also plays an important role in traffic monitoring. An algorithm is proposed for recognizing license plate candidates. The proposed work aimed to recognize the license plate of a car. Proposed work is designed in multilevel for more accurate License Plate (LP) recognition, At level 1 algorithm produced 93.5% accuracy and in level 3 algorithm gives 96% accuracy. For training and testing purpose, LP images were used from Medialab cars dataset, kaggle car dataset and goggle map images. These images in the dataset is formulated at various angles and illumination. Proposed algorithm for LP recognition is done by using the Block Binary Pixel descriptors (BBPS) and Linear Support Vector Classification (SVC). Proposed algorithm is novel and produces higher accuracy in minimal processing time of an average 0.42 milliseconds with 96% accuracy when compared with state-of-the art methods.**

***Keywords*—BBPS – Block Binary Pixel Sum; ALPR - Automatic License Plate Recognition; ROI - Region of Interest; SVC - Support Vector Classification; LP - License Plate***

## I. Introduction

ALPR is a active, popular and interesting topic in image processing [1], development in license plate recognition have received much attention towards English license plate recognition [2]. It also plays an important role in intelligent transport system [3] [4] [5], border control, toll collection, traffic management [6] [7], it is used in information and communication technology [8]. Computer vision plays an major part in extracting ROI from an image [9]; license plate recognition is done by extracting number plate from images [3] where the Region of Interest(ROI) is license plate of an vehicle, the main task is to extract characters from license plate [10], each vehicle number plate carries different font style and size [11].Surveillance camera is fixed everywhere, earlier it is fixed in important places like malls, hospitals and in sensitive places, but now it's fixed everywhere even people prefer to fix surveillance- camera in their places for their safety. Since many algorithms exists for ALPR but still it lags the accuracy due to various factors like images were captured at night time, tilted and blurred images, so there is a need for an efficient algorithm to improve accuracy. Existing algorithms were proposed using various pre-processing steps followed by feature extraction, machine learning and deep learning techniques. Before extracting features from image pre-processing techniques should be applied on image. Preprocessing the input image which includes conversion of input image to a gray scale image and pre-processing is applied to enhance the details present in the image, in order to highlight the ROI and prune the non-ROI. Then morphological operations like erosion, dilation, tophat, blackhat are applied to enrich input image feature, which supports system proficiency [12]. For recognizing character from image the popular method called optical character recognition with tesseract, an emerging concept is used. ALPR system follows the sequence of steps

- Acquisition of input image
- Localizing license plate
- Segmenting and slicing license plate characters
- Recognizing license plate characters

Since many algorithm exists for ALPR, but it still lags to recognize license plate accurately. So there is a need for an better algorithm to recognize license plate of vehicle accurately even the image captured in uneven lightning, blurred [13]. Proposed work overcomes all those existing disadvantages by recognizing license plate accurately since training of license plate recognition is done by extracting real time license plate images, which tends for more accurate precision. Proposed work is done in multilevel so as to improve the accuracy of proposed algorithm. Many existing algorithm exists for ALPR but its still have many unsolved queries like, it lags to work on images captured at night and dark environment, to recognize license plate of images taken from far distance, to work on skewed and blurred images. It also produces false positive, false negative results for many license plates and existing works fails to predict between characters of similar strokes. So has to solve existing algorithm flaws proposed algorithm is designed in a novel way to overcome all those existing flaws and to recognize license plate accurately from complex backgrounds.

## II. Related Work

ALPR plays an important part in border scrutiny, confirming safeguards, and dealing with vehicle-related crime. ALPR system makes use of deep learning techniques using a convolutional neural network (CNN). The CNN assigns significance to numerous features of the image and differentiating them from each other. According to researchers point of view CNN works good for LP character recognition [3]. ALPR received much consideration for the English license plate. ALPR addresses more frightening traffic dealing with scanty road-safety processes. Architecture of ALPR for predicting vehicle LP regions prior to Vehicle License Plate (VLP) in the ideal is proposed to eliminate false positives causing in higher prediction accuracy of VLP [2]. ALPR plays essential part in current transport organizations such as traffic monitoring and vehicle violation detection. In real-world scenarios, LP recognition faces many contests and is diminished by unrecognized interfering such as weather or lighting conditions. Machine learning based ALPR solutions has been wished-for to resolve such tests in recent years. However, most of the algorithm does not yield convincing results since their consequences are appraised on small or simple datasets that lack varied surrounds, or it require potent hardware to accomplish a practical frames-per-second in real-world solicitations [14]. In ALPR initially, license plate location will be determined. Then, in the second phase, enhancement can be done by applying Gaussian function for filtering. Next, edges in the image located so that LP location can be spotted. Then tilting and plate rotation and affine transformation are applied if the input image was captured in tilted and slanting position or angles .Then by neural network concepts LP characters extracted from LP images [15]. ALPR [16] system plays vital role in Intelligent Transport System (ITS) their major goal was traffic controlling. ALPR for vehicles is a main part of ITS. ALPR send images to a server for LP recognition. To decrease stays and bandwidth usage during images communication, an Edge-AI-based Real-time (ER) ALPR ER-ALPR was proposed, in which an AGX XAVIER entrenched system is embedded on the edge of a camera to attain real-time image input to an AGX edge device and to enable real-time automatic LP candidate recognition. To measure LP characters and styles in a precise setting, the ER-ALPR scheme smears the following methods: (1) image pre-processing (2) You Only Look Once v4-Tiny (YOLOv4-Tiny) [17] for LP prediction; (3) virtual judgment line for determining whether a license plate frame has passed; (4) the proposed modified YOLOv4 (M-YOLOv4) for LP candidate recognition; and (5) a logic ancillary ruling scheme for refining LP recognition. ER-ALPR system was complete in real-life test surroundings in Taiwan. Many state-of-art methodologies exists for recognizing text from various environment, but none of the algorithm recognizes accurately, it lags to differentiate between similar characters, existing algorithm lags to recognize license plate characters from complex environments.

## III. THE PROPOSED WORK

The proposed algorithm consists of three levels and each level has different phases
Level 1: Training phase

Input: Load various input font images.

- Initialize the BBPS Descriptor
- Train separately both character and digit classifier using Linear SVC to recognize LP accurately

Output: Dump the character and digit classifier as a file (pickle file).

**T**esting phase
Input: Load the pickle file (both character and digit classifier) and dataset images.

- Initialize the BBPS Descriptor

Output: Print the license plate text on the image

Level 2: gathering labeled data
This phase is used to train the character and digit classifier for more accurate prediction of LP characters.
Input: Localized license plate images

- Loop over the characters present in each localized LP Images
- Grab each character, create a directory and store it for advanced training

Output: Independent folder created for each character and digits; which is used for advanced training for LP recognition.
Level 3: Advance training phase
Input: Load the sample output generated in level 2(various segmented image stored in labeled folders)

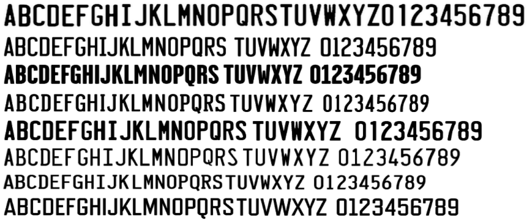
- Repeat level 1 training phase

Output: Dump the character and digit classifier as a file (advance pickle File). Testing phase
Input: Load the advance pickle file (both character and digit classifier) and dataset images and initialize the BBPS descriptor.
Output: Print the license plate text on the image

### A. The Proposed Work (Level 1)

Aim of the proposed work is to propose a powerful algorithm to recognize license plate candidates accurately. Here BBPS descriptor is used to propose an algorithm to extract features of license plate characters. Initially various font images were downloaded from web which is used to train the classifier. Linear SVC is used to train the digit and character classifier separately to obtain higher accuracy. Training dataset were constructed by segmenting LP characters from dataset Images and stored in various labelled folders, which is used in testing phase.

*1) Methodology:* The Objective of the proposed work is to recognize LP images. So a different font style characters and digit dataset images downloaded from web which look alike Fig 1 and used for training the algorithm. For training eight different font style applied image collected from web. Each font displays the characters A-Z and the numbers 0-9. Since we have eight images, we have eight examples for each letter and digit. Proposed algorithm extracts BBPS features from the characters and trained using linear SVC : one for letter recognition and a separate one for digit recognition.



ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
ABCDEFGHIJKLMNOPQRS TUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRS TUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRS TUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRS TUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRS TUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789

Fig. 1. Training set for License Plate Images

*2) Block-Binary-Pixel-Sum Descriptor(BBPS):* BBPS descriptor [18], is used for license plate character recognition. BBPS functions by separating an image into non-overlapping M x N pixel blocks, by applying BBPS descriptor on image ; image will be subdivided on three basis one is 3 x 3 regions, another one is 2 x 3 regions and last one is 3 x 2 sub division as shown in Fig 2. In BBPS target size of the image is fixed, canonical size of ROI is resized so that all images can maintain consistent representation and quantification of each character from dataset images. Input image is converted to a binary image, with pixels corresponding to the character having an intensity > 0, and pixels corresponding to the background set to 0.

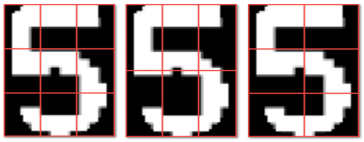
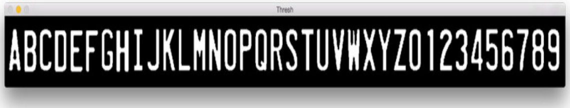


Fig. 2. Example for BBPS Descriptor

*3) Training Classifiers:* License plate has two groups of characters letters followed by digits. In order to obtain better accuracy, both the digit and character classifier should be trained separately. Here linear SVC is used to train a classifier for classification or prediction. Character contains alphabets from string a-z and the digits 0-9. In testing phase all font images will be loaded for training the classifier, convert it to a grayscale image, and threshold operator applied on image so the image look like Fig. 3.



ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

Fig. 3. Training Image after Applying threshold Operator

After thresholding image, contour detection was applied on image to sort our contours from left-to-right, bounding box computed for each of the sorted contours. For each of the sorted contours ROI extracted and passed to BBPS i.e., LP. Training dataset image has characters followed by digits so if value < 26 means it is character otherwise digit. Respective digit and characters updated in appropriate labeled lists. Each character and digit is classified using linear SVC, it will fit the data which is loaded into it will provide, a finest fitting hyperactive plane that rifts or classifies the statistics which feed in to it, linear SVC will dump the data along with its label value as a model. Linear SVC created model will be dumped into pickle file, at the end of training phase two separate pickle file will be created one for character and another for digit. Pickle file which is used for serializing and de-serializing an object structure, which maintains program state across sessions, or transport data over the network. Here we dumped both the classifier in pickle file in order to transmit training data to testing phase.Fig 5 depicts the work flow of level 1 training phase.

*4) Testing Phase:* In this phase two pickle file created in training phase will be loaded along with the testing dataset. From the localized license plate images [12], images were looped over the LP regions bounding box, for each of extracted characters from LP regions [19] applied BBPS descriptor, LP images always carry character followed by digits, So first character classifier is applied to predict features of LP characters followed by digit classifier, length of the bounding box characters were checked if length of the character greater than zero i.e., len(chars)>0 and compute the center of LP regions so has to print the detected strings parallel to the LP region. Fig. 6 depicts the work flow of level 1 testing phase.

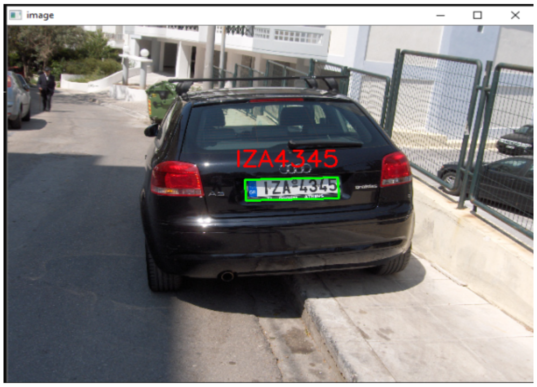


Fig. 4. License Plate Recognition via Proposed Algorithm

After applying the proposed algorithm, LP image look like Fig 4, proposed algorithm gives a good accuracy and prediction rate. Fig 12 shows the successfull LP recognition in level 1. In Fig 13 LP recognized falsely their character M is wrongly recognized has N since both characters looks similar our proposed classifier lags in differentiating nearest character. In Fig 13 digits were misclassification 1 is recognized has 9 so an enhanced proposed algorithm needed and for such wrong redictions. In order to overcome false prediction characters and to improve accuracy, training classifier were trained using real-time license plate images. License plate characters were extracted and labeled has a sample of license character examples from the license plate dataset, both digit and character classifier has been re-trained on top of the BBPS feature representations from the more “real-world” dataset, testing done with the re-trained classifiers which lead to obtain higher character identification accuracy.

## *B. The Proposed Work - Level 2*

- In order to improvise accuracy training classifier done using real-time license plate images.
- LP characters; extracted and labeled has a sample of license character examples from our license plate dataset.
- Both digit and character classifier will be re-trained on top of the BBPS feature representations from the more “real-world” dataset.
- Testing done with the re-trained classifiers which lead to obtain higher character identification accuracy.

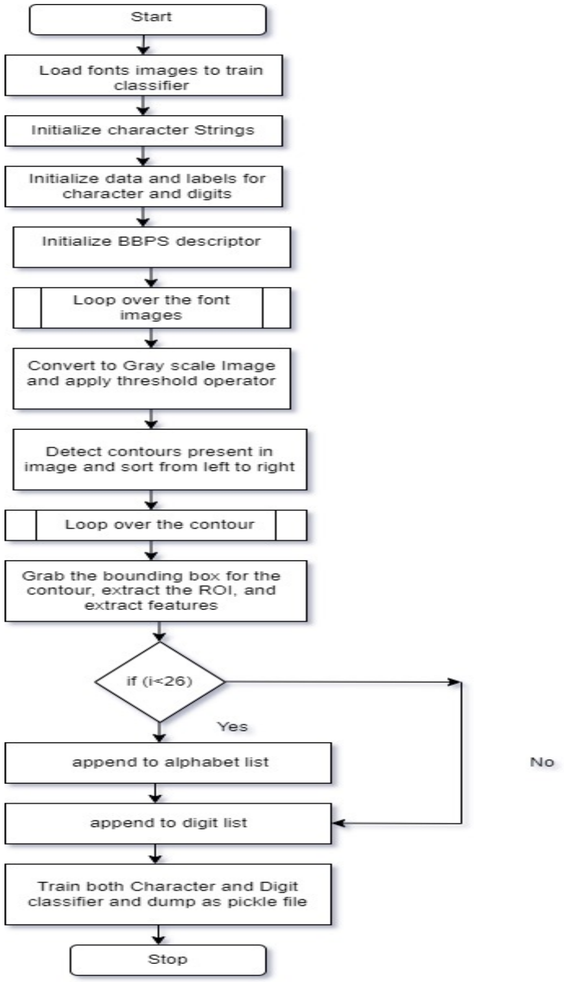

Start
Load fonts images to train classifier
Initialize character Strings
Initialize data and labels for character and digits
Initialize BBPS descriptor
Loop over the font images
Convert to Gray scale Image and apply threshold operator
Detect contours present in image and sort from left to right
Loop over the contour
Grab the bounding box for the contour, extract the ROI, and extract features
if (i<26)
Yes
No
append to alphabet list
append to digit list
Train both Character and Digit classifier and dump as pickle file
Stop

Fig. 5. Flow of Work of Level 1 Training Phase

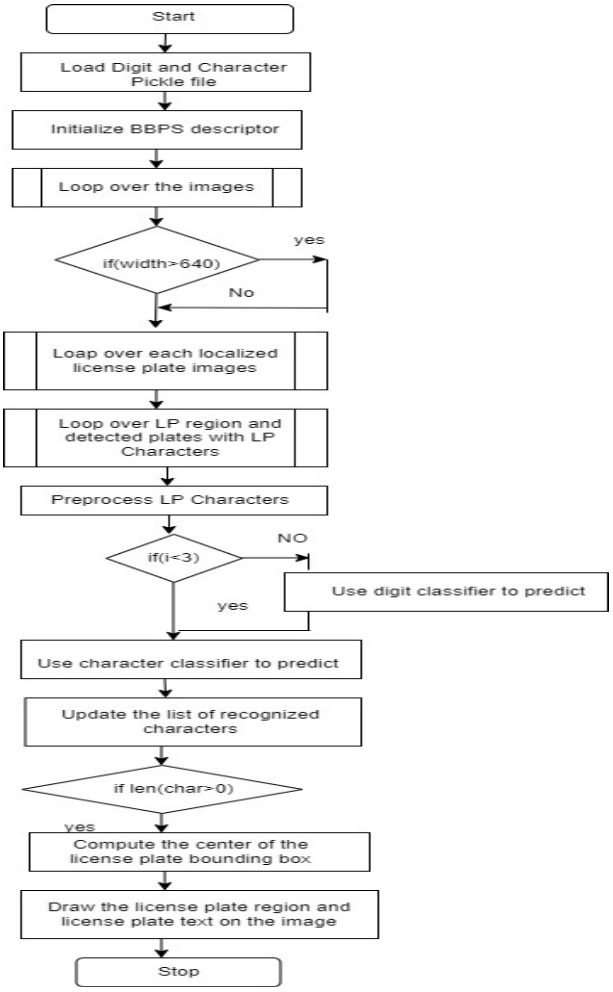

Start
Load Digit and Character Pickle file
Initialize BBPS descriptor
Loop over the images
if(width>640)
yes
No
Loap over each localized license plate images
Loop over LP region and detected plates with LP Characters
Preprocess LP Characters
if(i<3)
NO
Use digit classifier to predict
yes
Use character classifier to predict
Update the list of recognized characters
if len(char>0)
yes
Compute the center of the license plate bounding box
Draw the license plate region and license plate text on the image
Stop

Fig. 6. Flow of Work of Level 1 Testing Phase

Aim for retraining classifier is to obtain a higher accuracy; here classifier is retrained using real-world character examples using LP dataset. LP dataset which used to training is a real-world LP car image dataset so that it can recognize LP characters of real world vehicles. Fig. 7 depicts the work flow of level 2. From a localized bounding box generated images [12]. In second phase initially will randomly select 50% images for training and 50% images for testing. LP localized image with bounding box will be loaded [12], will loop over the LP region with bounding boxes and extract LP characters [19], in that each of the characters from LP images will be displayed and wait for key press event to manually label each of these LP characters like Fig 8.Once characters extracted from LP, it will wait for key press event. If the ‘/  (backtick/tilde) key was pressed, proposed work will ignore the character which is built to ignore falsely detected characters or “noise” in the LP image. Other key press represents confirmation of the character without labelled data; meanwhile appropriate key pressed for characters on screen those characters will be stored with correct label like Fig. 9. Once the LP character has been labelled, it will be written in to disk using a directory structure like filename followed by extension like Fig. 10.

Once characters extracted from license plate, once backtick or tilde key was pressed, proposed work will ignore the character which is built to ignore falsely detected characters or noise in the LP image. Other characters other than other key backtick or tilde key pressed which represents confirmation of the character without labelled data; meanwhile appropriate key pressed for characters on screen those characters will be stored with correct label like Fig. 8. Once the LP character has been labelled, it will be written in to disk using a directory structure like character name/example name.png

By pressing appropriate key while labelling data, character will be saved to folder as like Fig. 10. While comparing with our previous level contrived LP fonts samples these labelled LP characters are certainly representative of real-world LP characters like Fig. 11. In Level 2 labeled characters directory was created with real-time LP images, since web downloaded font image were trained and used in level 1 which lags to give exact recognition at some cases since there is lots of difference between web downloaded image and real LP images, so if training classifier also done with real LP images which gives good accuracy when testing with real LP images. In level 3 from LP dataset images using BBPS extract features, and retrain the character and digit classifier.

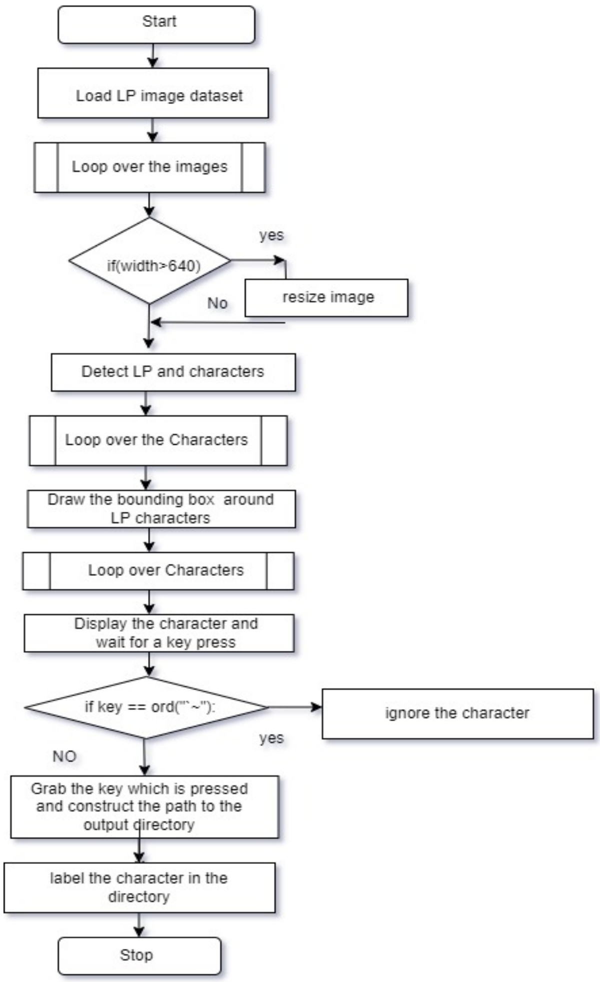

Start
Load LP image dataset
Loop over the images
if(width>640)
yes
resize image
No
Detect LP and characters
Loop over the Characters
Draw the bounding box around LP characters
Loop over Characters
Display the character and wait for a key press
if key == ord("`~"):
ignore the character
yes
NO
Grab the key which is pressed and construct the path to the output directory
label the character in the directory
Stop

Fig. 7. Flow of Work of Level 2 Gathering LP Characters

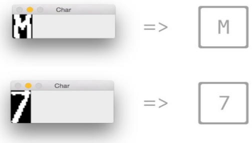


Fig. 8. Appropriate Key Press while Labeling LP Characters

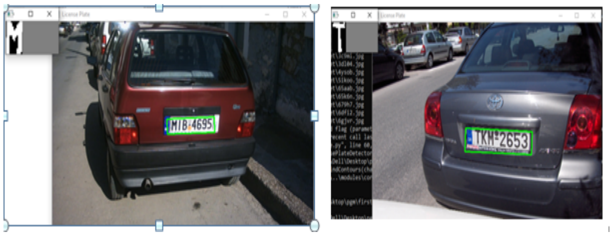


Fig. 9. Labeling LP Character Examples

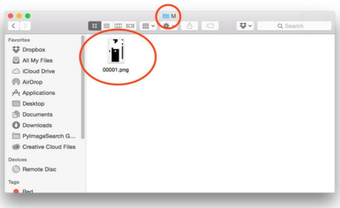


Fig. 10. Output ROI Stored in Subdirectory

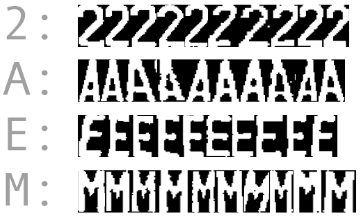


Fig. 11. Sample of Labeled LP Characters

## C. The Proposed Work - Level 3

From the gathered labeled directory (o/p of level 2). BBPS descriptor was applied to extract license plate characters. Re-trained letter and digit classifiers were used to re-train classifiers to recognize license plate characters accurately. As a beginning step in level 3, BBPS descriptor will be initialized and sample directory which is created in level 2 will be loaded, each of the images will be loaded, pre-processed and finally BBPS features extracted from those sample images. Accordingly like level 1 digit and character classifier trained and updated and dumped into separate pickle file.

As a beginning step in level 3, BBPS descriptor will be initialized and sample directory which is created in level 2 will be loaded, each of the images will be loaded, pre-processed and finally BBPS features extracted from those sample images. Accordingly like level 1 digit and character classifier trained and updated and dumped into separate pickle file.

# IV. Dataset

To recognize LP candidates, large collection of dataset required. Publishing large collections of license plates and any information regarding where the license plates were captured including Global Positioning System (GPS) coordinates, noticeable landmarks, street signs, etc. is actually considered an invasion of privacy in many countries [12]. It's really hard for ALPR standpoint where in google and their street view initiative have amassed one of the largest datasets of license plates in the entire world. The google street view cars have driven countless miles around the world, passing millions upon millions of cars and trucks all with license plates clearly visible; images where blur and license plate characters were looks so smudged [19]. For the proposed work dataset images taken from a popular dataset repository called as medialab which is maintained by a national university in greece, kaggle car dataset and google map images. Proposed work trained and tested with medialab images of 139 combined with kaggle car dataset 55 images and google map images 20, totally 214 images were trained and tested. Dataset images were localized [12], sliced and segmented [19] proposed work is a continuation of previously proposed license plate localization [12], segmentation and scissoring [19].

## V. Result and Discussion

The proposed work license plate recognition is done using medialab dataset, kaggle car image dataset and Goggle map images. Proposed algorithm build using BBPS descriptor implemented in anaconda python. The experiment is done in laptop with intel core i5, 2.70 GHz, 8 GB RAM windows 10-64-bit OS. On execution time all the images were rescaled and maintained the size of 640 pixels with unchanged aspect ratio. Proposed algorithm gives very good accuracy compared to existing works. Proposed work is novel and works better than the state-of-the-art methodologies. Existing works lags to recognize license plate from the blurred, sweked images captured in uneven lightning and fails to recognize license plate characters of images taken from longer distance, proposed algorithm works on all the above cases so its a novel and works better than state-of-the-art methodologies. Proposed algorithm is simple to understand, easy to implement in real time environment which gets executed in 0.42 milliseconds. Proposed algorithm is proposed in such a way to overcome all the flaws of existing methodologies.

Fig. 12 depicts the successful true positive result in level 1. Here license plate were successfully recognized whereas in Fig. 13 in left side image all candidates were recognized correctly except the second character candidate which is recognized has N instead of M. In Fig. 13 all character candidates were recognized correctly except a digit ‘1’ which is recognized has ‘9’. All these misclassification occur because all these candidates share similar features.

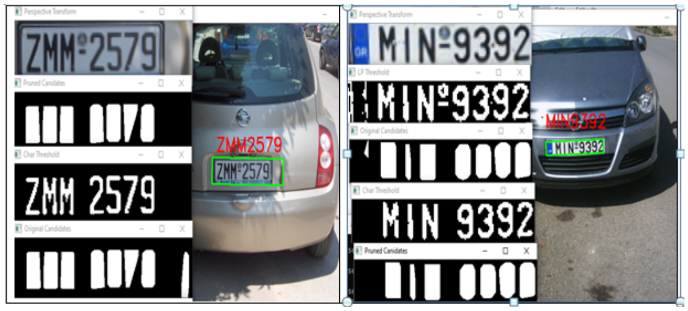


Fig. 12. Successful License Plate Recognition in Level 1

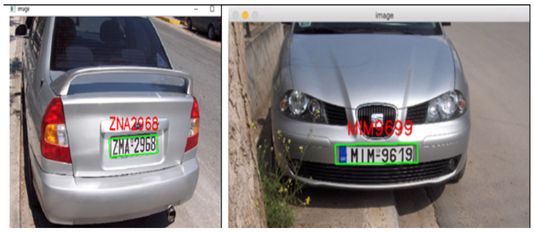


Fig. 13. False Prediction in Level 1

In level 1 training of LP characters were done using different font images downloaded from internet, level 1 yields good accuracy but it fails to recognize a few cases, in order to overcome such flaws, training of classifier is done by gathering characters from dataset images like Fig 14. In Fig. 15 depicts the wrongly predicted license plate images in level 1 which is correctly recognized in level 3.

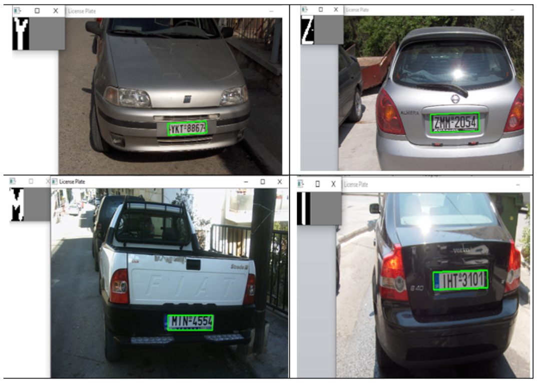


Fig. 14. Gathering LP Characters in Level 2

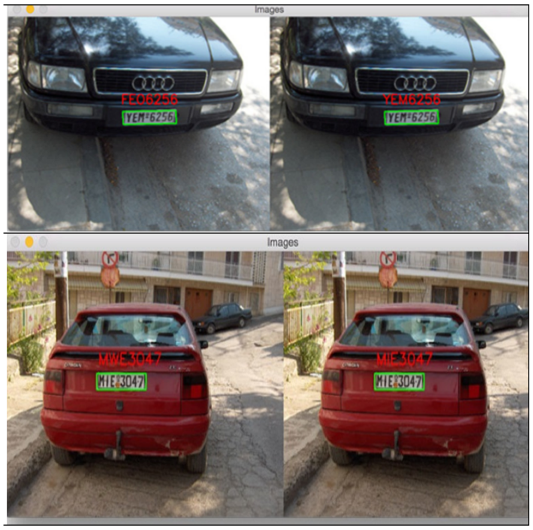


Fig. 15. Wrongly Recognized Vehicle LP in Level 1 Predicted Correctly in Level 3

## VI. Pseudocode

**Level 1: Training Phase**

```
initialize characters string, data and labels for the alphabet and digits
initialize BBPS descriptor
loop dataset images, convert to grayscale
and apply thresholding
    detect contours; sort from left to right
    loop over contours, grab bounding box of contour
    extract ROI
        if i <26
            it is character
        else
            digit
train character and digit classifier
dump the character and digit classifier to a file
```

**Testing phase**

```
def testing phase()
    load the character and digit classifiers created in training phase
    initialize BBPS descriptor
    loop dataset images
```

```
        if width > 640,
            resize(image)
        detect the license plates [12] and characters [19]
        loop over detected plates and characters
        preprocess it
            if i > 3
                call digit classifier
            else
                call character classifier
            update the text of recognized characters
        if len(chars)>0
            compute the center of LP bounding box
            draw the characters which are predicted above bounding box
```

**Level 2: gathering LP characters**

```
randomly select a portion of the images
loop dataset images
    if width> 640
        resize(image)
    detect the license plate [12] and characters [19]
        draw bounding box around recognized
        license plate
        loop over the characters
        display the character
        if actual != segmented character
            print(ignore)
            continue
        construct the path to the output directory
        if output directory not exist, create directory
            write the labeled character to file
```

**Level 3: Advanced training phase**

```
initialize the BBPS descriptor
initialize the data and labels for the alphabet and digits
loop over the sample dataset created in level 2
    extract the images
    loop over the images
        load the character, convert it to grayscale
        preprocess it
        if the character is digit
            append digit data and label
        otherwise
            append alphabet data and label
train both advanced character and digit classifier
dump the character and digit classifier to file
```

**Advanced testing phase**

```
Input: Load advanced character and digit classifier
call testing phase()
```

## VII. Performance Evaluation

In the proposed work in level 1 few LP character candidates were predicted wrongly, so to overcome such flaws training images were created using medialab, kaggle car dataset and Google map images. To evaluate the performance of proposed system accurately, here we have trained and tested the proposed system with three dataset images, which has images captured at various angles, various lightning condition. In a Tunisian dataset, [20] which is used for calculating accuracy for LP recognition, which comprised of true positive and true negative divided by true positive, true negative, false positive and false negative boundary boxes. Table 1 depicts different dataset images used, the performance metrics of proposed system is calculated in terms of measuring precision, recall, f-measure. Proposed algorithm recognize dataset images more accurately than the existing algorithm, it recognizes images captured at challenging environment and angles. Performances of proposed algorithm is produced in table format. If the license plate correctly predicted from license plate images which come under True Positive (TP), if non-license plate region recognized has license plate which called as False Positive (FP). Since the proposed work is done in multilevel performance evaluation has been done for each level independently in Tables 2, 3 and 4 depicts the true positive, false positive values from all the three levels of proposed work.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F - measure = \frac{2 * Precision * Recall}{Precision - Recall} \tag{3}$$

TABLE I. Different Dataset Images

| Dataset Name | No.of.Images |
|---|---|
| Medialab | 139 |
| Kaggle dataset | 55 |
| Google map images | 20 |

TABLE II. Confusion Matrix for Level 1

| | Predicted Negative (PN) | Predicted Positive (PP) |
|---|---|---|
| Actual Positive(AP) | 11 | 17 |
| Actual Negative(AN) | 0 | 204 |

TABLE III. Confusion Matrix for Level 2

| | Predicted Negative (PN) | Predicted Positive (PP) |
|---|---|---|
| Actual Positive(AP) | 0 | 7 |
| Actual Negative(AN) | 0 | 207 |

TABLE IV. Confusion Matrix for Level 3

| | Predicted Negative (PN) | Predicted Positive (PP) |
|---|---|---|
| Actual Positive(AP) | 3 | 13 |
| Actual Negative(AN) | 0 | 198 |

In prediction phase if both the license plate character candidate and predicted character candidate were same means its true positive, if non LP region predicted has LP candidate means its false positive, if non license plate region not recognized has license plate region its true negative, if non license plate region detected has license plate region means false negative. Equations for precision, recall , f-measure given in eq(1),(2),(3). Precision, recall, f-measure were metrices which is used to test performance of an algorithm. Precision is a measure which can be used when the count of false positive is less. Recall is a metrics which is used to improve the prediction rate when it lag to produce satisfying result in precision phase; this metrics can be used when the value of false negative is less. F1 measure can be used when the accuracy seek the balance between precision and recall, when class distribution is uneven i.e more actual negatives. Table 5 depicts the precision, recall, f-measure score of proposed work.Table 6 depicts the proposed methodology accuracy with existing works.

TABLE V. Precision, Recall, F-Measure Values

| | Precision | Recall | F1-score |
|---|---|---|---|
| Level-1 | 92.3 | 94.8 | 93.5 |
| Level-2 | 96.7 | 100 | 98.3 |
| Level-3 | 93.8 | 98.5 | 96 |

TABLE VI. Comparison with Existing Works

| S.no | Author's | Precision | Recall | F-measure |
|---|---|---|---|---|
| 1 | Proposed algorithm | 93.8% | 98.5% | 96% |
| 2 | Omar et al [10]., | 94.43% | 92.10% | 91.01% |
| 3 | Al-Shemarry et al [21]., | 91.6% | 87.1% | 89.33% |

Proposed work evaluated with popular dataset like medi-alab, kaggle, and Google map images. All images were used for both training the classifier and testing license plates. Fig. 16 depicts the final outcome by kaggle dataset images and Fig. 17 is a sample output for successful license plate recognition for goggle map images.

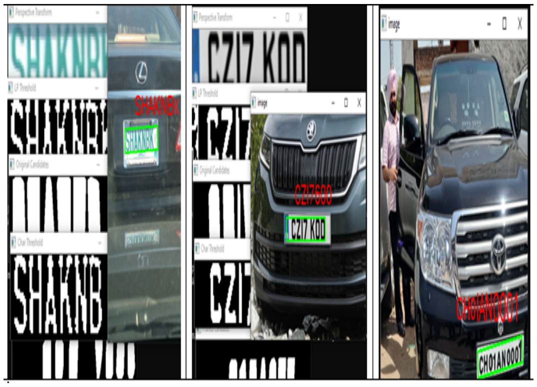


Fig. 16. Successfull License Plate Recognition using kaggle Dataset

## VIII. Conclusion

ALPR is an interesting hot topic, many algorithm exists for ALPR but it still lags on recognizing license plate characters from complex environment, by overcoming all the backdrops in existing system a novel and efficient algorithm produced here to recognize license plate characters accurately from complex environment. In this proposed work, license plate is recognized from media-lab,kaggle car dataset images and

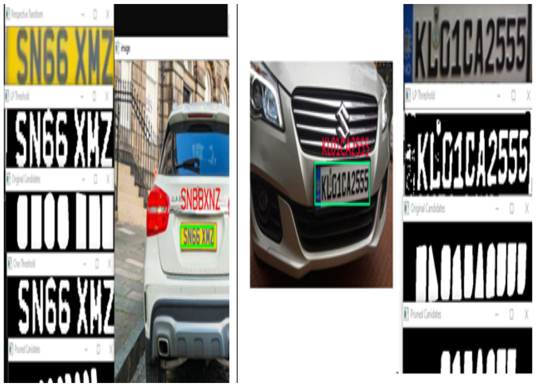


Fig. 17. Successfull License Plate Recognition using Goggle Map Images

from Google map images. Many existing algorithm exist for LP recognition but it lags to provide good accuracy due to various factors like images were captured in uneven lightning, taken at tilted position and blurred images. Existing algorithms are tedious, time consuming and it requires labeled data, those were expensive to compute, and highly time consuming. Proposed algorithm is novel, built using BBPS descriptor and linear SVC which is super-fast, run easily in real-time, and its simple to comprehend, its inexpensive to compute, and executes in 0.42 milliseconds. In level 1 proposed work produced 93.5% accuracy, In level 2 gathered training dataset images and which gave 98.3% in recognizing and training dataset images which is used as a input to level 3 . level 3 produced an enhanced better results than proposed level 1 and other existing methodologies. As a future work license plate of regional language images will be recognized.

## Acknowledgment

This research did not receive any specific grant from funding agencies in the public,commercial,or not-for-profit sectors.

## References

[1] Z. Selmi, M. B. Halima, U. Pal, and M. A. Alimi, "Delp-dar system for license plate detection and recognition," *Pattern Recognition Letters*, vol. 129, pp. 213–223, 2020.

[2] M. Onim, S. Hassan, H. Nyeem, K. Roy, M. Hasan, A. Ishmam, M. Akif, A. Hoque, and T. B. Ovi, "Blpnet: A new dnn model and bengali ocr engine for automatic license plate recognition," *arXiv preprint arXiv:2202.12250*, 2022.

[3] P. Kaur, Y. Kumar, S. Ahmed, A. Alhumam, R. Singla, and M. F. Ijaz, "Automatic license plate recognition system for vehicles using a cnn," *CMC-COMPUTERS MATERIALS & CONTINUA*, vol. 71, no. 1, pp. 35–50, 2022.

[4] J. Liang, G. Chen, Y. Wang, and H. Qin, "Egsanet: edge–guided sparse attention network for improving license plate detection in the wild," *Applied Intelligence*, vol. 52, no. 4, pp. 4458–4472, 2022.

[5] W. Al Faqheri and S. Mashohor, "A real-time malaysian automatic license plate recognition (m-alpr) using hybrid fuzzy," *International Journal of Computer Science and Network Security*, vol. 9, no. 2, pp. 333–340, 2009.

[6] R. Laroca, L. A. Zanlorensi, G. R. Gonçalves, E. Todt, W. R. Schwartz, and D. Menotti, "An efficient and layout-independent automatic license plate recognition system based on the yolo detector," *arXiv preprint arXiv:1909.01754*, 2019.

[7] K. Kluwak, J. Segen, M. Kulbacki, A. Drabik, and K. Wojciechowski, “Alpr-extension to traditional plate recognition methods,” in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2016, pp. 755–764.

[8] P. Agarwal, K. Chopra, M. Kashif, and V. Kumari, “Implementing alpr for detection of traffic violations: a step towards sustainability,” *Procedia Computer Science*, vol. 132, pp. 738–743, 2018.

[9] A. Ashrafee, A. M. Khan, M. S. Irbaz, A. Nasim, and M. Abdullah, “Real-time bangla license plate recognition system for low resource video-based applications,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 479–488.

[10] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, “Automatic license plate recognition (alpr): A state-of-the-art review,” *IEEE Transactions on circuits and systems for video technology*, vol. 23, no. 2, pp. 311–325, 2012.

[11] W. Al Faqheri and S. Mashohor, “A real-time malaysian automatic license plate recognition (m-alpr) using hybrid fuzzy,” *International Journal of Computer Science and Network Security*, vol. 9, no. 2, pp. 333–340, 2009.

[12] G. Lalitha and S. Gopinathan, “An effective algorithm for detecting and localizing license plate images,” *Journal of Fundamental & Comparative Research (Shodhsamhita)*, 2021.

[13] R. M. Khoshki and S. Ganesan, “Improved automatic license plate recognition (alpr) system based on single pass connected component labeling (ccl) and reign property function,” in *2015 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2015, pp. 426–431.

[14] S.-R. Wang, H.-Y. Shih, Z.-Y. Shen, and W.-K. Tai, “End-to-end high accuracy license plate recognition based on depthwise separable convolution networks,” *arXiv preprint arXiv:2202.10277*, 2022.

[15] O. Akbarzadeh, M. R. Khosravi, and L. T. Alex, “Design and matlab simulation of persian license plate recognition using neural network and image filtering for intelligent transportation systems,” *ASP Transactions on Pattern Recognition and Intelligent Systems*, vol. 2, no. 1, pp. 1–14, 2022.

[16] C.-J. Lin, C.-C. Chuang, and H.-Y. Lin, “Edge-ai-based real-time automated license plate recognition system,” *Applied Sciences*, vol. 12, no. 3, p. 1445, 2022.

[17] M. Usama, H. Anwar, M. M. Shahid, A. Anwar, S. Anwar, and H. Hlavacs, “Vehicle and license plate recognition with novel dataset for toll collection,” *arXiv preprint arXiv:2202.05631*, 2022.

[18] Y. Chang, H. Hsu, J. Lee, C. Chueh, C. Chang, and L. Chen, “License plate character recognition using block-binary-pixel-sum features,” in *International Conference on Computer, Networks and Communication Engineering (ICCNCE 2013)*. Atlantis Press, 2013, pp. 111–113.

[19] G. Lalitha and S. Gopinathan, “An enhanced algorithm for segmenting and slicing license plate characters via cc analysis,” *Journal of Education: Rabindra Bharathi University*, 2021.

[20] S. Ktata, T. Khadhraoui, F. Benzarti, and H. Amiri, “Tunisian license plate number recognition,” *Procedia Computer Science*, vol. 73, pp. 312–319, 2015.

[21] M. S. Al-Shemarry, Y. Li, and S. Abdulla, “An efficient texture descriptor for the detection of license plates from vehicle images in difficult conditions,” *IEEE transactions on intelligent transportation systems*, vol. 21, no. 2, pp. 553–564, 2019.