RESEARCH ARTICLE **SPECIAL ISSUE**

RSP Science Hub

International Research Journal on Advanced Science Hub
2582-4376
Vol. 05, Issue 05S May
www.rspsciencehub.com

Check for updates

http://dx.doi.org/10.47392/irjash.2023.S038

International Conference on intelligent COMPUting TEchnologies and Research (i-COMPUTER) 2023

Automatic License Plate Recognition System Using YOLOv4

Amogh Mohta[1], Anand Swaroop [1], Katkuri Fhanindra Reddy [1], Manjula R [2]

[1]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

[2]Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, India

Email: amoghmohta27@gmail.com

Article History

Received: 28 February 2023
Accepted: 19 March 2023

***Keywords**:*

ANPR;
Tesseract OCR;
YOLOv4;
OpenCV

Abstract

In this research paper, we'll talk about ALPR technology, which has gained popularity recently because of all the many ways it may be used. The fundamental benefit of this technology is that it may be utilized for a variety of purposes, including travel time analysis, intelligent parking, automated toll collection, intelligent transportation systems in smart cities, and traffic management. Automated License Plate Recognition (ALPR) reads the vehicle's registration number by first using YOLOv4 for object recognition following which we use OpenCV to enlarge the license plate image and identify the character boxes after which we use Tesseract optical character recognition to identify the various characters and form the license plate number. This system uses several image processing methods to recognize automobiles swiftly and automatically in video or picture material. As technology develops quickly with the introduction of machine learning and deep learning, the cost of computing falls, and the accuracy of used image processing methods rises, the usage of ALPR systems is becoming more widespread. In today's congested traffic system, a license plate detection system is crucial. It aids in monitoring compliance with traffic laws and other law enforcement operations.

There are many instances of reckless driving in India when vehicles break several traffic laws. As a result, a license plate detection system has been suggested and put into use throughout the years to assist with quick and simple traffic law enforcement by automobiles. This work offers a powerful method for character localization, segmentation, and identification inside the located plate. We are going to utilize tesseract OCR and the YoLo V4 approach to solve the License plate recognition system issue and deliver our suggested system with high accuracy.

1. Introduction

The focus of the proposed work is to detect and recognize the License plate of a moving vehicle from video and camera photos with high accuracy and detailed quality. (Kashyap et al.) The method known as "license plate detection and recognition" makes use of computer vision to find and identify a license plate from an input picture of an automobile. This technique has several applications. (Prabhakar, Anupama, and Resmi) It is used to track down vehicles that are violating traffic laws on the road. It is used in security to record the license plates of the cars entering and exiting certain locations. It is used to

record the license plates of the vehicles parked in parking lots. (Md, Asaduzzaman, and Islam) The license of uses for it is endless. In this research paper, we're going to suggest a system that automates the process of identifying offenders captured on camera, reducing the manual labor, cost, and time required to do that. It also helps with parking management at many locations where human interaction is necessary, so with the help of ANPR it can be reduced to zero. With the aid of rapid and simple license plate identification, we may also utilize it in auto tool booth payments. (Ganta, Srikanth, and Svsrk)

In our suggested system, we will use Python to achieve the answers to this kind of challenge. Python enables us to build our application for detecting and identifying license plates. (Kanagapushpavalli and devi) Three of its libraries—tesseract OCR, imutils, OpenCV, and Yolov4—are used to do this. There are three main procedures that software goes through in order to detect and identify a license plate: - Using a picture or video of a car as input: - The software receives as input the picture or video of the vehicle where the license plate is to be found.

(1) Processing the input: - The picture is processed to identify the area of the input that contains the license plate.

(2) Identifying the license plate: - The detected license values plate is taken from the video or photograph of the license plate.

2. Related Work

We've read through various research papers and other pieces that are pertinent to our work in this part. In some of the research papers they have used the Neural Networks for detection, segmentation, and recognition of the license plate of a vehicle and some of them also used the CNN algorithm for the recognition of license plate. In some of the research papers they used the various classifier-based datasets and convolution Neural Network for recognition. (Gnanaprakash, Kanthimathi, and Saranya)

Many methods are used to detect the license plate system in which CNN, KNN along with the classifier like haar-cascade and SVM is increased day by day. But in this paper, we are going to use the different methods which is YOLO V4, OCR method,Single Shot Multibox Detector and You Only Look once. (C. Patel, Shah, and A. Patel)

In YOLO, the grid division is in charge of detection and confidence loss; in YOLO V2, Anchor with K-means added, two-stage training, and full convolutional networks; in YOLOV3, multi-scale detection using FPN; and in YOLO V4,SPP, MISH activation function, data enhancement Mosaic/Mix-up, and GIOU (Generalized Intersection over Union) loss function; YOLO V4 is a deep learning object detection algorithm, which is very fast and more accurate than any other detection algorithm. (Badr et al.) It is trained on COCO dataset for training purposes and has one stage detector which makes it faster. The working of YOLO is based on first it divides the image in 13*13 of which each cell detects five bounding boxes that gives the confidence score which shows how confident the algorithm is that object to detect exists in the bounding boxes. YOLO takes comparatively less time to train the model than any other algorithm which makes it superfast and can be used on the single GPU. When employing multibox to detect numerous things present in an image, single shot detectors like YOLO only need one shot. It is a very accurate object detection technique that is much faster. (Sharma and Gajendra)

3. Proposed System

3.1. License Plate detection:

1. The first step of the procedure is to take the bounding box coordinates from YOLOv4 and the sub-image area within the box. As this picture is fairly little, we often use cv2.resize() to increase its size by three times.

FIGURE 1. License Plate Detection

2. The picture is then changed to grayscale, and a slight Gaussian blur is used to blur the edges.

6. As can be seen, this results in the discovery of several contours in addition to the contours of each character in the license number. A handful of parameters must be matched in order to approve a contour in order to filter out the undesirable regions. (Valdeos et al.) These parameters are

FIGURE 2. Grayscale of the License Plate

FIGURE 3. White text on dark background

FIGURE 4. Enlarged Image

FIGURE 5. Identifies all rectangular-shaped contours

the ratios of the height and width (i.e., the region's height must be at least one-sixth of the image's entire height). Many other criteria on region, region size, etc. are also placed. See the source code for specifics. This filtering reduces the remaining regions of interest to the individual characters of the license number. Tesseract is more precise with black text on a white background; therefore, we partition each sub image and apply a bitwise not mask to invert the image to black text on a white backdrop. (Kaur and Sarbjit) Then, a tiny median blur is applied to the picture before giving it to Tesseract for letter or number extraction. Illustration of how transformed letters seem in tesseract format.

4. Software Used

- Tesseract OCR
  - Python
  - Imutils
  - YOLO V4
  - Open CV

FIGURE 6. Filtering Image to remove spaces

FIGURE 7. Filtered Image

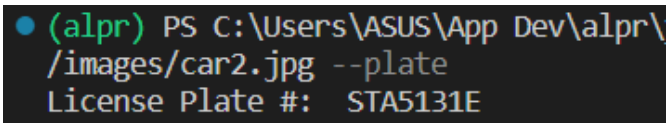

FIGURE 8. Code for License Identification

5. Ease of use

Your existing experience with computer vision and OCR technologies, as well as your knowledge with the underlying programming languages and libraries, all play a part in how simple it is to use OpenCV, Tesseract OCR, imutils, and Python for license plate detection.

A lot of tools for image processing and computer vision tasks are provided by the widely used computer vision. (Shariff et al.)

library OpenCV. If you have prior knowledge of computer vision or image processing, it is comparatively simple to use.

Tesseract OCR is a well-known and extremely accurate optical character recognition engine. It is simple to use for OCR tasks thanks to its interface with Python and OpenCV.

Imutils is a computer vision library that offers a selection of practical functions for fundamental image processing and computer vision applications. It is simple to use, especially for those who are familiar with Python, and it streamlines and simplifies routine computer vision operations. (Ap et al.)

Overall, your prior knowledge and expertise with the

libraries and programming languages involved will determine how simple it is for you to use OpenCV, Tesseract OCR, imutils, and Python for license plate detection. (Khinchi and Agarwal)

However, there is a sizable user community that can offer support and guidance, and the libraries are well known, frequently used, and well- documented.

6. Methods and Models

6.1. Abbreviations and Acronyms

YOLO ("You Only Look Once"), OCR ("Optical Character Recognition"), GIOU (Generalized Intersection over Union), ALRP (Automatic License Plate Recognition) (Bochkovskiy, Wang, and Liao)

6.2. Model Description

(I) Equations Used

1. height, width = im2.shape - This line gets the height and width of the grayscale image used for character segmentation.
2. if height / float(h) > 6: continue - This line checks if the height of the bounding box is at least 1/6th of the total height of the grayscale image. If the condition is not met, the loop skips to the next contour.
3. ratio = h / float(w) - This line calculates the height-to-width ratio of the bounding box.
4. if ratio < 1.5: continue - This line checks if the height-to-width ratio is greater than or equal to 1.5. If not, the loop skips to the next contour.
5. if width / float(w) > 15: continue - This line checks if the width of the bounding box is at least 1/15th of the total width of the grayscale image. If not, the loop skips to the next contour.
6. area = h * w - This line calculates the area of the bounding box.
7. if area < 100: continue - This line checks if the area of the bounding box is at least 100 pixels. If not, the loop skips to the next contour.
8. rect = cv2.rectangle(im2, (x1,y1), (x1+w1, y1+h1), (0, 255, 0), 2) - This line draws a rectangle around the current contour in the grayscale image used for character segmentation.
9. regofi = thresh [y1-5: y1+h1+5, x1-5:x1+w1+5] - This line extracts the region of interest (ROI) from the thresholded image using the coordinates of the bounding box. An additional 5 pixels are addedto each side of the ROI.
10. regofi = cv2.bitwise_not(regofi) - This line performs a bitwise notoperation on the ROI to make the characters black and the background white.
11. regofi = cv2.medianBlur(regofi, 5) - This line applies a median blur filterto the ROI to remove any remaining noise.
12. LicenseText = pytesseract.image_to_string(regofi, config='-c tessedit_char_whitelist=0123456789ABCDEFGHIJKLMNOPQR STUVWXYZ –psm 8 –oem 3') - This line uses the Tesseract OCR engine to recognize the characters in the ROI. The charactersare restricted to alphanumeric characters and are assumed to be in a single line (PSM 8) with a combination of LSTM and OCR engine modes (OEM 3).
13. clrText = re.sub('[\W_] +', '', LicenseText) - This line removes any non-alphanumeric characters from the recognized text using a regular expression.
14. NumPlate += clrText - This line appends the cleaned text to a string variable that will store the recognized license plate number.
15. if NumPlate != None: - This line checks if the license platenumber has been successfully recognized.
16. Print ("License Plate Number #: ", NumPlate) - This line printsthe recognized license plate number to the console.
17. return NumPlate - This line returns the recognizedlicense plate number as a string.

6.3. Algorithm Outline

1) Separate coordinates from box.
2) Get the sub image comprising the bounded region and add 5 pixels on either side.
3) Get the sub image comprising the bounded region and add 5 pixels on either side.
4) Comparison Table of different detectors is shown below in Table 1.

TABLE 1. The accuracy of each character detector

Detector	F1 - Score	IoU	mAP@0.5
YOLOv3 (416)	0.93	84.91	71.53
YOLOv4 (256)	1	94.25	98.36
YOLOv4 (608)	1	89.55	98.12

5) Use a gaussian blur to smooth the picture.
6) Use a gaussian blur to smooth the picture.
7) Generate a rectangle kernel for dilation.
8) Use dilation to make sections more visible.
9) Locate the contours of areas of interest inside the license plate.

10) Sequence contours from left to right.
11) Generate grey image duplicate.
12) Build an empty string to contain the license plate number.
13) Locate specific letters and numerals in a license plate by iterating through the shapes.
14) If height of box is not tall enough compared to total height, skip.
15) If height to width ratio is less than 1.5, skip.
16) If width of box is not wide enough related to total width, skip.
17) If area is less than 100 pixels, skip.
18) Create the rectangle.
19) Capture image's character area.
20) Perform bitwise not reverse picture to black text on a white backdrop.
21) Blur character region again.
22) Remove any undesired blank spaces from the tesseract text.

7. Results and Discussion

Comparison:

The object detection algorithm YOLOv4 performs better than its forerunner YOLOv3 in terms of accuracy. With an input size of 256, it achieves a perfect F1-score of 1, an IoU of 94.25%, and a mean average precision (mAP) of 98.36% at a threshold of 0.5. It receives a perfect F1-score of 1 and a mAP of 98.12% with an input size of 608.

TABLE 2. Computational time for each method

	GeForce RTX 2080 Ti		NVIDIA JETSON TX2	
Detector	Time (ms)	FPS	Time (ms)	FPS
SSD	51	19.6	74	13.5
SSD (TensorRT)	4.38	228.3	32.5	30.77
YOLOV4 (256)	9	111	105	9.5
YOLOv4 (608)	25	40	470	2.

The approach and hardware employed affect the computing time for object detection. The best performance comes from SSD with TensorRT optimization, with processing times of 4.38ms and

228.3 frames per second on GeForce RTX 2080 Ti and 32.5ms and 30.77 frames per second on NVIDIA Jetson TX2 respectively. On a GeForce RTX 2080 Ti, YOLOv4 with 256 inputs results in processing times of 9 ms and 111 frames per second, respectively, and 105 ms and 9.5 frames per second on an NVIDIA Jetson TX2. While YOLOv4 with input size of 608 takes longer to process, it does it in 470ms with 2.1 FPS on NVIDIA Jetson TX2 and 25ms with 40 FPS on GeForce RTX 2080 Ti.

Comparison Graph:

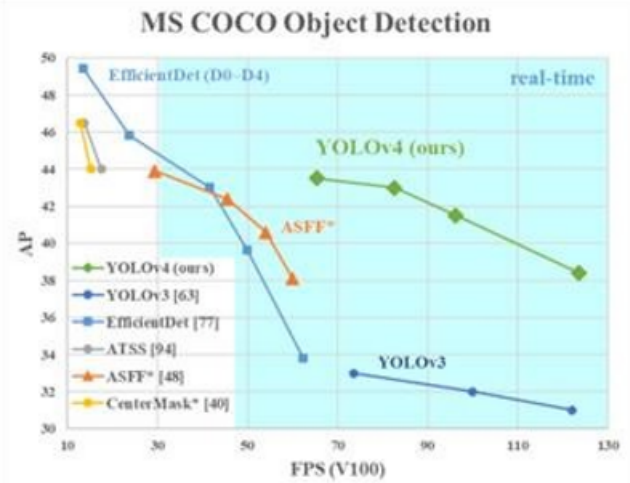

FIGURE 9. AP50 Vs FPS Graph Representing MS COCO Object Detection .

YOLOv4 increases the AP and FPS of YOLOv3 by 10% and 12%, respectively. While they operate at a pace of about 30 frames per second on a V100 GPU, the EfficientDet D4-D3 models obtain greater AP than YOLO v4 models, as can be shown. While maintaining very high accuracy, YOLO can operate at a significantly greater pace (> 60 FPS).

Output:

FIGURE 10. Example using a regular text license plate.

FIGURE 11. Example using license plate with similarly shaped characters (5 and S).

FIGURE 12. Example using a license plate in foggy conditions with 50% visibility.

8. Conclusion

In the Image extraction step of ALPR (Automatic License Plate Recognition), License plates are recovered from background photos or filmed videos of the cars. Additional phases include the segmentation phase and the recognition of segmented characters phase.

In ANPR, the license plate extraction phase is the most important phase, as it determines the accuracy and processing speed of the entire system, as the character segmentation and character recognition phases rely on the license plate extraction phase to provide the output as a number. In our suggested work, it works well with blurred and noisy photos, as well as dark and low-contrast images, thanks to the Yolo V4 approach, which creates high precision for our system and improves the quality of the license plate image so that it can produce effective results.

Further work on the Automatic License Plate Recognition (ALPR) system can concentrate on multi-style license plate identification, high-definition license plate recognition, and recognition rates for ambiguous and broken characters.

9. Acknowledgement

We would like to express our sincere gratitude to all those who have contributed to the success of this research project. Firstly, we extend our heartfelt thanks to our supervisor Prof Manjula R, whose guidance and support throughout the research process has been invaluable. We are grateful for the opportunity to have worked under her mentorship. Our sincere appreciation also goes to Vellore Institute of Technology (Vellore) management who have provided us with the necessary resources and support to carry out this research. Finally, we would like to express our thanks to the participants of this study whose willingness to share their time and information with us made this research possible.

We would also like to state that all information and ideas from external sources used in this research have been appropriately acknowledged through proper citations and references.

We have taken great care to ensure that this work is free from any form of plagiarism, and we hope that it will be of value to the academic community and beyond.

Authors' Note

There is no conflict of interest regarding the publication of this article. This paper is free of plagiarism.

References

Ap, N Palanivel, et al. "Automatic Number Plate Detection in Vehicles using Faster R-CNN". *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)* (2020): 16–16.

Badr, et al. "Automatic number plate recognition system". *Annals of the University of Craiova-Mathematics and Computer Science Series* 38.1 (2011): 62–71.

Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". (2020).

Ganta, L, Praveen Srikanth, and Svsrk. "A novel method forIndianvehicle registration number plate detection and recognition using image processing techniques". *Procedia Computer Science* 167 (2020): 2623–2633.

Gnanaprakash, V, N Kanthimathi, and N Saranya. "Automatic number plate recognition using deep learning". *IOP Conference Series: Materials Science and Engineering* 1084.1 (2021): 012027–012027.

Kanagapushpavalli, D and D Renuka devi. "Automaticlicenseplate recognition". *3rd International Conference onTrendz in Information Sciences & Computing (TISC2011)* (2011): 75–78.

Kashyap, Abhishek, et al. "Automatic Number Plate Recognition". *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*. IEEE, 2018. 838–843.

Kaur and Sarbjit. "An Automatic Number Plate Recognition System under Image Processing". *International Journal of Intelligent Systems and Applications* 8.3 (2016): 14–25.

Khinchi, Monika and Chanchal Agarwal. "A Review on Automatic Number Plate Recognition Technology and Methods". *2019 International Conference on Intelligent Sustainable Systems (ICISS)* (2019): 363–366.

Md, Atikuzzaman, Md Asaduzzaman, and Md Zahidul Islam. "Vehicle number plate detection and categorization using cnns". *In2019 International Conference on SustainableTechnologies for Industry 4.0 (STI)* (2019): 1–5.

Patel, Chirag, Dipti Shah, and Atul Patel. "Automatic Number Plate Recognition System (ANPR): A Survey". *International Journal of Computer Applications* 69.9 (2013): 21–33.

Prabhakar, Priyanka, P Anupama, and S R Resmi. "Automatic vehicle number plate detection and recognition". *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)* (2014): 185–190.

Shariff, A S, et al. "Vehicle Number Plate Detection Using Python and Open CV". *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. 2021. 525–529.

Sharma and Gajendra. "Performance Analysis of Vehicle Number Plate Recognition System Using Template Matching Techniques". *Journal of Information Technology & Software Engineering* 08.02 (2018): 1–9.

Valdeos, Miluska, et al. "Methodology for an automatic license plate recognition system using Convolutional Neural Networks for a Peruvian case study". *IEEE Latin America Transactions* 20.6 (2022): 1032–1039.

Embargo period: The article has no embargo period.

To cite this Article: Mohta, Amogh, Anand Swaroop , Katkuri Fhanindra Reddy , and Manjula R . "**Automatic License Plate Recognition System Using YOLOv4**." International Research Journal on Advanced Science Hub 05.05S May (2023): 280–286. http://dx.doi.org/10.47392/irjash.2023.S038