Applied Computer Science, vol. 16, no. 1, pp. 73–84
doi:10.23743/acs-2020-06

Submitted: 2020-03-05
Revised: 2020-03-07
Accepted: 2020-03-13

recognition software, car number plate, design

Mohanad ABDULHAMID[*], *Njagi KINYUA*[**]

SOFTWARE FOR RECOGNITION OF CAR NUMBER PLATE

Abstract

The purpose of this paper is to design and implement an automatic number plate recognition system. The system has still images as the input, and extracts a string corresponding to the plate number, which is used to obtain the output user data from a suitable database. The system extracts data from a license plate and automatically reads it with no prior assumption of background made. License plate extraction is based on plate features, such as texture, and all characters segmented from the plate are passed individually to a character recognition stage for reading. The string output is then used to query a relational database to obtain the desired user data. This particular paper utilizes the intersection of a hat filtered image and a texture mask as the means of locating the number plate within the image. The accuracy of location of the number plate with an image set of 100 images is 68%.

1. INTRODUCTION

Automatic Number Plate Recognition (ANPR) is a mass surveillance method that uses optical character recognition on images to read the license plate on vehicles using existing closed circuit television cameras or road-rule enforcement cameras, or ones specifically designed for the task-some systems commonly use infrared lighting to allow the camera to take the picture at any time of day. They are used for various tasks, including electronic toll collection on pay per use roads, restricted car identification access control schemes such as for pay parking-lots or for secured office compounds, monitoring traffic activity such as red light adherence in an intersection and for direct marketing. ANPR technology tends to be region specific, owing to plate variation from place to place. The first ANPR was invented in 1976 at the Police Scientific Development Branch in the UK.

[*] Al-Hikma University, Karada Kharidge, Baghdad, Iraq, moh1hamid@yahoo.com
[**] University of Nairobi, P.O. Box 30197-00100, Nairobi, Kenya, researcher12018@yahoo.com

Prototype systems were working by 1979 and contracts were let to produce industrial systems at the Computer Recognition Systems (CRS) in Wokingham, UK. Early trial systems were deployed on the A1 road and at the Dartford Tunnel (Badr & Abdelwahab, 2011; Sharma, 2018).

License Plate Recognition(LPR) has a wide range of applications, which use the extracted plate number and optional images to create automated solutions for various problems (Lin, Lin & Liu, 2018; Xie & Zhang, 2018), these include:

1. Parking – the plate number is used to automatically enter pre-paid members and calculate parking fee for non-members by comparing the exit and entry times.
2. Access control – a gate automatically opens for authorized members in a secured area, thus replacing or assisting the security guard. The events are logged on a database and could be used to search the history of events
3. Law enforcement – the plate number is used to produce a violation fine on speed or red-light systems. The manual process of preparing a violation fine is replaced by an automated process which reduces the overhead and turnaround time. Using a list of stolen cars or unpaid fines, an Automatic License Plate Recognition(ALPR) system deployed on the roadside, may perform a real-time match between the passing cars and those on the list, hence raise alerts for passing cars on the list.
4. Marketing tool – the car plates may be used to compile a list of frequent visitors for marketing purposes, or to build traffic profiles, such as the frequency of entry verses the hour or day.
5. Tolling – the car number is used to calculate the travel fee in a toll-road, or used to double-check the ticket.

Such routine tasks, possibly handling large volumes of traffic, maybe more efficiently done if automated than if done by humans. This would reduce associated costs of operation, increase processing speed and minimize errors that may result due to fatigue or monotony if a human operator were involved (Silva & Jung, 2018).

The automatic plate recognition system may be decomposed into three blocks: an image processing stage, an optical character recognition stage and the database.

2. PROBLEM SOLUTION AND DESIGN

The automatic license plate recognition application is developed using the license plate extraction, character segmentation and optical character recognition modules. The automatic number license plate presented in this work is developed in MATLAB 7.6. Implementation is grouped into image processing, optical character recognition and database blocks.

2.1. Image processing block

This block receives input images of the vehicles and produces cropped images which are passed to the optical character recognition block that succeeds it. The image processing block is based on image pre-processing and segmentation steps. First, the input color (RGB) images are converted into grayscale images. An RGB image, sometimes referred to as a *true-color* image, is stored as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location. A grayscale image is a data matrix whose values represent intensities within some range.

This conversion allows ease of thresholding, which creates a binary image using a certain threshold. Adaptive thresholding using the Otsu thresholding scheme; which assigns pixels with grey-level above a threshold (chosen so as to minimize the intra-class variance of the black and white pixels) in the grayscale scale image are assigned binary value "1" in the binary image, while those below are assigned binary value "0" are implemented using the inbuilt *graythresh* function.

Next, morphological operations opening and closing are used. The morphological closing operation is carried out using a rectangular structuring element of a size much greater than the inter-character spacing of the license plate characters, resulting in blurring of the plate characters in the license plate. Subtracting the resultant image from the original thresholded image, which constitutes top-hat filtering, removes large parts of constant intensity background, leaving plate characters and other fine details in the image intact. A morphological opening operation with a structuring element, of width less than the inter- character spacing, is then used to remove the unimportant fine details without affecting the characters themselves.

An image mask which isolates the regions with the largest difference in pixel intensities in a given neighborhood is then developed using a range filter. The function *rangefilt* is used to obtain an output array of same size as the input image, where each output pixel contains the range value, as the difference between maximum and minimum pixel intensity values in the 3-by-3 neighborhood around the corresponding pixel of the input image. The range filtered image is then binarized using an experimentally determined threshold of 0.4, which preserved as much license plate detail as possible, without including too many noise objects. A morphological closing operation is then done on the binary image with a 9-pixel, square structuring element to remove thin objects in image due to local transitions at the edges of the input image. Larger structuring elements merge objects, yielding large area masks that left many noise objects when overlaid with the top-hat filtered image, whereas smaller structuring elements yield fragmented characters. An area opening is then used to remove from the binary image all connected objects that had fewer than 300 pixels, the figure being determined by trial and error, to produce the binary masking image.

The image created by over-laying the top-hat filtered image with the mask by carrying out the logical AND of the two images is then labeled using the *bwlabel* function. The *bwlabel* function is used to search for connected components and labels them with unique numbers; it takes a binary input image and a value specifying the connectivity of objects. The function returns a matrix *L* of the same size as the input image, containing labels for the connected objects in that input image, and the number of connected objects found in the input image. The elements of *L* are integer values greater than or equal to 0.

The *regionprops* function is then used to measure object or region properties in an image and returns them in a structure array. When applied to an image with labeled components, it creates one structure element for each component; with the structure array having *area, centroid and bounding box* fields. The *area* field is scalar representing the actual number of pixels in the region. The *bounding box* field is a vector representing the smallest rectangle containing the region defined by co-ordinates of the upper left corner of the bounding box and the width of the bounding box along each dimension, in length and height. The *centroid* field is a vector that specifies the center of mass of the region; the first element of centroid is the horizontal coordinate (or x-coordinate) of the center of mass, and the second element is the vertical coordinate (or y-coordinate).

The *centroid* field of the *regionprops* function is used to determine the vertical coordinate of all labeled objects obtained from the logical AND of the top-hat filtered image and the texture map. Since few objects are members of either set, resultant intersection of both sets is composed largely of license plate characters. The y-coordinate values corresponding to the objects are observed to be randomly distributed over the image height, but objects corresponding to license plate characters had approximately the same centroid value. This variation in the centroid value is exhibited in the ones digit of the value; hence a rounding-off of the centroid values to the nearest ten-arrived at by trial and error eliminates this variation. The mode of the rounded vertical coordinates of objects in this intersection is thus the average height of the license plate characters on the image. Selecting all objects intersecting with a horizontal line, running across the image at this modal height yields the license plate characters. In very few cases, noise objects are included. The *area* field of the *regionprops* function is used to remove all noise objects with an area of less than 100 pixels, determined as the minimum area of license plate character-corresponding to the numeral 1.

Further filtering of noise objects requires the orientation of the major axis of the ellipse that has the same variance as the region of each object lie between 45 to 90 degrees from the horizontal axis. This allows removal of horizontally aligned noise objects resulting from dilation of fragmented license plate edges due to the closing operation. The Euler number of each object is also used to remove noise objects. The Euler number is a measure of the topology of an image. It is defined as the total number of objects in the image minus the number of holes in those objects. Character (B) has Euler number of -1, indicating that the number of holes

is greater than the number of objects; characters such as (A, O, P, D, Q) have an Euler number of zero, whereas all other simple characters have an Euler number of 1. Due to nails in the license plate, noise is introduced to character objects due to the connectivity used in labeling. An object having an Euler number of 1 in isolation would show an Euler number of two if such a nail object is introduced in its bounding box due to connectivity. As a result, all objects with an Euler number greater than 2 or less than -1 are considered to be noise objects and are removed. The remaining objects are subsequently considered to be characters and are passed to the optical character recognition block.

The *bounding box* field of the *regionprops* function is then used to obtain the smallest rectangles containing each object in the resultant image. These rectangles are then passed to the *imcrop* function which crops the required labeled objects corresponding to the license plate characters in the image, based on the rectangles' top-left coordinates, their widths, and heights respectively. It is these objects that are passed to the optical character recognition block. The quality of segmentation is strongly related to the choice of the structuring element's size on the plate enhancement phase, choice of the threshold used for image binarization, the relative angle between camera and plate and the quality of the image.

2.2. Optical character recognition block

2.2.1. Neural network classifier approach

A back-propagation neural network classifier is first adopted and is trained on thirty four vectors; corresponding to all possible characters; each having 150 elements; corresponding to image objects of resolution 15×10. Each target vector is a 34-element vector with a 1 in the position of the letter it represents, and 0's everywhere else. For example, the letter A is to be represented by a 1 in the first element (as A is the first letter of the alphabet), and 0's in elements two through thirty four. The network receives the 150 Boolean values as a 150-element input vector. It is then required to identify the letter by responding with a 34-element output vector. The 34 elements of the output vector each represent a letter. To operate correctly, the network would respond with a 1 in the position of the letter being presented to the network. All other values in the output vector would be 0. The neural network thus has 150 inputs and 34 neurons in its output layer to identify the letters. The network is a two-layer log-sigmoid network, with 10 neurons in the hidden layer. Training is done for 5000 iterations, for gradient descent with momentum. Gradient descent with momentum, implemented by the *traingdm* function, allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Momentum allows the network to ignore small features in the error surface, hence slide through shallow local minima. Without momentum a network can get stuck in such a minimum.

While the neural network classifier is usually preferred owing to its higher cognitive rate and its ability to give reasonable results when presented with new object which it has not been trained on, several shortcomings result in adoption of the template matching approach. With character recognition for license plate applications, the neural network classifier requires frequent retraining, preferably for each car image due to the large input vector size, having 150 binary elements for each character.

Accuracy improves as the number of neurons in the hidden layer and as size of input training vector increases. Increasing number of neurons in the hidden layer results in a speed penalty while increasing size of the input training vector is limited by memory constraints. Training requires either 5000 iterations or a steady-state error of less than 0.1, which make it slower than the template matching approach. In this case, the neural network classifier has a poor cognitive rate and misclassified well segmented and obvious characters.

2.2.2. The template matching approach

The template matching approach is then implemented. The templates used have a resolution of 42×24, hence rescaling the license plate objects prior to template matching is necessary. Character recognition is based on calculating the correlation metric, implemented using the *corr2* function, which computes the correlation coefficient between two matrices of the same size. All images from extracted objects and the template set are thus represented as 42×24 intensity matrices. The template images corresponding to the 34 possible characters A to Z and 0 to 9 are saved, and template matching is implemented by using the correlation between each extracted object from the image against all the images in the template. This removes the presumption that all license plates begin with letters (which would affect recognition of diplomatic license plates) or that license plate begins with letter K (which would affect recognition of foreign license plates). The correlation coefficients for each extracted object with the template set is ordered into a 34 element array, and the index of the element having the highest correlation coefficient used to identify the corresponding similarly indexed character. The characters corresponding to each extracted object are then concatenated to form a string, which is the detected vehicle registration number.

2.3. Database implementation

For a car identification access system, a database is needed. Since the basic segmentation and character recognition modules have been implemented, the output string from these modules is to be used to query a test database and extract hypothetical user data. While the original idea is to use a MYSQL database and use a PHP script to query the database, this is found to be unnecessary. MATLAB supports database queries from most databases since it implements Java Database

Connector (JDBC), and supports JDBC to Object Database Connector(ODBC) inter-conversion. This database support is implemented by the MATLAB database toolbox which requires that a data source should be set up first. The data source consists of data that the toolbox accesses and information required to find the data, such as driver, directory, server, or network names. Data sources interact with ODBC drivers or JDBC drivers. An ODBC driver is a standard windows interface that enables communication between database management systems and SQL-based applications. A JDBC driver is a standard interface that enables communication between applications based on Java and database management systems. The database toolbox software is based on Java. It uses a JDBC/ODBC bridge to connect to the ODBC driver of a database, which is automatically installed as part of the MATLAB Java Virtual Machine (JVM).

Thus a Microsoft access data source is set up, and a test database is created, having three fields; the car registration number, the car's color and its making or model; with the registration number field as the primary key. A MATLAB script is then used to connect to this database using the JDBC/ODBC driver bridge to obtain a connection object which is then used to pass SQL queries to the test database. In order to access the database, first a connection to the database is created using the database function. This function takes data source, username and password as arguments. The data source is the name of the data source in the ODBC for Windows. This has to be configured before running MATLAB. The username of the user has on the database to be accessed. The password is given with the defined username to access the database. These arguments are passed to the database function as strings.

Once the connection is established, queries to the database are performed using the fetch function. This function takes a connection pointer and an SQL query string as arguments. The connection pointer is the connection created with the database function. The SQL query string contains the desired SQL query. The queries used are based on the license number string, passed from the character recognition block. User data corresponding to the predefined fields is thus retrieved.

The database queries however fails if fewer than 4 characters have been misi-dentified due to deletion of the characters in the image processing block, or due to merging of characters. When used for giving specific vehicles access to a barrier area the decision to have an error rate of two characters is in the author's opinion, viewed as acceptable. This is because the likelihood of an unauthorized car having such a similar license plate with all detected characters in their right order is seen as quite small.

3. RESULTS

A set of 108 images, with a resolution of 640×480 pixels is obtained with the camera position set at a distance of between 0.5 and 2 meters from the vehicle. Of the 108 images, 100 images are used to test the developed software, and 68 cases are satisfactorily recognized.

The successful extraction of license plate characters is limited if image has large local variances in pixel ranges in regions other than the license plate, vehicle has bent or warped license plates, characters on the license plate are faded, or had very poor contrast relative to bright regions in the image, or if the vehicle in the image is too close to the camera. Faded plates and texture conflicts present the most difficult test cases. Of the 108 images, 7 images have faded plates, 4 images are obtained with vehicle too close to the camera, 3 images had significantly warped plates. Errors in recognition are largely caused by misclassification of the characters by the template matching algorithm. Efforts to improve its cognitive rate, by use of character standard deviations or Euler numbers to verify the recognized characters, are hindered by an intolerable increase in processing time. The recognition performances for simple visible plates is higher, at about 78%, *i.e.* 62 correct identifications in a sample of 80 images. Contention caused by misclassified digits is resolved by listing all probable license plates from the database.

Fig. 1 shows input image (left) and the resulting extracted license plate characters. The horizontal noise objects violate orientation requirements, and are thus not considered.

Fig. 1. Input image and the resulting extracted plate characters

Fig. 2 shows input image having large texture regions in areas other than the license plate, due to shadows on car windshield. In this case, extraction of license plate characters is satisfactory.

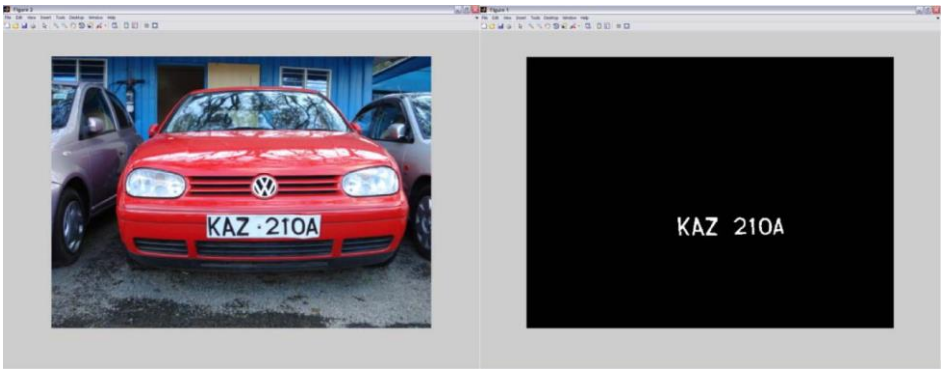

Fig. 2. Input image having large texture regions

Fig. 3 represents the limiting case, showing car on coarse stone background. In this case the texture mask is overwhelmed by the numerous and large changes in local pixel intensities.

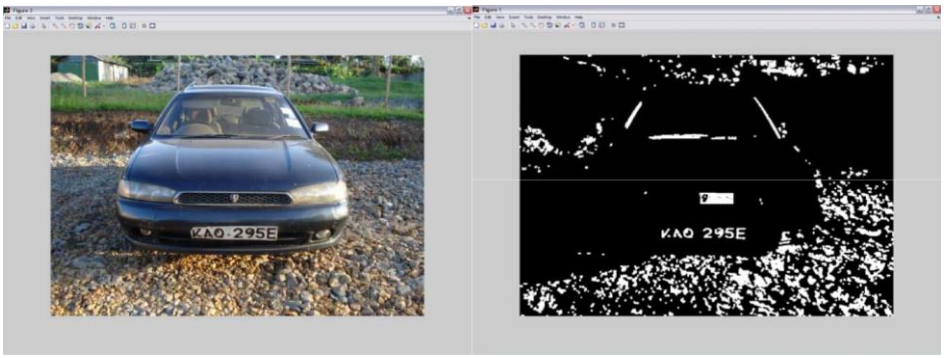

Fig. 3. Car on coarse stone background

Fig. 4 shows images for same vehicle in subsequent frames. While, noise objects imped proper extraction of license plate characters from the first frame, successful extraction from the subsequent frame is possible as shown in Fig. 5.

Fig. 6 shows image with warped plates occluding license plate characters. In this case, an insufficient number of characters (A and 1) are extracted to uniquely identify the vehicle.

Fig. 4. Images show same vehicle in subsequent frames

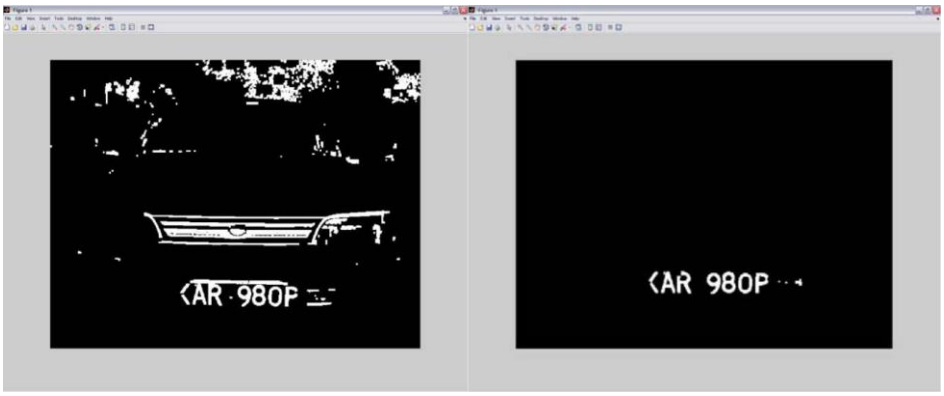

Fig. 5. Extraction of plate characters from subsequent frame

Fig. 6. Image with warped plates occluding plate characters

Fig. 7 shows images of vehicles with faded plates. The extracted license plate characters are too fragmented to be useful as shown in Fig. 8.

Fig. 7. Images of vehicles with faded plates

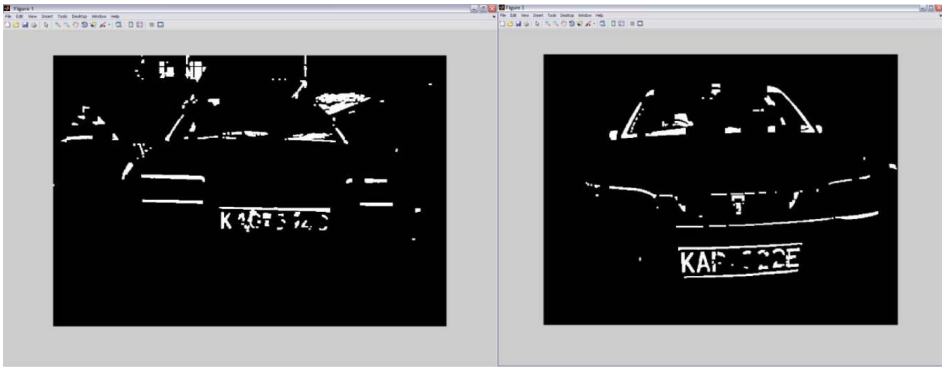

Fig. 8. Extracted license plate characters are too fragmented

Fig. 9 shows input image (left) is too close to the camera. The fixed size of the structuring element used in the top hat filter results in fragmented characters.

Fig. 9. Input image is too close to the camera

4. CONCLUSION

The car number plate recognition software, comprising of the license plate extraction, character segmentation and optical character recognition modules was designed and implemented. A suitable database with hypothetical user data was also incorporated to complement the system. The ANPR achieved an overall success rate of 68% when tested on 100 of the 108 images, with recognition performances for simple visible plates close to 80%. Results may be improved by refining the recognition stage and testing other classifiers. Different character templates could be used for such refinement of the recognition stage. Future work is intended to be done in improving and testing the system on a larger number of images.

REFERENCES

Badr, A., & Abdelwahab, M. (2011). Automatic number plate recognition system. *Annals of the University of Craiova, Mathematics and Computer Science Series*, 38(1), 62–71.

Lin, Ch.-H., Lin, Y.-S., & Liu, W.-Ch. (2018). An efficient license plate recognition system using convolution neural networks. In *2018 IEEE International Conference on Applied System Invention* (pp. 224–227). Japan: IEEE.

Sharma, G. (2018). Performance Analysis of Vehicle Number Plate Recognition System Using Template Matching Techniques. *Journal of Information Technology & Software Engineering*, 8(2), 1–9.

Silva, S. & Jung, C. (2018). License plate detection and recognition in unconstrained scenarios. In V. Ferrari, M. Hebert, C. Sminchisescu & Y. Weiss (Eds.), *European Conference on Computer Vision* (pp. 593–609). Springer, Cham.

Xie, F., & Zhang, M. (2018). A robust license plate detection and character recognition algorithm based on a combined feature extraction model and BPNN. *Journal of Advanced Transportation*, 2018, 1–14.