DEVELOPING NEW TECHNIQUES TO IMPROVE LICENCE PLATE DETECTION SYSTEMS FOR COMPLICATED AND LOW QUALITY VEHICLE IMAGES

A thesis submitted by

Meeras Salman Juwad Al-Shemarry

For the award of

Doctor of Philosophy

2020

Abstract

Intelligent transportation systems (ITSs) play a very important role in people's lives in many respects. One of the most important ITS applications is for automatic number plate recognition systems. Over the years, many algorithms have been developed for detecting licence plates (LPs) from vehicle images or from a sequence of images in a video. Many existing ITSs work only under good conditions or normal environments.

It is still challenging to find effective techniques to identify LPs under difficult conditions, such as low/high contrast, bad illumination, foggy, dusty, or distorted by high speed or bad weather. New techniques are needed to improve the performance of existing detection systems.

In this thesis, novel methods are developed for licence plate detection (LPD) systems to extract key features, and classify the LP region from complicated vehicle images based on preprocessing methods and machine learning algorithms with several types of texture descriptors.

In order to identify LPs from complicated vehicles images, four LPD methods were developed in this research. The first, is a three-level local binary pattern operator based on an ensemble of Adaboost cascades classifiers (3L-LBP_Adaboost) detection method. The second method, introduces a new texture descriptor based on a multi-level preprocessing stage with extended local binary pattern descriptor using an extreme learning machine classifier (MLELBP_ELM). The third, develops learning-based preprocessing methods using a local binary pattern and a median filter histogram of the oriented gradient with support vector machine classifier (LBP_MHOG_SVM) for detecting complicated LPs. Finally, for identifying distorted LPs using hybrid features, median robust extended local binary pattern and speeded-up robust with an extreme learning machine classifier (MRELBP_SURF_ELM). The experimental results show that both of the LBP_MHOG_SVM and MRELBP_SURF_ELM algorithms perform very well in LP detection accuracy rate compared with 3L-LBP_Adaboost and MLELBP_ELM algorithms. Also, the false positive rate (FPR) for both methods is better than those algorithms. The MLELBP_ELM and MRELBP_SURF_ELM methods carry out significant classification of different types of LP key features. The 3L-LBP_Adaboost approach takes much less execution time and produces high FPR compared to the three other methods. But it was a good technique for selecting suitable preprocessing and extraction methods, for detecting LPs from low quality vehicle images.

The experimental results proved the efficiency of the proposed approaches for detecting difficult regions of the LP inside a vehicle image with a high accuracy rate and low detection time. Whereas the overall performance evaluation for the 3L-LBP_Adaboost method in terms of detection, precision, and F-measure rates is 98.56%, 95.9%, and 97.19%, respectively, with an FPR of 5.6%. The average detection time per vehicle image was 2.001miliseconds.

In the MLELBP_ELM method, detection accuracy and FPR were improved by 0.54% and 0.56%, respectively, compared with the 3L-LBP_Adaboost approach. The classification and detection rates are 99.78% and 99.10%, respectively, with an FPR of 5%. The average execution time per vehicle image was 2.4530miliseconds.

The LBP_MHOG_SVM method yielded an excellent improvement compared with existing proposed methods, a 4% improvement for the FPR, and 1.50% for detection accuracy. The detection rate is 99.62%, with an FPR of 1.675%. The average of the processing time per vehicle image was 2.2187miliseconds.

Finally, the accuracy and detection rates are 97.92% and 99.71, respectively, with the FPR of 2.24% for the MRELBP_SURF_ELM method. The average of the execution time for the whole detection system per vehicle image was 2.108 milliseconds. This method was superior in the performance and execution time over the existing proposed methods in this research.

The findings suggest that the outcomes of this study can improve the performances of existing LPD systems. They can assist in law enforcement with an ITS system. Also, it can be effectively used to detect LPs in real-time applications under difficult conditions.

Certification of Thesis

This thesis is the work of Meeras Salman Juwad Al-Shemarry except where otherwise acknowledged, with the majority of the authorship of the papers presented as a Thesis by Publication undertaken by the student. The work is original and has not previously been submitted for any other award, except where acknowledged.

Student: Meeras Salman Juwad Al-Shemarry

Principal Supervisor: Prof. Yan Li

Associate Supervisor: Dr. Shahab Abdulla

Student and supervisors signatures of endorsement are held at the University.

Statement of Contribution

This section presents details of contributions by the various authors for each of the paper presented in this thesis by publication. The following detail is the agreed share of contributions for the candidate and co-authors in the published articles and the ones to be published.

Chapter 3, Al-Shemarry et al., (2018)

Al-Shemarry, M. S., Li, Y. & Abdulla, S. 2018. Ensemble of Adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images. *Expert Systems with Applications*, 92, 216–235. (Q1)

Authors	Percent contribution	Tasks Performed
Al-Shemarry, M. S.	70%	Designed the method, simulation, analysis, interpretation, wrote entire draft of the paper.
Li, Y. and Abdulla, *S.*	30%	Significantly improved the manuscript, interpretation, and analysis.

Chapter 4, Al-Shemarry et al., (2019)

Al-Shemarry, M. S., Li, Y. & Abdulla, S. 2019. 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, Feb. 2020. (Q1)

Authors	Percent contribution	Tasks Performed
Al-Shemarry, M. S.	65%	Designed the method, simulation, analysis, interpretation, wrote entire draft of paper.
Li, Y.	25%	Help with methodology design, significantly improved the writing of the manuscript, interpretation.
Abdulla, S.	10%	Suggested edits the manuscript, interpretation.

Chapter 5, Al-Shemarry and Li, (2020)

Al-Shemarry, M. S. & Li, Y. 2020. Developing Learning-Based Preprocessing Methods for Detecting Vehicle Complicated Licence Plates. *IEEE Access*, doi:10.1109/ACCESS.2020.3024625. (Q1).

Authors	Percent contribution	Tasks Performed
Al-Shemarry, M. S.	75%	Designed the method, simulation, analysis, interpretation, wrote entire draft of paper.
Li, Y.	25%	Significantly improved the manuscript, interpretation.

Chapter 6, Al-Shemarry and Li, (2020)

Al-Shemarry, M. S. & Li, Y. 2020. Distorted vehicle licence plates detection using hybrid feature descriptors and extreme learning machine classifier. *European Journal of Operational Research*, Q1, (Under review).

Authors	Percent contribution	Tasks Performed
Al-Shemarry, M. S.	70%	Designed the method, simulation, analysis, interpretation, wrote entire draft of paper.
Li, Y.	30%	Significantly improved the manuscript, interpretation.

List of Publications

1. Al-Shemarry, MS, Li, Y & Abdulla, S 2018, 'Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images', *Expert Systems With Applications*, vol. 92, pp. 216 - 235.

2. Al-Shemarry, MS, Li, Y & Abdulla, S 2019, '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, Feb. 2020.

3. Al-Shemarry, M. S. & Li, Y. 2020, 'Developing Learning-Based Preprocessing Methods for Detecting Complicated Vehicle Licence Plates', *IEEE Acess*, doi:10.1109/ACCESS.2020.3024625.

4. Al-Shemarry, M. S. & Li, Y. 2020, 'Distorted vehicle licence plates detection using hybrid feature descriptors and extreme learning machine classifier', *Submitted to the Journal.* (submitted).

Acknowledgement

I thank all who contributed to complete this thesis in one way or another.

First, I give deep thanks to Allah for protection and the ability to do this work.

Undertaking this PhD would not have been possible to do without the support and guidance that I received from my supervisors. So I would like to express my sincere gratitude to Prof. Yan, my principal supervisor. It would have been very difficult to complete my goal without her support, guidance, patience, and encouragement. I also want to present my deep thanks to associate supervisor Dr. Shahab for his expert advice and the valuable input to my project. It is my great honor to be their student.

I would like to show my deep personal gratitude to Tim Passmore for proofreading my thesis.

I would like to dedicate this work to my parents. My father who raised me with a love of the science and supported me in all my pursuits. My mother always in my dreams and heart, without her prayers, I could not continue to complete this work. I am greatly indebted to them for their unconditional support, love, and encouragement.

My respect goes to my husband and love of my life, Dr. Dawood, for keep going things even though there are many difficulties which faced us. He always shows how proud he is of me. Without his encouragement and support, I could not be able to finish my study. I am always proud of him.

The last words go to my wonderful daughters Kawther and Ghadeer and my lovely son Ali. You have given me extra strength and motivation to light my life and get things done. Thanks for your endless love, tolerance, and patience. Big thanks go to my sisters, brothers for their prayers and encouragement.

Also, I would like to dedicate this thesis to my beautiful little girl aged 9th months and three weeks, who passed away during the birth. At this time I faced very difficult an financial problems because of the unfair procedures of the ministry of higher education and scientific research in Iraq which is effected on my health.

I wish to introduce my appreciation to the University of Southern Queensland (USQ) and special thanks to the School of Sciences, Faculty of Health, Engineering, and Sciences for their

understanding of my difficult family situation. They have provided great support which is helped me to reduce the challenges during my academic study.

I want to express my sincerest gratitude to the Iraq Government, my sponsors for giving me the opportunity to come here and complete my PhD study. Also, my thanks extend to the Iraqi Cultural Attache in Canberra for facilitating procedures, overcoming obstacles, and providing the funding during the period of this study.

May the Almighty Allah richly bless all of you.

Table of Contents

List of Figures

(Excluding publications included in Chapters 3-6)

List of Tables

(Excluding publication included in Chapters 3-6)

Abbreviations

3L-LBP	Three-Level Local Binary Pattern
ANPR	Automatic Number Plate Recognition System
CCA	Connected Component Analysis
CLAHE	Contrast- Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
CPR	Car Plate Recognition
CRS	Computer Recognition System
DWT	Discrete Wavelet Transform
ECHE	Enhancement Cumulative Histogram Equalization
ECLACHE	Enhancement Contrast-Limited Adaptive-Cumulative Histogram Equalization
ELBP	Extended Local Binary Pattern
ELM	Extreme Learning Machine
FPR	False Positive Rate
GA	Genetic Algorithm
GST	Generalized Symmetry Transform
HL	Hue and Lightness
HLS	Hue, Lightness, and Saturation
HOG	Histogram of Oriented gradient
HSV	Hue, Saturation, and Value
ICA	Independent Component Analysis
KNN	K-Nearest-Neighbor
LDA	Linear Discriminant Analysis
LP	Licence Plate
LPD	Licence Plate Detection
LPR	Licence Plate Reader
MHOG	Median filter histograms of oriented gradients
MLELBP	Multi-Level Extended Local Binary Patterns
MLPR	Mobile Licence Plate Reader
MRELBP	Median Robust Extended Local Binary Pattern
NN	Neural Network
OCR	Optical Character Recognition
PCA	Principal Component Analysis

RGB	Red, Green, Blue
ROC	Receiver Operating Characteristic
ROI	Regions of Interest
SURF	Speeded Up Robust Feature
SVM	Support Vector Machine
VRI	Vehicle Recognition Identification

1

CHAPTER 1

INTRODUCTION

These days, intelligent transportation systems (ITSs) play an important role in different aspects of our daily life. Normally, these systems include two parts: the intelligent infrastructure system and the automatic number plate recognition system (ANPR) (Anagnostopoulos et al. 2006; Anagnostopoulos 2014). Using surveillance applications is necessary to observe and examine road traffic for law enforcement activities (Castello et al. 1999; Duan et al. 2004b; Sarfraz et al. 2013). In 1978, the first successful ANPR system was used in the UK to detect stolen cars. ANPRs have several names reflecting different purposes, like licence plate recognition (LPR), car plate recognition (CPR), licence plate reader (LPR), mobile licence plate reader (MLPR), and vehicle recognition identification (VRI). Some applications are: security systems (Sheldon 2013), highway road tolling systems (Song & Sarker 2014; Panahi & Gholampour 2017), parking management systems, enforcing move over laws for emergency vehicles (Roberts & Casanova 2012), traffic control (Dehghan et al. 2017), traffic management systems (He et al. 2017), and so on. Nowadays, the key requirements for a good ANPR system are high accuracy and better execution speed for any application (Angelova et al. 2015). All the ARPR applications still have difficulties for detecting licence plates (LPs) from vehicle images, or a sequence of images in a video, under difficult conditions. These difficulties impact on the main system stages, such as, extraction and detection. Therefore, the ARPR software should be able to deal with the following:

- **Poor resolution/ low-quality camera**, sometimes this problem appears due to either short or long distance between the vehicle and the camera, which results in having low/bad image quality. The system accuracy often depends on camera qualities, like type, resolution, shutter speed, light, and the installation method.
- **Blurry vehicle images** caused by the high-speed vehicle movement.

- **The location and tilt** the LP may be found in the different places of an image with different angles and conditions.
- **Different sizes, languages, and fonts** some countries do not allow to the use of different types of fonts, this could help to reduce some difficulties.
- **Occlusion problems** due to dirt during an image capture or inaccurate distance between the vehicle and the camera.
- **Color problems** due to different color between the vehicle body and the LP area.
- **Distortion problems** some characters in the LP, and the plate itself have screws and frames with a dirty background.
- **Environmental problems** poor illumination or lighting conditions because of weather, such as raining, snowing, and so on.
- **Low/high contrast problem** due to the vehicle headlights, and the different lighting resources during an image capture.

However, some of these problems could be corrected through the software. A robust licence plate detection (LPD) system is required to effectively work under all difficult conditions. Hence, developing new LP detection methods to solve those problems is a very important topic to improve the existing LPD systems. This thesis focus on the detection of LPs from vehicle images under complicated conditions, such as low/high contrast and poor lighting condition, dusk, fog, dirt, and distortion problems. This study proposes four algorithms for identification LPs from complicated and low quality vehicles images. These methods can identify and detect the LP and provide reliable and accurate detection results that will be useful to improve the performance of existing LPD systems in terms of accuracy rates and execution time. Finally, the outcomes of this study will help to enhance the quality of the life through using the ITSs with high quality and security.

1.1 Overview and Motivation of the Study

An LPD system has become a very important tool for many surveillance applications over the past decades (Kamat & Ganesan 1995; Kasaei et al. 2010; Angelova et al. 2015; Azam & Gavrilova 2017; Arafat et al. 2019). For detecting complicated LP problems, the preprocessing and feature extraction techniques should be selected and developed carefully. Selecting good software components plays an important role in the quality of the detection system. In particular, evaluation of a system performance is a very popular tool to determine LP problems

and test possible solutions to improve system weaknesses. In this study, an LPD system includes four stages: images acquisition, preprocessing, extraction, and detection. In chapter 2, a detailed description of those stages is provided. The two main key requirements of the LPD system are accuracy and runtime (Angelova et al. 2015). Recently, increased attention has been paid to efficient strategies to improve existing systems as an important topic in the field of machine learning. The main challenge is how to detect LPs from complicated vehicle images as fast, and with as high an accuracy as possible. Therefore, several detection methods were reported which identify different kinds of LP problems (Asif et al. 2016; Azam & Islam 2016; Boonsim & Prakoonwit 2016; Chen et al. 2017; Panahi & Gholampour 2017; Wang et al. 2018; Al-Shemarry et al. 2019). The detection of LPs under complicated conditions is still far from being achieved. A considerable amount of research is still needed. Therefore, in this thesis, the aim is to develop efficient and accurate methods to identify the LP area from distorted vehicle images.

1.2 Research Problems

The existing ANPR systems are far from satisfactory in detecting the LPs under difficult conditions. Therefore, developing new novel methods to detect LPs from complicated vehicle images is the main goal of this thesis. The performance of the developed approaches has been evaluated using several assessment tools. They are mainly object detection methods, such as detection and object localization metrics, and the receiver operating characteristic (ROC) curve. These measurement tools are used to check the detection system's ability and performance at identifying objects (Bashir & Porikli 2006; Kasturi et al. 2009). This study used very challenging and complicated data, compared with the existing databases used by other studies (Azam & Islam 2016; Azam & Gavrilova 2017; Liu et al. 2017; Panahi & Gholampour 2017; Silva & Jung 2018). Using supervised learning algorithms requires more images to produce accurate results. Therefore, the English LP database (EnglishLPDatabase-2001 ; MedialabLPRdatabase-2007) is extended to increase the number of vehicle images by adding many changes using an image photo editor application. Efficient preprocessing methods and powerful descriptors that are suitable to enhance the low-quality images have been used. This study focuses on answering the question :

How to enhance the performance of an LPD system under different conditions by developing advanced classification techniques?

This question leads to the following sub-questions:

a. What are the best preprocessing and extraction methods for the complicated LP detection?
b. How to enhance the performance of an LPD system through the developed methods?

The main objectives are:

1. To develop new methods for the LP detection, that will result in a good detection results (performance, accuracy, and processing speed) under difficult conditions;
2. To improve the working of existing ANPR systems and reduce the efforts required to efficiently detect unacceptable activities under complicated conditions in transport systems.

Based on the experimental results, the developed methods can achieve good system performance through identifying different types of LP problems from low-quality vehicles images, such as low/high contrast, bad illumination, foggy, dusty, and distortion. Also, they can be applied to different types of car LP databases. As there are no constraints in the proposed methods as to object shape, color, edge, and so on, due to the use of supervised learning techniques.

1.3 Contributions of the Thesis

In this thesis, four techniques are developed for detecting LPs from low-quality vehicles images with difficult conditions. Different types of LPs were detected successfully with high system performance. To investigate the performance of those proposed methods, they were compared with recently reported algorithms with different and/or the same databases. The following contributions have been made to answer the research questions and achieve the objectives:

1. Effective methods were developed for detecting LPs under complicated conditions, such as low/high contrast, bad illumination, foggy, dusty, and distorted by high speed and bad weather. They improved the detection system performance with less execution time and low false positive rate.
2. The developed methods were improved by presenting new preprocessing and extraction techniques that can improve the classification accuracy.

3. Investigating which method is better to achieve the main requirements of an LPD system under difficult conditions like distorted vehicle images, low/high contrast, and bad illumination.

The proposed methods have been implemented in Matlab R2018a. The database that is used in this study is the English LP (EnglishLPDatabase-2001 ; MedialabLPRdatabase-2007). It is a very popular database that is used by many researchers for detecting distorted vehicle images. Also, it contains three types of LP resolutions, which made this research very challenge and interesting. Moreover, each algorithm was evaluated using detection and object localization metrics. Those metrics are the false positive (FP), true positive (TP), false negative (FN), true negative (TN), recall rate (RR) or detection rate (DR), accuracy rate (AR), precision rate (PR) or positive prediction rate (PPR), and F-measure rate. The receiver operating characteristic (ROC) curve was used to evaluate the classification accuracy for the proposed algorithms. The ROC curve depends on four parameters which are the true positive rate (TPR) or RR, false positive rate (FPR), positive predictive value (PPV) or PR, and negative predictive value (NPV). A brief discussion about these contributions is provided below.

1.3.1 Identifying low quality licence plates from three-level local binary pattern features based on ensemble of AdaBoost cascade classifiers

This method employed an ensemble of AdaBoost cascade classifiers with a three-level local binary pattern (3L-LBP) operator for detecting LPs from vehicle images having low/high lighting and contrast conditions, fog, dusk and distorted. It includes two phases: testing and training. The same preprocessing and extraction techniques were applied for both phases. The images in the database include different types of noise with the texts referring to the false positive regions, such as dust, surface textures, distortion, and dirt. This noise increases the unwanted feature intensities in an image. For de-noising, a two-dimensional Gaussian filter and a contrast limited adaptive histogram equalization (CLAHE) method were used to filter out the noise and contrast problems. The enhancement steps could help to reduce the change in illumination and feature dimensions. The LBP is an effective operator for the illumination conditions. It can solve the occlusion and rotated LP problems. Therefore, it was used in this study as a powerful operator to extract features from three preprocessing levels. The first level is the extraction of LBP features from the grayscale image. The second one is extraction of LBP features from a Gaussian filtered image. The final level is the extraction of LBP features

from the CLAHE image. After that, the AdaBoost algorithm was used to train and classify the extracted LP features to produce strong cascade classifiers that consists of a large number of the weak classifiers or LBP features. An ensemble of cascade classifiers was used as detectors or trained models to detect LP objects. From the experiments in Chapter 3, it can be observed that the proposed method works very well compared with other existing methods. It produced a good detection accuracy with less execution time and good false positive rate (FPR), 98.56%, 0.780miliseconds, 5.6%, respectively. The preprocessing methods and adding LP images with different illumination conditions to the training dataset could help to reduce the FPR and increase system performance. The content of this chapter was published by *Expert Systems with Applications*, 92, 216–235.

1.3.2 Developing a texture descriptor to detect complicated licence plates based on an extreme learning machine classifier

This method provided many improvements to the method introduced in Section 1.3.1 above. An efficient texture descriptor was developed to make the detection system more robust with less processing time and a good detection rate. A multi-level extended local binary patterns (MLELBP) descriptor with a Gaussian filter and CLAHE method were used to extract different features from complicated LP images. Each LP produced four LP images from four preprocessing levels that were applied on the extended local binary (ELBP) descriptors. Therefore, the number of training LP images was increased. Several relevant features were extracted from those images. The English car LP database was extended by making many changes in the original database to reflect different difficult conditions. It helped to improve the performance of the detection system. The ELM classifier was used to train the extracted LP features and produced a strong features vector as a detector. The proposed method was tested on unseen data (distorted vehicle images). The experimental results were compared with existing LPD algorithms with the same database. The method has outperformed other algorithms in terms of classification, detection accuracies, and good detection time, 99.78%, 99.10%, and 0.735miliseconds, respectively. Moreover, many of the existing algorithms have only the testing phase (preprocessing stage) under some assumptions. This algorithm works without any assumptions, due to using two phases of testing and training. It could be used efficiently with real-time applications. However, it needs some further improvements to reduce the FPR and the time of the extraction stage using good preprocessing methods. The detail of this method is given in Chapter 4. The content of this chapter was published in *IEEE*

Transactions on Intelligent Transportation Systems, Digital Object Identifier 10.1109/TITS.2019.289799, 1524-9050 © 2019 IEEE.

1.3.3 Developing learning-based preprocessing methods for detecting complicated vehicle licence plates

A precise LPD system with a low FPR is very crucial to increase the efficiency and safety of any transportation system. This study developed an efficient preprocessing method to detect LPs from low-quality vehicles images. It contributed to improvements in the MLELPB_ELM method introduced in Section 1.3.2. It includes the combination of preprocessing methods, such as an enhancement cumulative histogram equalization (ECHE) and CLAHE. Those techniques were used for filtering unwanted LP regions or FP values and they reduced the dimensions of the features. The quality of normal vehicles images were kept during the enhancement stage. After that, the LBP and the histogram of the oriented gradient (HOG) were used as powerful descriptors for very sensitive and difficult conditions. They were used to extract the key features from three types of licence plate resolutions in the database. The same preprocessing and extraction techniques were used for both the training and testing phases. Then, the key extracted LP features were used as the inputs to feed the support vector machine (SVM) and ELM classifiers to build the trained models. A mean-shift algorithm was used with the detector to reduce redundant bounding boxes as well as FP values. For the performance assessment, detection metrics, object localization, and the ROC curve were used to evaluate this work. The experiments, compared the proposed method with the newest existing approaches. It achieved excellent results, in terms of detection accuracy, processing time, and FPR, 99.62%, 0.2408miliseconds, and 1.675%, respectively. The detail of this method was given in Chapter 5. The content of this chapter was published to *IEEE Access* journal.

1.3.4 Distorted vehicle licence plates detection using hybrid features and an extreme learning machine classifier

To make the LPD system more robust with high detection accuracy and less execution time, a new detection method was developed to add slight improvements in MHOG_LBP_SVM method introduce in section 1.3.3. The detection of LPs is similar to finding the ROIs that may contain the LP or non-LP. In this method, the preprocessing techniques of the MHOG_LBP_SVM method were developed to produce a new improvement technique,

enhancement contrast-limited adaptive-cumulative histogram equalization (ECLACHE). It was used to improve the distorted test images. Then, a new median robust ELBP (MRELBP) descriptor was used to extract different and difficult LP features from the improved images. This descriptor was developed recently by Liu et al. (2016) based on a median filter to make more improvements on the original ELBP descriptor. Also, a speeded-up robust feature (SURF) descriptor was used with the MRELBP descriptor to increase the extraction accuracy for more complicated LPs feature. Then, the ELM classifier with a mean-shift algorithm was used to classify the extracted MRELBP_SURF features to build strong LP detector. The performance of the MRELBP_SURF_ELM method can be observed in Chapter 6. The proposed method reduced both the range of the unwanted regions of the LP and the extraction time. The detection metrics using the confusion matrix and the ROC curve were used to evaluate the work. The experimental results show that the overall classification accuracy of the proposed algorithm is about 100% for all complicated LP conditions. The performance of the LPD system was very satisfactory, with good processing time and FPR, 99.71%, 0.323miliseconds, and 2.24%, respectively. The content of this chapter has been submitted to *the journal*.

1.3.5 Investigating which detection method is better to achieve the main requirements for detecting complicated vehicle licence plates

This thesis investigated which developed method has the best performance for LPD systems. This research compared the performances of four proposed methods, 3L-LBP_Adaboost, MLELBP_ELM, MHOG_LBP_SVM, and MRELBP_SURF_ELM, on two English car LP databases. The evaluations of the experimental results concluded that the last two methods, MHOG_LBP_SVM, and MRELBP_SURF_ELM were the best methods for detecting LPs from low-quality vehicle images. Those methods could be used for real-time applications. In this thesis, each proposed method presented a new idea to improve the performance of the LPD system by determining the drawbacks that face the detection system and implementing the solutions to fix it. The 3L-LBP_Adaboost method determined the best preprocessing and extraction techniques that should be selected to make LPD systems work very well under difficult conditions. After that, a new extraction and detection method, MLELBP_ELM, was developed to improve the weaknesses of the 3L-LBP_Adaboost method. This method achieved good detection results, but increased the extraction time. Moreover, the FPR was slightly improved, compared with that by 3L-LBP_Adboost method. This thesis developed new

methods to detect LPs in low quality and complicated vehicle images. These four approaches contribute to successful detection system performance. They can be used to improve the work of existing ANPR systems under difficult conditions. Hence they reduce the human effort to police activities in transport systems.

1.4 Structure of the Thesis

This thesis consists of seven chapters structured as follows:

Chapter 2 provides an overview of the ANPR system background. This chapter introduces briefly ANPR system applications and purposes, their components, phases, and concepts of classification, including its methods and the structure.

Chapter 3 introduces a new LP detection method based on extracted features from three preprocessing levels of the local binary pattern descriptor (3L-LBP). An ensemble of Adaboost cascade classifiers was used for detecting regions of the interest (ROI) for LPs. The results were compared with the results reported from the existing LP detection methods. This method can help improve the LPD systems for work under difficult conditions.

Chapter 4 presents a new extraction technique based on the multi-level preprocessing extended local binary patterns (MLELBP). The extreme learning machine (ELM) was employed for the classification task. The experimental results were compared with the results of the 3L-LBP_Adaboost method and the results of the existing LPs detection methods. This extraction method could help to improve the classification accuracy for the LPD system and produce a good detection rate under complicated conditions.

Chapter 5 focuses on developing an efficient preprocessing method for reducing the false positive rate (FPR) and increasing the accuracy rate for the LPD system. It includes the combination of an enhancement cumulative histogram equalization and a contrast-limited adaptive histogram equalization (ECHE_CLAHE) techniques. For extracting LP features, the combination of the local binary pattern and median filter histogram of the oriented gradient (LBP_MHOG) have been used. The support vector machine (SVM) classifier was used to train the extracted LPs features. This chapter investigates the performance of these techniques for improving the quality of the LPD system. It also investigates how much more efficient the SVM is in classifying the complicated features due to preprocessing techniques and good

descriptors. The results were compared with the results of the 3L-LBP_Adaboost, MLELBP_ELM methods, and the results reported from the newest existing LP detection methods.

Chapter 6 presents the modified version of the ECHE_CLAHE techniques. It includes the enhancement contrast-limited adaptive-cumulative histogram equalization (ECLACHE) technique. In addition, another combination of strong descriptors, the median robust extended local binary and the speeded up a robust feature (MRELBP_SURF) was used for extracting complicated LPs features. The ELM classifier was used to learn features and build the trained models, or detectors, to detect LP. This chapter investigates how the performance of the LPD system was improved by using this method.

Chapter 7 provides a summary and the findings of this research presented in this thesis. This chapter also discusses the ideas for future work.

Appendices A-D provide the simulation codes for the proposed approaches, which are presented in Chapters 3, 4, 5 and 6.

2

CHAPTER 2

OVERVIEW OF AN LPD SYSTEM AND ITS COMPONENTS

The goal of this thesis is to develop methods that are capable of detecting LPs from complicated vehicle images under difficult conditions. In order understand the detection system components, this chapter provides an overview of an LPD system structure. General concepts about detection system stages or components are discussed in Section 2.1. Section 2.2 introduces a core idea about the classification techniques of an LPD system. Section 2.3 overviews LPD system techniques that are currently used in each system stage. Section 2.4 provides brief details about LP detection methods used in each stage. Section 2.5 discusses different detection methods reported in the literature about LP identification. Finally, a brief summary of the detection system and its components and classification techniques for this thesis is given at the end of this chapter.

2.1 The background knowledge associated with LPD systems

The ANPR was firstly invented in 1976 in the UK at the Police Scientific Development Branch. The prototype of the systems worked in 1979, and the contracts were awarded to produce the industrial systems like the computer recognition systems (CRS) in the Wokingham, UK. Many early trial detection systems were deployed on the Dartford tunnel and the first arrest operation in 1981 was made through the detection of a stolen car (News 2005). However, ANPR systems did not become widely used until they became cheaper and easier to use during the 1990s. In the early 2000s, the collection of the ANPR data for future use to solve crime was reported (Taylor 2005). The first documented case in which an ANPR system was used to help to solve a murder occurred in 2005, in the Bradford, UK, where the ANPR system played an important role in locating the killers of Sharon Beshenivsky (News 2005). The next subsection describes system stages or components.

2.1.1 Detection system components

The software aspect for ANPR system runs on the standard hardware desktop computer and could be linked to the other applications with the same databases. It uses a series of image enhancement techniques to improve and normalize the input image with the number of an LP. Then, it uses many extraction and classification algorithms to extract and detect LP information. Generally, the ANPR systems are deployed in two basic ways. The first way allows for the entire detection process of LPs to be implemented in real time. The second way transmits all images captured by the system camera from many lanes to a remote computer location. Then, it implements the detection process with the delay in the time depending on the speed of processing methods used. There are four primary stages that the LPD system required for identifying an LP, which are listed below.

2.1.1.1 Vehicle images acquisition

The first stage is the vehicle image acquisition using a camera. The accuracy of any detection system depends on the quality of the camera, such as type, resolution, light, shutter speed, and the installation method. Sometimes many problems appear due to the short or long distance between the vehicle and camera which results in low/bad quality images.

2.1.1.2 Preprocessing stage

Once the vehicle image is captured, then further processing is carried out. It needs to improve the texture pattern by reducing the noise information in an LP background to enhance the processing speed for the extraction and detection stages. This stage has many steps applied to a vehicle image, such as resizing of the resolution, removal of the noise, and the conversion of the color from RGB to grayscale level or to the binary format (black and white) using several preprocessing techniques. Subsection 2.3.1 presents more details about those techniques.

2.1.1.3 Extraction stage

After the preprocessing stage, regions of interest (ROIs) are extracted from a vehicle image. This stage also influences the accuracy of the LP detection system. The vehicle image is cropped to the middle and the extraction methods are applied on every pixel of the rest vehicle image. This will reduce the processing time for the extraction stage, especially if those methods use unsupervised learning algorithms. But, the LP can exist anywhere in an image. Therefore, the extraction stage of the LP depends on some features, such as boundary, edges, color,

background as well as the texture features. These features can be used to identify the LP region using many extraction methods. In addition, two or more extraction techniques can be combined to extract further features. The detailed description of extraction techniques was given in Section 2.3.2.

2.1.1.4 Classification and detection stage

The third stage is to detect the LP area from the extracted features using many supervised learning algorithms. This stage depends on the quality of preprocessing and extraction methods to obtain the LP areas (Hongliang & Changping 2004a; Yousef et al. 2015). The relevant extracted features of the LP are classified by using classifiers to produce trained models. The detectors could detect different types of complicated LP features. There are many studies using unsupervised learning algorithms or preprocessing methods to detect two or more LP difficulties. This leads to increase the processing time for the detection system (Silapachote et al. 2005; Lee et al. 2013; Patel et al. 2013). The output result of this stage as a decision function to detect LP regions from vehicle image; whether it is LP or non-LP. Moreover, the aim of using supervised learning algorithms on this stage is to obtain high system performance with less execution time for the testing phase.

2.1.1.5 Segmentation and recognition stage

The final stage is to recognize the extracted characters from detected LP area. This stage uses many segmentation and recognition methods like the template matching techniques or classifiers, such as neural networks and fuzzy classifiers. In this stage, the LP number converts into machine-encoded text. Here optical character recognition (OCR) is used to recognize the plate numbers from the LP image.

Figure 2.1 shows the structure of the LPD system stages. The performance of the detection system relies on the robustness and reliability of each individual stage. This research focused only on the important stage of the LPD system, detection stage for detecting LPs from complicated vehicles images. Hence, the segmentation and recognition stages were not considered in this thesis.

The main objective of this study is to develop LP detection methods that yield better performance for detecting LPs from vehicle images having different difficult conditions, such as low/high contrast, dusk, fogy, distorted and so on (see Figure 2.2).

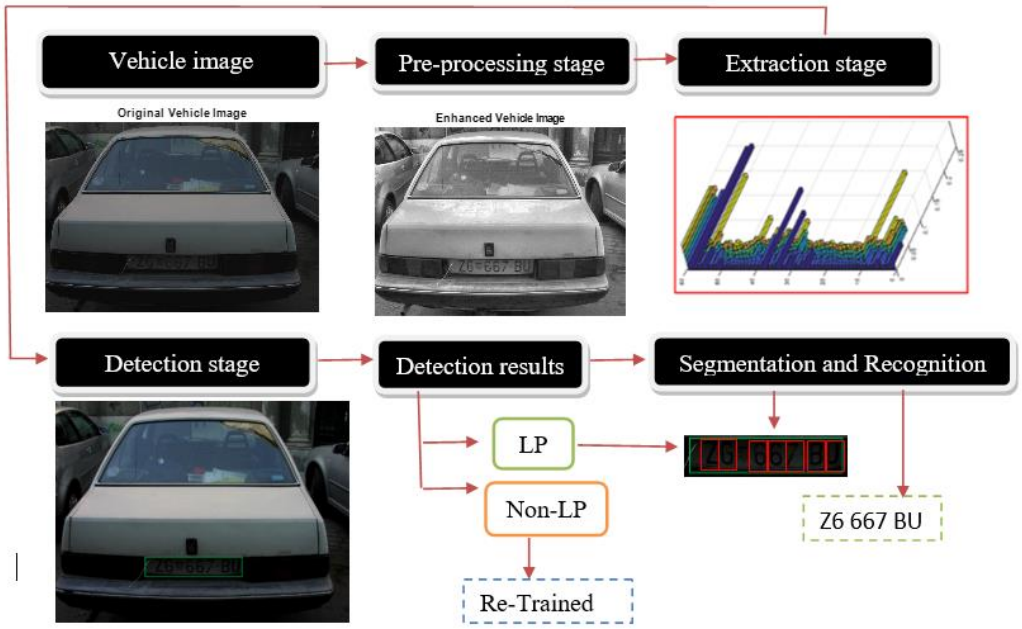


Figure 2.1. Stages involved in licence plate detection (LPD) system.

Figure 2.2. Examples of difficult conditions for licence plate detection.

2.2 Applications of LPD systems

The ANPR systems have become a very important tool in many surveilling applications over the past few decades. They are often used as a surveillance technique to identify LPs of vehicle images. There are several ANPR system applications, such as security systems (Sheldon 2013), highway road tolling systems (Song & Sarker 2014; Panahi & Gholampour 2017),

parking management systems, enforcing moveover laws for emergency vehicles (Roberts & Casanova 2012), traffic control (Dehghan et al. 2017), traffic management systems (He et al. 2017), and so on (see Figure 2.3).

The existing applications often work under some standard conditions, such as low/high lighting, rain, and limited day/night lighting. It is still very challenging to identify LPs from complicated vehicle images because of environmental effects.

Figure 2.3 Examples of ANPR system applications.

2.3 Overview of an LPD system classification

Classification techniques play an important role in the field of machine learning. As an LP contains many extracted data for analysis purposes, such as the classification. Therefore, it is very important to extract useful features from an LP image, and then use those features for the classification. The following subsections provide a detailed description of the LP classification methods in this research.

2.3.1 The classification concept

The classification task occurs throughout our daily life. It is a very essential means to make a decision, based on the available information. In the field of machine learning, the classification task denotes an algorithmic procedure. It works to assign one of a number of categories as input data (Brunelli 2009; Duda et al. 2012). The input data refer to instances and categories refer to classes. For example, the LP instance includes two classes or categories an LP or non-LP. This instance described by a vector of features which includes all known instance characteristics.

The main goal from the classification task is to assign the class labels for the extracted features among a set of data for a specific problem. The term classifier refers to a mathematical decision function which is implemented using the classification algorithm that maps the input data to it. The classifier is able to learn and identify the class of a features vector from training datasets. The training sets are included feature vectors with class labels.

The difficult LP has a large amount of different data, or values which describe relevant information of the LP area. These values, named "features", which are aggregated into a vector, the "features vector" (Makinacı 2005). Thus, the extracted feature process can be defined as a transformation operation to transform one or many LP features into a features vector. This operation helps to describe LP patterns and reduce the dimension of representation of those patterns. The LP classification means to classify different features of LPs and based on those features will be decided which class the LP belongs to. The output of the classification task is a decision function to present an LP or non-LP.

2.3.2 Types of classification techniques

There are two main types of machine learning algorithms: supervised and unsupervised (see Figure 2.4). In supervised learning, the observations of a group of data are related with the class labels (Zhu & Goldberg 2009; Duda et al. 2012). In unsupervised learning, the observations of a group or set of data are unlabelled or assigned to the known classes (Barlow 1989). The next subsection provides more details about those types of classification.

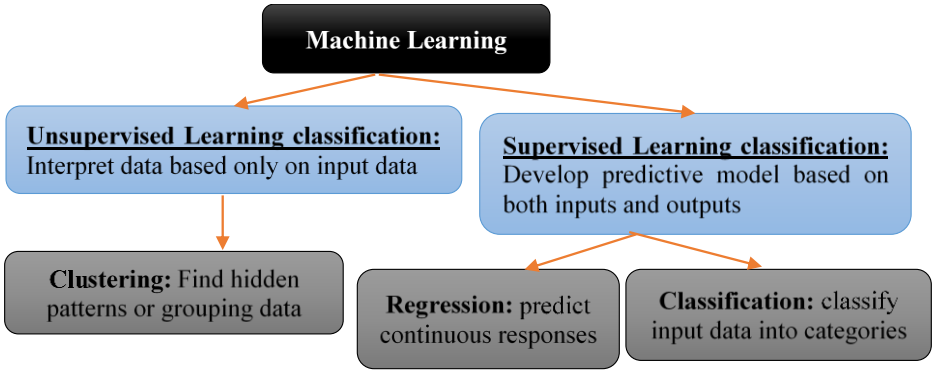


Figure 2.4: Types of classification learning algorithms

2.3.2.1 Supervised classification learning algorithms

Supervised classification is one of the algorithms associated with machine learning that deals with a set of data that have some information about the dataset. This type of classification, the

class labels information are given during the training dataset to train the classifier and produce the classification model (Brunelli 2009; Duda et al. 2012). The supervised approach proposes that a set from the training data have a set of relevant labelled instances which refer to correct output (Mohri et al. 2018).

In the supervised classification, given a set of the training examples N of the pairs form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$,where x_i is a set of features or the feature vector and y_i is the class label of the i-th example. The learning algorithm seeks about the function $g: X \rightarrow Y$, where X is the space of the input data and Y is the space of the output. For example, the spam filtering problem, the x_i refers to an email and y_i refers to either "spam" or "non-spam". Moreover, the class labels usually represent as an integer numbers y. Therefore, in the supervised classification learning, the aim found the transformation between the input feature space x and the output class label space y. If the output of class label has a known number of elements, such as $1, 2, \ldots\ldots, L$ then the problem was considered as classification task. In the case of classifying the LP problem, the classes labels are divided into two categories, such as the LP and non-LP which are represented as $y = \{-1, +1\}$ or $y = \{0, 1\}$. Figure 2.5 shows the structure of a supervised learning technique.

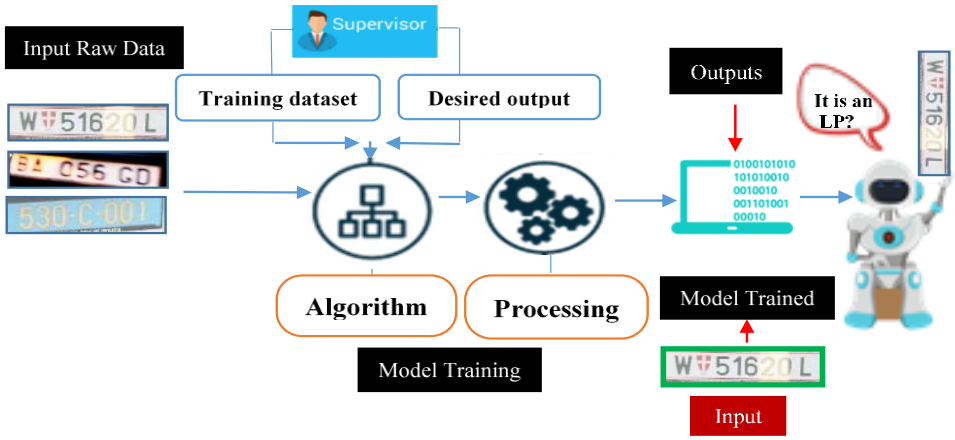


Figure 2.5: The structure of a supervised learning classification algorithm

There are many supervised classification learning algorithms, such as the support vector machine (SVM) (Cortes & Vapnik 1995), extreme learning machine (ELM) (Huang et al. 2004), Adaboost algorithms (Freund & Schapire 1995), linear discriminant analysis (LDA) (Fukunaga 2013), decision trees (Quinlan 1986), neural networks (NN) (Haykin 1994), logistic regression (Hosmer Jr et al. 2013), kernel estimation (Gasser & Müller 1979), linear regression (Seber & Lee 2012), Bayesian network classifier (Friedman et al. 1997), a fuzzy K-nearest-neighbor (kNN) (Keller et al. 1985), etc.

A database is usually divided into two sets, the training dataset and the testing dataset, in a typical supervised learning procedure. Using the training dataset, the classifier is constructed. After that, the performance of the trained classifier is evaluated by using the testing dataset. This evaluation process sometimes repeats for the different parameters of the constructed classifier. Therefore, the parameters of the classifier should be optimized in order to be ready for assigning the class labels to the features with unseen class labels. The main goal from the learning procedure is to maximize the testing accuracy on the testing dataset.

This study used the supervised learning algorithms to classify LPs regions to produce the trained models. During experiments, the English cars plate database is divided into two groups of data, which are the training and testing datasets as shown in Figure 2.6. The training dataset is used to train the classifier and build the trained model to detect an LP area, while the testing dataset is used for evaluating the performance of the trained model.

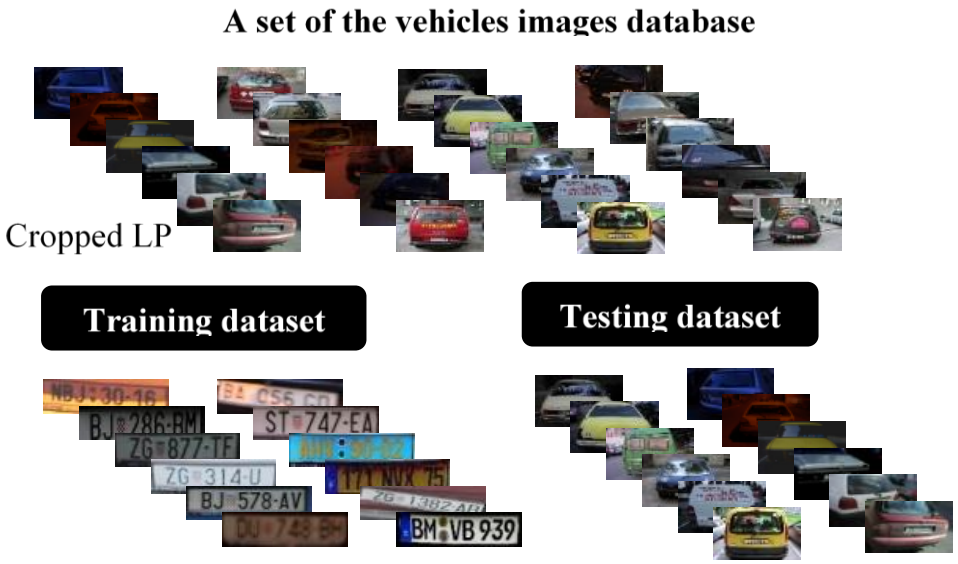


Figure 2.6: Dividing the vehicles images database into two groups: training and testing datasets.

2.3.2.2 Unsupervised learning classification algorithms

The unsupervised learning classification is the second approach of machine learning algorithms, that involves grouping of the unlabeled input data into classes to determine hidden patterns. This procedure assumes the training data have not been labeled and try to find the inherent patterns in the data to determine the correct output value for a new instance data (Brunelli 2009; Duda et al. 2012). In this type of learning, the class label information is not

provided even for a small number of data. Figure 2.7 shows the structure of an unsupervised learning technique.

The common examples of unsupervised learning algorithms, K-means clustering (Hartigan & Wong 1979), principal component analysis (PCA) (Jolliffe 2011), hierarchical clustering (Johnson 1967), kernel principal component analysis (Kernel PCA) (Schölkopf et al. 1997), independent component analysis (ICA) (Hyvärinen & Oja 2000), hidden Markov models (Rabiner & Juang 1986), categorical mixture model (Oberski 2016), and so on.

Combination of the two classification learning procedures (supervised and unsupervised) has been recently explored (Chapelle et al. 2009), is a semi-supervised learning procedure, which is used a combination of the small set of labeled data and a large set of unlabeled data.

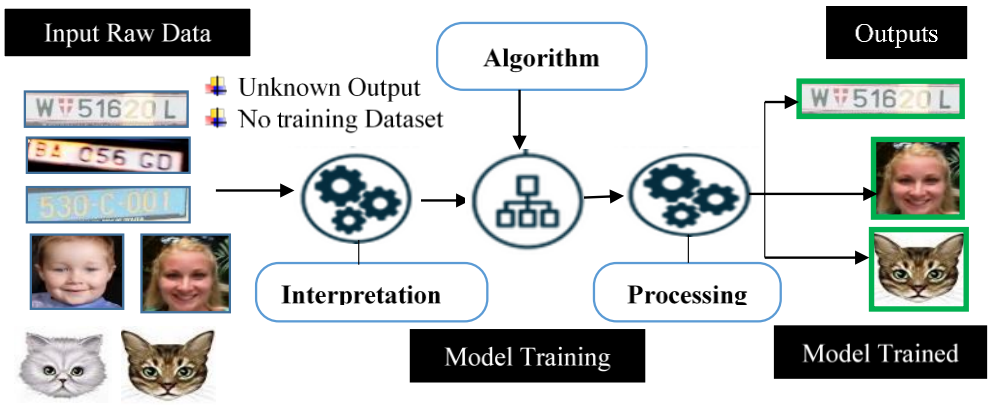


Figure 2.7: The structure of a unsupervised learning classification algorithm

2.3.3 Structure of a LPD classification

A classification process includes two stages: feature extraction and classification. The extraction for the most important LP features values is done at the extraction stage. The classification stage requires a classifier to determine the correct class of the LP area based on the extracted features. The concept of an LPD classification is provided in Figure 2.8. From this figure, it can be seen that appropriate LPs features were extracted from the LPs features space. At the LP features space, the LP features are divided into two classes, the LP and non-LP.

In this thesis, five extraction methods were used to extract complicated LP features under difficult conditions. Three levels local binary pattern, multi-levels preprocessing extended local binary pattern, median robust extended local binary pattern, median filter histogram of the oriented gradient, and the speeded-up robust feature extraction. While the

supervised learning algorithms are Adaboost, extreme learning machine, and support vector machine was used to perform the classification stage to classify and train the extracted LP features.

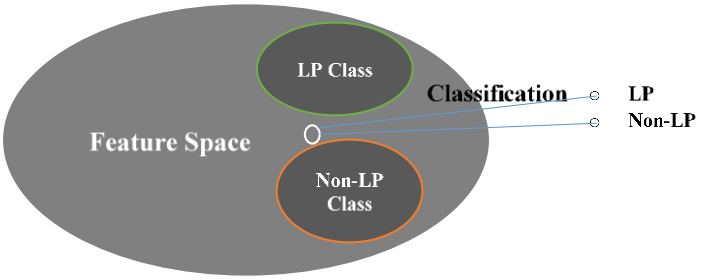


Figure 2.8: The process of the licence plate classification.

2.4 Overview of processing stages in licence plate detection methods

In this section the basic processing stages for developing a ANPR system are described.

2.4.1 Image preprocessing stage

Researchers have proposed different detection methods for the image preprocessing stage. While others have used this stage for both system phases: testing and training. Initially, the vehicle image is captured using a high-quality camera. After that, the preprocessing is applied to the captured image to improve the image and reduce unwanted features or noise. In the next step, the improved image is converted from a color RGB image to the grayscale image depending on a threshold value (Kaur & Kaur 2014). An image preprocessing stage includes the following methods.

2.4.1.1 Image conversion

The converting of a color RGB image to a grayscale one is a very important step for ANPR system stages. Sharma et al. (2014) used a wavelet transform, Gaussian filter, and then preserved the high-frequency component. Al-Shemarry et al. (2018) and Al-Shemarry et al. (2019) converted a color image into a grayscale one and used the preprocessing filter methods to filter out unwanted features or false positive values.

2.4.1.2 Binary processing

The next step is to convert a grayscale image to a binary representation. Samra and Khalefah (2013) suggested using a dynamic adaptive threshold method to avoid the illumination variation present in the vehicle image. There are many methods to convert a grayscale image

into binary, such as the Otsu algorithm, which is faster than the Sauvola and Niblack algorithms. Both methods are based on a thresholding value of an image and provided good results for the poorly illuminated images (Puloria & Mahajan 2015).

2.4.1.3 Noise removal

There is still some noise remaining in the grayscale image. Therefore, different filtering techniques are applied to remove this noise. The filters that are widely used in ANPR systems are Gaussian and median (Prabhakar et al. 2014; Karwal & Girdhar 2015; Azam & Islam 2016; Al-Shemarry et al. 2019). Kaur and Kaur (2014) suggested that the noise can be removed by using an iterative bilateral filter in the transport systems. Some researchers proposed blurring the image to filter out the noise (Beibut et al. 2014; Al-Shemarry et al. 2018).

The steps above are not fixed. Karwal and Girdhar (2015) proposed different steps to remove noise. The first step is to convert the captured image to grayscale. The second step is to apply a median filter to filter out the noise. The final step is to apply the Otsu method. Beibut et al. (2014) only used the Otsu algorithm for image binarization. Moreover, Kaur and Kaur (2014) in the image preprocessing stage implemented some different steps. After converting a color image RGB to a grayscale, they applied an iterative bilateral filter to remove the noise. Then the contrast of the image was enhanced using the contrast adaptive histogram equalization (CAHE) method (Sharma & Kaur 2011; Al-Shemarry et al. 2018, 2019). Selecting appropriate methods for the preprocessing stage is a very important process for LPD systems to obtain robust detection results in less time.

2.4.2 Extraction methods for LPD systems

In this stage, the LP regions from the image are being extracted out after the enhancement at the preprocessing stage. The LP extraction stage influences the accuracy of an LPD system. The input data to this stage is a vehicle image and the output is a part of the vehicle image that contains the potential LP region. The LP can exist anywhere inside the vehicle image, and it can be distinguished by its features. The LP color is one of the features since some countries have certain colors for their LPs. A rectangular shape of the LP boundary is another type of features that are used to extract the LP. Also, the color change between the LP characters and the LP background, which is known as the texture, is used to extract the LP region from vehicle images. The combination of two or more features can be identified as the LP. In the following subsections, the existing LP extraction methods are categorized depending on the type of features used.

2.4.2.1 Licence plate extraction using boundary/ edge information

Because of the shape of an LP is normally rectangular with known window size, it can be extracted through finding all the possible rectangles in a car image. Edge detection techniques are commonly and widely used to find these rectangles (Wang & Lee 2003; Hongliang & Changping 2004b; Zheng et al. 2005; Faradji et al. 2007). Many methods, Sarfraz et al. (2003), Zheng et al. (2005), Kanayama et al. (1991), Kamat and Ganesan (1995), and Sanyuan et al. (2004) used the Sobel filter to detect the LP edges. Due to the color transition between the LP and the vehicle body, the boundary of the LP is represented using edge characteristics. The LP edges have two horizontal and vertical lines when performing horizontal and vertical edges detection. Then, getting a complete rectangle when performing both edges at the same time. Al-Ghaili et al. (2008) proposed a new vertical edge extraction method. It showed that it is faster than the Sobel filter by about seven to nine times. The block-based with high edge magnitudes is identified as the possible LP areas. Since the block processing does not depend on the edges of the LP boundary, it can be applied to an input image with an unclear LP boundary (Lee et al. 2004). A boundary-based extraction uses the Hough transform (HT) (Kamat & Ganesan 1995; Duan et al. 2004a). It detected the straight lines in the vehicle image to locate the LP. It has the advantage of detecting the straight lines in an image with up to 30° inclination. Boundary line based methods also use the HT combined with a contouring algorithm (Duan et al. 2005). Kim and Chien (2001) used the generalized symmetry transform (GST) to extract the LP features. after getting the edge features the car image scanned in selective directions for detecting LP corners. Then, the GST used to detect a similarity between those corners and to form LP regions.

2.4.2.2 Licence plate extraction using global features

The connected component analysis (CCA) technique is important for a binary image processing (Matas & Zimmermann 2005; Qin et al. 2006; Wu et al. 2007; Anagnostopoulos et al. 2008). It scanned a binary image and labeled image pixels into components depending on pixels connectivity. The spatial measurements like area and the aspect ratio (M×N) are commonly used for LP extraction (Bellas et al. 2006). Chacon and Zimmerman (2003) applied a contour detection method on the binary image to identify the connected objects which are chosen to be the LP candidates due to having the same geometrical features. This method failed to detect LP in the case of the poor quality images because of the distorted contours. Miyamoto et al. (1991)

used the 2-D cross-correlation method to find the LP. This method with a pre-stored LP template is performed over the entire vehicle image to locate the most likely LP area.

Extracting LPs using correlation with the template is independent of the LP position in an image.

2.4.2.3 Licence plate extraction using texture features

This type of extraction method depends on characters present in the LP area. It results in significant changes for the greyscale level between the characters color and LP background color. Many texture extraction techniques are used by researchers (Yang & Ma 2005; Muhammad & Altun 2016b; Azam & Gavrilova 2017; Tsai et al. 2017; Al-Shemarry et al. 2018, 2019). Muhammad and Altun (2016b) employed the genetic algorithm (GA) method to perform the LP detection by repeatedly selecting extracted points of the HOG descriptor within the car image randomly. Then, it evaluated the LP regions at these points. Finally, it selected the regions of the LP which give the best similarity score. Tsai et al. (2017) used the modified histograms of oriented gradients (MHOGs) to extract the principal direction for each pixel in an image. Then, they determined the principal direction of each sample and its component cells. Al-Shemarry et al. (2018) extracted LPs feature from a three preprocessing levels using filtering methods with a powerful texture descriptor, local binary patterns (LBP). This method succeeded to extract different and difficult features from the distorted LP image (see chapter 3). Also, Al-Shemarry et al. (2019) used a new texture extraction method, MLELBP, to extract complicated features for LPs. This method used the new texture descriptor with preprocessing techniques to extract multi feature from three types of ELBP descriptors (see chapter 4). Deb et al. (2009) proposed an LP detection method based on using a sliding window and the histogram technique. Image transformations tools are also widely used in the LP extraction methods, such as the Gabor filter which is one of the main tools for the texture analysis (Caner et al. 2008). It has the advantage to analyze the texture in unlimited scales and orientations. Parisi et al. (1998) used the discrete Fourier transform (DFT) to detect the horizontal position in a row-wise fashion and the vertical position in a column-wise fashion of the LP.

The texture features make the classifier very invariant to the color, brightness, and size changes. Therefore, all the methods that are based on the texture features have the ability to detect the LP even if its the background and boundary are very deformed. Due to the many advantages of texture descriptors, this research focused on using this type of extraction method to achieve its objectives.

2.4.2.4 License plate extraction using color features

Since some countries have specific LP colors, this work requires the extraction of LPs by identifying their colors inside the car image. The simple and easy idea is that the combination of the LP colors and characters is unique. This combination appears almost only in the LP regions (Shi et al. 2005). According to a specific format of Chinese LPs, Shi et al. (2005) proposed that all the input pixels in an image are classified by using the hue, lightness, and saturation (HLS) model color into 13 categories. Lee et al. (1994) converted the RGB image color into the HLS, then used a neural network to classify the color feature of each pixel. The neural network outputs were green, red, and white colors for Korean LPs. The same LP color is projected horizontally and vertically to identify the highest color density of the LP regions. Pan and Li (2010) employed a fast mean-shift method to deal with the illumination variation problems related to the color based method. Wang et al. (2008) employed the hue, saturation, and value (HSV) color features space to extract the LP features. Wan et al. (2011) proposed a new method to localize the LP using the color barycenters hexagon model which is only slightly sensitive to the brightness conditions.

Extracting the LP using color features has the ability to detect the deformed and inclined plates. However, also it has several difficulties, especially for different illumination conditions.

2.4.2.5 License plate extraction using character features

There are many LP extraction methods proposed that are based on locating characters positioned inside the image. The existence of the character is examined using those methods. Then, the regions of characters are extracted as LP regions if the character is found. After that, the corresponding LP region is detected. Hontani and Koga (2001) extracted characters using the scale space analysis tool. It extracted the large size of blob type figures, which consists of the smaller line-type figures as the LP character candidates. The character regions are recognized through the difference between the character background and its region and the width of the characters. Then the LP is extracted by finding the distance of the inter-character space in the plate region (Cho et al. 2011). Yongchun and Jing (2012) introduced a method to find and extract all the characters that look like LP regions in a car image, instead of using the LP features directly.

These extraction methods using characters from the binary car image to define the LP region are time-consuming due to processing all the binary objects which look like an LP

object. Moreover, these methods produce many false positive values when there is other text inside the image.

2.4.2.6 License plate extraction combining two or more features

Many methods used a combination concept as an effective way to extract two or more features for LPs regions. In this case, the extraction methods called hybrid extraction methods (Le & Li 2006). The color and edges features were combined in Xu et al. (2004) and Wang et al. (2010) methods. Wu et al. (2009) applied the hue and lightness sub-band (HLS) feature of the 2-D discrete wavelet transform (DWT) to significantly highlight the vertical edges of the LP. The most probable candidate was selected by edge density verification and an aspect ratio constraint. Al-Shemarry and Li (2019b) used the combination of MHOG and LBP features to extract a multi LP features from complicated vehicle images (see Chapter 5). Also, Al-Shemarry and Li (2019c) introduce robust extraction method using the MRELBP descriptor combined with SURF features to extract LPs regions from the vehicle image under distorted conditions (see chapter 6). Mao et al. (2010) proposed a new extraction method using the wavelet analysis and improved HLS color to extract LPs regions.

2.4.3 LP classification and detection stage

In the detection stage, the LP area is being detected from the vehicle image using a classifier, detector, or trained model. There are different kinds of the classifiers that are used to classify extracted LP features, such as SVM, ELM, Adaboost, neural networks (NNs), convolutional neural network (CNN) and so on. Recently, deep neural networks (DNNs) (Masood et al. 2017) and CNN (Li & Shen 2016; Liu et al. 2018) were used to learn the key features of LPs. They showed a good detection accuracy. However, the learning mechanism for DNNs in difficult conditions, such as image rotation and scaling, cannot have robustness guaranteed unless the training dataset covers all the various LP conditions. The SVM classifier is widely used for LPD (Yuan & Cheu 2003; Ho et al. 2009; Sun & Watada 2015). Al-Shemarry et al. (2019) used the SVM with a mean-shift algorithm to train the combination of the MHOG and LBP features to detect the LP area for complicated vehicles images (see chapter 5). Also, many multiclass identification algorithms are used, such as NNs (Park et al. 1999; Yuan & Cheu 2003; Porikli & Kocak 2006) and the AdaBoost cascade classifier (Ho et al. 2009; Chen et al. 2015; Al-Shemarry et al. 2018). A cascade classifiers had shown good performance with other methods in terms of the detection process speed (see chapter 3). However, these algorithms are

not very robust if they are used without efficient enhancement techniques that should be applied to both system data phases: the training and testing. CNN is also used to learn features with ELM (Ding et al. 2017). This method yielded competitive results with less execution time compared with the DNN methods. Also, the ELM was used to detect LP regions with the HOG and by means of the maximally stable external region (Gou et al. 2014). The ELM classifier was used to classify a MLELBP features to detect complicated LPs (Al-Shemarry et al. 2019) (see chapter 4). Moreover, an ensemble of the ELM classifiers also was used to classify the combination of MRELBP and SURF features (see chapter 6). There are also many ensemble classifiers proposed to detect different LPs conditions (Zhao et al. 2010).

2.5 Existing methods of the LPD system

Over the last decade, many studies in the area of ANPR systems were conducted (Du et al. 2013). Some of the proposed detection systems depend on the color and language, some are sensitive to the illumination and complex background, while some are restricted to good weather conditions (Azam & Islam 2016). In addition, the camera distance and angle restrictions make detection systems less robust (Anagnostopoulos et al. 2008). According to the Azam and Islam (2016), detecting the LP area in hazardous conditions is not an insignificant task, especially when the complex background of the LP produces a number of the false positive values. However, this method proposed using different LPs parameters for different conditions. Recent studies between 2014 and 2016 used a fixed threshold value to maximize the method accuracy based on filtering criteria, such as edge density, counting intensity transition, color, and height and width (Samra & Khalefah 2013; Baharlou et al. 2015; Davis et al. 2015; Yousef et al. 2015). The car images used to evaluate detection systems typically had simple complexity in the background (Gerber & Chung 2016; Muhammad & Altun 2016a). Overall, most of the existing LPD methods did not consider filtering techniques comprehensively. Kusakunniran et al. (2014) used the SVM as the machine learning tool which was applied directly to the LP images without extracting any discriminative LP features. This makes the LPD system very sensitive to geometric transformation and noise. It reduces the detection accuracy and increases the training time. Baohua et al. (2010) proposed finding an LP location based on histogram equalization. This method addressed noise caused by light conditions. The histogram equalization technique works well when the LP image is affected by one noise source (e.g. light condition) but not with the different sources. Al-Ghaili et al. (2012) proposed a car LP detection method based on the vertical edge by using the Sobel gradient information. This method claims that it works very well with low contrast images conditions

captured using web cameras, but it may not be robust with complex backgrounds. Yu et al. (2015) introduced a method to find the LP location based on the wavelet transform with empirical mode decomposition (EMD) analysis tool. This method used the histogram equalization to improve the edge details of texts inside the image. However, it depended on heavily on the horizontal and vertical information of the LP characters. Panahi and Gholampour (2017) proposed a method for high-speed applications under dirty LP conditions. It used a specific device to capture vehicle images on the highway at night. Most of LP detection methods use one of the simple enhancement techniques like the histogram equalization for improving the information of the input images. This confirms that the enhancement step is very important for LP detection. Weng et al. (2015) presented a multi-spectral fusion algorithm for the degraded video frame text enhancement. It explored the RGB color information with statistical features to find the right combination for different feature bands. However, this method focused on non-uniform illumination effects but not with effects caused by other sources.

From studies listed in the literature, many LP extraction and detection methods have been developed. But there are still some limitations that must be considered. These limitations could be overcome using some further enhancement techniques. The work in this thesis aims to introduce methods to overcome these difficulties. These proposed methods enhance the existing LPD systems based on powerful preprocessing and extraction methods that are selected carefully to detect LPs from complicated vehicles images.

2.6 Chapter summary

An overview of the LPD system was provided in this chapter. Also, it presented the necessary background information about the steps of detecting the LP from vehicle images. Firstly, this chapter outlined of LPD system invention and development. Then, it discussed the system components and the fundamentals of the LPD system stages. The classification concept was discussed and the current methods reviewed in each stage of the LPD system. It identified and justified the methods that are used in this thesis to detect the LP from distorted vehicles images. Also, it reviewed the recent detection and classification methods for LPs.

From the literature, it can be concluded that there are still many limitations in the existing LP detection methods. Hence new detection algorithms are required to increase the reliability and accuracy in different ITS applications.

In the next chapters 3, 4, 5, and 6, new detection methods are developed to detect complicated LPs from low-quality vehicle images. Those methods could help to improve the work of existing ANPR systems under complicated conditions.

3

CHAPTER 3

ENSEMBLE OF ADABOOST CASCADES OF 3L-LBPS CLASSIFIERS FOR LICENCE PLATES DETECTION WITH LOW QUALITY IMAGES

3.1 Introduction

The content of this chapter is an exact copy of the published paper in the journal of *Expert Systems with Applications* by Al-Shemarry et al., (2018) 'Ensemble of Adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images', vol. 92, pp. 216-35.

The development of detection methods for ITS systems is essential in the field of machine vision. Many researchers work on finding reliable techniques to increase the performance of LPD systems under critical conditions. This chapter presents a new LP detection method for complicated vehicles images. It includes extracting features from three preprocessing levels of a local binary pattern descriptor using an ensemble of Adaboost cascade classifiers (3L-LBP_Adaboost). The aim of this study is to determine an optimal detection scheme with preprocessing methods to extract the ROI features of the LP object under various complicated conditions like low/high lighting and contrast, dusk, dirt, and foggy. This method was implemented on the English cars database (EnglishLPDatabase-2001 ; MedialabLPRdatabase-2007) for evaluation purposes and also it was compared with recently existing methods using the same database. The reason for selecting this database was that it contains a number of vehicles images that have different conditions which made it a strong, good database for testing different LPs difficulties (see Appendix E). The proposed method yields a high detection rate and a good processing time, compared with the state-of the art

approaches. Furthermore, the proposed method may improve the performance of existing LPD systems in detecting LPs under difficult conditions.

Appendix A provides the simulation Matlab code for the proposed LPD method.

Expert Systems With Applications 92 (2018) 216–235

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images

Meeras Salman Al-Shemarry[a,*], Yan Li[a], Shahab Abdulla[b]

[a] School of Agricultural, Computational and Environmental Sciences, Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia

[b] Open Access College, University of Southern Queensland, Australia, QLD 4350, Australia

ARTICLE INFO

Article history:
Received 4 May 2017
Revised 22 August 2017
Accepted 11 September 2017
Available online 21 September 2017

Keywords:
License plate detection (LPD)
Region of interest (ROI)
Adaboost Learning algorithm
Cascade classifier
Local binary pattern classifiers (LBP)

ABSTRACT

Due to the plate formats and multiform outdoor illumination conditions during the image acquisition phase, it is challenging to find effective license plate detection (LPD) method. This paper aims to develop a new detection method for identifying vehicle license plates under low quality images using image processing techniques. In this research, a robust method using a large number of AdaBoost cascades with three levels pre-processing local binary patterns classifiers (3L-LBPs) are used to detect license plates (LPs) regions. The method achieves a very high accuracy for detecting LP number from one vehicle image. The proposed method was tested and trained with the images from 630 and 400 vehicles, respectively. The images involve many difficult conditions, such as low/high contrast, dusk, dirt, fogy, and distortion problems. The experimental results demonstrate very satisfactory performance for LP detection in term of speed and accuracy, and were better than the most of the existing methods. The processing time for the whole testing LPD system was about 1.63 seconds to 2 seconds. The overall probability detection, precision, and f-measurement are 98.56%, 95.9% and 97.19%, respectively; with false positive rate 5.6%.

1. Introduction

Nowadays, intelligent transportation systems (ITSs) play a very important role in our daily life in many aspects. An ITS normally consist of two parts: a smart infrastructure system and an automatic number plate recognition system (ANPR) (Anagnostopoulos, 2014; Anagnostopoulos, Anagnostopoulos, Loumos, & Kayafas, 2006). It is necessary to examine and observe the road traffic to avoid unacceptable behaviors using surveillance applications (Castello, Coelho, Del Ninno, Ottaviani, & Zanini, 1999; Chakraborty & Parekh, 2015; Duan, Duc, & Du, 2004; Sarfraz et al., 2013). The first successful ANPR was recorded in 1978 for the detection of stolen cars in UK. Such an ANPR system is also named as optical character recognition (OCR), automatic license plate recognition (ALPR), or car plate recognition (CPR). An ANPR system has many different applications for a variety of purposes, such as for highway road tolling systems, security systems, parking management systems, and so on (Azad & Ahmadzadeh, 2014; Baharlou, Hemayat, Saberkari, & Yaghoobi, 2015; Dehshibi & Allahverdi, 2012). Currently, the ANPR still has big problems which are described below. Therefore, many researchers in the field of machine vision have tried to find modern and reliable methods to build an ITS. The main objective of an ANPR system is to identify a vehicle license plate from images or a sequence of images in a video. Those images are often captured from high quality cameras installed on the street lights, road traffic signs, high buildings or motorway overpass (Azam & Islam, 2016; Valera & Velastin, 2005; D. Zheng, Zhao, & Wang, 2005). LPD means extracting LP number from captured image which is the one of the most important stage of ALPR system (D. Zheng, Zhao, & Wang, 2005). The ANPR system involves four stages as shown in Fig. 1. The first one is the vehicle image acquisition using a camera. The accuracy of an ANPR system depends on the parameters of a camera, such as type, resolution, light, shutter speed, and the installation method. The capture vehicle image needs pre-processing stage to reduce the noise on LP background information and enhance the processing speed for detection and recognition stages. The key requirements for a high quality ANPR are high accuracy and processing speed for real-time application (Angelova, Krizhevsky, Vanhoucke, Ogale, & Ferguson, 2015). The second stage is to detect the LP area from acquired images. The LP detection stage depends on the quality of the images and the type of processing methods used to obtain the LPs images (Hongliang & Changping, 2004; Yousef, Al-Tabanjah, Hudaib, & Ikrai, 2015). In this stage, the LP region extracts from vehicle image as a region of interested and eliminates the unwanted background features by using many pre-processing algorithms and learning algorithms for

* Corresponding author.
E-mail addresses: MeerasSalmanJuwad.Al-Shemarry@usq.edu.au (M.S. Al-Shemarry), Yan.Li@usq.edu.au (Y. Li), Shahab.Abdulla@usq.edu.au (S. Abdulla).

https://doi.org/10.1016/j.eswa.2017.09.036
0957-4174/

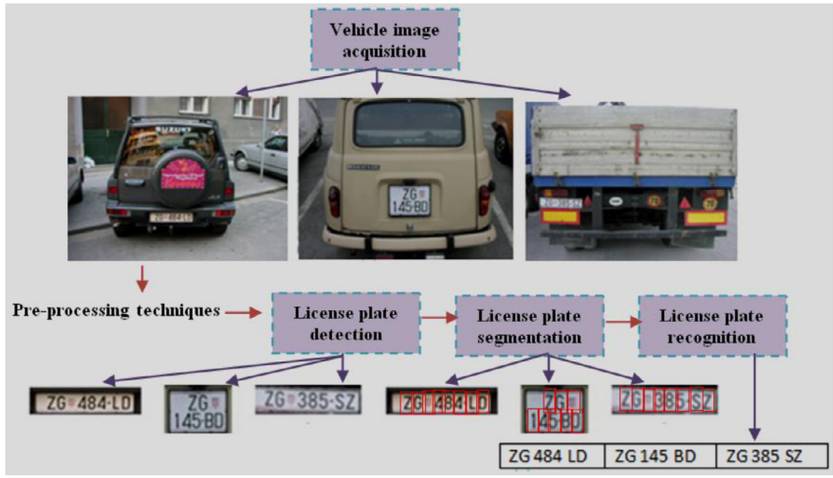


Fig. 1. ANPR system stages.

different features of LP, such as edge information, texture features and color features. The aims of using learning algorithm for this stage to obtain high performance in terms of the detection rate and processing time for testing phase. Moreover, the trained model from those algorithms can detect multi features values for different problems of LPs unlike the pre-processing algorithms (Lee, Han, & Ko, 2013; Patel, Shah, & Patel, 2013; Silapachote, Karuppiah, & Hanson, 2005). The third stage is to segment the LP area and extract the characters by using many techniques, such as projecting color information, labeling, or matching positions with templates. The final stage is to recognize the LP extracted characters by using template matching or using classifiers, such as boosting, extreme learning machine, neural networks and fuzzy classifiers (Baharlou et al., 2015; Chakraborty & Parekh, 2015; Han, Lee, Lim, & Chung, 2015). This stage needs many Samples of characters as inputs for training in advance. Then, the input image of segment characters is compared with the trained data to produce the output results. There are many surveys have been conducted by many authors (Atiwadkar, Mahajan, Lande, & Patil, 2015; Bhardwaj & Mahajan, 2015; Du, Ibrahim, Shehata, & Badawy, 2013; Panchal, Patel, & Panchal, 2016; Patel et al., 2013; M. Sarker, Mostafa, Yoon, Lee, & Park, 2013) related on ALPR system problems which are affected on detection and recognition stages:

i Low resolution problems related with camera quality and the distance between vehicle and camera.
ii Plate problems such as blurry, location, sizes, special symbols and fonts, occlusion, tilted, blurry LP backgrounds, distortions, and screws.
iii Environmental problems, such as lighting, rainy day, snow.
iv Illumination problems, such as vehicle headlights, and different lighting sources during image capturing.

In the past and recently time, many efforts have been done to develop a robust LPD system, but they missed most of LPs issues which make the LPD systems very limited for detecting LP level. Therefore, an efficient LP detection method is still needed to make a robust LPD system. In this paper, we focus only on the LP detection area from a vehicle image, so we not consider the segmentation and recognition stages of an ANPR system. The main objective of this study is to develop a LPD method that yields better performance for vehicles images having different difficult conditions, such as low/high contrast, dusk, fogy, and distorted. It employed a large number of AdaBoost cascades classifiers with three-levels LBPs (3L-LBP) features to detect the ROI area for LP from vehicles images. The paper is organized as follows: The first section introduces the ANPR system. The second section provides an overview of the related work about ANPR systems. Section 3 presents the proposed method. The experimental results are reported in Section 4. Finally, this study is concluded with some useful recommendations and suggestions for the future work.

2. Related work

Over the years, there are many algorithms being developed to extract LP features from one image or a sequence of images (video). Those features are used as the input to various classifiers such as cascade classifier, neural networks and fuzzy logic classifiers (Du et al., 2013). Different features, such as Haar-like feature, LBP features, ROI features, color features, boundary features, edge features and texture feature (Anagnostopoulos, Anagnostopoulos, Psoroulas, Loumos, & Kayafas, 2008; Azad, Davami, Jeo, & Shayegh, 2014; He, Zhang, Jia, Wu, & Hintz, 2007; Jia, Zhang, & He, 2007; Zheng, Zhao, Gu, & Hu, 2012), are used either separately or combined together to detect the LP region from images. In this paper the proposed method uses ensemble of AdaBoost cascades of 3L-LBPs classifiers for extracting ROI features from the LP area. It is usually one AdaBoost cascade being employed to detect the LP area. Throughout our literature review, many methods for the LP detection have been developed for real time LPDs (Gao & Lee, 2015; Li & Shen, 2016; Lienhart & Maydt, 2002; Porikli & Kocak, 2006;Sarker & Song, 2014; Song & Sarker, 2014). A brief overview about those existing methods is discussed below.

A strong classifier reported by Viola and Jones (2004) was trained using an AdaBoost algorithm and Haar-like features. It performed well for face detection. The study used “integral image” to calculate Haar-like features and used the AdaBoost algorithm to reduce the Haar-like features, and trained one cascaded classifier. The cascade classifier involved several stages to discard unwanted regions (non-face) from the image and saved the interested regions (face) for future processing. The accuracy rate by that algorithm was 96%. Ho, Lim, and Tay (2009) used two stages methods to extract the LP features. Several LP regions were identified in the first stage using a gentle AdaBoost classifier. In the second stage, the false positive rate was filtered using a support vector machine (SVM) classifier based on a scale-invariant feature transform (SIFT). The accuracy rate for the LP detection was 92%. A principal visual word (PVW) technique was developed by Zhou, Li, Lu, and Tian (2012) to locate the LP by local feature matching together with the PVW. The accuracy rate for the LP detection was 84.8%. Lim and Tay (2010) designed a character based method, maximally stable extremely regions (MSER) method to detect char-

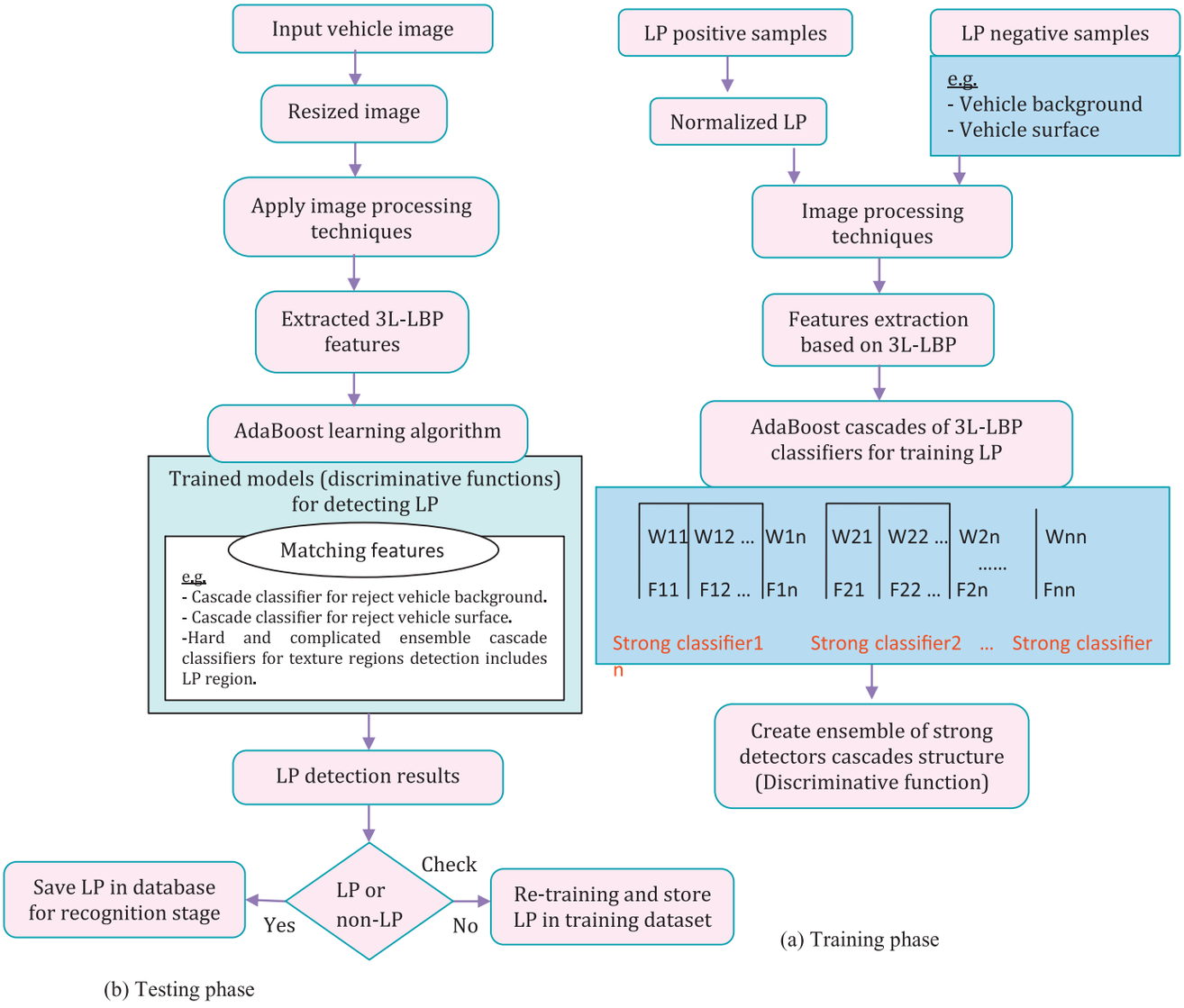


Fig. 2. (a) and (b). The framework of the proposed LPD system using the ensemble of AdaBoost cascades of the 3L-LBP classifiers for training and testing.

acters inside images and trained the SIFT-based unigram classifier using a core vector machine learning method to filter out the false alarms. The accuracy rate for the LP detection was 90.47%. Nguyen and Nguyen (2012) introduced an efficient algorithm for the real-time LP detection from images captured in real scenarios using a boosting technique for training an LP detector. A local binary patterns (LBPs) classifier was employed with traditional Haar features for discriminating LP image patches and also used a new mechanism to evaluate and improve the system performance through reducing false positive errors. The accuracy rate for the LPD was 85.2%. Wang, Sang, Wang, and Kuang (2013) used an AdaBoost classifier to detect LPs. A morphology based method was used to reduce some false positive rate and to identify LP regions. Then, the trained SVM classifier was used to verify the possible LP regions and to remove the non-candidate LP regions. The accuracy rate for the LPD was 88.28%. T. He, Yao, Zhang, Hou, and Han (2014) proposed a method to locate the regions of interest for multi-scale LPs in different inclination directions. This method used a blob technique with a filtering affine distortion method to detect the ROI area. The accuracy rate for the LP detection was 94.7%. Azam and Islam (2016) presented an effective LPD method to detect the LP area in an image under rainy conditions. A frequency domain mask was used to remove rain drops from the image. For handling indoor contrast, blurry, and night conditions, a new contrast improvement technique with a statistical binarization method was used. A random transform based on a tilt correction approach was applied to correct the tilted LP. The overall accuracy rate for the LPD was 94%. M. D. A. Asif, Tariq, Baig, and Ahmad (2014) introduced the YDbDr colour space to identify the blue regions, whereas a simple colour detection method used to identify yellow LP regions. The Otsu method used to gain binary image and the connected component analysis used to obtain on LP regions. The accuracy rate was 93.86%. Chen, Han, Ho, and Fan (2015) extracted the LP using a feature-salience theory with rectangle shape, texture, and colour features. The accuracy rate was 97.3%. Lee, Song, Ku, Jeon, Han, and Ko (2010) used local structure patterns that were calculated from the modified census transform (MCT) to extract the LP from vehicle images. The accuracy rate was 88.9% for the LP detection. He et al. (2007) also used an AdaBoost algorithm based on the both global statistical and local Haar features for LP detection. The accuracy rate for LPD was 96.4%. Hasan (2013) used canny edge, horizontal and Vertical edge

methods to detect LP regions with three stages Artificial Neural Network (ANN) to extracted features from LP regions. The accuracy rate for LPD was 92.7%. Panahi and Gholampour (2017) used vertical sobel edge operator and hough transform to identify LP area, and connected component algorithm (CCA) with two level SVM to extract LP features. The accuracy rate for LPD was 97%. Wafy and Madbouly (2016) used semi-symmetric corner points, morphological feature, and linear discriminated analysis (LDA) methods to extracted features from LP area. The accuracy rate for LPD was 98%.

There was no attempt to use the ensemble of AdaBoost cascades with multi levels pre-processing features extraction method, such as LBPs, Haar-like, and histogram of oriented gradients (HOG) classifiers to detect many difficult features of LP problems. The main contributions of this study, that many researchers used one level strong cascade classifier (sliding window) to detect ROI features for specific problems of the LP. This work uses multiple cascades classifiers with multi levels pre-processing stage to detect different features values under difficult conditions, such as low/high contrast, dusk, dirt, fogy, and distortion using an AdaBoost algorithm with 3L-LBP classifier. We used the best enhancement methods for pre-processing stage, such as two dimension Gaussian filter (2D-Gaussian) with standard deviation $\sigma = 0.25$ and contrast-limited adaptive histogram equalization(CLAHE) method with standard deviation $\sigma = 1$ for training and testing images. The LBPs classifier is employed for improving the accuracy rate because it has a discriminative power for extracting the ROIs from the LP area with a low processing time. The proposed method for the LPD is adaptive for different LP styles, colors, and languages. It can be used to enhance the performance of any existing ANPR system with different datasets. The next section provides more details about the proposed LPD method.

3. The proposed method

The proposed method involves two phases: the image pre-processing phase and the LP detection phase. The first phase aims to reduce the processing time through enhancing vehicle's images processes while preparing for the detection phase. To detect the ROIs area for the LP under different conditions, a large number of AdaBoost cascades of 3L-LBP classifiers are employed. The number of cascades classifiers depends on the LP variation problems. Each cascade classifier has different features values in order to solve one LP problem. The number of strong classifiers is obtained from the training phase. The proposed framework of the LPD system for training and testing phases is shown in Figs. 2(a) and 2(b).

3.1. Image pre-processing phase

This study uses a grayscale image by converting a color input image (24 bit) into a grayscale image (8 bit) as follows:

$$G(I,J) = 0.3 \times R(I,J) + 0.59 \times G(I,J) + 0.11 \times B(I,J) \quad (1)$$

where I and J are any pixel inside a grayscale image, G is a grayscale image and (R, G, B) are three color channels of red, green and blue for color images.

There are different resolutions involved in vehicles images, with large resolutions often require more processing time. Therefore, the grayscale images are resized to a desired size for the output image. To reduce the computational time, the images of sizes 1280×960, 1024×768 and 640×480 as shown in Fig. 3 were resized into a 400×300 resolutions.

The histogram for the grayscale image has 256 bins by default. It is a chart which presents the distribution of the grayscale image intensities. The (imhist) function in Matlab produces a histogram plot by defining (*X*) equally spaced bins, each bin representing a range of features values. After that, calculating the number of pixels (*Y*) within each range which represents the features

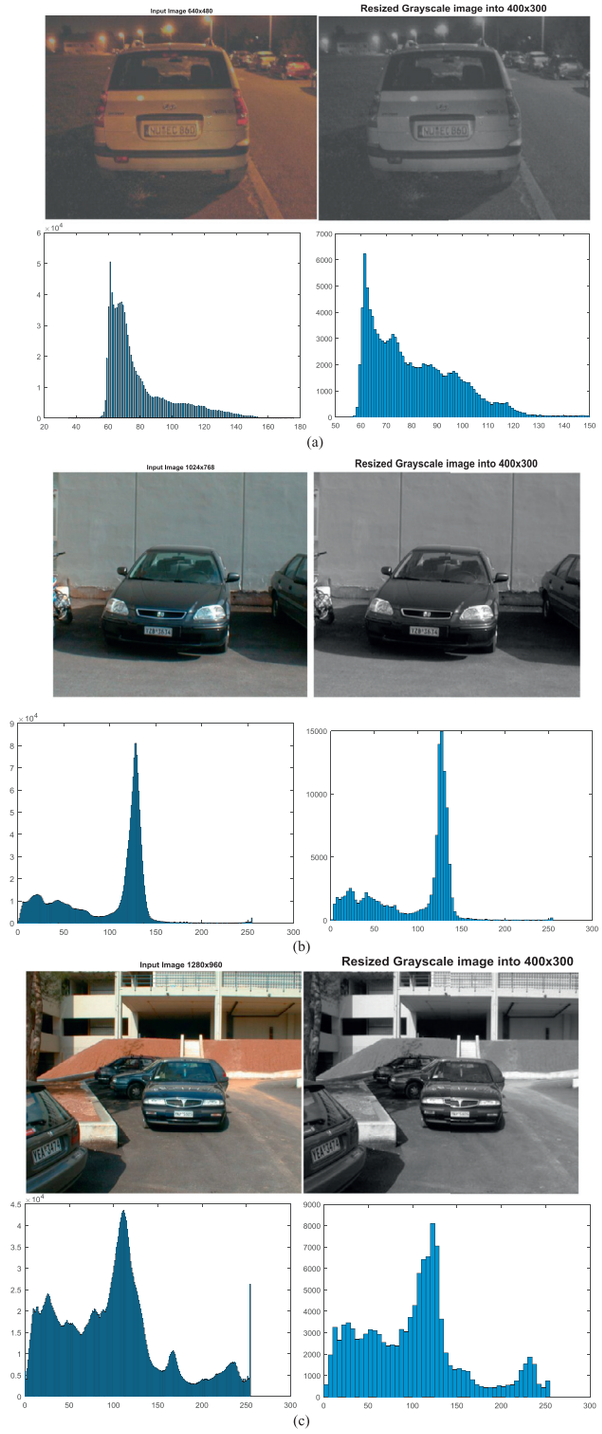


Fig. 3. Resized the different resolutions of original vehicles images with histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins). (a) Resized vehicle image from 640×480 into 400×300 resolution; (b) Resized vehicle image from 1024×768 into 400×300 resolutions; (c) Resized vehicle image from 1280×960 into 400×300 resolutions.

values appearance in each range of bins. Moreover, the information in a histogram can be used to choose an appropriate enhancement method to reduce the range of intensity features values. The proposed method uses texture features instead of color features to detect the LPs because the color features are very sensitive to the illumination conditions and noise (Azad et al., 2014). The LPD

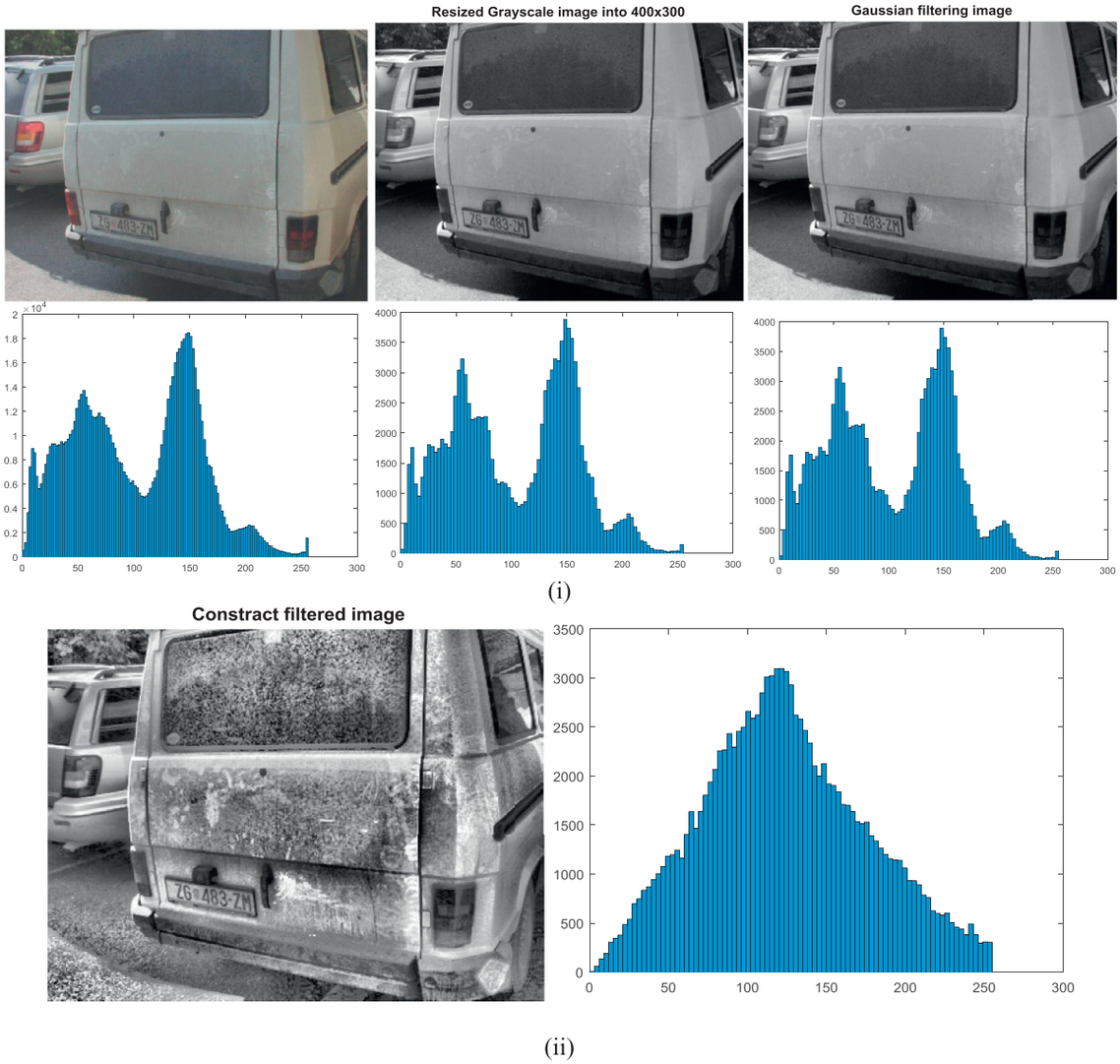


Fig. 4a. Example of vehicles images with dirt, and dust problems after applying Gaussian filter and CLAHE enhancement method to a grayscale image and their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

method employed an intensity transition on a vehicle image based on the 3L-LBP extracted features as initial localization for the LP. The intensity in the LP region is very high because various texts are included in the vehicle images. There are also various noises with the texts referring to non-LP regions, such as surface textures, dust, distortion, and small dirt (Fig. 4a, 4b). The noise conditions increase the unnecessary features intensities in an image. For denoising, a two dimension Gaussian filter with standard deviation $\sigma = 0.25$ (Lalimi, Ghofrani, & McLernon, 2013; Nixon & Aguado, 2012; Parker, 2010; Yogamangalam & Karthikeyan, 2013) has been used for this purpose. The Gaussian filter is defined as:

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

Some vehicles images are suffered from low illumination and weather conditions, such as low/high contrast (indoor/ night-light conditions) (Fig. 4c, 4f) and fogy (Fig. 4d, 4e, 4f). A contrast-limited adaptive histogram equalization (CLAHE) method with some improvement and standard deviation $\sigma = 1$ is applied on the grayscale filtered image to enhance the contrast condition. It is a common image enhancement method and widely used for image processing (Azam & Islam, 2016; Kaur & Kaur, 2014; Moustafa & Jaradat, 2015). The same pre-processing techniques are applied to positive and negative samples for the LPs at the training phase.

The enhancement steps of resizing, filtering, and contrasting images help in reducing changing day light effect and improving the contrast between the original image and enhanced image (Asif et al., 2014). The pre-processing stage has clearly effects to reduce the unnecessary features from images, so the histograms demonstrate their effects in each step (see Figs 0.4 steps (i, ii)). There a wide different between step (i) and step (ii) related to a high reduction on features bins range and space. The next section presents more details about the LPD method.

3.2. The LP detection method

3.2.1. 3L-LBP detectors for features extraction

There is no one detection method works for all types of images problems by using unsupervised learning algorithms (Silapachote et al., 2005). Therefore, the features extraction methods should be selected carefully to detect different types of problems. Some image problems, such as scale, rotation, and contrast should be considered for the detectors because each detector is

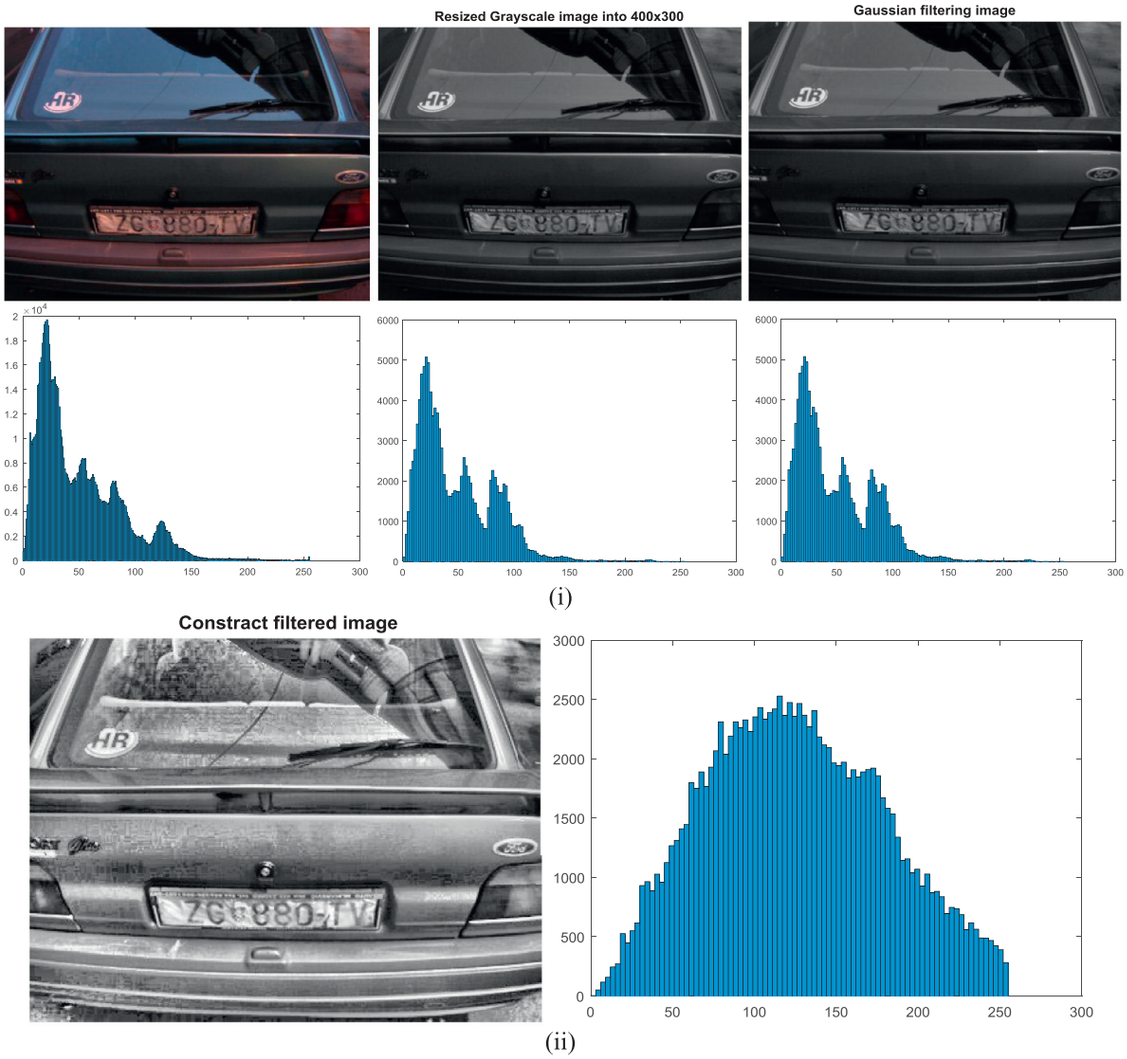


Fig. 4b. Example of vehicles images with distortion and dusk or low light problems after applying Gaussian filter and CLAHE enhancement method to a grayscale image and their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

designed for a specific kind of features. The proposed method uses supervised learning algorithms to employ a three levels of LBPs operator to extract different features for several reasons (Cvetković, Rajković, & Nikolić, 2016; Hussein, Porikli, & Davis, 2009; Krig, 2014; Lee et al., 2013; Li & Shen, 2016; Nguyen & Nguyen, 2012; Shan & Gritti, 2008). The first level is extracted LBPs features from LP grayscale image, the second one extracted LBPs features from filtered LP image, and the third level extracted LBPs features from contrasted LP image (see Fig. 5). Moreover, when the levels of LPs images changed in pre-processing stage, the significant features for different problems can be captured. Therefore, we utilize a maximum pooling strategy that selects the maximum values from the corresponding bins of 3L-LBPs histogram features concatenation at different scales of LP image. The AdaBoost algorithm compares a test vehicle image features with trained models features for LP and labels the LP test image features using the class that the training image with the highest similarity belongs to.

The LBP is an effective operator for various illumination conditions and can solve the occlusion and scale invariance problems. Ojala, Pietikäinen, and Harwood (1996) presented the first LBP operator by dividing an image into cells or regions and labeling the pixels for each region using a 3×3 thresholding neighborhood. The method involves comparing the center value with 8 neighbors and transferring the results to a binary number with weighted values as shown in Fig. 6. The output of the LBP operator can be presented in decimal form:

$$BP(x_{center}, y_{center}) = \sum_{n=0}^{7} s(i_n - i_{center}) 2^n \tag{4}$$

where n is the number of neighbors for the central pixel value *(center)*, i_{center}, i_n is the gray pixel value for *center* and the surrounding pixel values, and $s(x)$ is equal to 1 if $x \geq 0$, and 0 otherwise.

$$s(x) = \begin{cases} 1 \; x \geq 0 \\ 0 \; x < 0 \end{cases} \tag{5}$$

The pattern 11110001 is called uniform-patterns because it contains more than two transitions with a single label of a LBP operator, which are produced much less uniform-patterns without losing too much information. The most of the texture information are contained in the uniform-patterns, which are caused by the local primitives like corners and edges from the images. After labeling

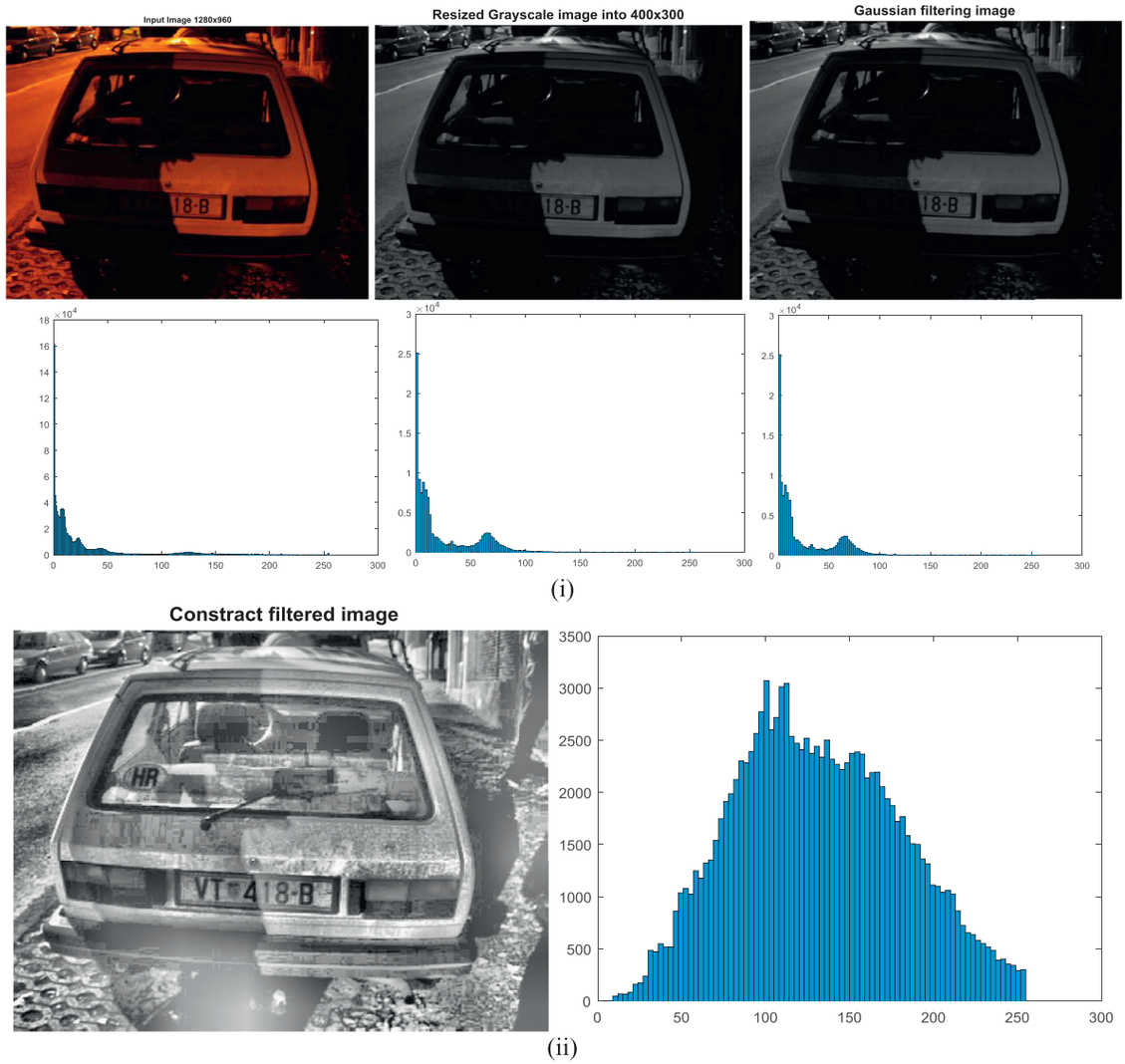


Fig. 4c. Example of vehicles images with night-light contrast problem after applying Gaussian filter and CLAHE enhancement method to a grayscale image with their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

each region image with the LBP operator, the histograms of the labeled image regions $R_1, R_2, \ldots, R_M$ can be presented as:

$$H_{i,j} = \sum_{x,y} I(f_i(x,y) = i) I((x,y) \epsilon\ Rj) \quad i = 0, \ldots,\ L-1,\ \ j = 0, \ldots, M-1 \tag{6}$$

where L is the number of different labels that are produced by the LBP operators, and M is the number of the LP regions.

This study uses two types of LP training samples, with 50×260 and 115×80 resolutions and being resized into 35×160 and 70×48, respectively which is the same size of LP testing images after resized it. The LPs training images are subdivided into 16×5 and 21×12 pixels to produce 80 and 252 cells, respectively, as shown in Fig. 7. The size of each cell is 3×3 pixels and the number of bins is 59. Therefore, the cascades classifiers have different LBP features range, but with the same number of the LBPs features. In this work, the training LP features are generated from 5 and 10 stages, respectively.

The quantized LBP weak classifier for every region in each level is aggregated into maximum pooling features histogram. The extracted features histograms perform as inputs to AdaBoost algorithm to produce the final LP detector. The LBP operator already removes the non-ROI or any noise from the training images (Jia et al., 2007). This is useful to reduce unnecessary features and the processing time for the training phase.

3.2.2. AdaBoost learning algorithm for the LP detection

Freund and Shapire in 1995 proposed an adaptive boosting (AdaBoost) algorithm to solve the difficulties in the Boosting method. The algorithm generates a strong classifier from a large number of weak learner classifiers to make the performance of a detection system better than randomly guessing (Freund & Schapire, 1995; Freund, Schapire, & Abe, 1999). The AdaBoost successfully provided good and accurate results in computer vision (Lienhart & Maydt, 2002). A weak learner or classifier that has the lowest error selects in each iteration stage. Then, the weak classifier works to increase the rate of the weighted error for misclassified training samples and decrease the rate of the weighted error for the classified training samples. This study uses the AdaBoost to train a large number of weak classifiers, and each weak classifier is designed to select a single LBP features histogram for best separating the positive and negative LPs samples. The basic AdaBoost learning algorithm works as follows:

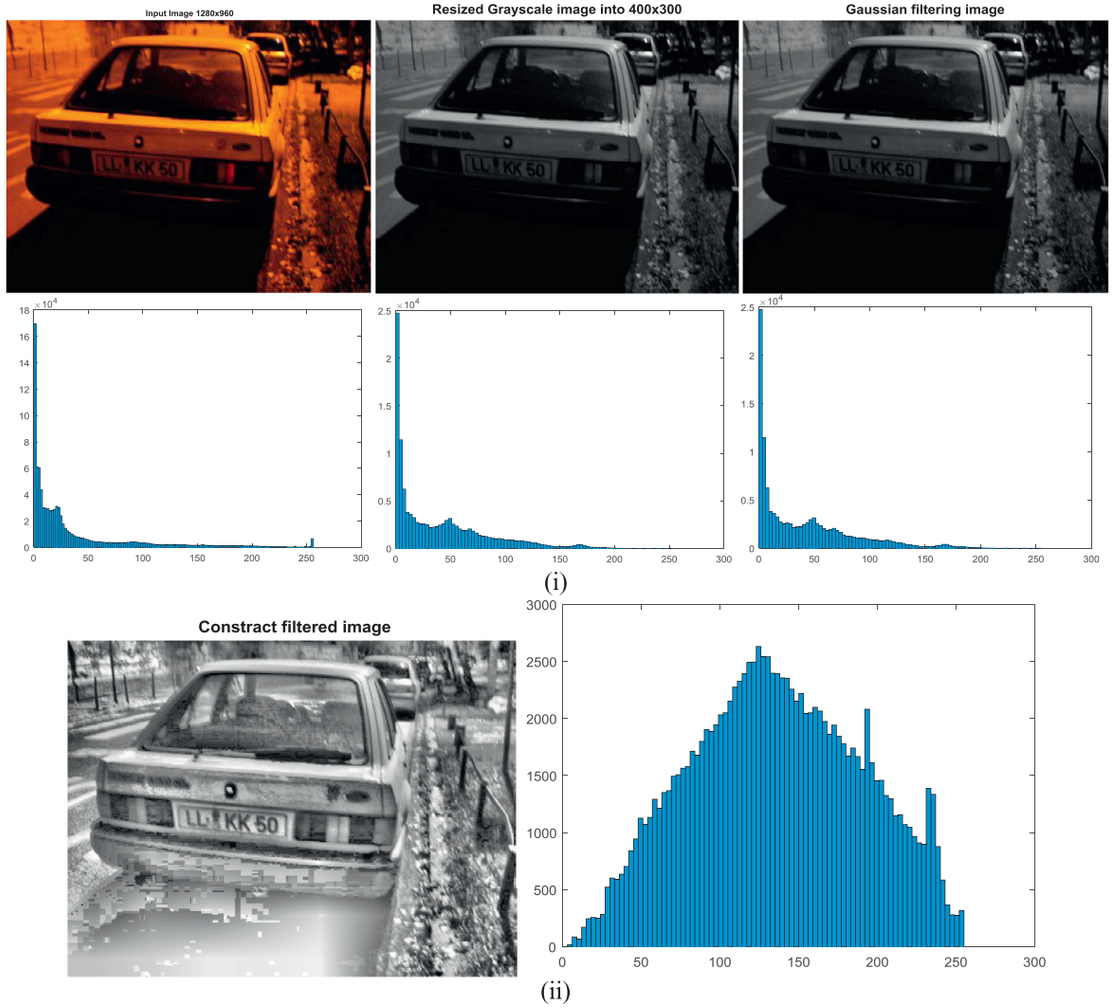


Fig. 4d. Example of vehicles images with dusk or low light problem after applying the Gaussian filter and CLAHE enhancement method to a grayscale image with their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

Provided the training dataset of $(X_1,Y_1),\ldots,(X_n,Y_n)$, where $y_i = 0$ for negative examples *(X)*, 1 for positive samples *(X)*; n the number of training samples, and i the number of the output for training samples, and given weights of $W_{1,i} = \frac{1}{2m} + \frac{1}{2l}$ for $y_i = 0$, 1 negative and positive samples, respectively, where m is the number of the negative samples and l is the number of positive samples. For each iteration t normalizes the samples weights, $W_{t,i} = \frac{W_{t,i}}{\sum_{j=1}^{n} W_{t,j}}$, where w_t is a probability distribution of samples. The trained classifier (h_j) for each feature *(j)* is limited to use a single feature. To evaluate the weighted error for each feature *(j)*,

$$W_t,\ \epsilon_j = \sum_i W_i \left| h_j(x_i) - y_i \right| \tag{7}$$

Then select the classifier h_t with the lowest error ε_t. To update the feature weights,

$$W_{t,i} = W_{t,i} + 1, \tag{8}$$

where $i = w_{t,i}\ \beta_t^{1-e_i},\ \mathrm{e_i} = 0$ if sample x_i is correctly classified, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Finally, to produce a final strong classifier *(Y(x))*:

$$Y(x) = \begin{cases} 1 \\ 0 \end{cases} \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \tag{9}$$

where $\alpha_t = log\frac{1}{\beta_t}$.

The learning algorithm works to re-train the samples with new weights, instead of re-sampling them. This means to start with regular weights on the training examples for *(T)* rounds or iterations, and to evaluate the weighted error for each LBP feature and pick the best one. After that, we update the feature weights by giving more weights for the LBP feature that is incorrectly classified and less weight for others. In the final step, this algorithm produces one strong cascade classifier for each level pre-processing LP which consists of the combination of the weak weighted classifiers with the weights depending on the error they have. The final ensemble of strong cascades classifiers for 3L-LBPs is used as trained models for the LP detection. Each strong classifier has a number of weak classifiers, which represents the LBPs features. The final step of the proposed algorithm can be represented as follows:

$$Y_i(x) = \begin{cases} 1 \\ 0 \end{cases} \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \tag{10}$$

where $i = 1\ldots..n$ is the number of strong cascades classifiers that are associated to the LP problems and each problem is related to one LBP histogram features.

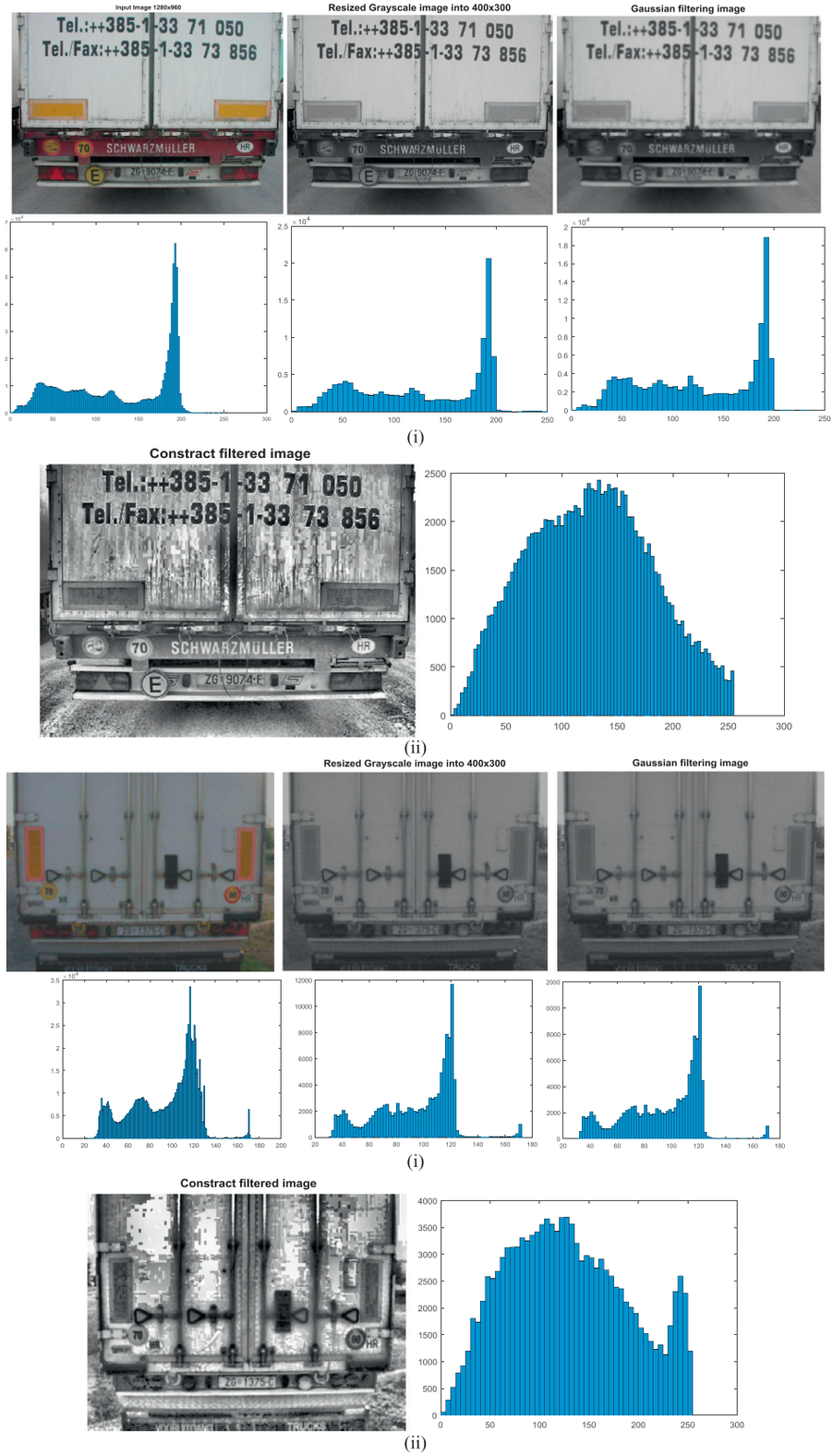


Fig. 4e. Example of vehicles images with dusk or low light and fog problems after applying the Gaussian filter and CLAHE enhancement method to a grayscale image with their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

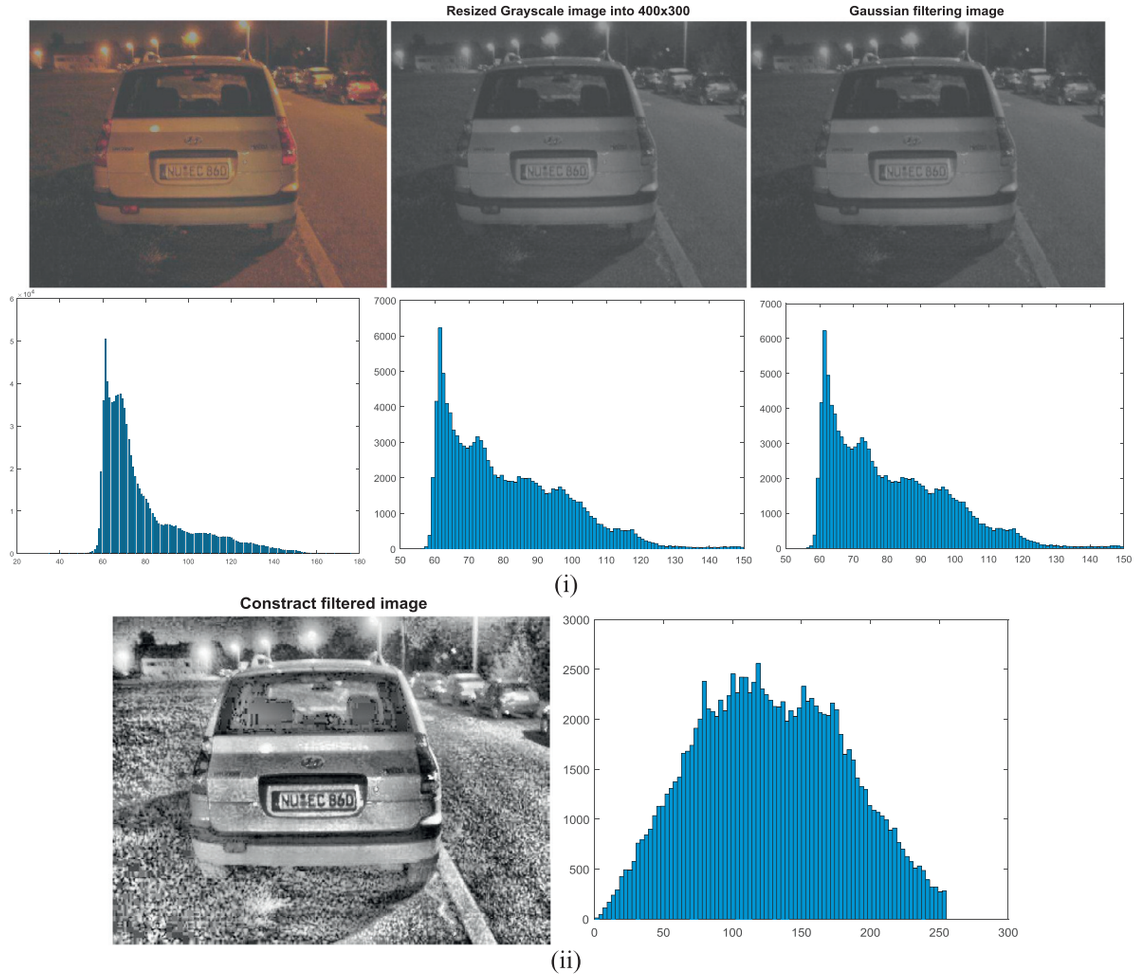


Fig. 4f. Example of vehicle image with low light and fog problems after applying the Gaussian filter and CLAHE enhancement method to a grayscale image with their histograms (For histogram: X axis = the range of features values in each bin, Y axis = no. of features values appearance in each range of bins).

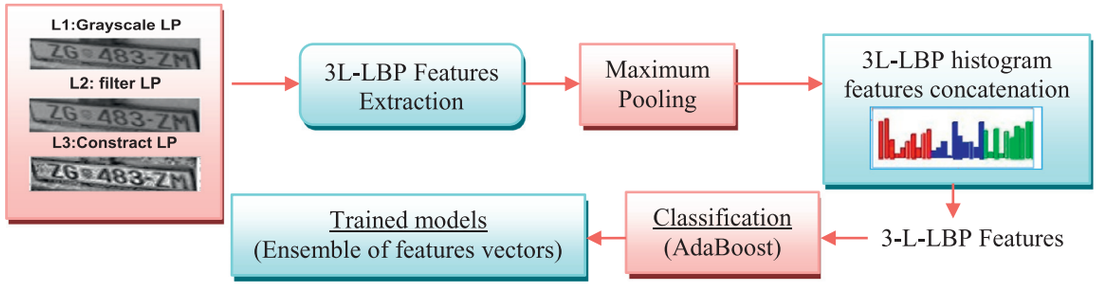


Fig. 5. The framework of 3L-LBP extraction features from LP vehicle image.

3.2.3. Cascade structure of boosted classifiers

A cascade structure is mainly developed for having high testing detection speed. The cascade classifier consists of a sequence of ensemble weak classifiers with binary nodes of features H_1,H_2,..., H_n, which are trained using a learning algorithm such as AdaBoost algorithm (Wu & Rehg, 2012). As a result of the rarity of interest objects, the most background areas in the tested image are filtered out using a cascade classifier. Only the objects of interest, which are found in a few areas of the tested image, can be detected quickly (Nguyen & Nguyen, 2012; Wu & Rehg, 2012). In the proposed method the training cascades classifiers with the 3L-LBPs filter out different types of scenes where they are identified definitely as non-LP areas, such as road surface, vehicle surface or highly textural areas. For example, the cascade classifiers for terrain can quickly reject a terrain area. The cascade classifiers for a vehicle surface can quickly reject a surface area. The hard and final complicated ensemble for the textural regions is used only for the remaining regions. For difficult types of vehicle images, this method is successful for detecting the LP regions using a large number of cascades classifiers with the strong LBP classifiers, and filtering out non-LPs regions in order to reduce the false positive rate. Let us assume the errors produced by the weak classifiers are

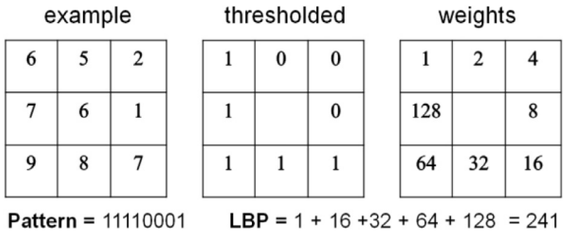


Fig. 6. Example of the extraction process for the LBP operator.

independent of each other for the image conditions. The final cascade has 20 weak classifiers and those classifiers have a high detection rate for the LP $d_i = 99.9\%$ and the false positive rate $f_i = 50\%$ for all i. The cascade classifier is capable of detecting $\prod_{i=1}^{20} d_i = 98\%$ for the LP with a false positive rate of $\prod_{i=1}^{20} f_i = 10^{-6}$. The algorithm for the cascade classifier structure is shown in Fig. 8.

The negative training samples are used to limit the number for training the weak classifiers. When those classifiers are trained with T iterations, the current ensemble of the cascades are applied to a large set of the negative vehicle image areas to find out the difficult examples that are incorrectly classified by all the existing weak classifiers in each cascade. For each round or iteration $T+1$ for training weak classifiers, these negative examples are added. The bootstrapping database for the negative examples does not need to be stored because it can be obtained by using the current cascade classifier on the images that do not have any objects of interest. The flow chart of the training strong cascades classifiers using AdaBoost algorithm is shown in Fig. 9.

In the training phase the number of the negative training samples must be larger than the positive training samples. The bootstrapping operation is implemented only on the negative training samples rather than the positive training samples. The bootstrapping process in each round is applied to the whole negative samples to randomly select the small part of the negative samples in order to obtain a small sub-window of all the samples in the training stage (Ma, Tan, & Yang, 2008). The second stage has the same number of the samples that came from the original or previous stage. In every round for the training phase each weak classifier gets a single LBP feature and adds it to the ensemble of the cascade classifiers. The stages in each cascade are built by the training classifiers using the AdaBoost learning algorithm. In our cascades structure each cascade has different stages, with each stage has different features, which is an important reason for the combination of the classifiers. Therefore, increasing the number of cascades in the training phase leads to the increase of the number of features. This means the important information do not get lost and can be used to solve difficult problems, such as deformable features (Hussein et al., 2009; Wu & Rehg, 2012). In Table 1, we explain how this work differs from (He et al., 2007; Nguyen & Nguyen, 2012; Wang et al., 2013) studies which are a very closed from this study.

4. Experimental results

4.1. Dataset

All the experiments were designed and conducted using a standard PC with 2.40 GHz Intel Core i5-4210 U and 4GB RAM, and using Matlab language version R2016a. This study uses various vehicles images captured by OLYMPUS C-2040ZOOM digital camera under different environmental conditions (cloudy weather, sunny day, night lighting, dusk). The database contains 530 vehicles images for different types of vehicles, such as trucks, passenger cars, and buses. This database is publicly available (EnglishLPDatabase) and with 640×480 resolutions. Each image contains a single license plate with the 640×480 resolution. The proposed method is also applied to another database that contains 300 images captured by an IR camera under different illumination, and each image has a single plate and/or multi-license plates with 1280×960 and 1024×768 resolutions (MedialabLPRdatabase). The online photo editor application used to increase the car English dataset through making difficult changes on the original dataset to evaluate the proposed method performance. The total number of the images in the database is 1030, which are divided into two groups with the testing phase contains 630 vehicle images as shown in Fig. 10, and the training phase contains 400 LPs images which are also used in testing phase.

In the training phase, the large numbers of the positive samples are required to capture the LP variations. For example, the training dataset contains the images of the rotated LPs in order to detect the various types of LPs rotated with different angles, such as $45°, \pm 18°$, $12°$ and $6°$. Those images can be detected using rotating LBP features detectors as shown in Fig. 11.

With the training datasets above, it is observed that the LPD system works very well because it can detect LPs in the images with a low false alarm rate. While it cannot detect LPs with a low light and dark condition, there are two ways to solve the problems. The first one is the image pre-processing stage and another one is to enhance the training datasets. To enable the LPD system to detect the LPs in low light conditions, LPs images with different illuminations are added to the training dataset. Examples of the images are shown in Fig. 12.

Fig. 7. LPs images are subdivided into 16×5 and 21×12 sub- regions.

1) *Input: a set of positive examples P, a set of initial negative examples N, and a database of bootstrapping negative examples D.*
2) *Input: the learning target Y.*
3) *i=0, H= Ø.*
4) *Repeat*
5) *i=i+1.*
6) *Classifier learning(learn Hi using P and N, and add Hi to H)*
7) *Reject correctly classified non-LP area from N.*
8) *Implement the current cascade H on D; add any false detection to N until N reaches the same size as the initial set.*
9) *Until the learning target Y is satisfied.*
10) *Output: strong cascade* ($H_1, H_2, \ldots, H$).

Fig. 8. The algorithm of a cascade classifier structure (Wu, Brubaker, Mullin, & Rehg, 2008).

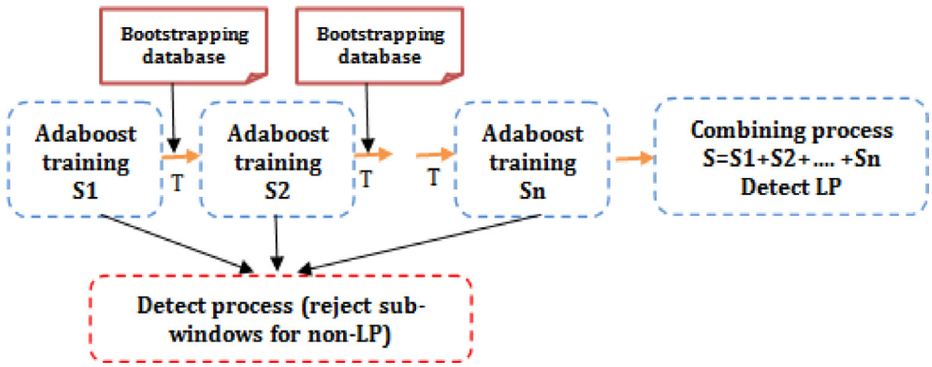


Fig. 9. The flowchart of learning cascades AdaBoost structure for the LPD.

Table 1
The differences between Nguyen and Nguyen (2012); Wang et al. (2013), He et al. (2007) methods, and the proposed method.

Ref.	Methods	Difference
Nguyen and Nguyen (2012)	Boosting + two combined features(LBP + Haar-like) to produce one strong cascade classifier)	Used the original learning algorithm which selected features based on random guessing that led to many irrelevant features selected. It caused high false positive rate and low accuracy results under good conditions, such as fixed resolutions, good illumination conditions.
Wang et al. (2013)	Adaboost + Morphology-based method + SVM classifier	Used an advance boosting method that selected features based on the best weighted feature values (best weak classifier) to reduce the dimensionality of the features space and provided good results depending on the classifier used. It took more computational time because it used the SVM classifier to extract features from the LP area and produced one strong cascade classifier. It also used the Morphology based method to reduce some false positive rate. The detection accuracy of the method was low.
He et al. (2007)	AdaBoost + global statistical + local Haar features.	Time consuming using AdaBoost with global statistical features to identify edges for the ROI features in order to reduce the space of extracted features by using local Haar features. Not consider weather and illumination conditions. -The cascade classifiers for this algorithm worked only in invariant conditions, such as colour, brightness, size, and position of LP.
Proposed	Ensemble of AdaBoost cascades of LBP classifiers.	The better techniques were used in terms of accuracy and processing time for training, extracting, and selecting features from LP vehicles images under low/high contrast, fog, rotation, and the deformation problems. The 3L-LBP is used to extract multi-level pre-processing features to detect difficult and complicated regions for LP area. The LBP was used by (He et al., 2007; Hussein et al., 2009; Krig, 2014; Li & Shen, 2016; Shan & Gritti, 2008) due to: it has a discriminative power to extract the best relevant features and not need more time for extracting. Also this classifier can reduce the false positive rate without using any other methods. We selected the advance version of boosting learning named AdaBoost for training and selection features and obtained good accurate results (Freund & Schapire, 1995; Freund et al., 1999).

The training dataset for the negative samples contains 600 non-LPs images that were collected from vehicles images with the background having road, building, tree, and ground as shown in Fig. 13.

In this study, we made the number of negative samples more than the number of positive samples in order to reduce the error rate for the LP detection stage. The training images were processed to the same size of the LPs patterns for the testing images.

4.2. Detection results

Some results by the proposed method based on the vehicles images in the dataset are shown in Fig. 14. It can be noticed that all the LPs were detected although there were bad features appeared with different LP variations conditions. The detailed information for the LPs on all the datasets with detection or recall rate are shown in Table 2. In Table 2 the low detection results are ap-

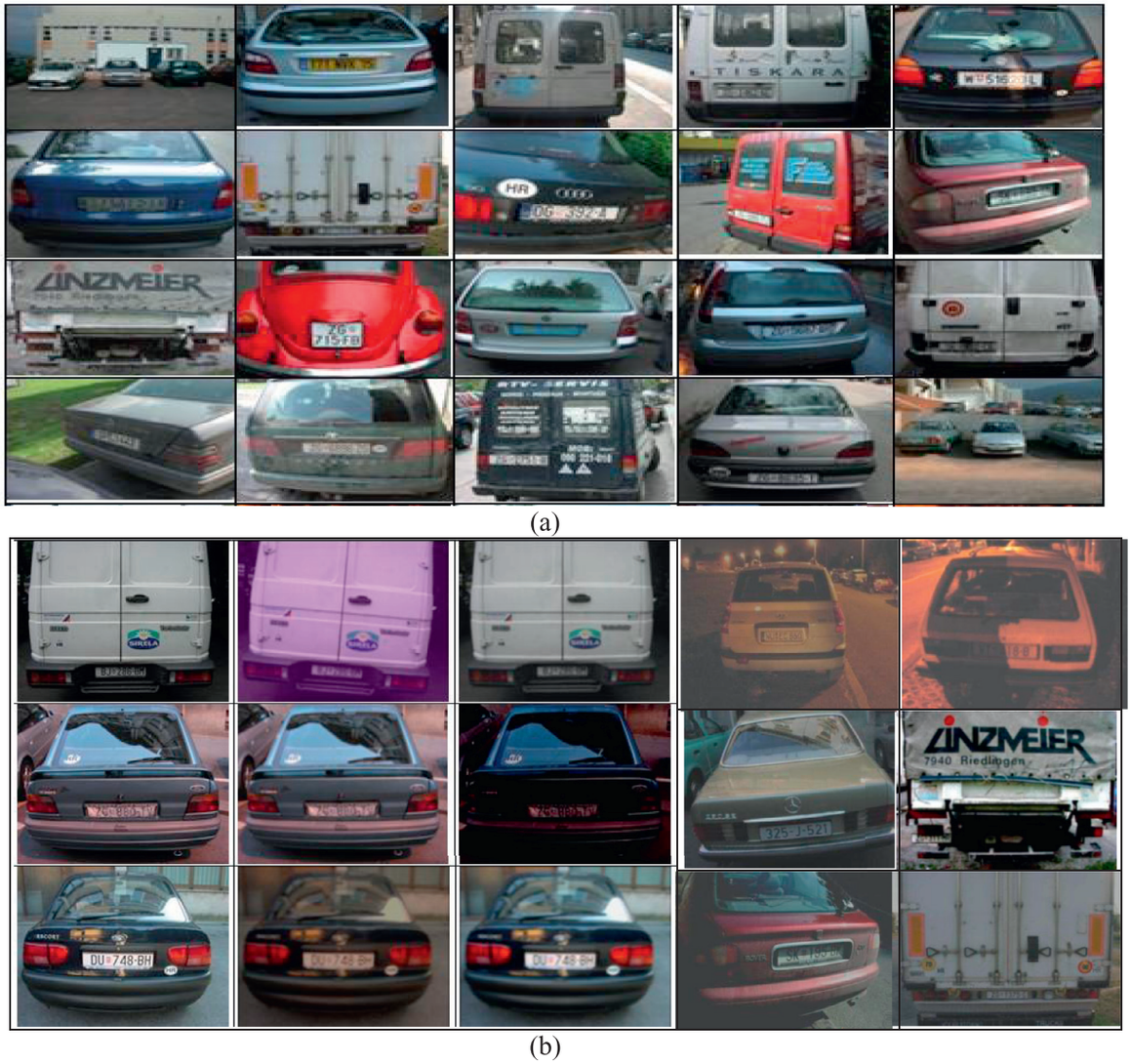


Fig. 10. Some examples from the testing vehicles images in the dataset. (a) Some vehicles images from original dataset. (b) Some vehicles images with difficult changes using online photo editor application.

Fig. 11. Some examples of LPs images from the training dataset with rotation.

peared at the dusk time compared with the daytime or night-time. The reason is that the lighting conditions is very poor during the dusk time; therefore, making LP hard to identify.

Moreover, some false positive (FP) are noticed when several objects in the vehicle images as shown in Fig. 15, look like a LP (such as commercial signs and vehicle logo), are detected with a low trust value.

4.3. Processing time

The detection rate also can be referred as the accuracy rate of the proposed method is 98.53%. The processing time is an important indicator of system performance. The proposed 3L-LBP classifier does not need more processing time and achieves a very good accuracy compared with other existing classifiers. Therefore, the average of processing time per one vehicle image for the whole system stages pre-processing, extraction and detection was 1.82 s which is between 1.63 and 2 seconds as shown in Table 2 and Fig. 16. In Table 2, the complicated vehicles images for dusk time, fogy daytime, and distortion need more processing time to detect the difficult LP features. Some vehicles images have high false positive rate 7% at sunny daytime because they have more logos and commercial signs on them which have clearly vision at this time on vehicle mage. However, the proposed method can solve some of the errors in the pre-processing phase. Form Table 4 the reported time is much better compared with other existing method to solve many difficult problems. The efficiency and the accuracy of

Fig. 12. Some examples of LPs images with different illumination conditions are added to the training datasets.

Fig. 13. Some examples of the negative images in the training dataset.

Table 2
The description of dataset and the performance results.

Condition	No. of vehicle image	True LP	False LP	False Positive Rate	Detection Rate	Average Processing Time(s)
Dusk time	226	210	16	4.5%	97.6%	2.00
night-time(low/high contrast)	135	130	5	4.3%	98.7%	1.63
Sunny daytime	294	294	0	7%	100%	1.65
Fogy daytime	210	202	8	6.5%	98.8%	1.85
Distortion	165	156	9	5.6%	98.2%	1.98
Total	**1030**	**992**	**38**	**5.6%**	**98.56%**	**1.82**

the proposed method make the LP detection reliable and possible for large-scale or real-time applications.

4.4. The performance evaluation

This study follows the popular evaluation measurements for objects detection in natural scene, such as precision, recall and F-measure, and the processing time (Bashir & Porikli, 2006; Kaur & Kaur, 2014; Li & Shen, 2016). These measurements consider the true positive (TP) detection rate from the number of the positives as the ground truth that is related to the number of the FP rate (Jia et al., 2007). Those measurements can be defined as follows:

$$Positive\ prediction\ or\ Precision\ rate(PR) = \frac{TP}{TP + FP} \tag{11}$$

$$Detection\ or\ Recall\ rate(RR) = \frac{TP}{TP + FN} \tag{12}$$

$$F - measure(Fm) = 2 * \left(\frac{RR * PR}{RR + PR}\right) \tag{13}$$

where *(Fm)* is the trade-off harmonic mean between the recall and precision rates; *FN* is a false negative rate which means the number of images in the ground truth at least has one object, but the system confirms that there are no objects inside the images.

The FP rate is 5.6% using the ensemble of AdaBoost cascades with 3L-LBP classifiers and 7.5% using one AdaBoost cascade of

Table 3
Comparisons of the LPD results by different methods, the proposed cascades LBPs classifiers produced the best detection rate, both with the highest recall and f-measure rates.

Ref.	FP	Precision (%)	Recall (%)	F-measure (%)
Azam and Islam (2016)	6.3%	NR	98.15%	NR
Ho et al. (2009)	NR	90.18%	92.07%	91.10%
Zhou et al. (2012)	4.5%	95.50%	84.80%	89.83%
Lim and Tay (2010)	≈13%	83.73%	90.47%	86.97%
Wang et al. (2013)	NR	81.68%	88.28%	84.84%
He et al. (2014)	19.6%	92.7%	94.7%	93.68%
Asif et al. (2016)	6.5%	87.15	93.86%	90.38%
Proposed (1) One cascades with LBP classifiers	**7.5%**	**93.6%**	**89.3%**	**91.4%**
Proposed (2) Ensemble of cascades with 3L-LBP classifiers	**5.6%**	**95.9%**	**98.56%**	**97.19%**

NR: Not Reported.

LBP classifiers with a detection rates of 98.56% and 89.3%, respectively. The comparison results of the performance evaluation of this work with some reported methods in the related work are shown in Table 3. It achieved a recall rate of 98.56%, which are 4.67% higher than M. R. Asif, Chun, Hussain, and Fareed (2016) method and 0.38% than Azam and Islam (2016) method. The F-measure of the proposed method is 97.19%, which is also the best, with 6.42% higher than the method by Asif et al method. The FP rate is a little bit than Asif et al. which is 0.7%. Also, the proposed method achieved 4.1% and 3.12% for the recall rate and F-measure, respectively, higher than He et al. (2014) method with 19.6% FP

Fig. 14. Successful vehicles images detection results using our proposed LPD method with different views point (a, h, k, l), dirt and low light (f, h, i), fog and dusk (a, g, e, i, l), low/high contrast (b, f, j, m) and distortion(b, c, f, j) problems.

rate which is very high 14% compared with the proposed method. Wang et al. (2013) achieved an 88.28% and 84.84% recall rate and F-measure, respectively, which is less than the proposed method with 10.52% and 12.04% for recall rate and F-measure, respectively. Moreover, the proposed method achieved 6.7%, which is higher than Ho et al. (2009) method for recall rate and 5.7% for F-measure rate. Lim and Tay (2010) achieved a 90.47% and 86.97% recall rate and F-measure, respectively, which is less than the proposed method with 8.09% and 10.22% for recall rate and F-measure, respectively, with 7.4% FP rate. Zhou et al. (2012) achieved 84.80% and 89.83% recall rate and F-measure, respectively, which is less than the proposed method with 13.76% and 7.36% for recall rate and F-measure, respectively, with FP rate 4.5% which is only 1.1% higher than proposed method under complicated vehicles images. Finally, the proposed method gained 95.9%, 98.56%, and 97.19%, respectively, for precision, detection, and F-measure rates using the ensemble of AdaBoost cascades of 3L-LBPs classifiers. While the results of one AdaBoost cascade yields 93.6%, 89.3% and 91.4%, respectively. Based on the performance evaluation measurements which are described above, the proposed method outperforms all the existing methods in terms of recall rate, F-measure, and FP rate for LP detection.

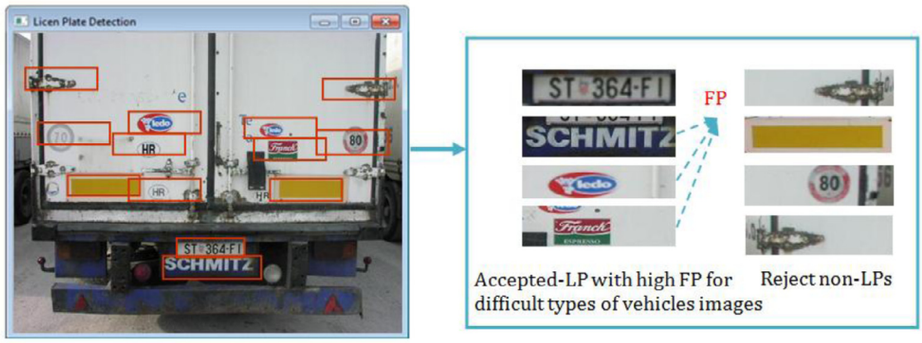


Fig. 15. Examples of detect LP with false positive value.

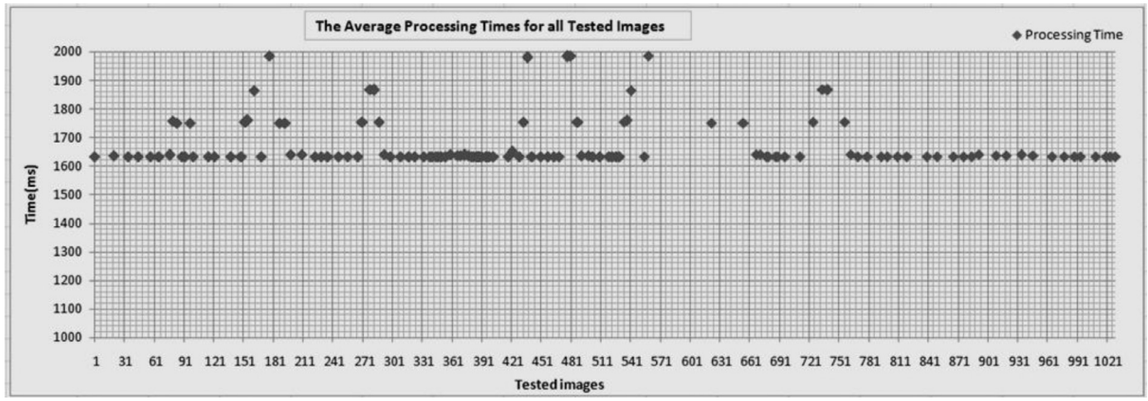


Fig. 16. Average of the processing times for all the tested images.

4.5. Comparison with existing methods

The proposed method is compared using the same dataset that was used by (Azam & Islam, 2016; Hasan, 2013; Panahi & Gholampour, 2017; Wafy & Madbouly, 2016) studies and other different datasets in (Asif et al., 2016; He et al., 2014; Ho et al., 2009; Lee et al., 2013; Lim & Tay, 2010; Zhou et al., 2012) under less difficult conditions. The presented LPD method by Azam and Islam (2016) shows good performance over few of the previous state-of-the-art techniques as mentioned in the literature. Also, the detection time 0.45 s which is less than the proposed method 0.33 s with a little bit detection rate 0.41%. But, it only considers easy tilted LP under good conditions and low difficult contrast night conditions using many unsupervised learning methods with 325 English car image database using MATLAB 7.12. Hasan (2013) method has detection rate 92.7% which is less than the proposed method 5.86% under good conditions also by using many unsupervised learning methods for detection stage with 69 English car images using MATLAB 7.1. It does not consider the tilted LP, noise and low/high contrast image environment. Panahi and Gholampour (2017) also show good performance method, the detection rate is a little bit less 1.56% than the proposed method with the same hardware platform 2.40 GHz Intel Core i5 and 4GB RAM. It only considers medium quality plates not solve the tilted LP, noise, fogy, and high contrast image environment. They used many unsupervised learning methods with 500 English car image dataset using C++ without reported the detection time. The detection method presented by Wafy and Madbouly (2016) shows good detection rate 98% which is a little bit less than the proposed method 0.56% and 0.22 s detection time. They only consider easy and good conditions using unsupervised time consuming methods with 405 English car database using Open CV&C++, the detection time not reported. However, the dataset used in this study was more complicated compared with existing studies. So, we compared it with other different datasets for existing methods and the experimental results are summarized in Table 4. The reason for selecting different datasets, such as Malaysia, Chinese, Korea, and Caltech cars LPs to see the main difference between the proposed method and those existing methods in term of performance because of our algorithm targets to detect LP from a large amount of very distorted and complicated images. The detection time on Ho et al. (2009), Zhou et al. (2012), and Lee et al. (2010) methods is much higher than the others and our method due to applying the time consuming methods to detect and extract LP regions as well as the detection rate. The detection rate of the proposed LPD method is 98.56% with 1030 good and complicated vehicles images under difficult conditions, and the average running time for the whole LPD system is only 2 s. The memory complexity of those methods is O ($N \times M$); where N and M is the dimension of the input tested image. The average of memory usage for the proposed LPD is not a big implementation issue nowadays compared with existing methods. From the Table 4, we can see that the performance of the proposed LPD method is better than the others methods. In the large scale and the real-time implementation, this proposed method will show a good performance with high performed hardware platform involved, for example, high quality capturing device, high processing device, and faster network connection.

In summary, the proposed LPD method successfully detects LP area in the difficult conditions, and performs better than the four existing LPD methods with the same datasets.

Table 4
Comparison results between the proposed methods with other existing methods.

Ref.	Methods	LP Format	Dataset size	Platform	Detection rate	Detection time(s)
Azam and Islam (2016)	Frequency domain mask, contrast improvement technique, statistical binarization, Radon transform, and entropy vector.	English car	325	MATLAB 7.12, Intel Core 2 Duo CPU T6600, 2.2 GHz, 2 GB RAM	98.15%	0.450s
Wafy and Madbouly (2016)	semi-symmetric corner points, morphological feature, linear discriminated analysis (LDA)	English car	405	Intel Pentium 4, 3 GHz CPU, 2 MB cache, 2 GB RAM, Open CV&C++	98%	1s
Panahi and Gholampour (2017)	vertical Sobel edge operator and Hough transform, ConnectedComponent Algorithm, 2L-SVM	English car	500	Intel core i5 2.2 GHzCPU, 4 GB RAM, C++	97%	NR
Hasan (2013)	Canny edge, Horizontal and Vertical edge, three stages Artificial Neural Network (ANN)	English car	69	MATLAB 7.1	92.7%	NR
Ho et al. (2009)	AdaBoost + (SIFT + SVM)	Malaysia car	79	NR	92.07%	5s
Zhou et al. (2012)	Principal Visual Word (PVW)	Chinese car	410	MATLAB, 4GB RAM, 2.53-GHz	84.8%	D(1.06 s) + E(6.13 s)
Lim and Tay (2010)	MSER + Heuristic + CVM	Caltech Car	126	OpenCV	90.47%	NR
He et al. (2014)	Blob for candidate detection, filtering affine distortion, saliency detection, post-processing	Chinese car	200	NR	94.7%	NR
Asif et al. (2016)	YDbDr color space + Otsu method	Chinese car	1511(300R)	MATLAB 2013b,a Pentium® Dual-Core, 3.06 GHz, 2 GB RAM	93.86%	0.33s
Lee et al. (2013)	local structure patterns + MCT + color based method + position based method	Korea car	Video	MATLAB	88.9%	3.293s
Proposed(1)	One cascade with LBP classifiers	English car	1030	2.40 GHz Intel Core i5-4210 U and 4GB RAM	89.3%	D(1.325 s) + E(2.43 s) + P(0.10 s)
Proposed(2)	Ensemble of cascades with 3L-LBP classifiers	English car	1030	2.40 GHz Intel Core i5-4210 U and 4GB RAM	98.56%	D(0.78 s) + E(1.12 s) + P(0.10 s)

E: Extraction time, D: detection time, P: pre-processing time, NR: Not Reported, R: rejected LP.

5. Conclusion

In this paper, we proposed a new LPD which includes two phases. At each phase, we used better approaches capable of handling different difficult conditions. The aims of this study to learn the ensemble of cascades for 3L-LBPs classifiers due to its discriminative power using AdaBoost learning algorithm. The strong cascade classifier contains a large number of weak classifiers with different types of 3L-LBPs features values to detect different LP regions. The proposed method is implemented and tested on 1030 vehicles images having different difficult conditions, such as low/high contrast, fogy, tilted LP, and distortion. The overall performance evaluation for detection, precision and F-measure rates are 98.56%, 95.9%, and 97.19%, respectively, with FP rate 5.6%. We also compared the experimental results against existing LPD methods which are presented in the literature. We find that the proposed LPD method outperforms many existing LPD methods which have the same and different datasets in terms of detection probability and running time under difficult condition. The average of processing time per one vehicle image was 1.82 seconds for whole LPD system. Moreover, the proposed method works without any limitations due to using two phases testing and learning reverse other existing methods which use only testing phase with pre-processing stage under some limitations. In the training phase, we extracted LP features using three level LBPs classifiers, the first one from LP grayscale image, the second from filtered LP image, the third from enhancement LP image in order to detect all difficult features values from LP area. Due to many vehicle images in the dataset have many commercial signs and logos which are lead to increase the FP rate unwanted features values and take more processing time, in the future work, we intend to improve the proposed LPD to solve these problems and work on the whole ANPR system for real time applications through complete the recognition stage. In addition, the overall processing time of a detection system can be reduced through high speed hardware and software. Also, we can add more difficult issues like weather condition to increase the performance of the LPD method. Finally, the experimental results on selected dataset demonstrated that the proposed method has a good performance compared with other existing methods, and can be implemented efficiently for real-time applications.

Acknowledgments

The authors thank Mohammed Diykh and Dawood Sallem Hussian, and the anonymous reviewers for their valuable comments.

References

Anagnostopoulos, C.-N. E. (2014). License plate recognition: A brief tutorial. *IEEE Intelligent transportation systems magazine, 6*(1), 59–67.

Anagnostopoulos, C.-N. E., Anagnostopoulos, I. E., Psoroulas, I. D., Loumos, V., & Kayafas, E. (2008). License plate recognition from still images and video sequences: A survey. *IEEE Transactions on Intelligent transportation systems, 9*(3), 377–391.

Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E. (2006). A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent transportation systems, 7*(3), 377–392.

Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A. S., & Ferguson, D., Real-Time Pedestrian Detection with Deep Network Cascades, In *BMVC*, 2015, September, 32-1.

Asif, M. D. A., Tariq, U. U., Baig, M. N., & Ahmad, W. (2014). A novel hybrid method for text detection and extraction from news videos. *Middle-East Journal of Scientific Research, 19*(5), 716–722.

Asif, M. R., Chun, Q., Hussain, S., & Fareed, M. S. (2016). Multiple licence plate detection for Chinese vehicles in dense traffic scenarios. *IET Intelligent Transport Systems, 10*(8), 535–544.

Atiwadkar, A., Mahajan, S., Lande, T., & Patil, K. (2015). Vehicle license plate detection: A survey. *International Research Journal of Engineering and Technology (IRJET), 2*(8), 354–360.

Azad, B., & Ahmadzadeh, E. (2014). Real-time multiple license plate recognition system. *International Journal of Research in Computer Science, 4*(2), 11–17.

Azad, R., Davami, F., Jeo, J., & Shayegh, H. R. (2014). New framework based on complementary methods for efficiency and accuracy of license plate recognition system. *Paper presented at the Information and Knowledge Technology (IKT), 2014 6th Conference on*.

Azam, S., & Islam, M. M. (2016). Automatic license plate detection in hazardous condition. *Journal of Visual Communication and Image Representation, 36*, 172–186.

Baharlou, S. M., Hemayat, S., Saberkari, A., & Yaghoobi, S. (2015). Fast and adaptive license plate recognition algorithm for Persian plates. *Paper presented at the Pattern Recognition and Image Analysis (IPRIA), 2015 2nd International Conference on*.

Bashir, F., & Porikli, F. (2006). Performance evaluation of object detection and tracking systems. In *Paper presented at the Proceedings 9th IEEE International Workshop on PETS*.

Bhardwaj, D., & Mahajan, S. (2015). Review Paper on Automated Number Plate Recognition Techniques. *International Journal of Emerging Research in Management &Technology, 4*(5), 319–324.

Castello, P., Coelho, C., Del Ninno, E., Ottaviani, E., & Zanini, M. (1999). Traffic monitoring in motorways by real-time number plate recognition. In *Paper presented at the Image Analysis and Processing, 1999. Proceedings. International Conference on*.

Chakraborty, S., & Parekh, R. (2015). An Improved Template Matching Algorithm for Car License Plate Recognition. *International Journal of Computer Applications, 118*(25), 16–22.

Chen, Y.-N., Han, C.-C., Ho, G.-F., & Fan, K.-C. (2015). Facial/license plate detection using a two-level cascade classifier and a single convolutional feature map. *International Journal of Advanced Robotic Systems, 12*(183), 1–16.

Cvetković, S., Rajković, B., & Nikolić, S. V. (2016). Real-time image classification using LBP and ensembles of ELM. *Scientific Publications of the State University of Novi Pazar Series A: Applied Mathematics, Informatics and mechanics, 8*(1), 101–109.

Dehshibi, M. M., & Allahverdi, R. (2012). Persian vehicle license plate recognition using multiclass Adaboost. *International Journal of Computer and Electrical Engineering, 4*(3), 355–358.

Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013). Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology, 23*(2), 311–325.

Duan, T. D., Duc, D. A., & Du, T. L. H. (2004). Combining Hough transform and contour algorithm for detecting vehicles' license-plates. In *Paper presented at the Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*.

EnglishLPDatabase. http://www.zemris.fer.hr/projects/LicensePlates/english/baza_slika.zip *accessed July 2016*.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal–Japanese Society For Artificial Intelligence, 14*, 771–780 1612.

Freund, Y., & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. *Paper presented at the European conference on computational learning theory*.

Gao, Y., & Lee, H. J. (2015). Vehicle Make Recognition Based on Convolutional Neural Network. *Paper presented at the Information Science and Security (ICISS), 2015 2nd International Conference on*.

Han, B.-G., Lee, J. T., Lim, K.-T., & Chung, Y. (2015). Real-time license plate detection in high-resolution videos using fastest available cascade classifier and core patterns. *ETRI Journal, 37*(2), 251–261.

Hasan, M. (2003). *Real time detection and recognition of license plate in Bengali* (pp. 1–16). UC Riverside: Bourns College of Engineering, Retrieved from: http://escholarship.org/uc/item/0m27j18m.

He, T., Yao, J., Zhang, K., Hou, Y., & Han, S. (2014). Accurate multi-scale license plate localization via image saliency. *Paper presented at the Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*.

He, X., Zhang, H., Jia, W., Wu, Q., & Hintz, T. (2007). Combining global and local features for detection of license plates in a video. In *Paper presented at the Proceedings of Image and Vision Computing New Zealand*.

Ho, W. T., Lim, H. W., & Tay, Y. H. (2009). Two-stage license plate detection using gentle Adaboost and SIFT-SVM. *Paper presented at the Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*.

Hongliang, B., & Changping, L. (2004). A hybrid license plate extraction method based on edge statistics and morphology. In *Paper presented at the Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*.

Hussein, M., Porikli, F., & Davis, L. (2009). Object detection via boosted deformable features. *Paper presented at the Image Processing (ICIP), 2009 16th IEEE International Conference on*.

Jia, W., Zhang, H., & He, X. (2007). Region-based license plate detection. *Journal of Network and computer Applications, 30*(4), 1324–1333.

Kaur, S., & Kaur, S. (2014). An efficient approach for number plate extraction from vehicles image under image processing. *International Journal of Computer Science and Information Technologies, 5*(3), 2954–2959.

Krig, S. (2014). *Computer vision metrics: Survey, Taxonomy, and Analysis*. New York: Springer.

Lalimi, M. A., Ghofrani, S., & McLernon, D. (2013). A vehicle license plate detection method using region and edge based methods. *Computers & Electrical Engineering, 39*(3), 834–845.

Lee, Y., Han, D. K., & Ko, H. (2013). Reinforced adaboost learning for object detection with local pattern representations. *The Scientific World Journal, 2013*, 1–14.

Lee, Y., Song, T., Ku, B., Jeon, S., Han, D. K., & Ko, H. (2010). License plate detection using local structure patterns. *Paper presented at the Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*.

Li, H., & Shen, C. (2016). Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs. arXiv preprint arXiv: 1601.05610.

Lienhart, R., & Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Paper presented at the Image Processing. 2002. Proceedings. 2002 International Conference on*.

Lim, H. W., & Tay, Y. H. (2010). Detection of license plate characters in natural scene with MSER and SIFT unigram classifier. *Paper presented at the Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010 IEEE Conference on*.

Ma, C., Tan, T., & Yang, Q. (2008). Cascade boosting lbp feature based classifiers for face recognition. *Paper presented at the Intelligent System and Knowledge Engineering, 2008. ISKE 2008. 3rd International Conference on*.

MedialabLPRdatabase. http://www.medialab.ntua.gr/research/LPRdatabase.html *accessed July 2016*.

Moustafa, A. A., & Jaradat, M.-I. R. M. (2015). A New Approach for License Plate Detection and Localization: Between Reality and Applicability. *International Business Research, 8*(11), 13–25.

Nguyen, T.-T., & Nguyen, T. T. (2012). A real time license plate detection system based on boosting learning algorithm. *Paper presented at the Image and Signal Processing (CISP), 2012 5th International Congress on*.

Nixon, M. S., & Aguado, A. S. (2012). *Feature extraction & image processing for computer vision*. London: Academic Press.

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition, 29*(1), 51–59.

Panahi, R., & Gholampour, I. (2017). Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications. *IEEE Transactions on Intelligent Transportation Systems, 18*(4), 767–779.

Panchal, T., Patel, H., & Panchal, A. (2016). License Plate Detection Using Harris Corner and Character Segmentation by Integrated Approach from an Image. *Procedia Computer Science, 79*, 419–425.

Parker, J. R. (2010). *Algorithms for image processing and computer vision*. Hoboken, New Jersey: John Wiley & Sons.

Patel, C., Shah, D., & Patel, A. (2013). Automatic number plate recognition system (anpr): A survey. *International Journal of Computer Applications, 69*(9), 21–33.

Porikli, F., & Kocak, T. (2006). Robust license plate detection using covariance descriptor in a neural network framework. *Paper presented at the Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*.

Sarfraz, M. S., Shahzad, A., Elahi, M. A., Fraz, M., Zafar, I., & Edirisinghe, E. A. (2013). Real-time automatic license plate recognition for CCTV forensic applications. *Journal of Real-Time Image Processing, 8*(3), 285–295.

Sarker, M., Mostafa, K., Yoon, S., Lee, J., & Park, D. S. (2013). Novel License Plate Detection Method Based on Heuristic Energy. *The Journal of Korean Institute of Communications and Information Sciences, 38*(12), 1114–1125.

Sarker, M. M. K., & Song, M. K. (2014). Real-Time Vehicle License Plate Detection Based on Background Subtraction and Cascade of Boosted Classifiers. *The Journal of Korea Information and Communications Society, 39C*(10), 909–919.

Shan, C., & Gritti, T. (2008). *Learning Discriminative LBP-Histogram Bins for Facial Expression Recognition*. Paper presented at the BMVC.

Silapachote, P., Karuppiah, D. R., & Hanson, A. R. (2005). *Feature selection using adaboost for face expression recognition*. Retrieved from.

Song, M. K., & Sarker, M. M. K. (2014). Modeling and implementing two-stage AdaBoost for real-time vehicle license plate detection. *Journal of Applied Mathematics, 2014*.

Valera, M., & Velastin, S. A. (2005). Intelligent distributed surveillance systems: A review. *IEE Proceedings-Vision, Image and Signal Processing, 152*(2), 192–204.

Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International Journal of Computer Vision, 57*(2), 137–154.

Wafy, M., & Madbouly, A. M. (2016). Efficient method for vehicle license plate identification based on learning a morphological feature. *IET Intelligent Transport Systems, 10*(6), 389–395.

Wang, R., Sang, N., Wang, R., & Kuang, X. (2013). Novel License Plate Detection

Method for Complex Scenes. *Paper presented at the Image and Graphics (ICIG), 2013 Seventh International Conference on*.

Wu, J., Brubaker, S. C., Mullin, M. D., & Rehg, J. M. (2008). Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(3), 369–382.

Wu, J., & Rehg, J. M. (2012). *Object detection* ensemble machine learning (pp. 225–250). New York: Springer.

Yogamangalam, R., & Karthikeyan, B. (2013). Segmentation techniques comparison in image processing. *International Journal of Engineering and Technology (IJET), 5*(1), 307–313.

Yousef, K. M. A., Al-Tabanjah, M., Hudaib, E., & Ikrai, M. (2015). SIFT based automatic number plate recognition. *Paper presented at the Information and Communication Systems (ICICS), 2015 6th International Conference on*.

Zheng, D., Zhao, Y., & Wang, J. (2005). An efficient method of license plate location. *Pattern Recognition Letters, 26*(15), 2431–2438.

Zheng, K., Zhao, Y., Gu, J., & Hu, Q. (2012). License plate detection using haar-like features and histogram of oriented gradients. *Paper presented at the Industrial Electronics (ISIE), 2012 IEEE International Symposium on*.

Zhou, W., Li, H., Lu, Y., & Tian, Q. (2012). Principal visual word discovery for automatic license plate detection. *IEEE Transactions on Image Processing, 21*(9), 4269–4279.

Meeras Salman Al-Shemarry received her B.Sc Degree in Computer Science from Babylon University, Iraq in 2002 and M.Sc Degree in Information Technology (IT) from University Utara Malaysia (UUM) in 2010. She is a Lecturer in Computer Department, College of Science, University of Karbala, Iraq. She is currently a P.h.D Student in the Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia. Her research interests include image processing, system analysis using UML diagrams, Database management system and artificial intelligence.

Yan Li is currently an Associate Professor in Computer Sciences in the Faculty of Health, Engineering and Sciences at the University of Southern Queensland (USQ), Australia. Her research interests are in the areas of Big Data Technologies, Artificial Intelligence, Biomedical Engineering, and Signal/Image Processing.

Shahab Abdulla received his B.Sc and M.Sc Degrees from University of Technology Baghdad and PhD from USQ. He is currently a Lecturer in Language Centre, University of Southern Queensland, Australia. His research interests are in the areas of biomedical engineering, complex medical engineering, networked system, intelligent control, computer control systems, robotics and mathematics research, etc.

4

CHAPTER 4

AN EFFICIENT TEXTURE DESCRIPTOR FOR THE DETECTION OF LICENSE PLATES FROM VEHICLE IMAGES IN DIFFICULT CONDITIONS

4.1 Introduction

The content of this chapter is an exact copy of the published paper in the journal of *IEEE Transactions on Intelligent Transportation Systems* by Al-Shemarry et al., (2019) 'An Efficient Texture Descriptor for the Detection of License Plates from Vehicle Images in Difficult Conditions'.

In Chapter 3, the developed method, 3L-LBP_Adaboost, achieved a good detection accuracy with a acceptable time for detecting LPs from low-quality vehicles images. During the experiments, there are many objects that look like LPs inside vehicle images, such as commercial signals plates and logos, which increased the false positive rate (FPR) and took more processing time. But this method was a good schema to search for further performance improvements in LPD systems.

This chapter proposes an efficient extraction method, based on preprocessing methods, to improve an extended local binary pattern (ELBP) descriptor during the extraction stage. A Gaussian filter and a contrast-limited adaptive histogram equalization (CLAHE) enhancement method are used to build the enhancement texture descriptor, multi-level extended local binary pattern (MLELBP).

It extracted multiple complicated features, through an increased size of the training dataset during the preprocessing stage, by introducing some noisy data into training set using a Gaussian filter and a CLAHE method. The English car database was extended using an online

photo editor application to make different changes to the original database and to increase the number of vehicle images in the database in order to improve the accuracy of the LPD system. The extracted key features are then used as inputs to the extreme learning machine classifier (ELM) for multiclass identification under difficult conditions. This chapter discusses the performance of the MLELBP_ELM algorithm for the LP extraction and classification stages and compares it with the 3L-LBP_Adaboost algorithm. Also, the performance of this method was compared with other existing approaches reported in the literature. The MLELBP_ELM method helps to improve the LPD system through achieving a high detection rate with a fast computational speed for difficult vehicle images.

Appendix B provides a Matlab code for the proposed LPD method.

An Efficient Texture Descriptor for the Detection of License Plates From Vehicle Images in Difficult Conditions

Meeras Salman Al-Shemarry, Yan Li, and Shahab Abdulla

***Abstract*— This paper aims to identify the license plates under difficult image conditions, such as low/high contrast, foggy, distorted, and dusty conditions. This paper proposes an efficient descriptor, multi-level extended local binary pattern, for the license plates (LPs) detection system. A pre-processing Gaussian filter with contrast-limited adaptive histogram equalization enhancement method is applied with the proposed descriptor to capture all the representative features. The corresponding bins histogram features for a license plate image at each different level are calculated. The extracted features are used as the input to an extreme learning machine classifier for multiclass vehicle LPs identification. The dataset with English cars LPs is extended using an online photo editor to make changes on the original dataset to improve the accuracy of the LPs detection system. The experimental results show that the proposed method has a high detection accuracy with an extremely high computational efficiency in both training and detection processes compared to the most popular detection methods. The detection rate is 99.10% with a false positive rate of 5% under difficult images. The average training and detection time per vehicle image is 4.25 and 0.735 s, respectively.**

***Index Terms*— Extreme learning machine, local binary pattern, extended local binary pattern, license plate detection.**

I. Introduction

THE automatic license plate recognition (ALPR) is a well-known topic in the field of intelligent transportation systems (ITS). It is a surveillance technique to detect and recognize vehicle license plates (LPs) for many security and service purposes. For example, for observing and examining the roads traffic to prevent unacceptable behaviors, highway tolling systems, security systems, and parking management systems [1]–[4]. The core part of an ANPR system is to identify a vehicle LP from an image or a sequence of images in a video. One of the most important factors in a license plate detection (LPD) system is feature extraction. For easy recognizing, LPs are always designed in specific shapes and colors. It is often problematic to extract features of license plates with difficult conditions, such as lighting variations, dirt, dusk, viewpoint variations, and distortions, as shown in Fig. 1. Some descriptors like the texture features operators and local binary patterns (LBPs) are effective for various illumination conditions. They can partly solve the occlusion and scale invariance problems [5].

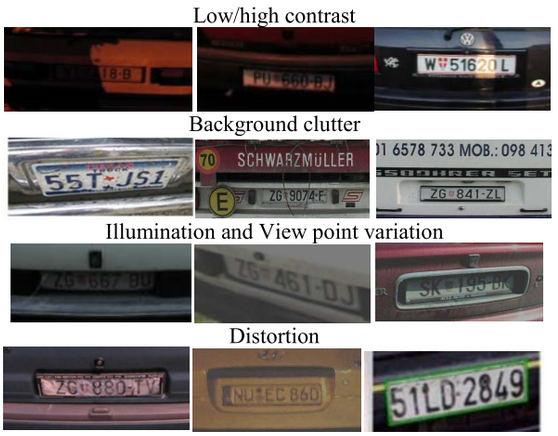


Fig. 1. Examples of the difficult conditions for license plates.

There are a large number of LBP variants proposed by researchers to improve its robustness and distinctiveness. For example, the completed local binary patterns (CLBPs) [6], extended local binary patterns (ELBPs) [7], and completed local derivative patterns (CLDPs) [8]. Those LBP descriptors are robust to gray-scale and rotation variations. To improve efficiency, Guo *et al.* [8] proposed a scale-selective local binary pattern (SSLBP) that firstly extracted scale-sensitive local features and then applied a global operator to achieve the scale-invariance. In [8], the scale-invariant feature extraction scheme achieved a good texture classification performance, but it was a high-dimensional descriptor.

Various studies revealed that ensembles of multiple classifiers [9]–[12] were able to produce better performance than a single classifier. However, those ensemble based methods faced an issue of imbalance between the number of positive and negative training samples due to underlying mechanism of binary classification. Those methods are likely to reach a local optimum or an over-fitting solution. Recently, deep neural networks (DNNs) [13] and

Manuscript received December 18, 2017; revised July 5, 2018, October 11, 2018 and January 4, 2019; accepted January 31, 2019. Date of publication February 22, 2019; date of current version February 3, 2020. This work was supported by the University of Southern Queensland and the Ministry of Higher Education and Scientific Research of Iraq. The Associate Editor for this paper was S. S. Nedevschi. *(Corresponding author: Yan Li.)*

M. S. Al-Shemarry is with the School of Agricultural, Computational and Environmental Sciences, Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia, and also with the Computer Department, Science College, Karbala University, Karbala 56001, Iraq (e-mail: meerassalmanjuwad.al-shemarry@usq.edu.au).

Y. Li is with the School of Agricultural, Computational and Environmental Sciences, Faculty of Health, Engineering and Sciences, University of Southern Queensland, Toowoomba, QLD 4350, Australia (e-mail: liyan@usq.edu.au).

S. Abdulla is with the Open Access College, University of Southern Queensland, Toowoomba, QLD 4350, Australia (e-mail: shahab.abdulla@usq.edu.au).

Digital Object Identifier 10.1109/TITS.2019.2897990

convolutional neural networks (CNNs) [14], [15] have been used as an automatic way to learn the key features in LPs. The DNN algorithms can combine the feature extraction and classification into one unified neural network framework. They have shown a higher detection accuracy. However, the features learning mechanism in DNNs cannot guarantee robustness in difficult image conditions, for example, rotation and scaling, unless the training samples can cover various observation conditions. Furthermore, their computational costs during both training and detection processes are expensive. With a high speed of vehicles, not only the accuracy but also the computational speed are the key factors for real-time applications. In this paper, a multi-level extended local binary patterns (*ML-ELBP*) descriptor uses a Gaussian filter [16] and contrast-limited adaptive histogram equalization (*CLAHE*) enhancement method [17] to extract different features from license plate images. After extracting the key features, the extreme learning machine (ELM) classifier [18] is used for multiclass identification. The proposed method can achieve a high detection accuracy with a fast computational speed. The major contributions of this work are:

1) Proposed an efficient new descriptor, the *ML-ELBP*, to extract key LP features from vehicle images with difficult conditions like lighting, foggy, rotations, blurry, darkness, and complex backgrounds.
2) Introduced two phases of an LPD system development: testing and training using *ELM* classifier as a good trainer.
3) Achieved a high detection accuracy with an extremely high computational efficiency in both training and detection processes compared to the most popular detection methods.
4) Increased the size of the training dataset during the pre-processing stage by introducing some noisy data into training set using a Gaussian filter and a *CLAHE* method.
5) Extended the English cars dataset using an online photo editor to make changes on the original dataset to improve the accuracy of the LPD system.

The rest of the paper is organized as follows. Section II reviews the related work on the LPD. Section III introduces the framework of this proposed method. Section IV presents details about the extraction of *ML-ELBP* features and *ELM*-based classification. Section V shows the experimental results. Finally, the conclusion with future works for this study are provided in Section VI.

II. Related Work

Many studies on developing LPDs have been reported both recently and in the past [9], [13], [15], [19]–[23]. An LPD system consists of three stages: 1) image preprocessing; 2) feature extraction; and 3) classification. The image preprocessing stage is important to improve features' robustness and detection accuracy. Thus, various preprocessing methods have been proposed [24]–[27]. With illumination changes and low/high contrast variations, some methods normalize the input images in a different color space, for example, RGB color space [28]–[30] or gray space [31], [32]. Other methods convert input images from the RGB color space into the HSV color space [33], [34].

Transformations like translation, rotation, and scaling, can been made on training images to improve robustness in feature extraction and detection [35], [36]. Those transformed images can represent various observation conditions. An accurate LPD system is important for security and management purposes. Some researchers used those features that are sensitive to light changes to represent LPs. For example, descriptors based on global and local features [37], [38], local binary patterns (LBPs) [39], histogram of oriented gradients (HOG) [40], [41], scale-invariant feature transform (SIFT) [19], [42], and Gabor features [43], [44] have been used.

As an LBP descriptor on each cell is normalized over several of its neighbors, it includes more discriminative neighboring information than other descriptors. The features based on gray level statistics [45] are used to describe the textures in LPs and have shown good discrimination results. Various combinations of several different descriptors were also proposed for LPD in an attempt to complement each other [37], [39], [40], [43]. However, it would lead to high feature dimensions. A number of techniques [46], [47] have been designed to reduce dimensionality. Recently, several methods were reported to quantify local features using coding techniques. They then concatenated those coded features into a global features representation over the whole image using pooling techniques, and spatial pyramid matching [48], [49]. An one-to-all strategy with a binary support vector machine (SVM) as the base classifier [3], [19], [50] is widely used for LPD. Other multiclass identification techniques are also used, such as neural networks (NNs) [3], [51], [52] and AdaBoost cascade classifier [19], [53]. A cascade classifier has shown a comparable performance with other popular methods in terms of the computational speed of the detection process. However, the detection accuracy for the cascade classifier is not very high if it does not use some efficient enhancement techniques for the training and testing datasets. Deep neural networks (DNNs) based methods have a large number of tuning parameters. Meanwhile, due to the multi hidden-layer structure, its computational cost is very high. In [54], a method using CNN to learn features and using an *ELM* as the classifier was reported. That method obtained competitive results with less computation time compared with those DNN methods. In another study, the ELM was used for LPD with HOG and the means of maximally stable external region [55]. Several ensemble classifiers were also proposed [9], [56] to detect various problems.

In this paper we propose a new descriptor, *ML-ELBP*, for features extraction and use an ensemble of *ELMs* as the classifier. We increase the size of the training dataset by using better enhancement preprocessing methods through three level processing steps. The Gaussian filter and *CLAHE* method are used to introduce extra noise into the training set to avoid the overfitting problem. We extend the English cars plates' dataset to improve the detection accuracy of the LPD system. The proposed method is robust to different LP styles, colors,

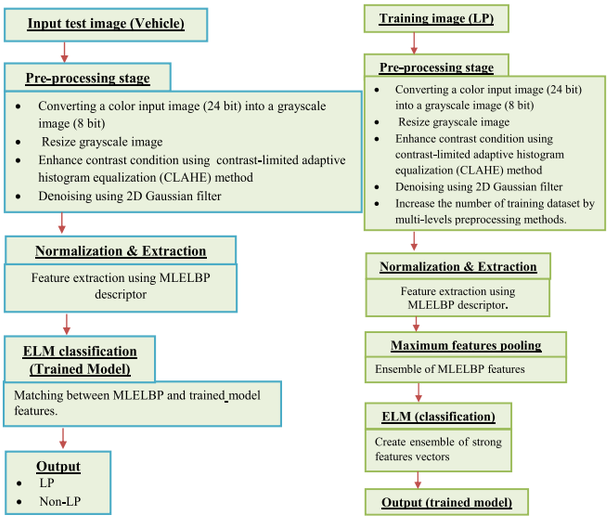


Fig. 2. Framework of the proposed LPD method. (a) Testing Phase and (b) Training Phase.

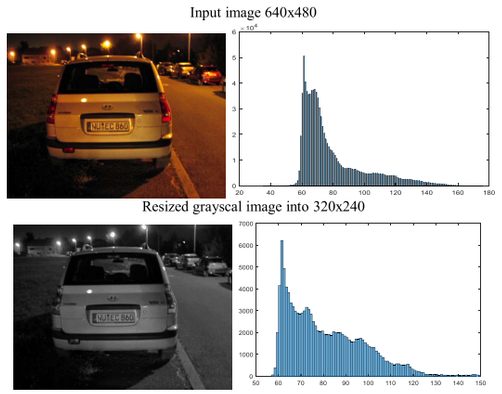


Fig. 3. Examples of the resized original vehicle images from 640×480 into 320×240 resolutions with histograms (X axis = the range of features values in each bin; Y axis = the numbers of features values appeared in each range of bins).

and languages. It can be used to enhance the performance of any existing ANPR systems with different datasets.

III. The Proposed Method

The proposed method involves two phases: the image pre-processing phase and the LP detection phase. The first phase is to enhance vehicle images for better performance while preparing for the detection phase. The second phase includes two stages: 1) feature extraction stage; and 2) *ELM* classification stage.

In the feature extraction stage, the *ML-ELBP* descriptor is employed to extract features from a given input image through three level pre-processing processes. The details of this descriptor is described in Section IV. The second stage applies an *ELM* classifier composed of a single hidden layer feedforward network (*SLFN*) to train the *ML-ELBP* features. An ensemble of strong features vectors are obtained as the trained models.

The training phase uses the *ELM* algorithm to estimate the connection weights in the *SLFN* for all training samples. The proposed framework of the LPD system for training and testing phases are shown in Figs. 2(a) and 2(b).

This study converts a color input image into a grayscale image by formula (1):

$$Gray(i, j) = 0.3 \times R(i, j) + 0.59 \times G(i, j) + 0.11 \times B(i, j) \quad (1)$$

where i and j are any pixel inside a grayscale image. $R(i, j)$, $G(i, j)$ and $B(i, j)$ are three channels of Red, Green, and Blue, respectively, for color images.

Large resolutions need more processing time. The vehicles images are resized from 640×480 to 320×240 as shown in Fig. 3. The proposed method uses texture features instead of color features to detect the LPs because the color features are very sensitive to the illumination conditions and noise [57]. The LPD method employs an intensity transition on a vehicle image based on the extracted three level *ELBP* features as an initial localization for the LP.

The intensity in the LP region may be very high because various texts are included in the vehicle image. There are various noise mixed with the text referring to non-LP regions, such as surface textures, dust, distortion, and small amounts of dirt. Noise incurs unnecessary features in an image. Image enhancement methods are used to reduce noise and improve the lighting conditions. Some LPs may not be recognized if the vehicle images contain too much mixed noise and too distorted (too dirty and dark etc).

IV. The Proposed ML-ELBP Descriptor

The LBP [58] is one of the most popular texture descriptors in the field of the computer vision and image analysis. It has many advantages, such as invariance to illumination conditions, low computational cost, and ease of implementation [47]. A large number of *LBP* variants [6]–[8], [46], [47], [59] have been proposed to improve its discriminative power, robustness, and applicability. However, they are not so effective for processing images with difficult conditions.

In this paper we propose a new descriptor, *ML-ELBP*, based on combining a *CLAHE* method and a Gaussian filter for vehicle LP image feature extraction. It can extract representative texture features from distorted images and images under difficult conditions. The proposed descriptor is shown in Fig. 4. It has a similar structure as the scale space *ELBP* (*SSELBP*) descriptor proposed by Hu *et al.* [46], but with different preprocessing methods. Next we will briefly introduce the concept behind the *ELBP* [7], [46], [47] before explaining the block diagram of the proposed descriptor.

A. Review of the ELBP

Whereas the *LBP* encodes only the relationship between the central point and its neighbours, *ELBP* was initially designed to encode distinctive spatial relationships in a local region. It contains more spatial information. The *ELBP* [7] consists

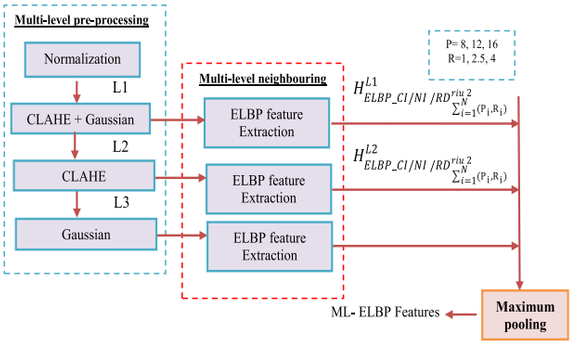


Fig. 4. The block diagram of the proposed *ML-ELBP* descriptor.

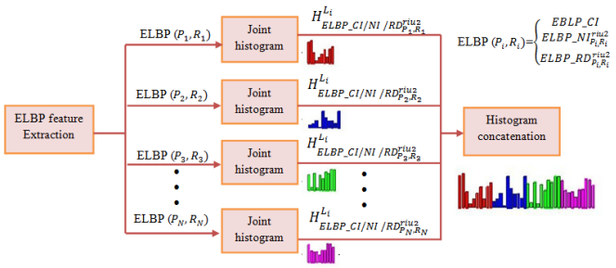


Fig. 5. The block diagram of the ELBP descriptor.

of three *LBP*-like descriptors: the intensity value of the central pixel (*ELBP_CI)*, the intensity values of its neighboring pixels in radial directions with radius R *(ELBP_NI)*, and the intensity difference values of the central pixel with its neighboring pixels in radial directions *(ELBP_RD)*. The ELBP framework is shown in Fig. 5. The central pixel x_C with the intensity value g_C are given to encode the intensity value of x_C. The *ELBP_CI* descriptor compares the g_C value with the mean of the neighboring pixels, denoted, $\beta_{P,R} = \frac{1}{P}\sum_{N=0}^{P-1} g_{P,R,N}$, which is defined as

$$ELBP\_CI\ (x_C) = L\ (g_C - \beta), \quad L(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & if\ x < 0. \end{cases} \tag{2}$$

The *ELBP_CI* descriptor generates an *one-bit* binary pattern in an image for each pixel. In addition to *ELBP_CI*, the *ELBP* involves the *ELBP_NI* descriptor to extract features from the intensities of the neighboring pixels P. The P neighbors of the central pixel value are distributed on the circle with the radius R and have the intensities values denoted as $g_{P,R,N=0,1,\ldots,P-1}$. By comparing the neighboring pixels with their average value denoted as $\beta_{P,R}$, the *ELBP_NI* descriptor encodes the intensity values as follows:

$$ELBP\_NI_{P,R}(x_C) = \sum_{N=0}^{P-1} L(g_{P,R,N} - \beta_{P,R})\ 2^N \tag{3}$$

The *ELBP_NI* generates a *P-bit* binary pattern in each comparison which has been transferred into a decimal value. The third descriptor involved in the *ELBP* is the *ELBP_RD* descriptor, which encodes the intensity differences of the pixels on two circles with the radius R and *R-1* along the radial

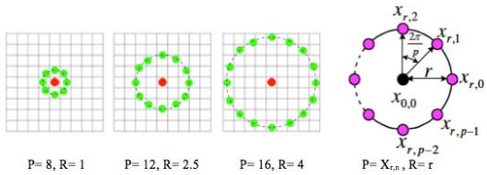


Fig. 6. Pixel relations in radial directions.

directions (see Fig. 6). It is similar to *ELBP_NI* descriptor, it generates a *P-bit* binary pattern and converts to the decimal value, the *ELBP_RD* defined as

$$ELBP\_RD_{P,R-1,R}(x_C) = \sum_{N=0}^{P-1} L\ (g_{P,R,N} - g_{P,R-1,N})\ 2^N \tag{4}$$

Liu *et al.* [7] found that the $ELBP^{riu2}_{r,p}$ led to a good texture classification performance. The operators of *ELBP_NI* and *ELBP_RD* can produce 2^N different binary patterns. We further apply rotation-invariant and uniform mappings on *ELBP_NI* and *ELBP_RD* to remove the rotation effect, and, to reduce the pattern dimension. The updated operators are denoted as $ELBP\_NI^{riu2}_{P,R}$ and $ELBP\_RD^{riu2}_{P,R}$. Where the superscripts, *"ri"* and *"u2"*, represent the rotation-invariant and uniform mappings, respectively.

B. Multi-Level Pre-Processing

A new descriptor uses multi-level pre-processing steps through applying a *Gaussian* filter and the *CLAHE* method. Firstly, we convert an LP image into a grayscale image, and resize it from 50×260 to 25×100 resolution. Then, we build a multi-level pre-processing features space to grayscale image (Y) as follows:

$$L_i = \begin{cases} Y, & i = 0, \\ CLAHE(\sigma) \times G(\sigma), & i = 1, \\ CLAHE(\sigma), & i = 2, \\ G(\sigma) & i = N \end{cases} \tag{5}$$

where L_i is the level of pre-processing steps, $i = 1, 2, \ldots, N$, and N is the total number of the levels. In this paper, $N = 4$. $G(\sigma)$ is the *Gaussian* filter with the standard deviation $\sigma = 0.25$. $CLAHE(\sigma)$ defines as a contrast-limited adaptive histogram equalization method with a standard deviation $\sigma = 0.01$ (see Fig.7).

As a new image is produced in each pre-processing step, one LP image is related to four images now. Therefore, the training LP images database has been expended. Various representative features can be captured from those images. The same pre-processing steps are applied to the testing images, but just with the first step *L1*. After that, we normalize each LP image *(Y)* to ensure that it has zero mean and unit variance.

C. ML-ELBP Features Extraction

The *ELBP* descriptor depends only on one or two local neighboring circles. It is not sufficiently robust to classify texture images with scale variations. To solve this problem,

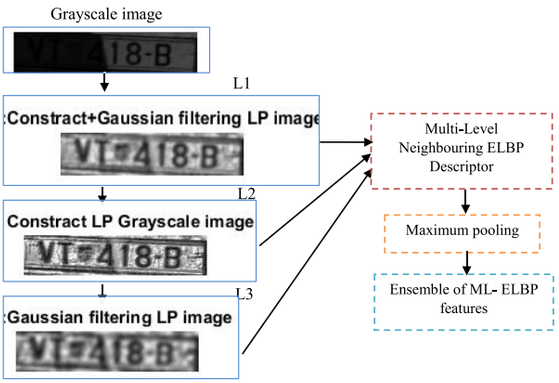


Fig. 7. The multi-level preprocessing steps are applied on a training image.

we use more neighboring circles for the *ELBP* descriptor with the Gaussian filter and *CLAHE* enhancement methods. The training images are increased as the result of multi-level image pre-processing. The next stage is to extract strong weighted features using the *ELBP* descriptor with multi neighboring pixels and different radius. As shown in Fig. 6, with different *(P, R)* choices, the output group of ELBPs are denoted as *ELBP (Pi, Ri)*, $i = 1, 2, \ldots, N$. Where N is determined based on the size and the complexity of the LPs images. To combine patterns in *ELBP* (P_i, R_i), the joint histogram brings the first concatenating patterns, after that it calculates the corresponding histogram. The combination scheme can be considered as the conversion from a joint multi-dimensional histogram to one dimensional histogram. Based on Hu *et al.* [46] the joint histogram of *ELBP_CI*, $ELBP\_NI^{riu2}_{r,p}$ and $ELBP\_RD^{riu2}_{r,p}$ are denoted as $H_{ELBP\_CI/NI/RD^{riu2}_{P_i,R_i}}$. The joint histograms of *ELBP* (P_i, R_i) are calculated and the *ML- ELBP* features are obtained and denoted $H_{ELBP\_CI/NI/RD^{riu2}_{L(i)}}$, where $L(i) = \sum_{i=1}^{N} (P_i, R_i)$.

The descriptor, firstly, divides an LP image into 16 cells (2×8), then labels pixel P for each cell based on different thresholding neighborhood circles and radius *((P = 8, R = 1), (P = 12, R = 2.5)* and *(P = 16, R = 4))*. The different descriptors, *ELBP_CI*, *ELBP_NI*, and *ELBP_RD*, are used in this paper to capture key features for the images with difficult conditions (like distorted, dark, rotations, and dirty images). The feature extraction involves to compare the center pixels values with different neighbor's circles and transfer the decimal results to binary results with weighted values.

D. Maximum Pooling

The maximum pooling strategy is utilized for selecting the maximum values from the corresponding bins of the multi-level *ELBP* histogram features at different scales of an LP image. For each level L_i, the same (P_i, R_i) set is used to calculate the corresponding level of *ELBP* histogram features, denoted as $H^{L_i}_{ELBP\_CI/NI/RD^{riu2}_{L(i)}}$. The significant features of LPs images at different levels can be captured by a parameter pair in the (P_i, R_i) set based on the multiple choices of (P_i, R_i) in the *ML-ELBP*. When the levels of LPs images change, the remaining key features can be captured by the next pair of histogram features. Their mathematical expression is shown as follows:

$$H_{ELBP\_CI/NI/RD_{L(i)}} = \max_{i=1,2,\ldots,N} (H^{L_i}_{ELBP\_CI/NI/RD^{riu2}_{L(i)}}) \quad (6)$$

Next an *ELM* classifier [18] is applied to distinguish the extracted *ML-ELBP* histogram features, and to build the trained models.

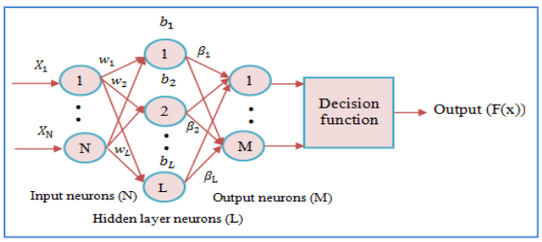


Fig. 8. Architecture of the SLFN.

E. ELM Classifier

The ELM is a machine learning algorithm with a single hidden layer feedforward network (*SLFN*) [18]. It is with high efficient and easy to implement. It normally contains three layers: input layer, hidden layer and output layer [60], [61]. The architecture of an *ELM* is shown in Fig. 8, containing N input neurons, L hidden neurons and M output neurons. For N different input data $\{x_i\}$, $i = 1, 2, \ldots, N$, the output $H(x)$ of the hidden layer can be expressed as shown in formula (7).

$$H(x_{i,}) = \hat{A}(wx + b) \quad (7)$$

where $\hat{A}(x)$ represents the hidden layer activation function of the *ELM*, w is the weight matrix between the hidden layer and the input layer, and b is the bias of the hidden neurons.

The output of the neurons in the output layer can be expressed as in formula (8).

$$H(x_i)\beta = Y_i^T, \quad i = 1, 2, \ldots, N \quad (8)$$

The default activation function, *sigmoid* function for the hidden neurons in the *ELM* is applied in this study. Y represents the training data target matrix. Formula (8) can be abbreviated as:

$$H(x_{i,})\beta = Y \quad (9)$$

where

$$H(x_{i,}) = \begin{bmatrix} \hat{A}(w_1, b_1, x_1) & \hat{A}(w_2, b_2, x_2) \ldots \hat{A}(w_L, b_L, x_1) \\ \hat{A}(w_1, b_1, x_2) & \hat{A}(w_2, b_2, x_2) \ldots \hat{A}(w_L, b_L, x_2) \\ \vdots & \vdots \\ \hat{A}(w_1, b_1, x_N) & \hat{A}(w_2, b_2, x_N) \ldots \hat{A}(w_L, b_L, x_N) \end{bmatrix}_{N \times L}$$

and

$$\beta = \begin{bmatrix} \beta_1^T \\ \beta_2^T \\ . \\ . \\ . \\ \beta_L^T \end{bmatrix}_{L\times m}, \quad \mathrm{Y} = \begin{bmatrix} y_1^T \\ y_2^T \\ . \\ . \\ . \\ y_N^T \end{bmatrix}_{N\times m} \tag{10}$$

The input weights w and bias b are randomly assigned in the *ELM* algorithm. The speed of the *ELM* is significantly faster than other training algorithms. For multiclass classification, the *ELM* aims at minimizing the training error and the norm of the output weights, which results in a less computational time for training the *SLFN* [62]. Thus, in *ELM* the following parameters are minimized:

$$Min.\ err. = \begin{cases} \|H\beta - Y\|^2 \\ \|\beta\| \end{cases} \tag{11}$$

In the implementation, the minimal norm least square method is used in this study. Therefore, only the training error is minimized and the solution is unique. The input layer is connected to the input feature vector x, (i.e., *ML-ELBP*) for the LP image. The dimension number of x is denoted as D. The connection between the input and hidden layers is actually a function of feature mapping from a D space to an L-dimensional space. Given an input feature x, its mapped feature vector can be denoted as:

$$V(x) = [\hat{A}(w_1, b_1, x_1), \ldots, \hat{A}(w_L, b_L, x_N)]. \tag{12}$$

For the output layer, the number of the output nodes M is equal to the number of *ML-ELBP* features for three levels. The output weight between the *ith* hidden node and the *jth* output node is denoted as $\beta_{i,j}$, where $j = 1, \ldots, M$. The value of an output node j can be calculated as:

$$F_j(x) = \sum_{i=1}^{L} \beta_{i,j} \times \hat{A}\,(w_L, b_L, x_N) \tag{13}$$

β is the weighting vector between output and hidden layers.

Thus, for the input sample x_i, its output vector at the hidden layer can be written as

$$F(x) = [f_1(x), \ldots, f_M(x)] = V(x)\beta \tag{14}$$

Tamura and Tateishi [63] and Huang [64] pointed out that a *SLFN* with N sigmoid hidden nodes could learn exactly N distinctive ways. In this paper, the number of the hidden nodes is 550 for getting the optimal detection results with the extracted *ML-ELBP* features. It is set based on the detection performance with the testing data set. The cell size is 9×9 for different thresholding neighborhood circles (8, 12, and 16) and the average dimension of the feature vectors is 710.

V. Experimental Results

A. Database

In this study, various vehicles images captured by digital cameras under different environmental conditions (cloudy weather, rainy day, night lighting, dusk) are used. The database contains different types of 471 vehicles images, such as trucks, passenger cars, and buses. This database is publicly available [65]. An *online photo editor* [66] is used to increase the vehicle images in the English license plates database. The photo editor makes various complex changes on the original images (see Fig. 9), and the resultant images are included in the database. The final total number of vehicles images used in this study is 1500. It contains rotated images in order to detect various types of the rotated LPs with different angles, such as 45°, 30°, 20°, 15°, and 5° as shown in Fig. 10. To enable the LPD system to detect the LPs from low quality images, the LPs images with different illumination conditions are added to the training dataset. The final LPs images in the database are divided into two groups: the testing group containing 1000 vehicle images and the training group having 500 LPs images (see Fig. 11). Both original and changed vehicles images are used in the testing set. The experimental results showed that the detection accuracy was not noticeably increased when more images were put in the training set. This demonstrated that all the key characteristics have been

Fig. 9. Examples of testing vehicles images in the database. (a) Vehicles images from the original database. (b) Vehicles images with difficult changes using the online photo editor.

Fig. 10. Some examples of LPs training images with rotations.

Fig. 11. Some examples for difficult training LPs images.

TABLE I
THE CLASSIFICATION RESULTS BY LBP, ELBP, SSELBP, AND THE PROPOSED ML-ELBP DESCRIPTORS WITH THE ELM CLASSIFIER ON ENGLISH PLATES CARS DATABASE, WHERE THE TRAINING IS DONE WITH FIVE ROTATION ANGLES

Method	(P, R)	FD	RA					AA
			45°	30°	20°	15°	5°	
ELM+LBP	(8,1)	350	89.34	92.45	93.66	95.50	98.27	94.19%
ELM+ELBP	(8,1)	400	88.6	90.13	94.59	97.40	99	94.89%
	(16, 2)	680	95.42	96.29	97.38	99.48	99.72	94.98%
	(24, 5)	1000	96.04	96.92	97.67	96.54	97.79	97.01%
	(8,1)+ (16, 2)+ (24, 5)	1300	97.35	97.81	97.89	96.93	98.80	97.74%
ELM+SSELBP	(8,1)	500	94.13	95.16	96.49	98.09	98.50	96.47%
	(12, 1.5)	880	95.41	96.01	99	97.94	97.82	97.23%
	(16, 2)	940	96.50	96.78	96.90	98.50	98.97	97.53%
	(8,1)+ (12, 1.5)+ (16, 2)	1010	97.45	97.56	98.54	99.02	99.50	98.41%
Proposed ELM+ML-ELBP	(8,1)	540	97.52	97.84	98.52	99.10	99.87	98.57%
	(12, 2.5)	670	97.65	96.64	98.45	99.18	99.32	98.24%
	(16, 4)	750	98.15	97.91	98.57	99.40	99.89	98.78%
	(8,1)+ (12, 2.5)+ (16, 4)	**710**	**99.01**	**99.34**	**98.54**	**99.95**	**100**	**99.16%**

FD: Features Dimension; RA: Rotation Angle; AA: Average Accuracy.

captured from the 500 original LP images and their three level pre-processing images in the training data.

B. Features Extraction and Classification

As this study focuses on features extraction rather than classification, the *ELM* classifier has been used as a trainer to distinguish the extracted histogram features for the LP regions. In the proposed method, to build the multi-level processing, we use a *Gaussian* filter with the standard deviation $\sigma = 0.25$, and *CLAHE* (σ) with the standard deviation $\sigma = 0.01$, and set the pre-processing step to three. For all the levels, we extract *ML-ELBP* features using the same set (P_i, R_i), $i = 1, \ldots, N$, where N is determined based on the size and the complexity of LPs images. We set i depending on the use of different neighborhood circles $i = 1, 2, \ldots, 8$; $i = 1, 2, \ldots, 12$; or $i = 1, 2, \ldots, 16$ with $R = 1; 2.5; 4$, respectively. The LPs features from the images with different angles and descriptors are classified using the *ELM*.

For comparison, in addition to our proposed *ML-ELBP*, other descriptors, *LBP*, *ELBP*, and *SSELBP*, along with the *ELM* classifier are also applied to the same vehicle LP images. All the experimental results are listed in Table I. It is noticed that the best classification accuracy is from the *ML-ELBP* descriptor with radius of *(1; 2.5; 4)*. Through our experiments and as reported in [7], the performance of the *ELBP* increases when the neighborhood size is 11×11. The best performance is achieved by $ELBP^{riu2}_{((8,1)+(16,2)+(24,5))}$ for all the training images and with different angles. The classification accuracy is 97.89% when the features dimension is 1300. In general, the *LBP* had the worst performance among the descriptors. The central pixel also provides useful discriminative information. Neglecting the central pixel would clearly result in information lost, which is consistent with the conclusion reported by Guo *et al.* [6] and Varma and Zissermam [67]. Thus it is better to explicitly include the information from the central pixel in *ELBP* based descriptors. Better results by the *SSELBP* descriptor at the neighborhood size of 9×9 are obtained. Combining $SSELBP^{riu2}_{((8,1)+(12,1.5)+(16,2))}$ can achieve an accurate result of 98.41%, for the five angles at 45°, 30°, 20°, 15°, and 5°. With all the training images and different angles the classification accuracy is 98.76%. The features dimension is 1010. The SSELBP performs better than the ELBP and LBP descriptors. However, the proposed *ML-ELBP* descriptor produces the best classification results for the five angles at 45°, 30°, 20°, 15° , and 5°. It combines all the features of the $M\text{-}ELBP^{riu2}_{((8,1)+(12,2.5)+(16,4))}$ and can capture all important textures information in an LP area. The average classification accuracy for the five LP angles is 99.16%. The classification accuracy for all datasets with different angles and resolutions is 99.78%. The features dimension is 710 for a 9×9 cell size. Also, it was observed that the good results obtained by using different neighboring pixels with different radius values. We acknowledge that different neighboring analysis increase the histogram features dimension. However the dimensionality of 710 for three resolutions is not a significant problem in relation to our proposed descriptor. Finally, the *ML-ELBP* produces robust classification results compared with the *ELBP* and *SSELBP* descriptors.

The *ELM* constructs several models in the hidden layer from the extracted features. Each training image is split into 16 cells with 8×2 subsamples. The 16 cells histograms are merged into one model histogram for the *ML-ELBP* descriptor. Moreover, from one designated training image, each training sample has three other physically different samples generated by the three preprocessing levels. After that, the classifier produces binary values or predicted value of "0" for non-LP and "1" for LP. It can become a detector based on the predicted value from the classifier. It is trained with different rotation invariant textures for the LP with five angles (45°, 30, 20°, 15° and 5°) and different neighborhood circles with radius selection P and R. We set the number of angles depending on the vehicle images dataset which contains only those angles. The number of overlaps depends on the locations from the LP region, with an average of 2∼6 bounding boxes. Therefore, the trained model detects different numbers of LP regions per vehicle image.

TABLE II

THE AVERAGE CLASSIFICATION RESULTS FOR ALL THE TRAINING DATASET WITH DIFFERENT RESOLUTIONS BY LBP, ELBP, SSELBP, AND THE PROPOSED ML-ELBP DESCRIPTORS

Method	IS	CS	FD	CA
ELM+LBP	28×140	3×3	350	94.55%
ELM+ ELBP $^{riu2}_{((8,1)+(16,2)+(24,5))}$	28×140	11×11	1300	97.89%
ELM+ SSELBP $^{riu2}_{((8,1)+(12,1.5)+(16,2))}$	28×140	9×9	1010	98.76%
Proposed ELM+ MLELBP $^{riu2}_{((8,1)+(12,2.5)+(16,4))}$	**28×140**	**9×9**	**710**	**99.78%**

CA: Classification Accuracy; FD: Feature Dimension; IS: Image Size; CS: Cell Size.

This study used one single classifier for all the rotated images. During the experiments, firstly five classifiers were used to train all the LP images rotated with angles of 45°, 30°, 20°, 15°, and 5°, separately. Then, five trained models were built and computed with results. After that, one classifier was trained for all the rotated images and the results were obtained. We observed that the results from both cases were very similar. One single classifier, therefore, was finally used for efficiency. There were two trained models: one was the normal LP, and another one was for the rotated images with difficult problems. Each trained model included two classes (LP *vs* non-LP). In this work, we tried to increase the number of pre-processing levels to reduce features dimensions. But this did not help improve the accuracy for classification process. All the training and testing images were conducted under different conditions like low/high lighting, dirt, rotation, foggy, and distortion.

Due to the illuminant varied, some LP samples included significant large grayscale distortions. Therefore, the English cars database is more challenging than other databases which were used in the previous studies [22]. Table II presents the average classification results for our proposed descriptor, compared with other existing state-of-the-art descriptors [5], [7], [8].

C. Impact of the Number of the Hidden Neurons

In the experiments, the only parameter to be determined for the *ELM* is the number of the hidden neurons with *'sigmoid'* activation function. Fig. 12 shows the detection rates with different numbers of neurons for the English plates cars database. With a small number of the hidden neurons, the detection rate is very low. The detection rate keeps improving as the number of the neurons increases from 100 to 550. Adding more hidden neurons does not help to further boost the performance beyond 550. Thus 550 neurons in the hidden layer was set for the *ELM* in the experiments. We also tested other activation functions, such as *'sin'* and *'hardlim'*. We found that the sigmoid function gave better results. The whole process can finish in one time period without iterations with a minimum training error. We can notice from Fig. 12 that the performance of the *ELM* is very stable for the large number of the hidden neurons. Also, the performance tends to become worse, when it has too few or many nodes generated randomly [68]. However,

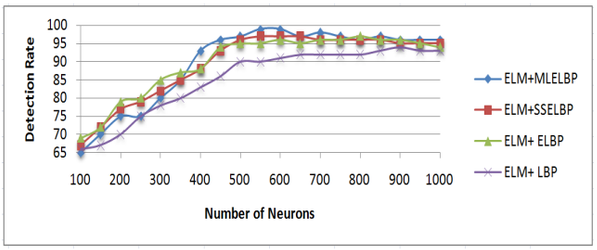


Fig. 12. Detection rate and the number of the hidden neurons in the *ELM* with different *LBP* based descriptors.

TABLE III

THE COMPARISONS OF THE LP DETECTION PERFORMANCES BY DIFFERENT METHODS

Method	FP	PR (%)	RR (%)	F-m (%)
ELM+LBP	6.5%	95.9%	93.73%	94.80%
ELM+ELBP	5%	98.6%	97.12%	97.85%
ELM+SSELBP	4.2%	97.45%	98.83%	98.13%
Proposed (ELM+ML-ELBP)	**5%**	**98.2%**	**99.10%**	**98.65%**

the *ELM* with the *ML-ELBP* can achieve a good detection rate at 99.10% with 550 hidden nodes, which is slightly higher than the *SSELBP* with 650 nodes and with a detection rate of 98.76%. It can also be observed that the *ELM* with the *LBP* achieves a low detection rate of 94.55% with 900 nodes compared to the *ELBP* that achieved 97.89% detection rate with 800 hidden nodes. The proposed method outperforms all the existing *LBP* based descriptors along with the *ELM* in terms of the detection rate. It appears to be suitable in real-time applications that, need fast prediction results and response capability.

D. LP Detection Performance

The detection accuracies for the English car plates' dataset are shown in Table III. In this study, using different descriptors to produce more advanced features brings the detection rate to nearly 100%. For the real world scenario database, using those features helps improve the detection rate significantly and produce good results. To evaluate the performance of the proposed method, common measurement, such as detection or recall rate *(RR)*, precision rate *(PR)*, and F-measure rate *(FR)* are used in this paper. These measurements consider the true positive *(TP)* detection rate from the number of the positives as the ground truth that is related to the number of the false positive *(FP)* rate [32]. Those measurements can be defined as follows:

$$Positive\ prediction\ or\ Precision\ rate\ (PR) = \frac{TP}{TP + FP} \tag{15}$$

$$Detection\ or\ Recall\ rate\ (RR) = \frac{TP}{TP + FN} \tag{16}$$

$$F\text{-}measure\ (Fm) = 2^* \left(\frac{RR\ ^* PR}{RR + PR} \right) \tag{17}$$

TABLE IV
COMPARISON RESULTS FOR THE PROPOSED LPD METHOD AND OTHER METHODS. THE PROPOSED METHOD PRODUCED THE BEST DETECTION RATE WITH THE LOWEST FP RATE

Ref.	Method	FP	PR (%)	RR (%)	F-m (%)
Al-Shemarry et al.[56]	Ensemble of AdaBoost+3L-LBP	5.6%	95.9%	98.56%	97.19%
Azam and Islam [22]	Frequency domain mask, contrast improvement technique, statistical binarization, Radon transform, and entropy vector.	6.3%	NR	98.15 %	NR
He et al. [70]	Blob for candidate detection, filtering affine distortion, saliency detection.	19.6%	92.7%	94.7%	93.68%
Ho et al. [19]	AdaBoost+(SIFT+SVM)	NR	90.18%	92.07%	91.10%
Asif et al. [69]	YDbDr color space+ Otsu method	6.5%	87.15	93.86%	90.38%
Proposed	**ELM+ML-ELBP**	**5%**	**98.2%**	**99.10%**	**98.65%**

NR: not reported

where *(Fm)* is the trade-off between *RR* and *PR*; *FN* is the false negative rate, which is equal to the number of vehicles images in the ground truth that have LP, but the output of LPD system shows that there is no LP inside those images.

Along with the *ELM* classifier, the *ML-ELBP* outperforms other *LBP* descriptors. It takes less than one second to process 640×480 images under difficult lighting conditions. From Table III we can observe that the proposed *ML-ELBP* descriptor produces the best recall and F-measure rates compared with the *LBP*, *ELBP*, and *SSELBP* descriptors. The *ELBP* descriptor achieves 98.6% precision rate which is 0.4% higher than the *ML-ELBP* descriptor. The comparison results of the performance evaluation with some existing methods are shown in Table IV. The proposed method in this study achieves a recall rate of 99.10%, which is higher than the results by Asif, *et al.* [69] and Azam and Islam [22]. The F-measure of the proposed method is 98.65%, which is also the best compared with the method used by Asif *et al.* The *FP* rate is less than those by Asif *et al.* and Azam and Islam. Al-Shemarry *et al.* [56] introduced an efficient detection method that achieved a 98.56% and 97.19% for recall rate and f-measure, respectively, with 5.6% FP rate. The proposed method in this paper has a slightly higher recall and F-measure rate, respectively, and with a slightly lower *FP* rate than those reported in [56]. This work obtained a 98.2%, 99.10%, and 98.65% for precision, detection, and F-measure rate, respectively, with an *FP* rate of 5%. The results are much better than those by He *et al.* [70] and Ho *et al.* [19].

Based on the above performance evaluation measurements, the proposed method outperforms all the existing methods in terms of precision, recall, F-measure, and *FP* rates for LP detection. The *ELM* is significantly faster than other classification methods, which makes the proposed method more feasible for large-scale real-time applications.

E. Detection Results

Some detection results of the proposed method based on the vehicles images in the database are shown in Fig. 13. It can be noticed that all the LPs were detected although some unwanted features appeared with different LP variations conditions. The reason is that the lighting conditions are very poor with dusk. Therefore, these conditions are making the LP hard to identify. Moreover, some false positives were noticed when several objects that look like a LP (such as commercial signs and vehicle logo) in the vehicle images are detected with a low trust value.

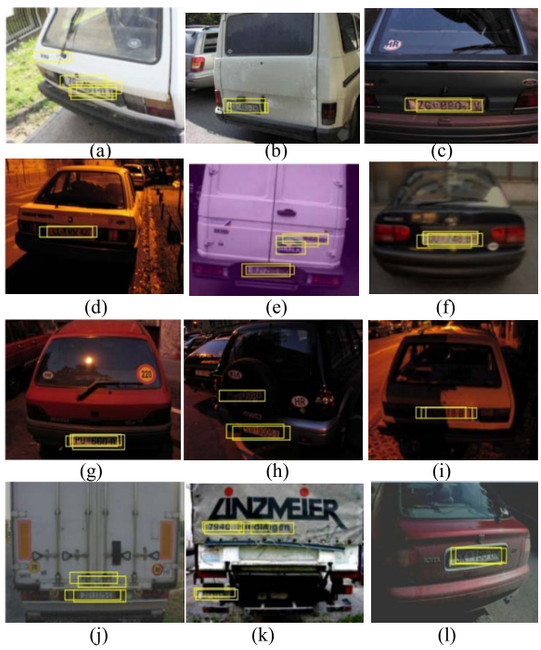


Fig. 13. Examples of successful vehicles images detection results using the proposed method for the cars images with dirt and low light (b), (e), (j), (k); with different views point (a), (b), (d), (l); with fog and dusk (e), (f), (j); with distortion (c), (f), (i) and with low/high contrast (d), (g), (h), (i) problems.

F. Computational Cost

The computational costs for the LP detection using the *ELM* with different *LBP* descriptors for the training and testing phases are shown in Table V. The reported time is the averaged time for both phases. All the experiments were conducted using a computer with 3.4 GHz Intel Core i7-4770, 16GB RAM, using MATLAB, R2017b version. The processing time is an important indicator of a system performance. The proposed descriptor does not need much processing time and achieves a very good accuracy compared with other existing descriptors. The average of detection time per one vehicle image is 0.735s. The vehicles images under dusty, foggy, night-time and distortion conditions normally need

TABLE V
THE AVERAGE COMPUTATIONAL TIME FOR THE TRAINING AND TESTING PHASES. THE TRAINING TIME IS IN SECONDS (S) AND TESTING TIME IS IN MILLISECONDS (MS)

Method	Training time per image (s)		Testing time per image (ms)		Detection rate (%)
	Minimum training time	Maximum training time	Minimum testing time	Maximum testing time	
ELM+LBP	1.520	2.20	0.923	2	93.73%
ELM+ELBP	2.110	3.35	1.123	1.5	97.12%
ELM+SSELBP	3.150	4.13	1.067	1.2	98.83%
Proposed (ELM+ML-ELBP)	**2.008**	**4.25**	**0.459**	**0.735**	**99.10%**

more processing time to detect the complicated LP features. Some vehicle images include other logos and commercial signs on them, which are easily mixed up as the LP area. The proposed method is robust and can capture the significant LP features during the pre-processing phase. It produces a low *FP* rate. From Table V the computation time for the *ML-ELBP* descriptor with the *ELM* is much shorter, compared with other existing descriptors to solve many difficult problems.

VI. CONCLUSION

This study proposed a new ML-ELBP descriptor to extract different LP features from a multi-level preprocessing stage by a Gaussian filter and the *CLAHE* method using an *ELBP* descriptor. We increased the number of training vehicle images through the pre-processing stage. The English car plate database was extended using an online photo editor to make different changes on the original vehicle images to reflect various difficult conditions. It helped improve the accuracy of the LPD system. The *ELM* classifier was used to classify and learn the *ML-ELBP* features in order to produce an ensemble of strong features vectors or trained network models as a detector to detect different LPs. This work used a feature vector of 710 dimensions to represent the LP regions, which was trained using a *SLFN* with 550 hidden nodes. The output neurons depend on the number of LP classes in the training dataset. The proposed method was tested on further distorted images (unseen data) taken under difficult conditions, such as low/high contrast, foggy, and rotated LPs. The overall performance evaluation for the detection, precision and F-measure rate is 99.10%, 98.2%, and 98.86%, respectively, with an *FP* rate of 5%. The experimental results of the proposed method were also compared with several existing LPD methods that used the same database. It outperformed those methods in terms of the detection rate and efficiency. The average detection time per vehicle image was 0.735s. Many existing methods used only the testing phase with the pre-processing stage under some assumptions. This proposed method works well without assumptions due to the use of two separate phases of testing and learning.

The experimental results demonstrated that the proposed method could be used efficiently for real-time applications. In the future, we intend to improve the proposed LPD to further reduce the false positive, and to better adopt weather conditions. At the same time, the overall processing time of the detection system will be reduced through high speed hardware and software selections.

REFERENCES

[1] S. L. Gomes *et al.*, "Embedded real-time speed limit sign recognition using image processing and machine learning techniques," *Neural Comput. Appl.*, vol. 28, pp. 573–584, Dec. 2017.

[2] M. S. Sarfraz, A. Shahzad, M. A. Elahi, M. Fraz, I. Zafar, and E. A. Edirisinghe, "Real-time automatic license plate recognition for CCTV forensic applications," *J. Real-Time Image Process.*, vol. 8, no. 3, pp. 285–295, Sep. 2013.

[3] F. Yuan and R. L. Cheu, "Incident detection using support vector machines," *Transp. Res. C, Emerg. Technol.*, vol. 11, nos. 3–4, pp. 309–328, Jun./Aug. 2003.

[4] B. Azad and E. Ahmadzadeh, "Real-time multiple license plate recognition system," *Int. J. Res. Comput. Sci.*, vol. 4, no. 2, pp. 11–17, 2014.

[5] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognit.*, vol. 29, no. 1, pp. 51–59, 1996.

[6] Z. Guo and D. Zhang, "A completed modeling of local binary pattern operator for texture classification," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1657–1663, Jan. 2010.

[7] L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth, "Extended local binary patterns for texture classification," *Image Vis. Comput.*, vol. 30, no. 2, pp. 86–99, Feb. 2012.

[8] Z. Guo, X. Wang, J. Zhou, and J. You, "Robust texture image representation by scale selective local binary patterns," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 687–699, Feb. 2016.

[9] Y. Zhao, J. Gu, C. Liu, S. Han, Y. Gao, and Q. Hu, "License plate location based on Haar-like cascade classifiers and edges," in *Proc. 2nd WRI Global Congr. Intell. Syst.*, vol. 3, Dec. 2010, pp. 102–105.

[10] B. Zhang, H. Pan, Y. Li, and L. Xu, "Reliable license plate recognition by cascade classifier ensemble," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.*, 2014, pp. 699–706.

[11] N. Boonsim and S. Prakoonwit, "Car make and model recognition under limited lighting conditions at night," *Pattern Anal. Appl.*, vol. 20, no. 4, pp. 1195–1207, Nov. 2016.

[12] J.-H. Jo and D.-J. Kang, "An ensemble classifier based method to select optimal image features for license plate recognition," *Trans. Korean Inst. Elect. Eng.*, vol. 65, no. 1, pp. 142–149, 2016.

[13] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz. (2017). "License plate detection and recognition using deeply learned convolutional neural networks." [Online]. Available:https://arxiv.org/abs/1703.07330

[14] Y. Liu, H. Huang, J. Cao, and T. Huang, "Convolutional neural networks-based intelligent recognition of Chinese license plates," *Soft Comput.*, vol. 22, no. 7, pp. 2403–2419, Apr. 2018.

[15] H. Li and C. Shen. (2016). "Reading car license plates using deep convolutional neural networks and LSTMs." [Online]. Available: https://arxiv.org/abs/1601.05610

[16] M. S. Nixon and A. S. Aguado, *Feature Extraction & Image Processing for Computer Vision*. New York, NY, USA: Academic, 2012.

[17] K. Zuiderveld, "Contrast limited adaptive histogram equalization," in *Graphics Gems IV*, New York, NY, USA: Academic, 1994, pp. 474–485.

[18] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Netw.*, vol. 61, pp. 32–48, Jan. 2015.

[19] W. T. Ho, H. W. Lim, and Y. H. Tay, "Two-stage license plate detection using gentle adaboost and SIFT-SVM," in *Proc. 1st Asian Conf. Intell. Inf. Database Syst.*, Apr. 2009, pp. 109–114.

[20] M. Hasan, "Real time detection and recognition of license plate in bengali," Bourns College Eng., Riverside, CA, USA, 2013. [Online]. Available: http://escholarship.org/uc/item/0m27j18m

[21] M. Wafy and A. M. M. Madbouly, "Efficient method for vehicle license plate identification based on learning a morphological feature," *IET Intell. Transp. Syst.*, vol. 10, no. 6, pp. 389–395, Aug. 2016.

[22] S. Azam and M. M. Islam, "Automatic license plate detection in hazardous condition," *J. Vis. Commun. Image Represent.*, vol. 36, pp. 172–186, Apr. 2016.

[23] R. Panahi and I. Gholampour, "Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 767–779, Apr. 2017.

[24] D.-S. Kim and S.-I. Chien, "Automatic car license plate extraction using modified generalized symmetry transform and image warping," in *Proc. IEEE Int. Symp. Ind. Electron.*, vol. 3, Jun. 2001, pp. 2022–2027.

[25] S. Kim, D. Kim, Y. Ryu, and G. Kim, "A robust license-plate extraction method under complex image conditions," in *Proc. Object Recognit. Supported User Interact. Service Robots*, vol. 3, Aug. 2002, pp. 216–219.

[26] T. D. Duan, D. A. Duc, and T. Le Hong Du, "Combining Hough transform and contour algorithm for detecting vehicles' license-plates," in *Proc. Int. Symp. Intell. Multimedia, Video Speech Process.*, Oct. 2004, pp. 747–750.

[27] C.-T. Hsieh, Y.-S. Juan, and K.-M. Hung, "Multiple license plate detection for complex background," in *Proc. 19th Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, vol. 2, Mar. 2005, pp. 389–392.

[28] G. Li, R. Yuan, Z. Yang, and X. Huang, "A yellow license plate location method based on RGB model of color image and texture of plate," in *Proc. 2nd Workshop Digit. Media Appl. Museum Heritages (DMAMH)*, Dec. 2007, pp. 42–46.

[29] A. H. Ashtari, M. J. Nordin, and M. Fathy, "An iranian license plate recognition system based on color features," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 4, pp. 1690–1705, Aug. 2014.

[30] A. H. Ashtari, M. J. Nordin, and S. M. M. Kahaki, "A new reliable approach for Persian license plate detection on colour images," in *Proc. Int. Conf. Elect. Eng. Inform. (ICEEI)*, Jul. 2011, pp. 1–5.

[31] M. K. Song and M. M. K. Sarker, "Modeling and implementing two-stage AdaBoost for real-time vehicle license plate detection," *J. Appl. Math.*, vol. 2014, Aug. 2014, Art. no. 697658.

[32] W. Jia, H. Zhang, and X. He, "Region-based license plate detection," *J. Netw. Comput. Appl.*, vol. 30, no. 4, pp. 1324–1333, Nov. 2007.

[33] X. Wang, M. Zhou, and G. Geng, "An approach of vehicle plate extract based on HSV color space," *Comput. Eng.*, vol. 17, no. 30, pp. 133–135, 2004.

[34] H. Wang, X. Wang, W. Li, and X. Jia, "Color prior knowledge-based license plate location algorithm," in *Proc. 2nd Workshop Digital Media Appl. Museum Heritages (DMAMH)*, Dec. 2007, pp. 47–52.

[35] C.-C. R. Wang and J.-J. J. Lien, "Automatic vehicle detection using local features—A statistical approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 83–96, Mar. 2008.

[36] D.-M. Tsai and C.-H. Chiang, "Rotation-invariant pattern matching using wavelet decomposition," *Pattern Recognit. Lett.*, vol. 23, nos. 1–3, pp. 191–201, Jan. 2002.

[37] X. He, H. Zhang, W. Jia, Q. Wu, and T. Hintz, "Combining global and local features for detection of license plates in video," in *Proc. Image Vis. Comput. Conf. New Zealand*, Dec. 2007, pp. 288–293.

[38] M. Hussein, F. Porikli, and L. Davis, "Object detection via boosted deformable features," in *Proc. 16th IEEE Int. Conf. Image Process. (ICIP)*, Nov. 2009, pp. 1445–1448.

[39] T.-T. Nguyen and T. T. Nguyen, "A real time license plate detection system based on boosting learning algorithm," in *Proc. 5th Int. Congr. Image Signal Process.*, Oct. 2012, pp. 819–823.

[40] K. Zheng, Y. Zhao, J. Gu, and Q. Hu, "License plate detection using Haar-like features and histogram of oriented gradients," in *Proc. IEEE Int. Symp. Ind. Electron.*, May 2012, pp. 1502–1505.

[41] S. Lew, C.-S. Yi, W.-J. Lee, B.-R. Lee, K.-W. Min, and H.-C. Kang, "Extraction of the license plate region using HoG and AdaBoost," *J. Digit. Contents Soc.*, vol. 10, no. 4, pp. 597–604, 2009.

[42] M. Zahedi and S. M. Salehi, "License plate recognition system based on SIFT features," *Procedia Comput. Sci.*, vol. 3, pp. 998–1002, 2011.

[43] Y. Wang, H. Zhang, X. Fang, and J. Guo, "Low-resolution Chinese character recognition of vehicle license plate based on ALBP and Gabor filters," in *Proc. 7th Int. Conf. Adv. Pattern Recognit.*, Feb. 2009, pp. 302–305.

[44] S. Ktata, F. Benzarti, and H. Amiri, "License plate localization using Gabor filters and neural networks," *J. Comput. Sci.*, vol. 9, no. 10, p. 1341, Oct. 2013.

[45] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, Feb. 2013.

[46] Y. Hu, Z. Long, and G. AlRegib, "Scale selective extended local binary pattern for texture classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 1413–1417.

[47] L. Liu, S. Lao, P. W. Fieguth, Y. Guo, X. Wang, and M. Pietikäinen, "Median robust extended local binary pattern for texture classification," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1368–1381, Mar. 2016.

[48] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 2169–2178.

[49] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3360–3367.

[50] K. S. Durgesh and B. Lekha, "Data classification using support vector machine," *J. Theor. Appl. Inf. Technol.*, vol. 12, no. 1, pp. 1–7, Feb. 2010.

[51] F. Porikli and T. Kocak, "Robust license plate detection using covariance descriptor in a neural network framework," in *Proc. IEEE Int. Conf. Video Signal Based Surveill.*, Nov. 2006, p. 107.

[52] S. H. Park, K. I. Kim, K. Jung, and H. J. Kim, "Locating car license plates using neural networks," *Electron. Lett.*, vol. 35, no. 17, pp. 1475–1477, Aug. 1999.

[53] Y.-N. Chen, C.-C. Han, G.-F. Ho, and K.-C. Fan, "Facial/license plate detection using a two-level cascade classifier and a single convolutional feature map," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 12, p. 183, Dec. 2015.

[54] S. Ding, L. Guo, and Y. Hou, "Extreme learning machine with Kernel model based on deep learning," *Neural Comput. Appl.*, vol. 28, no. 8, pp. 1975–1984, Aug. 2017.

[55] C. Gou, K. Wang, Z. Yu, and H. Xie, "License plate recognition using MSER and HOG based on ELM," in *Proc. IEEE Int. Conf. Service Oper. Logistics, Inform.*, Oct. 2014, pp. 217–221.

[56] M. S. Al-Shemarry, L. Yan, and S. Abdulla, "Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images," *Expert Syst. Appl.*, vol. 92, pp. 216–235, Feb. 2018.

[57] R. Azad, F. Davami, J. Jeo, and H. R. Shayegh, "New framework based on complementary methods for efficiency and accuracy of license plate recognition system," in *Proc. 6th Conf. Inf. Knowl. Technol. (IKT)*, May 2014, pp. 171–176.

[58] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[59] Y. Hu, Z. Long, and G. AlRegib, "Completed local derivative pattern for rotation invariant texture classification," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 3548–3552.

[60] G. Feng, G.-B. Huang, Q. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009.

[61] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognit.*, vol. 38, no. 10, pp. 1759–1763, Oct. 2005.

[62] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[63] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: Four layers versus three," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 251–255, Mar. 1997.

[64] G.-B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 274–281, Mar. 2003.

[65] *LPDatabase*. Accessed: Jul. 2016. [Online]. Available: http://www.zemris.fer.hr/projects/LicensePlates/english/baza_slika.zip

[66] *OnlinePhotoEditor*. Accessed: Jul. 2017. [Online]. Available: https://www.freeonlinephotoeditor.com/

[67] M. Varma and A. Zisserman, "A statistical approach to material classification using image patch exemplars," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2032–2047, Nov. 2009.

[68] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, 2006.

[69] M. R. Asif, Q. Chun, S. Hussain, and M. S. Fareed, "Multiple licence plate detection for Chinese vehicles in dense traffic scenarios," *IET Intell. Transport Syst.*, vol. 10, no. 8, pp. 535–544, Oct. 2016.

[70] T. He, J. Yao, K. Zhang, Y. Hou, and S. Han, "Accurate multi-scale license plate localization via image saliency," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 1567–1572.

Meeras Salman Al-Shemarry received the bachelor's degree in computer science from Babylon University, Iraq, in 2002, and the master's degree in IT from University Utara Malaysia, Malaysia, in 2010. She is currently pursuing the Ph.D. degree with the Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia. She is also a Lecturer with the Computer Department, Science College, Karbala University, Iraq. Her research interests include system analysis using UML diagrams, image processing and objects detection, artificial intelligence, and database management system.

Yan Li is currently a Professor in computer science with the Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia. Her research interests are in the areas of signal and image processing, biomedical engineering, artificial intelligence, big data analytics, and computer networking technologies.

Shahab Abdulla received the B.Sc. and M.Sc. degrees from the University of Technology, Baghdad, Iraq, and the Ph.D. degree from the University of Southern Queensland (USQ), Australia. He is currently a Lecturer with the Language Centre, USQ. His research interests include computer control systems, intelligent control, complex medical engineering, and in the areas of biomedical engineering, networked system, robotics, and mathematics research.

5

CHAPTER 5

DEVELOPING LEARNING-BASED PREPROCESSING METHODS FOR DETECTING COMPLICATED LICENCE PLATES

5.1 Introduction

The content of this chapter is an exact copy of the published paper in the journal of *IEEE Access* by Al-Shemarry, M.S. and Li, Y, (2020) 'Developing Learning-Based Preprocessing Methods for Detecting Complicated Vehicle Licence Plates'.

In Chapter 4, the MLELBP_ELM method succeeded to solve some weakness that faced the detection system in chapter 3, such as increased the detection accuracy, reduced the execution time, and a slightly improved FPR.

This chapter focuses on the preprocessing techniques that use a combination of powerful descriptors for distorted images. At the preprocessing stage the method now consists of a Gaussian filter, an enhanced version of the cumulative histogram equalization (ECHE), and a contrast-limited adaptive histogram equalization (CLAHE) techniques. These enhancement techniques are very useful to filter out the unwanted LP regions, to reduce feature dimensions and save the processing time at extraction stage. At the extraction stage, the combination of powerful descriptors, a median-filter histogram of oriented gradient (MHOG), and LBP descriptors are used for extracting complicated feature values. The extracted features use as inputs to the support vector machine (SVM) classifier to construct the detector that identifies the LP area. The SVM detector successfully removed the redundant bounding boxes, or unwanted sliding windows, which lead to increased FPR, by using the mean-shift algorithm. The proposed method was tested on a very challenging database published in Chapter 4. The

performance of the SVM detector was evaluated with the ELM detector through a 5-fold cross-validation procedure and using the receiver operating characteristic (ROC) curve. Also, the proposed approach was evaluated using several performance measurement metrics for object detection systems. The outcomes were compared with the recently reported algorithms in Chapters 3 and 4, and with other existing detection algorithms. From the experimental results, the proposed method outperformed other existing methods with the same database in terms of the FPR, detection accuracy rate, and time processing.

The Matlab code of this method is provided in Appendix C.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Developing Learning-Based Preprocessing Methods for Detecting Complicated Vehicle Licence Plates

Meeras Al-Shemarry[1, 2] and Yan Li[1]
[1]School of Sciences, Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia
[2] Computer Department, Science College, Karbala University, Karbala 56001, Iraq

Corresponding author: Meeras Al-Shemarry (e-mail: meerassalmanjuwad.al-shemarry@usq.edu.au).

This work was supported by the University of Southern Queensland and Ministry of Higher Education and Scientific Research of Iraq.

ABSTRACT A licence plate detection (LPD) system is an important tool in several roadway traffic applications. This study aims to develop an advanced detection system that works well in complicated scenarios. It proposes a robust preprocessing enhancement method for accurately detecting the licence plates from difficult vehicle images. The proposed method includes the combination of a Gaussian filter, an enhancement cumulative histogram equalization method, and a contrast-limited adaptive histogram equalization technique. The local binary pattern and median filter with histogram of oriented gradient descriptors are used as powerful tools to extract key features from three types of licence plate resolutions. The extracted features are used as input to support vector machine classifier. Processing methods, such as a position-based method are used with the detector to reduce unwanted bounding boxes, as well as false positive values. Four databases consisting of 2050 vehicle images under different conditions are used. Various detection metrics, object localization, and the receiver operating characteristic (ROC) curve are used to evaluate the performance of the proposed method. The experimental results on vehicles databases in several languages, including English, Chinese, and Arabic number plates, show that the proposed method has achieved significant performance improvements. It outperforms the state-of-the-art approaches in terms of both the detection rate and the processing time. The detection rate when trained with 1520 LP images is 99.62% with a false positive rate of 1.675% for complicated images. The average detection time per vehicle image is 0.2408 milliseconds.

INDEX TERMS Histogram of oriented gradient, Licence plate detection, Local binary patterns, Support vector machine.

I. INTRODUCTION

Automatic number plate recognition (ANPR) systems have become a very important tool in many surveilling applications over the past few decades. They are often used as a surveillance technique to identify licence plates of vehicles and are very useful for security systems, highway road tolling systems, traffic sign systems, tracking, and parking management systems [1-5]. The existing systems often work under some standard conditions, such as low-high lighting, rain, and limited day-night lighting. It is still very challenging to identify licence plates (LPs) from complicated vehicle images because of environmental effects.

A robust licence plate detection (LPD) system is desirable to effectively work under all sorts of difficult conditions, such as night, dusk, rain, fog or snow; with images that are blurred, rotated, low-high lighting, distorted, with complex backgrounds and different colors. A number of examples for problematic licence plate (LP) images are shown in Fig. 1.

FIGURE 1. Examples for complicated images of an LP included in the databases.

Some examples for ANPR applications are shown in Fig. 2. As an LPD system is rather difficult, the feature extraction techniques for doing this task should be developed carefully to extract and classify relevant features from regions of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

interest. Several extraction methods are widely used individually or combined together for the LP detection, such as local binary patterns (LBPs) [6], global and local features [7], Haar-like features [8], scale invariant feature transform (SIFT) [9], histogram of oriented gradient (HOG) [10], and so on.

FIGURE 2. Examples of ANPR applications.

The robustness and distinctiveness of the LBP descriptor is usually used to capture the important information that is sensitive to the illumination and rotation conditions [11]. But especially with distorted images, the descriptor needs to be further improved to focus on important information. Since the HOG focuses more on the edge information of an image, it is used widely to detect objects inside an image [10]. Therefore, the HOG features have been used with LBPs to obtain good extraction results. The main contributions of this study are:

1) Developing a new pre-processing method that includes the combination of a Gaussian filter, the enhancement cumulative histogram equalization (ECHE) method, and contrast-limited adaptive histogram equalization (CLAHE) technique to filter out unwanted LP regions to improve the accuracy of the detection system.
2) Improving the work of the HOG descriptor by using a median filter and combined it with the LBP descriptor to produce a powerful feature extraction method for complicated image environments, such as low/high contrast, fogginess, blurriness, rotated LPs, and dark, or complex backgrounds.
3) Increasing the classification accuracy by applying a support vector machine (SVM) and extreme learning machine (ELM) as an effective classifier, separately, with a new updated descriptor MHOG and LBP descriptor.
4) Removing redundant bounding boxes, which increase the false positive rate, using processing methods, such as a position-based method (mean-shift) with detectors.
5) Evaluating the proposed method using several performance measurement metrics and comparing its performance with the newest existing LPD methods.

This paper is organized as follows: Section II reviews the related research work. Section III introduces the proposed method, the details about the HOG and LBP descriptors, and the SVM and ELM classifiers. Section IV presents the databases. Section V shows the experimental results. Section VI presents the comparison with other existing methods. Finally, Section VII describes the conclusions and future work.

II. RELEVANT WORK

The main goal of developing an LPD system is to identify the licence plate number from the regions of interest (ROIs) in vehicle images. Many LPD methods in the literature were proposed. Although the LPD systems have been studied for many years, it is still very challenging to detect LPs from low quality images. Some methods developed depended on a specific color or language, or were limited to fine weather conditions, while others were sensitive to the lighting and complex backgrounds [12-14]. In addition, the angle of camera and the distance constraint make an LPD system less robust [15]. The detection of LPs in hazardous conditions is not easy, especially with complex backgrounds, which often produces a number of non-LP regions. For example, the proposed method by Azam and Islam [16] was not robust for angle invariant nor with distance. The texture-based methods are widely used by many researchers due to the significant texture change in the pixels greyscale level. The support vector machine (SVM) classifier is one of the supervised machine learning algorithms that is commonly used for classification and regression [17]. It can be seen as a type of single layer forward neural network (SLFN), called a support vector network [18]. The output be very good using a SVM if it is used to detect objects with good pre-processing and extraction techniques. The method by Kusakunniran, et al. [19] used a SVM as the machine learning tool, and fed the candidate license plate (CLP) images as the input to the SVM during the training and testing phases. In that study, the SVM was applied directly to the input images without any discriminative features being extracted. It made the detection system very sensitive to noise and geometric transformation. As a result, the detection accuracy was reduced to 80% and the training time was increased. Recently, deep neural networks (DNNs) [20] and convolutional neural networks (CNNs) [21] have been used as an automatic means to learn the key features in LPs. The DNN algorithms can combine the feature extraction and classification into one unified neural network framework. They have shown a higher detection accuracy. However, the features learning mechanism in DNNs cannot guarantee robustness in difficult image conditions, for example, rotation and scaling, unless the training samples can cover various observation conditions. Furthermore, their computational costs during both training and detection processes are expensive due to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

the multi-hidden-layer structure. In [22], a method using a CNN to learn features and an ELM as the classifier was reported. That method obtained competitive results with less computation time compared with those with DNN methods. However, the performances of those methods were compromised with the real conditions of a complex background and changing outdoor light conditions. Therefore, using a combination of textures features with good pre-processing methods to detect the LPs from complicated images can result in a better system performance. With the high speed of vehicles, not only the accuracy but also the computational speed are the key factors for real-time applications.

This paper proposes an efficient framework for improving the performance of an LPD system. It includes a pre-processing method, containing a Gaussian filter, enhancement cumulative histogram equalization (ECHE) and contrast-limited adaptive histogram equalization (CLAHE) methods. This combination works well for all conditions, and is suitable for different LP colors, styles, and languages. It can be applied to enhance any existing ANPR system with different databases. After the pre-processing stage, this study employs two powerful descriptors, HOG and LBP [23, 24]. A median filter was used with HOG (MHOG), descriptor to reduce noise during the extraction stage. The MHOG and LBP descriptors are used to extract several significant features from LP images. Finally, the SVM is used to build the trained model to detect the LP regions and also an ELM classifier was used for evaluation purposes. An English car database which was reported by Al-Shemarry et al. [25] was used in this study. This database contains many complicated vehicle images with different conditions, including low/high lighting, dusk, dirt, fog, and distorted images.

III. PROPOSED METHODOLOGIES

The structural diagram of the proposed LPD methodology is illustrated in Fig.3. It consists of two stages: training and testing. The training stage employs SVM and ELM learning algorithms, separately. Both training and testing phases use the same pre-processing and extraction methods. At the pre-processing stage, the ECHE and CLAHE techniques are applied to enhance the problematic part of vehicles images, while keeping the quality of the normal images during the enhancement process. Through the feature extraction stage, this study carefully selects the effective descriptors, MHOG and LBP, which are suitable for difficult conditions such as under low/high contrast, dirt, dusk, fog, and distortion problems. Finally, the detection stage uses a SVM and an ELM classifier, separately, as trained models to detect the LP region from the tested input vehicle image. The output results are saved for the recognition stage to obtain a complete ANPR system. The details about the proposed method are presented in the next sections.

A. ENHANCEMENT STAGE FOR VEHICLE IMAGES

The proposed framework uses the texture and gray level features to detect the LP regions instead of color features

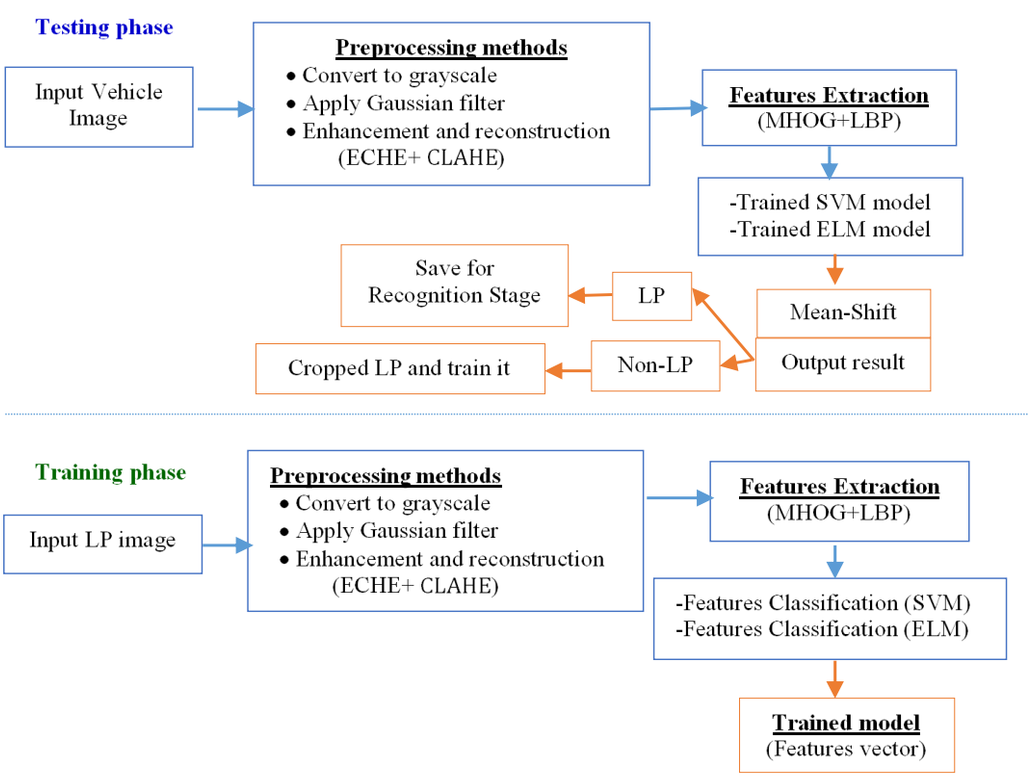


FIGURE 3. The structural diagram of the proposed LPD method for both testing and training phases.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

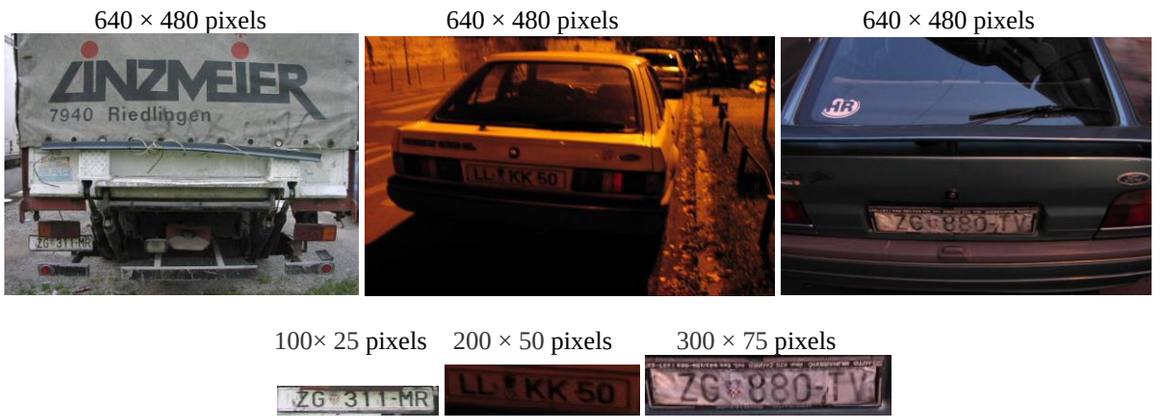


FIGURE 4. **The different resolution for LP in the training dataset.**

which are very sensitive to the illumination and noise problems. The main purpose of this study is to detect the LP numbers under complicated conditions, for example under different illumination. Most previous studies converted the color images into grayscale ones for RGB reduction [26]. According to Saravanan [27] a color image IMRGB (i, j) has M × N dimensions. Where M is the height or the number of rows and N is the width or the number of columns in the image, and $0 \le i \le (M-1)$ and $0 \le j \le (N-1)$. IMRGB (i, j) was converted to a grayscale image, Gray (i, j), by Eq. (1) [27] :

$$Gray\,(i,j) = 0.2989 \times R\,(i,j) + 0.5870 \times G\,(i,j) + 0.1140 \times B\,(i,j) \quad (1)$$

where R(i, j), G(i, j), and B(i, j) are the three channels of colors, red, green and blue, respectively.

A large image resolution needs more time processing. In this study, an English car database has a consistent resolution of 640×480 pixels. In this study, a sliding window (M×N) will be used to scan an image from M×N resolution instead of scanning the whole image. The sliding window will be started to detect LP regions from M = 200 and N = 30. This process leads to reduce the processing time and obtain better detection results. As shown in Fig. 4, the LPs in the training dataset have three resolutions, 100×25, 200×50, and 300×75 pixels.

Referring to non-LP regions, there are various sources of noise along with the text in a vehicle image, such as surface textures, distortion, dusk, low/high lighting, and dirt. Those noises increase the bounding boxes during the detection stage.

In this study, we propose an enhancement pre-processing algorithm to reduce noise and improve the lighting conditions for complicated images without affecting on the quality of the images in normal conditions. The steps of the ECHE combined with the CLAHE algorithm are as follows:

1) Apply a Gaussian filter with the standard deviation σ= 0.25,
2) Apply the cumulative histogram equalization (CHE) method,

$$P_x(i) = P(x = i) = n_i/n \ , 0 \le i < L \quad (2)$$

where L is the total number of grayscale levels in an image which is typically 256 and n is the image pixels. $P_x(i)$ is the image's histogram of the pixel value i, which is normalized to [0, 1].

3) Calculate the histogram of the cumulative distribution function (CDF), which is also an image accumulated normalized histogram and defined as

$$CDF_x(i) = \sum_{j=0}^{i} P_x(j) \quad (3)$$

4) Calculate the new values of the histogram through the general histogram equalization formula,

$$CDF_y(i) = iC \quad (4)$$

where C is a constant in the range of [0-L] , which is also needs a linearized CDF across the new value range y.

5) Apply the contrast-limited adaptive histogram equalization (CLAHE) method with the standard deviation σ= 0.02, and
6) Build a new enhanced image by replacing each gray value in the image with the new gray values.

The performance differences between the CHE, CLAHE, and the proposed pre-processing methods (ECHE+CLAHE) can be observed on Figs.5 (a), (b), and (c), respectively. From Figs. 5(a), (b), and (c), note that the proposed pre-processing method reduced the range of feature dimensions. Some results applied on testing vehicle images by the algorithm are shown in Fig. 6.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

The grayscale image histogram has 256 bins by default. It represents the distribution function of image intensities. The *imhist* function in Matlab shows the histogram plot *(X, Y)*, where *X* is the spaced bins, and each bin represents the range of feature values, *Y* is the number of pixels within each range. From the Fig. 6, we can observe the results by the enhancement method for different types of both, clear and complicated vehicle images. Using the histogram information helps decrease the value ranges of the unwanted features. The next subsection presents more details about the strong descriptors used in this study.

B. THE HOG AND LBP DESCRIPTORS

This study uses two powerful descriptors, HOG and LBP, to extract key features from complicated LP images. They are used in this work for many reasons as described in this paper.

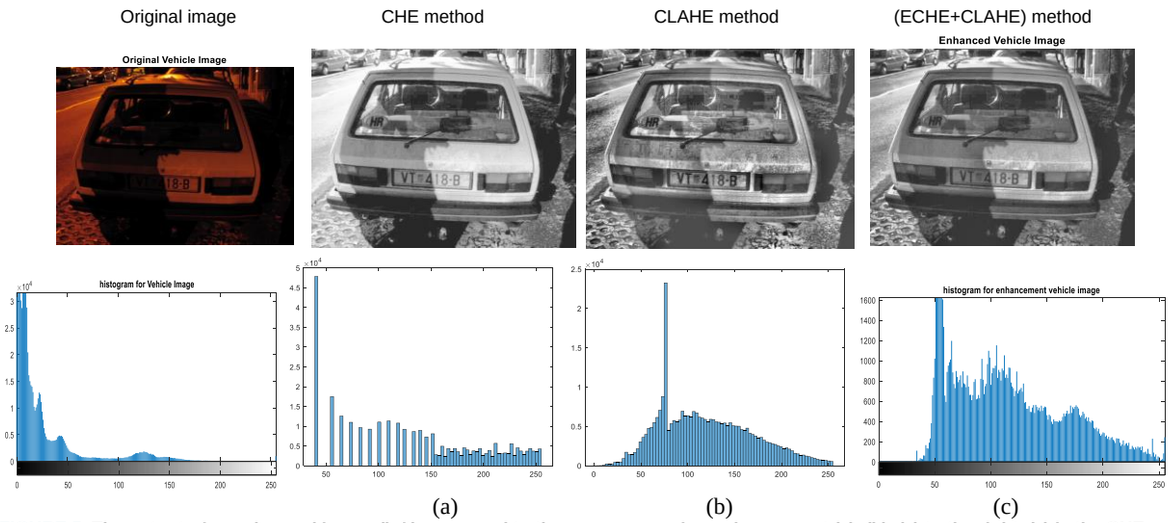


FIGURE 5. The output of unenhanced image (left) compared to three preprocessing enhancement (a), (b), (c) to the right, (a) is the CHE method, (b) is the CLAHE method, and (c) is the proposed preprocessing method (ECHE+CLAHE) (X axis = the range of features' values in each bin, Y axis = the number of features' values appearing in each bin range).

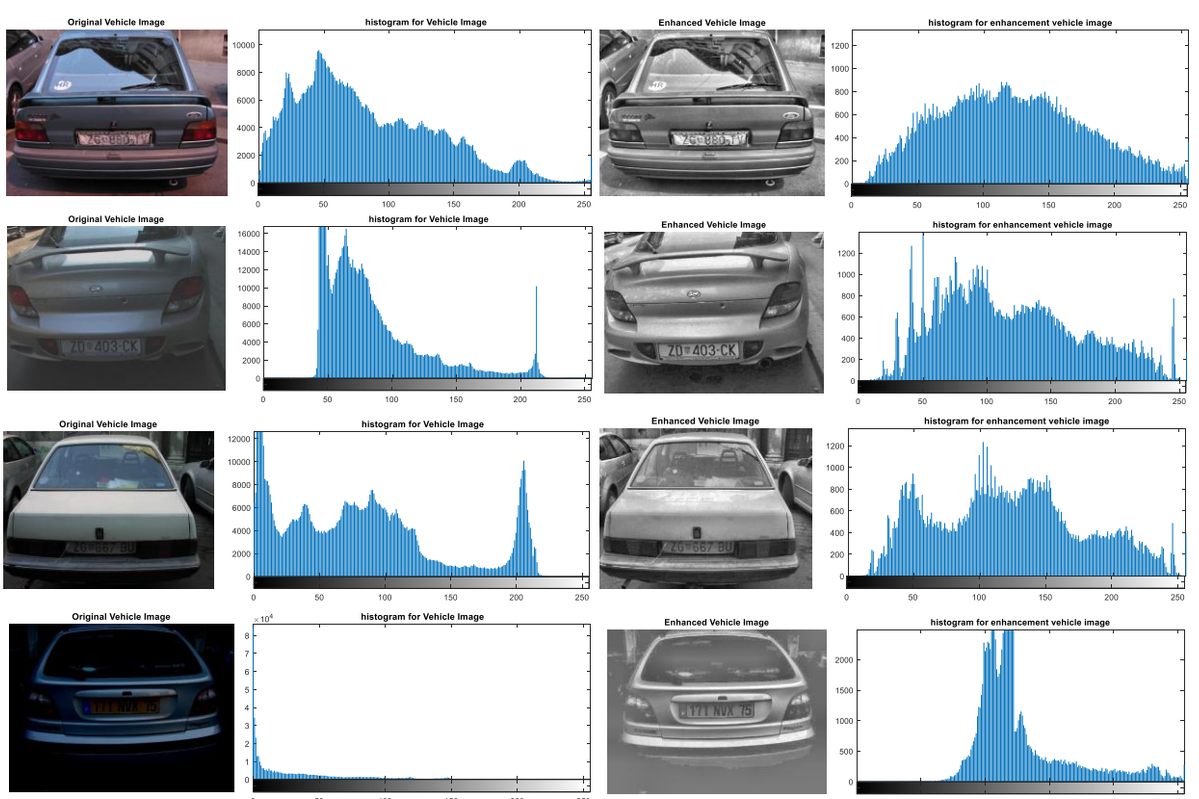


FIGURE 6. The output with the histogram for the testing complicated vehicle images in the dataset. The histogram displays how the enhancement algorithm works (X axis = the range of features values in each bin, Y axis = the number of feature values appeared in each bin range).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

The description about the extraction descriptors is given in this section.

1) The HOG descriptor

The HOG descriptor is one of the common imaging descriptors in the area of the computer vision [28]. It delivers good results in different computer vision applications, such as face detection [29], vehicle detection [30], and text extraction [31]. In addition, it is not sensitive to lighting changes and small offsets. It can effectively describe the edge features of an object. Therefore, it provides a rough estimation of the object appearance and shape. The steps for the extraction of the HOG features were as follows:

a. Used the median filter with HOG and assumed that the input to the MHOG descriptor is a window *G* from the enhancement LP grayscale image. The first step for MHOG is to divide *G* into non-overlapping blocks of 8 × 2 pixels. Each block is divided into small regions or called cells (8 × 8 pixels). The cells are combined into adjacent blocks of 2 × 2 cells, and we concatenate those four cells histogram into one block features. The horizontal and vertical gradients are obtained for each pixel inside the cell. The simplest technique to do that is by using the 1D Sobel operators ([-1, 0, 1] and [-1, 0, 1] T) [32]:

$$G_x(x, y) = G(x+1, y) - G(x-1, y) \quad (5)$$
$$G_y(x, y) = G(x, y+1) - G(x, y-1) \quad (6)$$

where G_x (x, y) is the horizontal gradient, and G_y (x, y) is the vertical gradient. x and y are the row and column indexes, respectively.

b. After that, the gradient is transformed into the polar coordinates of *x* and *y* directions with the angle set to between 0 and 180 degrees. The gradient magnitude μ and the direction of pixel θ are calculated according to Eqs. (7) and (8).

$$\mu(x, y) = \sqrt{G_x^2 + G_y^2} \quad (7)$$
$$\theta(x, y) = \frac{180}{\pi}(\tan_2^{-1}(G_x, G_y) \bmod \pi) \quad (8)$$

where $\tan_2^{-1}$ is the inverse tangent for the quadrant, which produces values between $-\pi$ and π.

c. The MHOG histogram in each cell is computed, and all the values are joined into 9 bins, meaning the cell is divided into nine gradient directions from 0° to 180° orientations. In this way, we can gain different gradient orientations due to different contrasts among images. Therefore, the block gradient histogram should be normalized. The block normalization technique is a mid-solution for changes in illumination conditions. The cell histograms need to be normalized to reduce the contrast changes between the images for the same object. The normalization process can be done on the histogram vector by using L1-norm or L2-norm. The L1-norm provides lower reliability than L2-norm [28]. This study uses the L1-norm for the normalization of the MHOG features vector.

d. Finally, we collect the MHOG for all overlapping blocks features in the detection window, and combine them into a final MHOG features vector for classification. The steps of generating the MHOG descriptor from an LP image are shown in Fig. 7. The extracted features' histograms are used as the input to the SVM and ELM classifiers, separately, to build the final LP detector.

2) The LBP descriptor

The second powerful descriptor used in this study is the LBP. It is employed to extract different features for several reasons [24]. The LBP is effective for different illumination

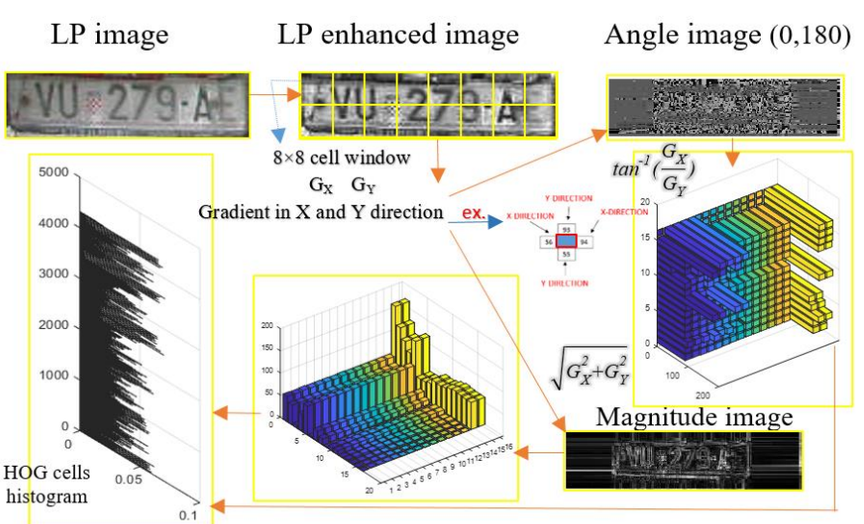


FIGURE 7. Steps of generating the MHOG descriptor for a LP image patch.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

conditions and can solve the scale invariance and occlusion problems. The first LBP descriptor was presented by Ojala et al. [24] . The study divided the trained image into cells and labelled the pixels in each cell using a 3 × 3 window, then selected the center pixel value of the cell as a threshold. The center pixel value was then compared with the gray values from neighboring 8 pixel cells. If it was smaller than the neighbouring pixel value, the location of the pixel was marked by 1, otherwise 0. Therefore, the 8 neighboring values in the 3×3 window could produce 8 binary numbers. The 8 binary numbers are usually converted to the decimal numbers which are the LBP code. In total there were 256 grayscale values. The LBP values of the window for the central pixel are utilized to reflect the texture information in the region (see Fig. 8). The LBP code value of the center pixel is calculated by Eq. (9):

$$LBP(x_c,y_c)=\sum_{n=0}^{n-1} f(p_n-p_c)2^n \qquad (9)$$

where p_c is the brightness value of the center pixel value (x_c,y_c), p_n is the brightness value of the n point in the i neighbouring domain. $f(x)$ function is defined as:

$$f(x)=\begin{cases}1, & x\geq 0\\ 0, & x<0\end{cases} \qquad (10)$$

For example, the pattern of 11001111 includes more than two transitions. Therefore, it is called uniform patterns. The single LBP descriptor can produce much fewer uniform patterns without loss of useful features. After the LBP operator labelled image, $f_l(x, y)$ has been obtained. The histogram of the LBP can be defined as:

$$H_i=\sum_{x,y} I\{f_l(x, y)=i\},\ i=0, \ldots, n-1, \qquad (11)$$

in which n is the number of labels that are produced by the LBP descriptor, and $I\{f(x)\}$ is 1 if $f(x)$ is true and 0 if $f(x)$ is false.

The features from each LBP region are concatenated into a maximum pooling features histogram vector. The steps for generating the LBP descriptor from an LP image are shown in Fig. 8. Finally, the extracted features histograms are used as the input to the SVM or ELM classifier to build the final trained LP detector. The LBPs are useful to remove the unwanted regions or noise from images [33].

C. FEATURE SELECTION USING SVM AND ELM CLASSIFIERS

Despite most of the learning algorithms being able to do the same job, their performance is heavily dependent on the pre-processing and extraction methods used. In this research, a deep learning classifier, such as a CNN, was also used. But it did not yield good classification results, especially with the HOG descriptor. A CNN classifier normally requires a fixed-resolution for input images [34], while there are three resolutions included in our training dataset. Thus it is not easy to conclude which learning algorithm is better than others. In this study, two popular classifiers were used, SVM and ELM, to evaluate the performance of the proposed method. All the extracted features using the MHOG and LBP procedures are fed to the classifiers separately, and build an ensemble of strong LP detectors to detect different LP features. To make the trained model computationally efficient in terms of the processing time and accuracy, most discriminative features should be extracted, and redundant information and noise are removed. This paper applies a SVM or ELM classifier to train and classify the extracted features. The descriptions of the classifiers are given in this section.

1) *Support Vector Machine (SVM)*

The SVM is one of the supervised machine learning algorithms, that is commonly used for classification and regression [17]. It can be seen as a type of artificial intelligence network, called a support vector network [18]. The main objectives of a binary SVM is to separate the

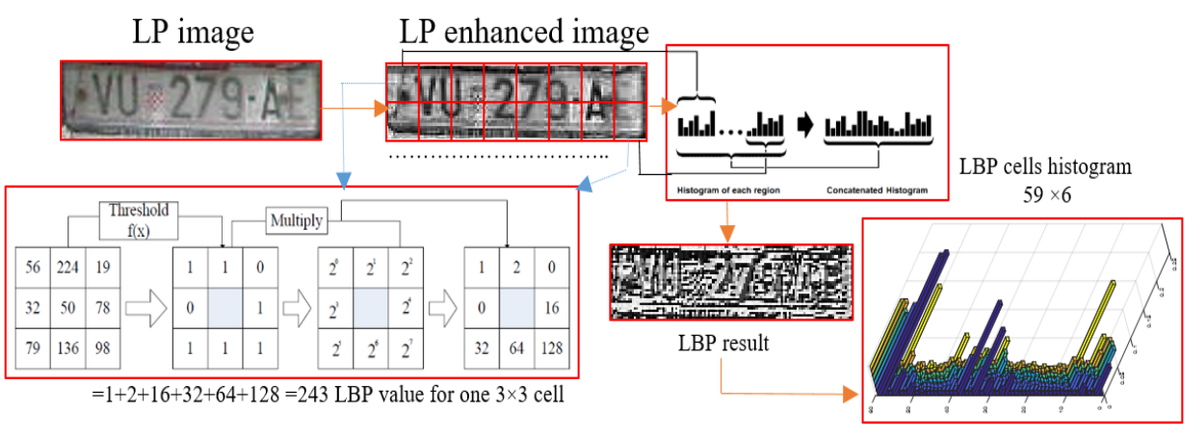


FIGURE 8. Steps of generating the LBP descriptor for LP images.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

margins in the feature space of the two different classes α_i, positive and negative (see Fig. 9).

In general, linear functions are utilized as a separating hyperplane in the feature space x_N. It is used in pattern recognition and object detection for the generalized linear classifier. Then, it gives a decision surface maximized by employing an optimization approach. A kernel function K (x_N, x), such as linear, nonlinear, gaussian kernel, sigmoid, polynomial, radial basis function etc., is used to achieve better classification performance. When using a kernel function K, the scalar output $y_N \alpha_N$, can be implicitly calculated in the kernel feature space N with a threshold bias value b. Refer to the references in [17] for more details about the SVM classifier.

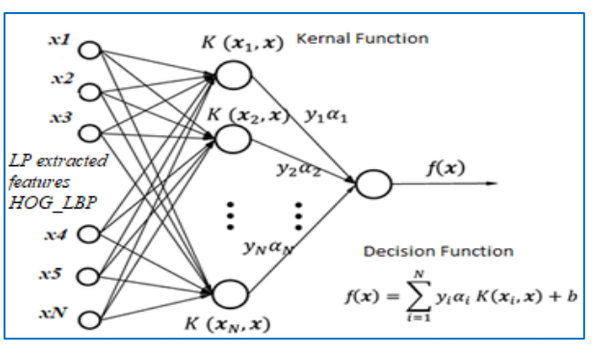


FIGURE 9. The workflow of the support vector machine network.

2) Extreme Learning Machine (ELM)

Although the neural network and support vector machine are widely used [35, 36] , those types of algorithms have a low learning speed for a reasonable classification accuracy [37]. The ELM is a single hidden-layer feedforward neural network (SLFN), initially proposed by Huang et al [38]. It can be viewed as a variant of a random vector functional-link (RVFL) network classifier [39] without direct links and bias terms. It ignores the need for tuning parameters in the training phase and hence minimizes the training time compared to the traditional neural networks. The ELM workflow is illustrated in Fig. 10, where the network input *(IW)* vector obtained from the input extracted features of the HOG and LBP multiplied by the input weight *IW* matrix. The *IW* and the hidden-layer bias *b* values are randomly assigned, while the output weights *OW* are analytically calculated. After that, the results are added to the bias vector *b* to create the input to the *"sigmoid"* activation function and produce a good output layer. The *b* value as the threshold helps the input layer to decide the activation of neurons and increases the flexibility of the training model. Without bias, the neurons do not pass to the other network layers. Finally, the output of the *"sigmoid"* function is multiplied with the network output weight *OW* matrix to produce the final results y_m or decision function. Refer to reference [40] for more details about the ELM classifier.

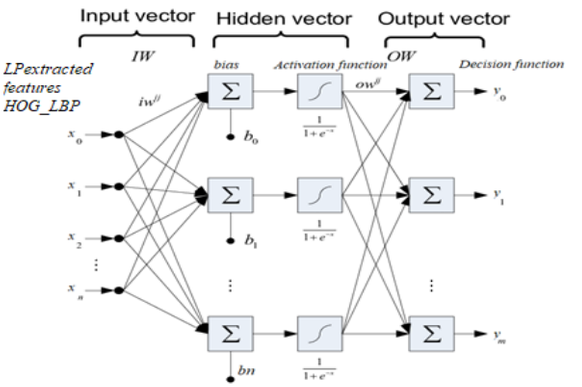


FIGURE 10. Extreme learning machine workflow.

D. THE OVERLAPPING AND REDUNDANT BOUNDING BOXES

In this study, a sliding window is applied to scan vehicle images. The MHOG and LBP features are extracted at each stage of the sliding window. After that, the trained classifiers, SVM or ELM, are applied to detect the LPs. If the classifier detects the LP, the bounding box records the region of interest. When the scan process is completed on the whole testing image, many bounding boxes are detected around the LP region in the vehicle image. Therefore, a suppression technique is applied in order to remove the overlapping and redundant bounding boxes from the detection area. Fig. 11 shows an example of the overlapping bounding boxes problem with both classifiers, SVM and ELM.

As shown in Figs. 11 (a) and (b), the vehicle image has six and 15 overlapping bounding boxes with SVM and ELM, respectively. This is an open problem, no matter which detection method is used, whenever the LP area was correctly detected. Here all those redundant boxes refer to the same LP, a technique to suppress the smaller bounding boxes and keep the larger bounding boxes is required, as shown in Figs. 12 (a) and (b).

There are many ways to solve the overlapping bounding boxes problem. A position-based method, or mean-shift, is a generalized as the way to detect an object by searching the candidate objects which have the highest similarity with the detected one [41] . In this study, the proposed method applies the mean-shift algorithm to reduce the similar candidate objects or overlapping bounding boxes for LPs. It is used to capture multiple regions in the space of the bounding boxes by utilizing the coordinates of the redundant boxes, *(x, y)*, as well as the current scale of the tested image (logarithm). A mean-shift tracking technique is used to track the detected LP in a vehicle image. The process of the mean-shift method is shown in Fig. 13.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

FIGURE 11. (a) The output of the proposed method using the SVM without the mean-shift algorithm; (b) The output of the proposed method using the ELM without the mean-shift algorithm.

FIGURE 12. (a) The output of the proposed method using the SVM with mean-shift algorithm, (b) The output of the proposed method using the ELM with mean-shift algorithm.

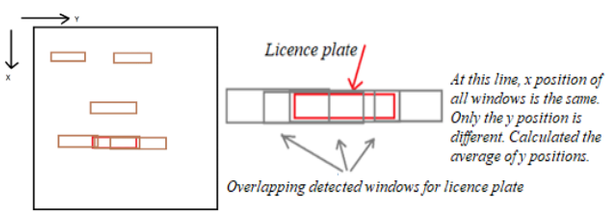


FIGURE 13. Steps of the mean-shift algorithm.

This method starts with searching around the overlapping windows of the LP region. The windows have the same *x* coordinate as the LP frame. Then, the average of *y* coordinates which have different positions to the start point *x* is calculated and removed. The results from using this algorithm are very satisfactory, with low false positive values and higher accuracy rates. Also, this algorithm relies on the effective classifier of the SVM or ELM that is selected to produce good results. Refer to reference [41] for more details about the mean-shift algorithm.

IV. DATABASE

The proposed method is tested on two groups of databases: 1) An English car database including 510 vehicle images under different conditions [42]. 2) An extended English car database by Al-Shemarry et al. [25] covering 1540 vehicle images under complicated conditions, such as too dark, blurry, distortion, and low/high contrast environments (see Fig. 14). The second database is an extension of the first English car database as Al-Shemarry et al. [25] changed the lighting, blurry, distortion conditions using online photo editor application to make it more challenging [43]. From the two databases, 530 vehicle images were randomly selected for the testing and 1520 were used for the training LPs dataset. The total number of vehicle images is 2050. In addition, this method is also applied to different vehicle images with Arabic or Chinese language on the licence plates. The vehicle images are downloaded from the Internet and resized into 480 ×640 resolution and are used to evaluate the performance of the LPD system as shown in Fig. 15.

The extended database also includes the rotated images with different LP angles, such as 45˚, 30˚, 20°, 15°, and 5˚. Moreover, the training dataset contains various illumination

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

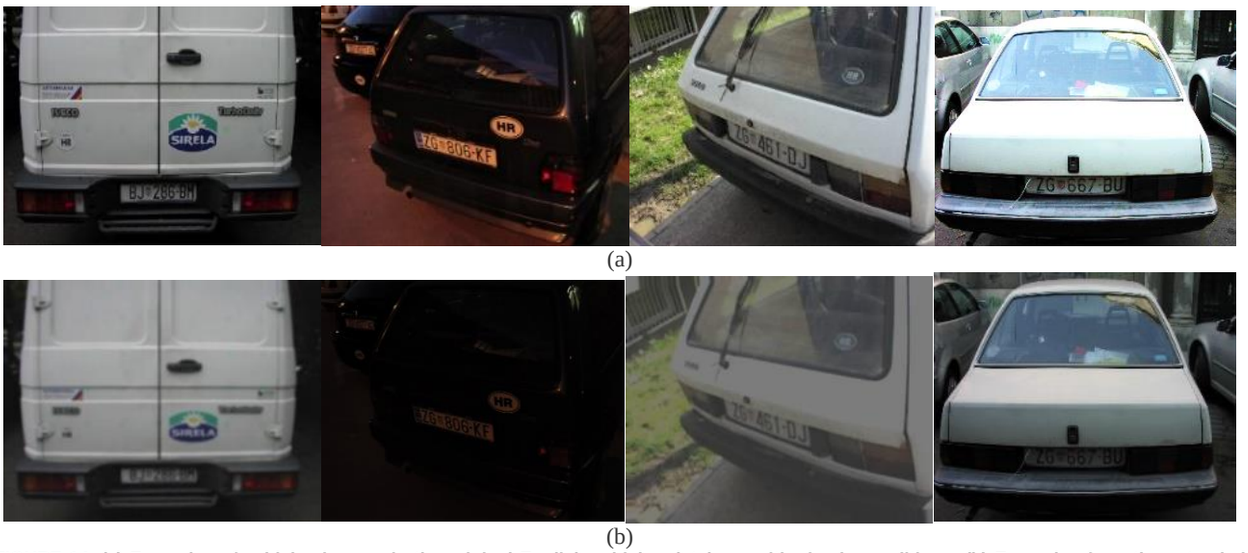


FIGURE 14. **(a) Examples of vehicles images in the original English vehicles database with simple conditions; (b) Examples from the extended English vehicles database after photo editing.**

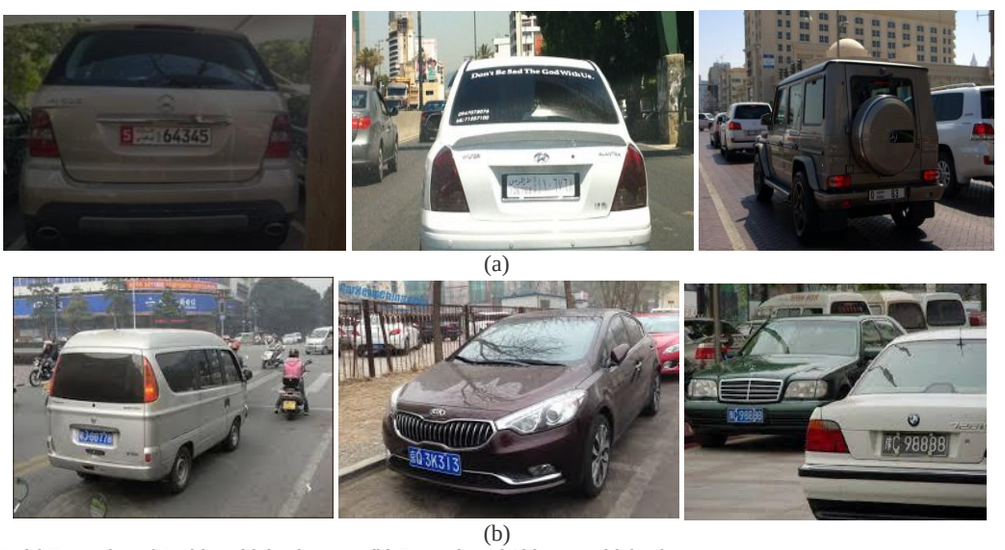


FIGURE 15. **(a) Examples of Arabic vehicles images; (b) Examples of Chinese vehicles images.**

FIGURE 16. **Examples of LPs training images under simple and complicated conditions.**

conditions with three LPs resolutions of 100×25, 200×50, and 300×75 to enable the LPD system to capture LPs from complicated or low-quality images. Fig. 16 shows some of the training LPs images under difficult and simple conditions. The experimental results showed that all the key characteristics have been captured from the 1520 training images.

V. EXPERIMENTS AND RESULTS

All the experiments by the proposed method are implemented on a computer with 3.4 GHz Intel Core i7-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

4770, 16GB of RAM using MATLAB, version R2017b. In this section, the descriptions of MHOG and LBP features analysis and features selection processes by SVM and ELM classifiers are firstly presented. Secondly, the performance evaluation of the proposed method is evaluated using the detection and object localization metrics. Thirdly, the performance comparison using the receive operating characteristic (ROC) curve between the SVM_MHOG_LBP and ELM_MHOG_LBP is provided. Finally, the detection accuracy and time efficiency of the proposed method are compared with the newest detection methods that also used the same database [25, 44, 45] as the one used in this study.

A. FEATURE ANALYSIS AND SELECTION

Feature analysis is a very important step for any classification problems. This study combines the MHOG and LBP features into a features vector. The SVM and ELM classifiers are used separately, to detect LPs from vehicle images. With the MHOG features, each bin includes the feature magnitude for 8×8 cells for a specific direction. High feature values indicate the most discriminative bin. The complete MHOG operator contains the magnitude values in all 180 directions. Therefore, it is a high value around the 20, 60, 100, 140, and 180 degrees. For the LBP descriptor each bin contains features values for 3×3 cells. The extracted features for the three LP resolutions of 100×25, 200×50, and 300×75 using the MHOG and LBP descriptors are shown in Fig. 17.

The dimensions of the extracted feature values for the three resolutions above are 1517, 1784, and 1582, respectively. The linear SVM and the kernel ELM classifiers were applied to select strong subset feature values. The information about features selection for both classifiers are displayed in Table 1.

In this study, the linear kernel function for the SVM was used for learning the classes of LPs and non-LPs. The SVM classifier was tested with 5-fold cross validation for three dimensions features values as well as three LPs resolutions of 313, 547, and 348, respectively. From Table 1 the training dataset includes three detectors or training models with different dimension features values. Each SVM trained model with 5-fold cross validation contains five ensemble classes for LPs features. The weighting values for three SVM training models are 6.591, 5.605, and 6.321, respectively, with different bias values. The bias is a special parameter in the SVM. The classifier without this value would always go to the origin. Also, the SVM does not give the separating hyperplane for the maximum features margin without the bias term. The ELM was tested with 550 input hidden neurons, and the input neurons for the three LP resolutions are 640, 851, and 1202, respectively. The ELM classifier produced three training models for the three LP resolutions with a training accuracy rate of 100%.

TABLE 1. The dimension of training models using SVM and ELM classifiers.

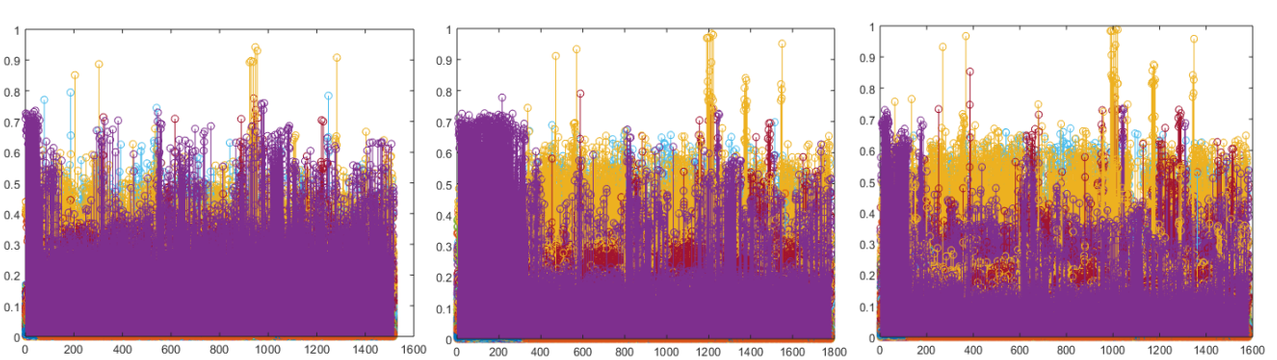
Method	TM1 (100×25)	TM2 (200×50)	TM3 (300×75)	Average TA
SVM_MHOG_LBP	features	features	features	99.786%
Class 1	62	110	71	
Class 2	64	111	71	
Class 3	61	105	66	
Class 4	62	112	72	
Class 5	64	109	68	
Total features	313	547	348	
ELM_MHOG_LBP	*640*	851	1202	100%

B. PERFORMANCE EVALUATION

The proposed method is evaluated using detection and object localization metrics.

1)Detection Metrics for LPD

Several assessment measures are used to check the performance of the proposed detection system. The assessment metrics include the number of objects that are correctly detected, falsely detected, or miss-undetected by the system [46]. Also, the detection based metrics are used to evaluate the system under test (SUT) performance. For those metrics, all the objects are validated to see if there is a matching between SUT and the ground truth (GT). The detection metrics used in this study to evaluate the LPD system are as follows:



(a) (b) (c)

FIGURE 17. (a) The output of the extracted MHOG_LBP features for an LP with 100×25 resolution; (b) The output of the extracted MHOG_LBP features for an LP with 200×50 resolution; (c) The output of the extracted MHOG_LBP features for an LP with 300×75 resolution (For histogram: X axis = the range of features values in each bin, Y axis = the features values appearance in each bin range).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

- FP (false positive): means that the LP is detected by the SUT, but it is not in the GT.
- FN (false negative): indicates that the LP exists in the GT, but it is not detected by the SUT.
- TP (true positive or correct detection): means that the LP exists in both the GT and the SUT.
- TN (true negative): indicates that the LP does not exist in the GT and by the SUT.

We added in the car English car database 50 vehicle images without the LP that means the TN which refers to the vehicle image without the LP object inside it. This is a very important step to validate the detection system performance.

TABLE 2. The detection results for the proposed method using the SVM and ELM classifiers.

Method	No. of Testing Images = 530, TN= 50 Images					
	FP	TP	FN	TN	DR	AR
SVM_LBP	15	483	47	40	91.13%	89.40%
SVM_MHOG	25	478	52	44	91.57%	87.14%
SVM_MHOG_LBP	**9**	**528**	**2**	**48**	**99.62%**	**98.12%**
ELM_LBP	14	480	50	42	90.56%	89.07%
ELM_MHOG	21	473	57	39	89.24%	86.77%
ELM_MHOG_LBP	**13**	**515**	**15**	**43**	**97.16%**	**95.22%**

AR: Accuracy Rate

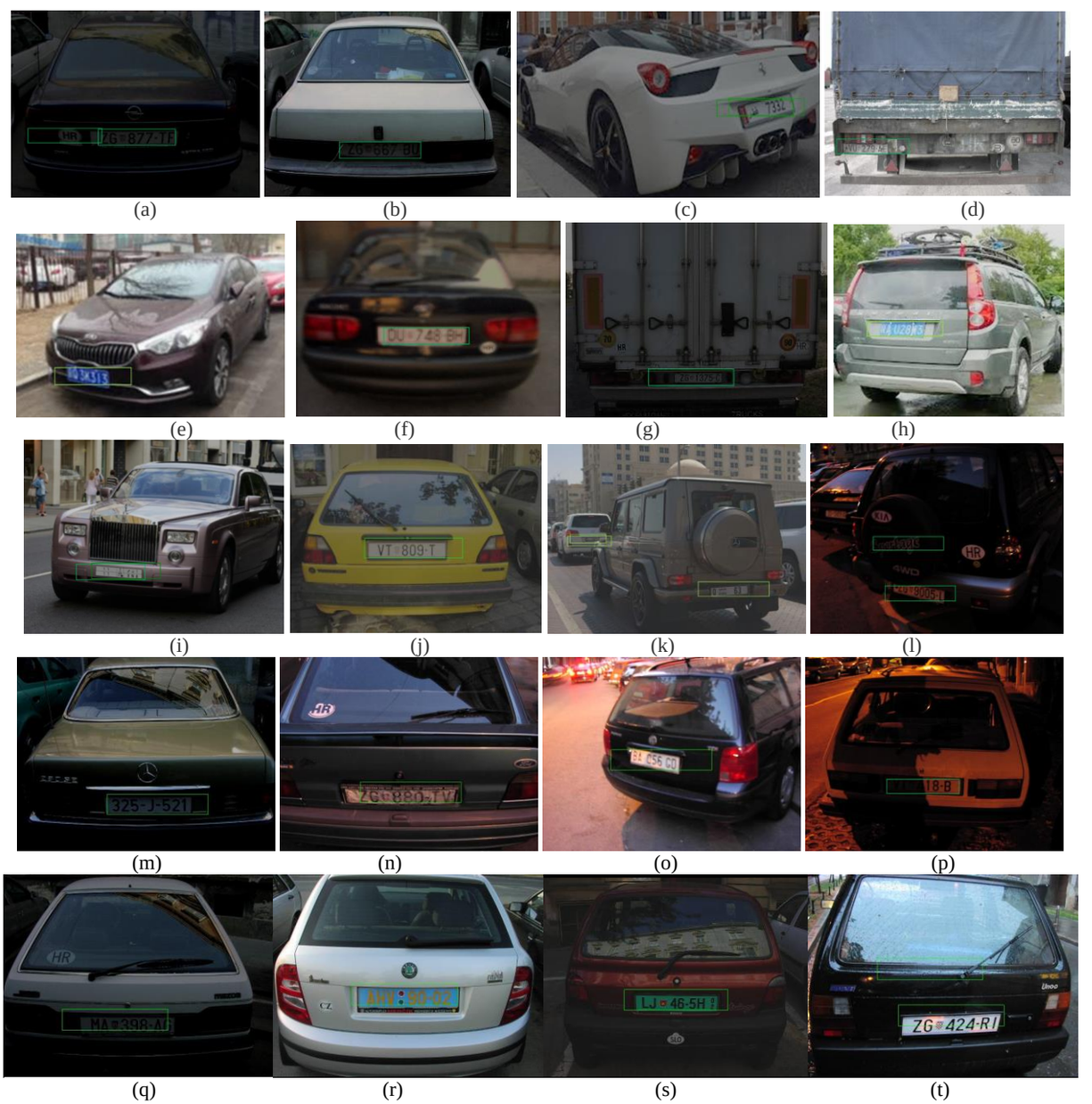


FIGURE 18. Examples of the successful detection results by the proposed method for complicated cars images with low light and dirt (a), (b), (d), (e), (q); with various views points (c), (h), (i), (k), (l), (o); with dusk and fog (a), (e), (f), (i), (m); with distortion (d), (e), (k), (p), (n), with different color (h), (r), (s), (t), and with low/high contrast (f), (l), (m), (o), (p), (r) problems.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

The detection and accuracy rates can be calculated as follows:

$$Detection\ rate\ (DR) = TP/(TP+FN) \quad (12)$$

$$Accuracy = (TP+TN)/(TP+TN+FP+FN) \quad (13)$$

The *DR* is the proportion of the true positive result that is truly predicted as a positive result by the detector, while the accuracy rate is the proportion of the true results for both positive and negative LP objects in the *GT*. The detection results are shown in Table 2.

From Table 2, it is noticeable that the combination of the extracted features from both the MHOG and LBP improves the detection results. The SVM_MHOG_LBP method outperforms other methods in terms of the detection accuracy rates. Some examples of the detection results for the proposed method are shown in Fig.18.

It was observed that all LPs were detected with very low FPs values, and fewer redundant bounding boxes appeared under difficult conditions. Also, some FPs were noticed when some objects in the vehicle image looked like an LP (for example, commercial signs and the vehicle logos).

2)Object Localization Metrics

The recall (RR) or detection rate (DR), precision (PR), and F-measure (F-m) rates [47] at a matching confidence of $\delta \geq 0.5$ are used to evaluate the LP localization performance. Those metrics are defined as follows:

$$RR = TP/(TP+FN)\ when\ \delta \geq 0.5 \quad (14)$$

$$PR = TP/(TP+FP)\ when\ \delta \geq 0.5 \quad (15)$$

$$F\text{-}m = 2(PR \times RR)/(PR+RR) \quad (16)$$

PR is the proportion of the actual negative objects that the detection system predicted correctly as negatives [47]. The matching confidence assumption δ is defined as follows:

$$\delta = \{Bb \cap GT/Bb \cup GT, GT \subseteq Bb\} \quad (17)$$

where *Bb* denotes the predicted area of the LP bounding box. The detection performance is analyzed with the confidence $\delta \geq 0.5$. It means that *Bb* encloses to the related *GT* while the area of the former is twice of the latter. If δ is too big, the predicted *Bb* would be much larger than the *GT*, which does not make any sense to the LP detection ratio. The confidence value determines directly from the strong classifier. Table 3 shows the object localization metrics of the proposed method.

TABLE 3. The performance results by the proposed LPD system.

Method	RR	PR	F-m
SVM_LBP	91.13%	96.98%	93.96%
SVM_MHOG	91.57%	95.02%	93.26%
SVM_MHOG_LBP	**99.62%**	**98.32%**	**98.96%**
ELM_LBP	90.56%	97.16%	93.74%
ELM_MHOG	89.24%	95.74%	92.66%
ELM_MHOG_LBP	**97.16%**	**97.53%**	**97.34%**

We can observe from Table 3 that the results of object localization metrics for *SVM_MHOG_LBP* are better than other methods' results.

C. THE RECEIVE OPERATING CHARACTERISTIC (ROC) CURVE

This study uses a combined features vector extracted by the HOG and LBP, and employs a SVM or an ELM classifier to detect the LPs from complicated vehicle images. The efficiency of the proposed method is evaluated using the receiver operating characteristic (ROC) curve. The ROC curve is a useful tool for organizing the work of the classifiers and displaying their quality of the performance [48]. It is known as a performance metric for comparing and evaluating algorithms [48, 49]. The ROC curve depends on four parameters, true positive rate (*TPR*) or *RR* rate (see Eq. 9), false positive rate (FPR), positive predictive value *(PPV)* or *PR* rate (see Eq. (15)), and negative predictive value *(NPV)*. Those parameters are defined as follows:

$$FPR = FP/(FP+TP) \quad (18)$$

$$NPV = TN/(TN+FN) \quad (19)$$

The ROC results from the SVM and ELM classifiers are reported in Figs. 19, 20, and Table 4.

The ROC curve for the SVM with MHOG_LBP features is better than that for the ELM with MHOG_LBP. The TPR and FPR from the SVM_MHOG_LBP are improved by 2.46% and 0.787%, respectively, in comparison to those by the ELM_MHOG_LBP. The area under curve (AUC) values are between 0 and 1.

The high value of the AUC is a better measure for evaluating the performance of the proposed method than the accuracy rate [49]. The AUC for the SVM_MHOG_LBP is better than the ELM_MHOG_LBP of 99.99% and 99.20%, respectively.

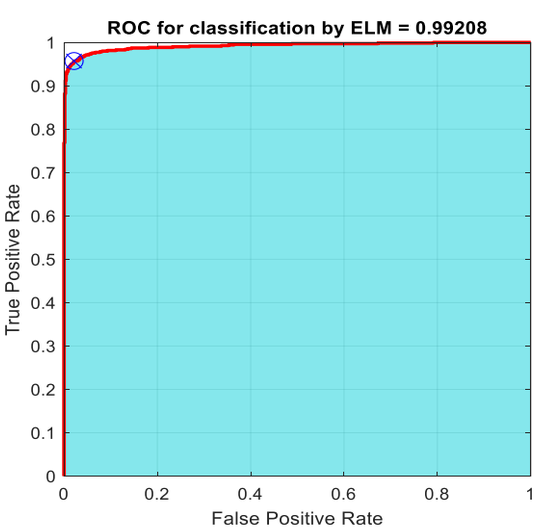


FIGURE 19. The ROC curve for the MHOG_LBP features classification using the ELM.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

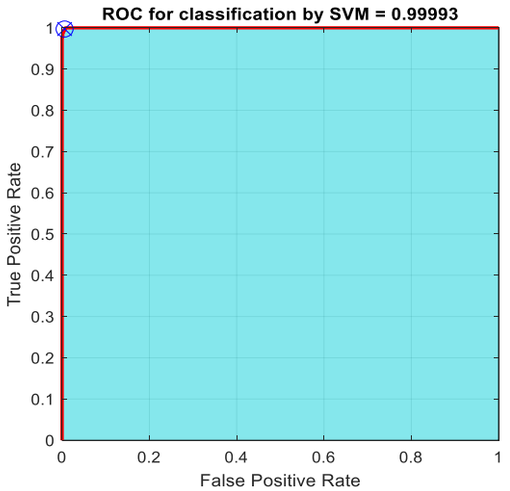


FIGURE 20. The ROC curve for the MHOG_LBP features classification using the SVM.

TABLE 4. The performance results from SVM and ELM classifiers with the MHOG_LBP features, using ROC curve parameters.

Method	SVM_HOG_LBP	ELM_HOG_LBP
AUC	99.99%	99.20%
FPR	1.67%	2.46%
TPR	99.62%	97.16%
PPV	98.32%	97.53%
NPV	96%	74.17%
CA	99.78%	96.92%

CA: Classification Accuracy

D. RUN TIME

The runtime is a key indicator of a system performance. Table 5 shows the average running time per vehicle image for the total and the three stages of preprocessing, extraction and detection by the proposed LPD system. The implementation time for the SVM_MHOG_LBP was much shorter than that for the ELM_MHOG_LBP method under both ordinary and complicated conditions. The proposed method makes the LPD system reliable for real-time applications through removing the unwanted bounding boxes and decreasing the running time.

TABLE 5. The average run time for the proposed method for each stage in the SVM_MHOG_LBP and ELM_MHOG_LBP schemes.

Stages	Proposed method	
	SVM_MHOG_LBP	ELM_MHOG_LBP
Pre-processing	0.0519*ms*	0.0519*ms*
Extraction	1.7260*ms*	1.7260*ms*
Detection	0.4408*ms*	1.0089*ms*
Total test time	2.2187*ms*	2.7868*ms*
Training Time	201.835*ms*	116.087*ms*

VI. COMPARISON WITH EXISTING METHODS

In this section, the proposed method is compared with the state-of-the-art license plate detection methods that used the same database of English car plates [25, 44, 45, 50]. The performances are compared using the measures of the *RR* and *PR* rates as well as the F-m rate. Azam and Gavrilova [45] reported a genetic algorithm depending on the HOG features with a mixture of binary classifiers to classify LP and non-LP images. To improve the classification performance, the genetic algorithm was applied to select the best features subset. The method used a mixture of binary classifiers of k-nearest neighbor, SVM, decision tree, and linear discriminant analysis. It achieved some good results under different conditions, but it required a high-contrast preprocessing method to improve images. This leads to increase the false positive rate. Raghunandan, et al. [44] proposed a mathematical model using a Riesz fractional operator to enhance the details of LP edges' information. That method worked well as long as there was a clear LP shape in the image. It was not robust for distorted images and it was time-consuming with a low *RR* rate. Yousif, et al. [50] presented a novel methods based on genetic algorithm (GA) to identify LPs under limited conditions. Henry, et al. [51] designed a deep ALPR system to identify multinational LPs. Their proposed system included three steps: LP detection, LP recognition, and multinational layout LP detection. It achieved good results under good conditions but it was not robust or perform satisfactorily for complicated vehicle images. Also, Al-Shemarry, et al. [25] produced a new and efficient descriptor, with multi-level extended local binary patterns (MLELBP) to extract difficult features' using three neighbouring feature dimensions under complicated conditions. The input image was resized to make the proposed method work without any limitations related to standard vehicle image resolution and save the processing time. The ELM was used to build the trained model and obtain good detection results. In this method, the multi-levels preprocessing in the extraction stage leads to getting high feature dimensions which causes increases in the false positive rate. In addition, the resizing image process is not always successful especially with distorted images that could lead to loss of important feature values. Therefore this study proposed a new enhancement preprocessing method and used the mean-shift technique with a detector to reduce false positive values and decrease the processing time. The performance results of the proposed detection method and latest detection methods are reported in Table 6. Note from Table 6 that the proposed method achieved the best detection results compared with the existing methods in terms of *RR*, *PR*, and *F-m* rates.

TABLE 6. Comparisons of the proposed method with the existing state-of-the-art methods in terms of the *RR*, *PR*, and *F-m* rates.

Method	*RR*	*PR*	*F-m*
Azam and Gavrilova [45]	91.3%	NR	NR
Raghunandan et al. [44]	79.4%	84.6%	81.9%
Al-Shemarry et al. [25]	99.10%	98.2%	98.65%
Yousif, et al. [50]	85.43%	97.86%	91.22%
Henry, et al. [51]	99.76%	98.85%	99.30%
Proposed SVM_MHOG_LBP	**99.62%**	**98.32%**	**98.96%**

VII. CONCLUSIONS AND FUTURE WORK

This study proposed a new preprocessing method to improve the LPD system performance for complicated vehicle images by a Gaussian filter and the ECHE with the CLAHE algorithm. The MHOG and LBP descriptors were used to extract the representative LP features. The English car plates' database, Al-Shemarry et al. database, Arabic and Chinese vehicles images databases were used to evaluate the system performance under more difficult and complicated image conditions. The SVM and ELM classifiers were used to classify the extracted features, separately, for comparison purposes. An ensemble of strong detectors or trained models for three types of LP resolutions, 100×25, 200×50 and 300×75, were developed. From the experimental results, the SVM with the combined MHOG and LBP descriptors outperformed the ELM and the other methods with a single descriptor in terms of the detection accuracy rate. The proposed method was tested using different databases with simple and complicated conditions, such as fogy, low/high contrast, distorted and rotated LPs. It yielded excellent results. The detection and accuracy rates are 99.62% and 98.12%, respectively. The overall performance evaluation for the object localization metrics of the recall, precision, and F-measure rates are 99.62%, 98.32%, and 98.96%, respectively, with an FPR of 1.675%. Also, the ROC curve was used to compare and evaluate the results of the proposed method. The classification results of the ROC curve were very good for the SVM and ELM methods at 99.78% and 96.92%, respectively. The proposed method was also compared with the existing LP detection methods that used the same English car database. It showed that the proposed method performed better than those by other methods in terms of the efficiency and detection rate. The average runtime for the detection stage per vehicle image was 0.2408ms. The experimental results demonstrated that the proposed technique could be applied efficiently for real-time applications. We plan, in future work, to enhance the proposed detection method through using high-quality hardware and software components for reducing the overall detection time of the LPD system, that currently is 2.2187ms in total.

REFERENCES

[1] Z. Huang, Y. Yu, J. Gu, and H. Liu, "An efficient method for traffic sign recognition based on extreme learning machine," *IEEE transactions on cybernetics*, vol. 47, no. 4, pp. 920-933, 2017.

[2] D. Selmanaj, M. Corno, and S. M. Savaresi, "Hazard detection for motorcycles via accelerometers: A self-organizing map approach," *IEEE transactions on cybernetics*, vol. 47, no. 11, pp. 3609-3620, 2016.

[3] P. Pramkeaw, M. Ketcham, W. Limpornchitwilai, and N. Chumuang, "Analysis of Detecting and Interpreting Warning Signs for Distance of Cars using Analyzing the License Plate," in *2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*: IEEE, pp. 1-8.

[4] P. Gao, R. Yuan, F. Wang, L. Xiao, H. Fujita, and Y. Zhang, "Siamese attentional keypoint network for high performance visual tracking," *Knowledge-Based Systems*, vol. 193, p. 105448, 2020.

[5] P. Gao, Q. Zhang, F. Wang, L. Xiao, H. Fujita, and Y. Zhang, "Learning reinforced attentional representation for end-to-end visual tracking," *Information Sciences*, vol. 517, pp. 52-67, 2020.

[6] M. S. Al-Shemarry, Y. Li, and S. Abdulla, "Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images," *Expert Systems With Applications*, vol. 92, pp. 216-235, 2018.

[7] H. Zhang, W. Jia, X. He, and Q. Wu, "Learning-based license plate detection using global and local features," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, vol. 2: IEEE, pp. 1102-1105.

[8] K. Zheng, Y. Zhao, J. Gu, and Q. Hu, "License plate detection using haar-like features and histogram of oriented gradients," in *Industrial Electronics (ISIE), 2012 IEEE International Symposium on*, 2012: IEEE, pp. 1502-1505.

[9] W. T. Ho, H. W. Lim, and Y. H. Tay, "Two-stage license plate detection using gentle Adaboost and SIFT-SVM," in *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, 2009: IEEE, pp. 109-114.

[10] S. Lew, C.-S. Yi, W.-J. Lee, B.-R. Lee, K.-W. Min, and H.-C. Kang, "Extraction of the License Plate Region Using HoG and AdaBoost," *Journal of Digital Contents Society*, vol. 10, no. 4, pp. 597-604, 2009.

[11] L. Liu, L. Zhao, Y. Long, G. Kuang, and P. Fieguth, "Extended local binary patterns for texture classification," *Image and Vision Computing*, vol. 30, no. 2, pp. 86-99, 2012.

[12] L. Xie, T. Ahmad, L. Jin, Y. Liu, and S. Zhang, "A New CNN-Based Method for Multi-Directional Car License Plate Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 507-517, 2018.

[13] M. Molina-Moreno, I. González-Díaz, and F. Díaz-de-María, "Efficient Scale-Adaptive License Plate Detection System," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1-13, 2018.

[14] W. Weihong and T. Jiaoyang, "Research on License Plate Recognition Algorithms Based on Deep Learning in Complex Environment," *IEEE Access*, vol. 8, pp. 91661-91675, 2020.

[15] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *IEEE Transactions on intelligent transportation systems*, vol. 9, no. 3, pp. 377-391, 2008.

[16] S. Azam and M. M. Islam, "Automatic license plate detection in hazardous condition," *Journal of Visual Communication and Image Representation*, vol. 36, pp. 172-186, 2016.

[17] S. R. Gunn, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5-16, 1998.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273-297, 1995.

[19] W. Kusakunniran, K. Ngamaschariyakul, C. Chantaraviwat, K. Janvittayanuchit, and K. Thongkanchorn, "A Thai license plate localization using SVM," in *Computer Science and Engineering Conference (ICSEC), 2014 International*, 2014: IEEE, pp. 163-167.

[20] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks," *arXiv preprint arXiv:1703.07330*, 2017.

[21] Y. Liu, H. Huang, J. Cao, and T. Huang, "Convolutional neural networks-based intelligent recognition of Chinese license plates," *Soft Computing*, pp. 1-17, 2017.

[22] S. Ding, L. Guo, and Y. Hou, "Extreme learning machine with kernel model based on deep learning," *Neural Computing and Applications*, vol. 28, no. 8, pp. 1975-1984, 2017.

[23] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008: IEEE, pp. 1-8.

[24] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971-987, 2002.

[25] 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*, 2019.

[26] A. A. Gooch, S. C. Olsen, J. Tumblin, and B. Gooch, "Color2gray: salience-preserving color removal," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 634-639, 2005.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3024625, IEEE Access

[27] C. Saravanan, "Color image to grayscale image conversion," in *2010 Second International Conference on Computer Engineering and Applications*, 2010: IEEE, pp. 196-199.
[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, vol. 1: IEEE, pp. 886-893.
[29] H. Tan, B. Yang, and Z. Ma, "Face recognition based on the fusion of global and local HOG features of face images," *IET computer vision*, vol. 8, no. 3, pp. 224-234, 2014.
[30] D. Sun and J. Watada, "Detecting pedestrians and vehicles in traffic scene based on boosted HOG features and SVM," in *Intelligent Signal Processing (WISP), 2015 IEEE 9th International Symposium on*, 2015: IEEE, pp. 1-4.
[31] Y. Feng, Y. Song, and Y. Zhang, "Scene text localization using extremal regions and Corner-HOG feature," in *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, 2015: IEEE, pp. 881-886.
[32] L. Mao, M. Xie, Y. Huang, and Y. Zhang, "Preceding vehicle detection using Histograms of Oriented Gradients," in *2010 International Conference on Communications, Circuits and Systems (ICCCAS)*, 2010: IEEE, pp. 354-358.
[33] W. Jia, H. Zhang, and X. He, "Region-based license plate detection," *Journal of Network and computer Applications*, vol. 30, no. 4, pp. 1324-1333, 2007.
[34] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, 2019.
[35] Y. Wang, H. Yu, D. Sylvester, and P. Kong, "Energy efficient in-memory AES encryption based on nonvolatile domain-wall nanowire," in *Proceedings of the conference on Design, Automation & Test in Europe*, 2014: European Design and Automation Association, p. 183.
[36] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349-360, 2009.
[37] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95-99, 1988.
[38] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, 2004, vol. 2: IEEE, pp. 985-990.
[39] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76-79, 1992.
[40] Y. Yang and Q. J. Wu, "Extreme learning machine with subnetwork hidden nodes for regression and classification," *IEEE transactions on cybernetics*, vol. 46, no. 12, pp. 2885-2898, 2015.
[41] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603-619, 2002.
[42] EnglishLPDatabase-2001, " http://www.zemris.fer.hr/projects/LicensePlates/english/," *accessed July 2016*.
[43] Online.Photo.Editor. ""https://www.freeonlinephotoeditor.com/," accessed April, 2017." (accessed.
[44] K. Raghunandan *et al.*, "Riesz fractional based model for enhancing license plate detection and recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.
[45] S. Azam and M. Gavrilova, "License plate image patch filtering using HOG descriptor and bio-inspired optimization," in *Proceedings of the Computer Graphics International Conference*, 2017: ACM, p. 1.
[46] F. Bashir and F. Porikli, "Performance evaluation of object detection and tracking systems," in *Proceedings 9th IEEE International Workshop on PETS*, 2006, pp. 7-14.
[47] D. François, "Binary classification performances measure cheat sheet," ed: Volume, 2009.
[48] A. Slaby, "ROC analysis with Matlab," in *Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on*, 2007: IEEE, pp. 191-196.
[49] A. Godil, R. Bostelman, W. Shackleford, T. Hong, and M. Shneier, "Performance Metrics for Evaluating Object and Human Detection and Tracking Systems," 2014.
[50] B. B. Yousif, M. M. Ata, N. Fawzy, and M. Obaya, "Toward an Optimized Neutrosophic k-Means With Genetic Algorithm for Automatic Vehicle License Plate Recognition (ONKM-AVLPR)," *IEEE Access*, vol. 8, pp. 49285-49312, 2020.
[51] C. Henry, S. Y. Ahn, and S.-W. Lee, "Multinational License Plate Recognition Using Generalized Character Sequence Detection," *IEEE Access*, vol. 8, pp. 35185-35199, 2020.

Meeras Salman Al-Shemarry is a Lecturer in Computer Department, Science College, Karbala University, Iraq. She has a Bachelor of Computer Science from Babylon University, Iraq in 2002. She received her Master degree in IT 2010 from University Utara Malaysia (UUM), Malaysia. Currently, she is a PhD Student in the Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia. Her research interests include system analysis using UML diagrams, image processing and objects detection, artificial intelligence, database management system.

Yan Li is currently a Professor in Computer Science in the school of sciences, University of Southern Queensland, Australia. Her research interests are in the areas of Artificial Intelligence, Big Data Analytics, Signal and Image Processing, Biomedical Engineering, Artificial Intelligence, Big Data Analytics and Computer Networking Technologies.

6

CHAPTER 6

DISTORTED VEHICLE LICENCE PLATE DETECTION USING HYBRID FEATURES DESCRIPTOR AND EXTREME LEARNING MACHINE CLASSIFIER

6.1 Introduction

The content of this chapter is an exact copy of a submitted paper to the *journal* (2020) for publication 'Distorted vehicle licence plates detection using hybrid feature descriptors', (submitted).

The detection of LPs is similar to find the regions of interest (ROIs) that may contain the LP (true positive value) or non-LP (false positive value).

In Chapter 5, an efficient preprocessing method was introduced for the purpose of reducing the FPR and execution time for identifying LPs from complicated vehicle images. It was showed a large improvement in reducing the FPR and system runtime.

This chapter makes a further enhancement to modify the previous version of the preprocessing method in Chapter 5 and increase the system detection accuracy while reduce the testing time and keeping the previous improvement for in FPR. It applies the enhancement contrast-limited adaptive-cumulative histogram equalization (ECLACHE) technique. At the extraction stage, a combination of a median robust extend local binary pattern (MRELBP) and speeded up robust feature (SURF) descriptors as used. Those descriptors have the same powerful advantages that were described for previously descriptors in Chapters 3, 4, and 5. They extracted LP features under distorted conditions. The ELM with a mean-shift algorithm was used to train the extracted features and built a strong detector.

There is no one detection method working to solve all types of images problems and satisfy all system requirements. This method succeeded in achieving good detection accuracy results with a large improvement for system runtime. But the FPR was slightly increased compared with the method in Chapter 5 due to large improvements in preprocessing methods. The proposed method outperforms other reported methods in Chapters 3, 4, 5, in terms of the detection accuracy and execution time.

The Matlab code of this method is provided in Appendix D.

Distorted vehicle licence plate detection using hybrid feature descriptors

Meeras Salman Al-Shemarry [a, b,*] and Yan Li [a]
[a] School of Agricultural, Computational and Environmental Sciences
Faculty of Health, Engineering and Sciences
University of Southern Queensland, Australia
[b] Computer Department, Science College, Karbala University, Karbala 56001, Iraq

[*]Correspondence author
E-mail addresses: MeerasSalmanJuwad.Al-Shemarry@usq.edu.au (M. S. Al-Shemarry), Yan.Li@usq.edu.au (Y. Li)

Abstract

Intelligent transportation systems (ITSs) play a very important role in people's lives in many respects. One of the most important ITS applications is A licence plate detection system (LPD). In this paper, a new LPD framework is proposed. It includes a novel technique for the preprocessing, extraction, and detection stages to detect LPs from distorted vehicle images. An efficient preprocessing method is developed in this study which proposes an enhanced contrast-limited adaptive histogram equalization technique for filtering the unwanted LP regions. At the extraction stage, strong hybrid features of a median robust extended local binary pattern and speeded-up robust feature descriptor are applied to extract complicated features from LPs. Those hybrid features can enhance the useful information, and thus the detection system performance under difficult scenarios. In the detection stage, the trained model of an extreme learning machine classifier, with a mean-shift algorithm, is used as a detector to make a decision for output results. The performance of the proposed method was compared in terms of true-positive and false-positive rates with other classifiers and existing detection methods. The experiments on an English car plates database show that the proposed method made significant improvements in accuracy and runtime speed for the LPD system under difficult scenarios.

Keywords

Transportation systems; Licence plate detection (LPD); Hybrid features; Extreme learning machine (ELM); Confusion matrix; Receiver operating characteristic curve (ROC).

1. Introduction

There is rapid growth in automatic systems development including the licence plate detection (LPD) system that helps to automatically identify the license plate (LP) of vehicles as quickly as possible. It is a difficult challenging to identify LPs from images complicated due to environmental effects. Therefore, a good LPD system is required to effectively work under difficult scenarios, for example, low/high lighting, night, or with blurred, foggy, rotated, dusty, distorted, and with complex backgrounds, etc. For an LP to be identified, the test image is passed through three stages as shown in Fig. 1.

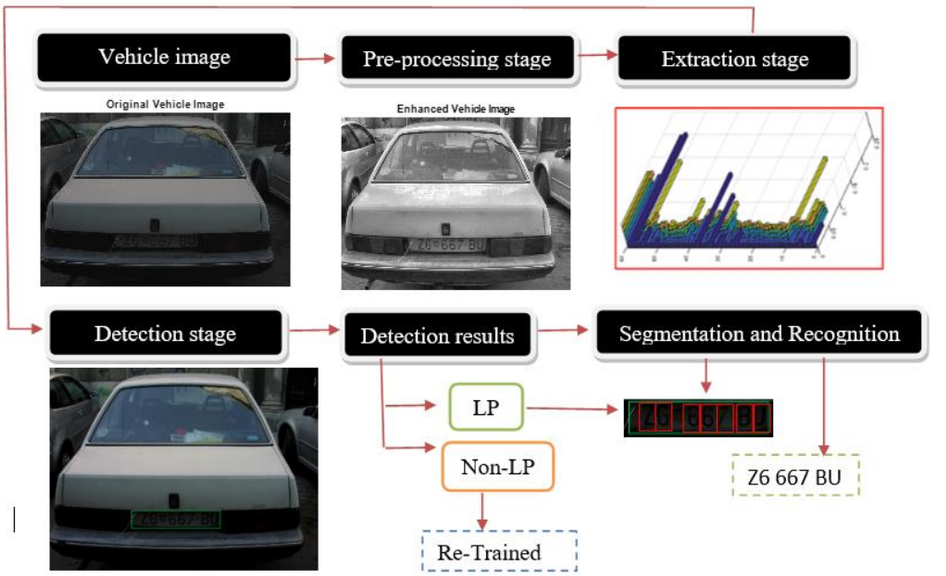


Fig. 1. Stages involved in the licence plate detection (LPD) system.

In the first stage, the quality of a captured image is improved using some preprocessing methods. There are many enhancement techniques to do that, like filtering, contrast enhancement, histogram equalization, binarization, and so on. In the second stage, the region of the interest (ROI) is extracted by using various extraction descriptors. These extracted regions may or may not be the true LP regions. This stage is called the extraction stage for LPs. In the third stage, the extracted regions are used as inputs for a good classifier in order to build a strong detector, or a trained model, for LP localization. This stage is called as LP detection. The performance of the detection system relies on the robustness and reliability of each individual stage. This paper focuses only on the detection stage for detecting LPs from distorted vehicle images. Therefore, the segmentation and recognition stages were not considered in this work. In this stage, the number from the LP converts into machine-encoded text. Then, the optical character recognition (OCR) is used to recognize the plate numbers from the LP image. Many methods have been proposed in the past years to detect LPs. They can be categorized as color-based (Ashtari, Nordin, & Fathy, 2014; Shi, Zhao, & Shen, 2005), edge-based (Ascar Davix, Oshin, & Shamili, 2016; Azad, Azad, & Shayegh, 2014; Ha & Shakeri, 2016; J. Wang, Bacic, & Yan, 2018), character-based (Li & Wang, 2016; Soora & Deshpande, 2016), texture features-based (Al-Shemarry, Li, & Abdulla, 2018, 2019), and filter-based (Tadic, Popovic, & Odry, 2016; L. Zhang, Shi, Xia, & Mao, 2013) methods. But the task of identifying an LP properly is still a very challenging task. The preprocessing stage plays an important role in enhancing and building up detection system performance. Descriptors, like local binary pattern (LBP), works well for different illumination conditions. It can partly solve scale invariance problems (Ojala, Pietikäinen, & Harwood, 1996). Many researchers proposed a large number of LBP variant descriptors to improve the weakness of LBP operators and achieved good classification performance (El Khadiri, Kas, El Merabet, Ruichek, & Touahni, 2018). For example, the extended local binary pattern (ELBP)

(Liu, Zhao, Long, Kuang, & Fieguth, 2012), the completed local binary pattern (CLBP) (Guo, Zhang, & Zhang, 2010), multi-level extended local binary pattern (Al-Shemarry et al., 2019), completed local derivative patterns (CLDPs) (Hu, Long, & AlRegib, 2016), scale-selective local binary pattern (SSLBP) (Guo, Wang, Zhou, & You, 2016), and median robust extend local binary pattern (MRELBP) (Liu, Fieguth, Pietikäinen, & Lao, 2015). The speeded up robust feature descriptor (SURF) is better than other features for object detection inside an image (Arróspide & Salgado, 2014; Bauer, Sünderhauf, & Protzel, 2007; Bay, Tuytelaars, & Van Gool, 2006a). It is usually used to capture the important information which is sensitive to illumination and rotation conditions (Hanbay, Alpaslan, Talu, & Hanbay, 2016; Khan, McCane, & Wyvill, 2011; Z. Luo, Chen, Takiguchi, & Ariki, 2015; Panchal, Panchal, & Shah, 2013; Pang, Li, Yuan, & Pan, 2012; Rajesh, Kaushik, & Jangra, 2016). For illumination problems, the SURF descriptor has a discriminative power to extract contrast lighting features with or without rotation problems (Bay et al., 2006a; Pang et al., 2012). One method is often not sufficient to solve several problems for a robust system. Recent studies showed that combining multiple descriptors enables good extraction compared with using a single descriptor for multiple problems (Y. Chen, Zhao, Lv, & Zhang, 2018; Kim, Song, Kim, & Park, 2019; Lalimi, Ghofrani, & McLernon, 2013; Y. Wang, Zhang, Fang, & Guo, 2009; H. Zhang, Jia, He, & Wu, 2006). In supervised learning, the extraction features are trained by the classifier to produce trained models and these models learn to discriminate different LP problems. Classifiers used for detection of LPs include ELM (G.-B. Huang, Zhu, & Siew, 2006), Adaboost (Hastie, Rosset, Zhu, & Zou, 2009), SVM (Gunn, 1998) and CNN (Xie, Ahmad, Jin, Liu, & Zhang, 2018). The purpose of this work is to develop an LPD framework for complicated vehicle images. It includes a new enhanced preprocessing method, the enhancement contrast-limited adaptive-cumulative histogram equalization (ECLACHE), the combination of the MRELBP and SURF descriptors, and an extreme learning machine (ELM) classifier with a mean-shift algorithm to reduce a false-positive rate and improve the classification accuracy.

The main contributions of this work are as follows:

i. Updating our previous work (Al-Shemarry et al., 2018), the pre-processing method to improve the performance of the LPD system by filtering unwanted LP regions. The new method is an enhancement contrast-limited adaptive-cumulative histogram equalization (ECLACHE).
ii. Utilizing hybrid features, MRELBP and SURF, as extraction methods. Based on previous studies in the literature those methods very strong and suitable descriptors to extract several features under different environments like low/high contrast, blurry, foggy, rotated LPs, complex backgrounds, dark, and so on.
iii. Applying a mean-shift algorithm with ELM classifier to reduce the false-positive values and improve the classification accuracy.
iv. Using complicated and much challenged English car databases (Al-Shemarry et al., 2019; EnglishLPDatabase-2001) in this work.
v. Evaluating the proposed object detection method using very important factors for performance measurement metrics.

vi. Comparing the performance of the proposed method with new existing detection methods.

This paper is organized as follows: Section 2 introduces the framework of the LPD system. Section 3 shows system database. Section 4 presents the experimental results. Section 5 displays the comparison with other existing methods. Finally, in section 6 the conclusion and future work are discussed.

2. The proposed method

In this research, the proposed LPD system for vehicle images under difficult environmental conditions is presented. The framework of the proposed method is depicted in Fig. 2. It is consists of three main stages: preprocessing, extraction and detection. For the preprocessing stage, a good enhancement method, the ECLACHE, is developed. It overcomes all difficulties related to high/low lighting, dusk, blurry, and foggy conditions. At the second stage, the features are extracted from the enhanced image using a combination of the two powerful descriptors, MRELBP and SURF. These descriptors were carefully selected for being suitable for difficult conditions. Finally, the ELM classifier is used to build the trained model from the extracted MRELBP and SURF features. The LPD system consists of two phases of training and testing. The same methods in the preprocessing and extraction stages are used in both phases. The detected LP images are cropped and stored for the recognition stage to obtain a complete licence number plate recognition system. The detail of the proposed framework is described in the following sections.

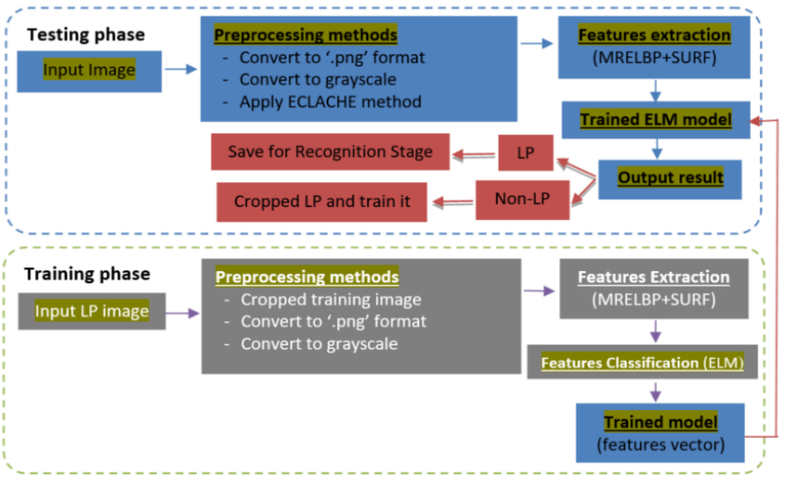


Fig. 2. The framework for the proposed LPD system

2.1 The preprocessing stage

The detection of an LP is related to find the ROI that may be LP or non-LP. In this system, the texture and gray scale features are used to reduce the feature dimension and the processing time for the next stage. The input image format of ".jpg" is converted to ".png" in order to save the quality of the image instead of resizing. Then, it converts to the grayscale image of *GIMG* (Saravanan, 2010) by Eq. (1).

$$GIMG\{X\} = 0.2989 \times R\{X\} + 0.5870 \times G\{X\} + 0.1140 \times B\{X\} \quad (1)$$

where *R {X}*, *G{X}*, and *B {X}* are the red, green, and blue channels of colors, respectively, and *X* refer to the image pixels.

There are different sources of noise in a distorted vehicle image, leading to false-positive detections, like surface textures, low/high lighting, distortion, dirt, and dust. Therefore, the enhancement preprocessing algorithm, ECLACHE, is used to reduce the noise and enhance the lighting conditions. The steps of the ECLACHE algorithm are shown in Fig. 3.

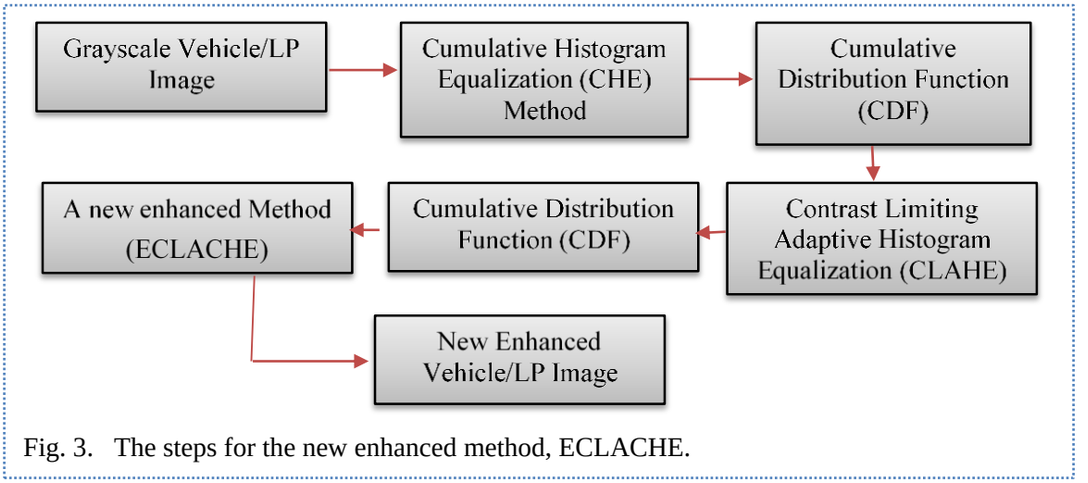


Fig. 3. The steps for the new enhanced method, ECLACHE.

The first step is applied the cumulative histogram equalization (CHE) method to the grayscale image, *GIMG {X}*, and let n_i is the number of pixels occurrences on gray-level *i*. The probability of an occurrence pixel of level *i* in the *GIMG {X}* image is

$$P_x(i) = P(x = i) = n_i/n \ , 0 \leq i < L \qquad (2)$$

where *L* is the total number of grayscale levels in an image which is typically 256 and *n* is the image pixels. $P_x(i)$ is the image's histogram of the pixel value *i*, which is normalized to [0, 1]. The second step is calculated the cumulative distribution function (CDF) for CHE method, which is also an image accumulated normalized histogram and defined as

$$CDF_x(i) = \sum_{j=0}^{i} P_x(j) \qquad (3)$$

The third step is created a transformation form, *Y = T(x)* to produce a new image *{Y}*, with a new values of the flat histogram, which is also need a linearized CDF across the new value range and defined as

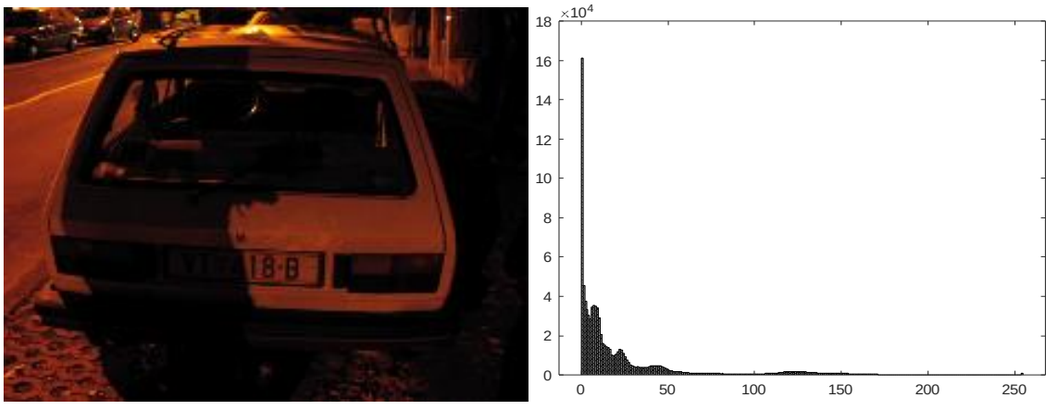
$$CDF_y(i) = iC \qquad (4)$$

where *C* is some constant in the range *[0-L]*. Notable that *T* maps the new values back into the original range, since it used a normalized histogram of *{X}*. A more detailed is provided by Pizer et al. (1987). Fourth step is applied the contrast limiting adaptive histogram equalization (CLAHE) method on the new image *{Y}*, and repeated the 2nd and 3rd steps in order to produce a new enhanced image that contains the new grayscale values.

The performance of the proposed preprocessing method, ECLACHE, can be observed from Fig. 4. From Fig. 4, it can be noticed the difference between the improved method by Al-Shemarry et al (2019) and ECLACHE method. Also, the range of the feature dimensions is reduced by this method giving a good image quality. Moreover,

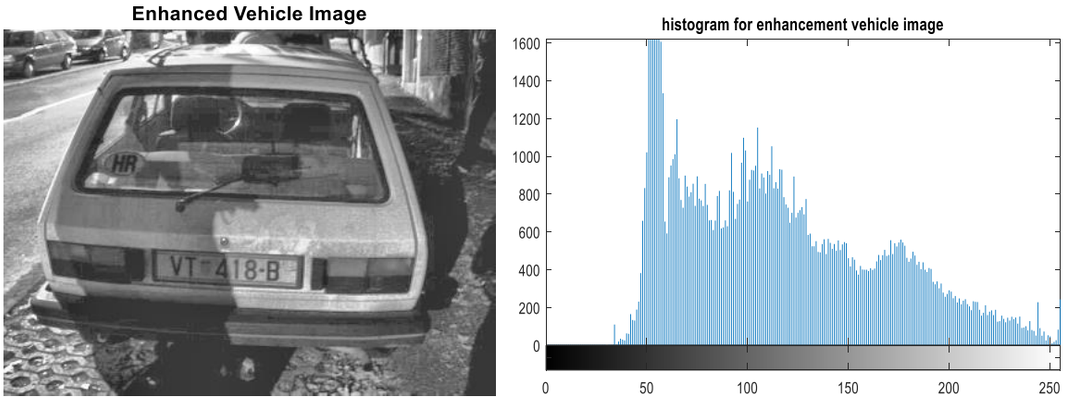
the histogram information helps to display the difference between the original image and the enhanced image through decreasing the range of the unwanted features.

Original vehicle image



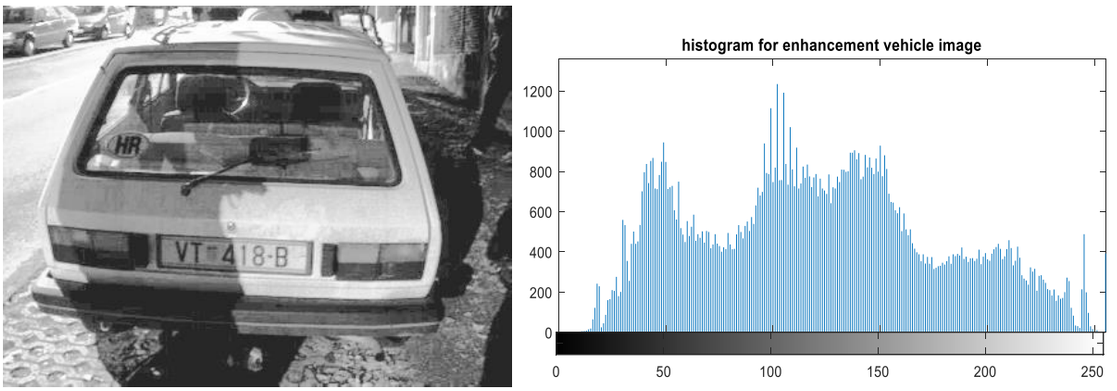
(a)

Improved image by Al-Shemarry et al. (2018)



(b)

Improved image using ECLACHE



(c)

Fig. 4. (a) The original licence plate image; (b) The output for the preprocessing enhancement method by Al-Shemarry et al (2019); (c) The output for the ECLACHE method with the histogram. The histogram shows how the ECLACHE method works (X axis is the range of the feature values in each bin; Y axis is the number of feature values appearing in each bin range).

2.2 The features extraction stage

This research uses the combination of MRELBP and SURF descriptors to extract the key features from low-quality vehicles images. There are many reasons for us to use those descriptors as described in the introduction.

2.2.1 The speeded-up robust feature (SURF) descriptor

The scale invariant feature transform (SIFT) descriptor is the most widely used, but it has a high computational cost. In 2006, Bay et al. (2006a) developed a SURF variant descriptor of the SIFT. This section presents a brief summary of the SURF construction process, strong interest point localization and interest point descriptor computation.

2.2.1.1 Interest point localization

The SURF descriptor is depended on the Hessian matrix. Given an image I a point $x= [x, y]$, the Hessian matrix $H(x, \sigma)$ of the point x at the scale σ is defined as

$$H(x,\sigma) = \begin{bmatrix} CG_{xx} & CG_{xy} \\ CG_{xy} & CG_{yy} \end{bmatrix}, \quad (5)$$

where $CG_{xx}(x, \sigma)$ is the convolution of the Gaussian order derivative $\frac{\partial}{\partial_x^2}\, g(\sigma)$ for the image I in the point x, and similarly for $CG_{xy}(x, \sigma)$ and $CG_{yy}(x, \sigma)$. In the contrast to the SIFT descriptor, which approximates laplacian of Gaussian with the difference of Gaussians, the SURF approximates second derivatives Gaussian order with box filters. An example of the lowest scale analysed by this filters in Fig. 5. Image I convolutions with box filters were computed rapidly using the integral image (Viola & Jones, 2001).

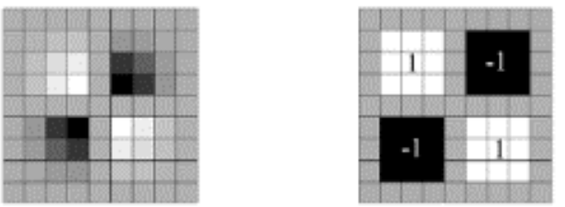


Fig. 5. Left: the Gaussian second order derivative in the *xy-direction*. Right: the corresponding box filter approximation.

The scale and location of interest points are selected based on the determinant of the Hessian matrix. Interest points are localized in the scale and image I space through applying the non-maximum suppression in the $3\times3\times3$ neighbourhood. For more details, see the reference by Bay, Tuytelaars, and Van Gool (2006b).

2.2.1.2 Interest point descriptor

The first step of the SURF descriptor is constructed circulars regions around the strong detected interest points to assign a unique orientation and gain invariance to the image rotation. The orientation is calculated by using the Haar wavelet response in both directions, *x* and *y*. It can be computed through the integral image quickly, similar to the second derivatives Gaussian order with box filters. The dominant orientation in the interest point's information is estimated and included. The next step is extracted the square regions around interest points and divided into *4×4* sub-regions. In each sub-region, the Haar wavelet response is computed at horizontal *d(x)* and vertical *d(y)* directions. Each sub-region is represented on four-dimensional descriptor that contains the sum of absolute values of the *d(x)* and the *d(y)*, which is summarized in Eq. 5 (Bay et al., 2006a).

$$V = \left(\sum d(x), \sum d(y), \sum |d(x)|, \sum |d(y)| \right) \quad (6)$$

The result of the SURF descriptor for all *4×4* sub-regions is about 64 features dimension. Through the experimental the main advantage of the SURF descriptor is good and fast extraction process. It is claimed that, the SURF descriptor is invariant to the scale and rotation of the object inside an image (Han, Virupakshappa, & Oruklu, 2015). The input value for the SURF descriptor is an enhanced image and the output is strong key points or features from the regions of interest. As shown in Fig. 6 (a) the dimension of the extracted feature values for the SURF descriptor and Fig. 6 (b) the location of the strongest key points in the regions of interest for the feature extraction process.

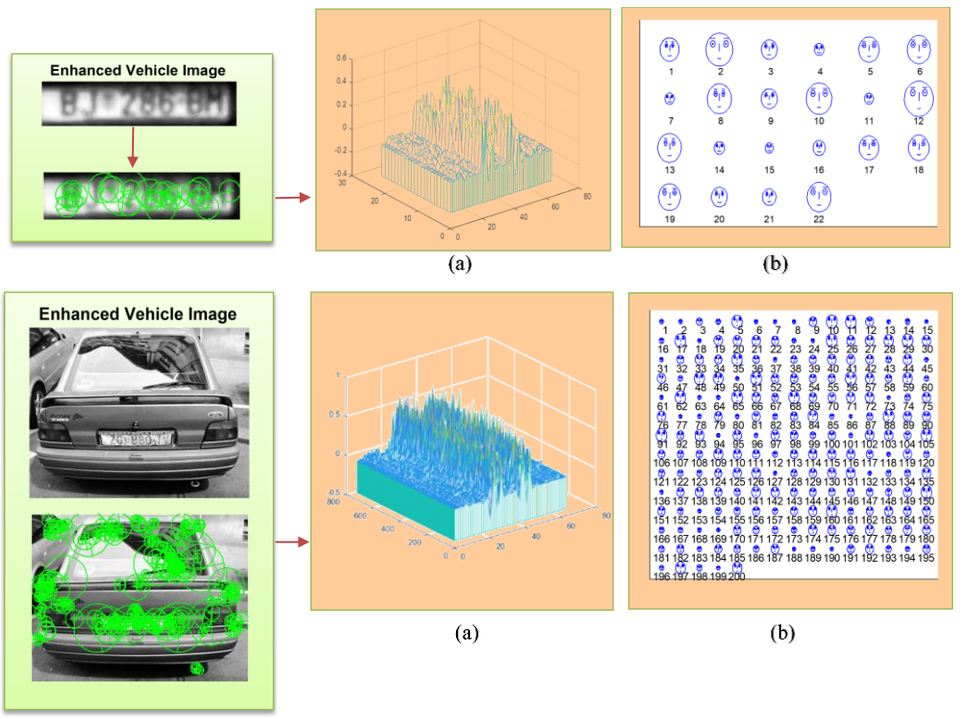


Fig. 6. The SURF descriptor output: (a) The dimension of the extracted features values for the SURF descriptors; (b) the location of the strongest key points in the regions of interest for the feature extraction process.

2.2.2 Median robust extend local binary pattern (MRELBP)

The texture LBP descriptors, produced good performance texture classification results, however there are still some significant drawbacks if they are used alone without improvement methods. The drawback of the ELBP (Liu et al., 2012) descriptor is that it is very vulnerable to the image noise. The first strategy is to replace the individual pixel gray-scale values for the sample points with the simple filter responses. The MRELBP (Liu et al., 2016) descriptor is simple conceptually, high-quality, and a computationally effective approach, based on the combination of a median filter with multi-feature resolution support. This descriptor offers the gray-scale invariance, noise robustness, rotation invariance, and powerful discrimination. It was compared against many other LBP descriptors (Liu et al., 2016) and produced good extraction results. The ELBP is modified by replacing the individual pixel intensities with filter responses φ, as shown in the Fig. 6. The image is normalized to zero mean and unit variance, using standard encoding scheme (*riu2*). If the center pixel value is X_C and the patch filter φ, the MRELBP_CI, MRELBP_NI and MRELBP_RD descriptors are defined as

1) MRELBP of the center pixel:

$$MRELBP\_CI\ (X_C) = S\ (\varphi\ (X_{C,w}) - \mu_w) \qquad (7)$$

The result from applying the filter φ to $X_{C,w}$ is that the local patch of the size $w \times w$ (neighbouring feature space) centered on the center pixel X_C , and μ_w denoting the mean of the $\varphi\ (X_{C,w})$ over the whole normalized image.

2) MRELBP of the neighbors' pixels:

$$MRELBP\_NI_{r,\ p}^{riu2}\ (X_C) = \sum_{n=0}^{p-1} S(\varphi\ (X_{r,p,wr,n}) - \mu_{r,p,wr}) 2^n \qquad (8)$$

$$\mu_{r,p,wr} = \frac{1}{p} \sum_{n=0}^{p-1} \varphi\ (X_{r,p,wr,n}) \qquad (9)$$

where $X_{r,p,wr,n}$ refers to the neighbouring features space or the patch size of the $wr \times wr$ which is centered on the $X_{r,p,n}$.

3) MRELBP of the radial difference:

$$MRELBP\_RD_{r,\ r-1,\ p,\ wr,\ wr-1}^{riu2}\ (X_C) = \sum_{n=0}^{p-1} S(\varphi\ (X_{r,p,wr,n}) - \varphi\ (X_{r-1,p,wr-1,n})) 2^n \qquad (10)$$

where $X_{r,p,wr,n}$ and $X_{r-1,p,wr-1,n}$ refer to the centered patches at the neighboring pixels $X_{r,p,n}$ and $X_{r-1,p,n}$, respectively. The formula $\{X_{r,p,n}\}_n^p$ denotes the circular of the neighbors pixels for the center pixel *Xc* at radius *r*.

Referring to Fig. 7, this study used multi-feature space, MRELBP *(r, p, w)* descriptors to extract difficult features from complicated enhanced images. The MRELBP used the median filter remove the noise or unwanted regions from the vehicle image. The results are four local binary pattern images. Liu et al. (2015) fixed the number of neighbour pixels at 8 on each MRELBP descriptor to reduce the dimension of the extracted features and obtain good extraction results. The main parameters for MRELBP descriptor are the sampling radius r, the size of the center patch $wc \times wc$, and the sizes of the neighboring patches $wr \times wr$. Choosing *(r, p)* as *(2,8)+(4,8)+(6,8)+(8,8)* gave very good results. Also, each LBP image has a joint histogram, which refers to the strong extracted features. The four joint histograms merge together to produce the concatenation histogram for all the MRELBP descriptors.

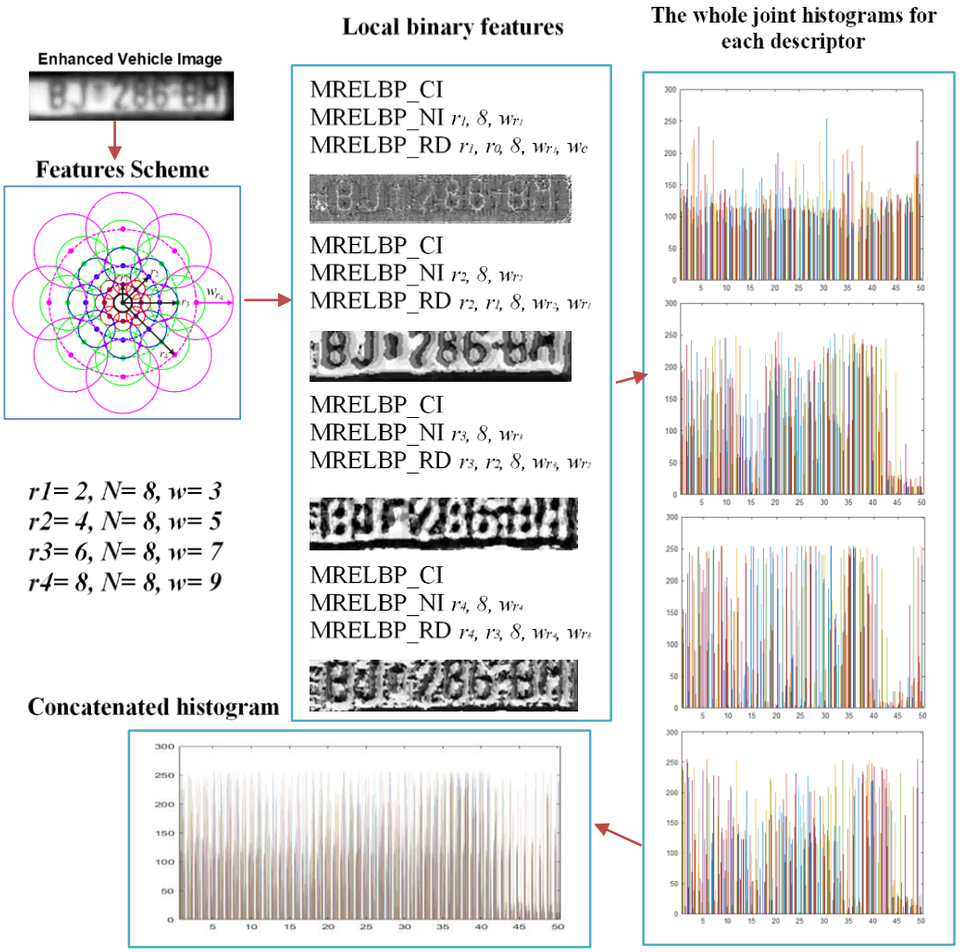


Fig. 7. The overview of the MRELBP descriptor. For illustrating the work of the MRELBP an example features scheme is given with their neighbouring features space. Each circle represents a neighbors' pixels over the center pixel point which are computed to replace the gray value of the central point based on the LBP formula.

2.3 Detection stage

The ELM classifier (G.-B. Huang et al., 2006; G. Huang, Huang, Song, & You, 2015) was used at the detection stage. The speed of the ELM is significantly faster than other training algorithms. It can be viewed as a variant of a random vector functional-link (RVFL) network classifier (Pao & Takefuji, 1992; Scardapane, Wang, Panella, & Uncini, 2015) without direct links and bias terms. It is widely used as a strong classifier for the pattern classification task. As shown in the Fig. 8 the combination of the extracted features *(n)*, or input neurons, feeds the ELM. For optimal LP detection results, in this paper, the number of hidden neurons *(L)* is 1000 which is set depending on the detection performance with the testing data set. The default activation function for the hidden neurons in the ELM that is applied in this work is a 'sigmoid' function. Also, other activation functions, such as 'sin' and 'hardlim' are tested, but the experiments found that the sigmoid function gave better results. The ELM starts to train extracted features, MRELBP_SURF with hidden nodes to obtain output neurons *(m)*. After that, the

strong detector was built as a decision function to detect the LP regions from the vehicle image. For more detail about ELM using the references above.

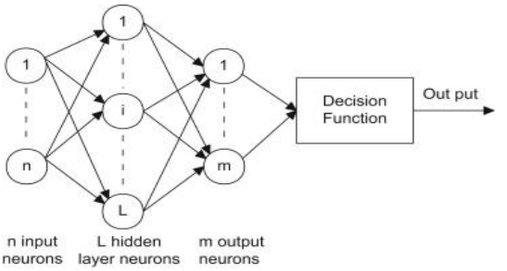


Fig. 8. Architecture of the ELM.

3. Database for experiment

In order to evaluate the performance of the proposed method, we used the same database that presented by Al-Shemarry et al. (2019) and (EnglishLPDatabase-2001). An English car database that contains 2050 images, with various scenes, and size of 640×480. Some of these images are captured by an OLYMPUS C-2040ZOOM digital camera under different environmental conditions (EnglishLPDatabase-2001), such as cloudy weather, night

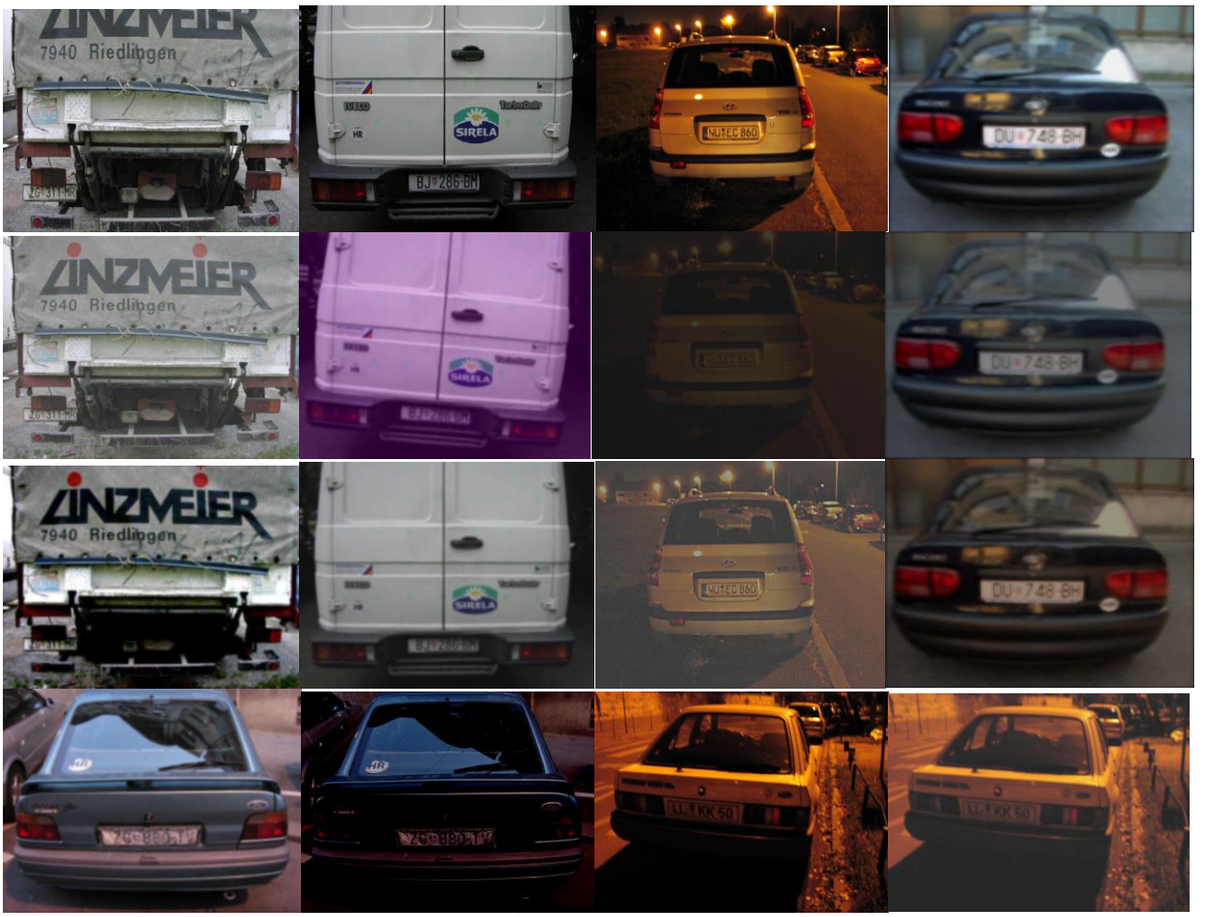


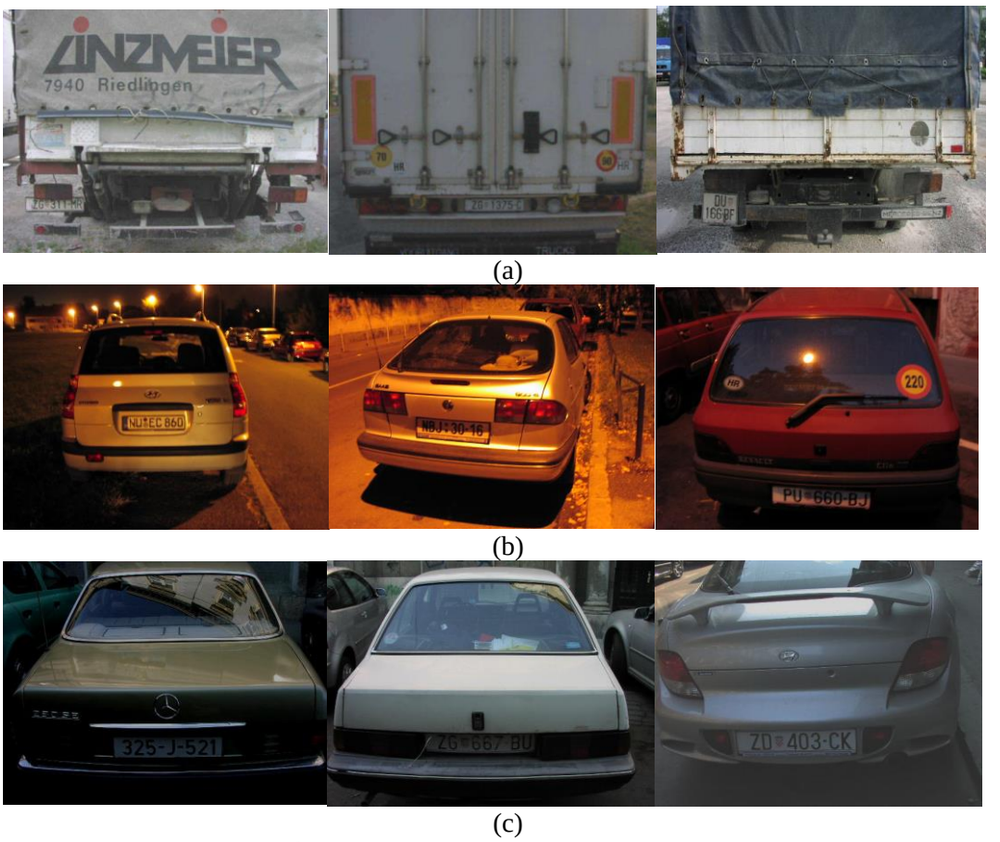
Fig. 9. Some examples of complicated scenes for each vehicle image in the database.

lighting, sunny day, and dusk. The remaining vehicle images one from Al-Shemarry et al. (2019) under blurry, low/high contrast environments, and distortion conditions. We divided the database into datasets, 1700 for training and 350 for testing. A very important aspect of this database is that the LP inside vehicle image has different sizes, 25×100, 50×200, and 75×300. This makes the work is very challenging and the result more reliable. In addition, each vehicle image in this database has various complicated scenes, in order to capture all LP difficulties (see Fig. 9). Some samples of the training and testing data sets are shown in Figs. 10 and 11.

The experimental results showed that all the key characteristics have been captured from the 1700 training images in the data set.

4. Experimental results

This section presents experimental results of the proposed work. The data is trained and tested on the following platform: Desktop and laptop computer with 3.4 GHz Intel Core i7-4770, 16GB of RAM using MATLAB programming language, version R2018a. The performance of the ELM classifier is evaluated using the confusion matrix and receive operating characteristic (ROC) curve.



(a)

(b)

(c)

Fig. 10. Some examples of testing images in the dataset: (a) Vehicle images with LPs 25×100 in size; (b) Vehicle images with LPs 50×200 in size; (c) Vehicle images with LPs 75×300 in size.

4.1 Extraction and classification results

This work focuses on the preprocessing and the features extraction methods rather than the classification methods. The ELM classifier is used to classify the MRELBP and SURF features of the LP regions. The MRELBP descriptor used a median filter with four circular neighbourhood or the features space which contain different radius r and window size $w \times w$ with the same number of neighbouring pixels p. For all the circulars, we extract the MRELBP features using the set of $(P\ i, Ri)$, $i = 1,..., N$, where the value of N is identified based on the LP image size and its complexity. We set i depending on the use of different neighborhood circles for four MRELBP descriptors, $i = 0, 1, 2,...,7$ means 8 pixels neighbours with $R = 2;4;6;8$, respectively, and different windows size w = 3×3; 5×5; 7×7; 9×9 respectively, (see Figure 7). From our experimental results, it is noticed that the combining of the four MRELBP descriptors can achieve good extraction results. The SURF descriptor extracted features from the LP images with 60 strong points as shown in Fig. 6. The combination of MRELBP and SURF was used as input to the ELM classifier to build a strong detector. This detector produces a binary predicted value of "1" for the LP and "0" for non-LP. The number of the overlapping bounding boxes depends on the location of the LP regions, with an average of ∼2 is very good due to use the mean-shift algorithm with the ELM detector. The elapsed time for training phase with 1700 LP images is 74.501691ms. The classification results are shown in Table 1.

Table1
The average of the classification results for the proposed method descriptors.

Method	IS	SP	WS	FD	CA
ELM_SURF	25×100 50×200 75×300	60	4×4	341	95.58%
ELM_MRELBP $^{Riu2}_{(((8,2))+(8,4)+(8,6)+(8,8))}$	25×100	-	3×3	223	93.89%
	50×200		5×5	256	95.06%
	75×300		7×7	252	96.98%
			9×9	250	98.30%
ELM_SURF_MRELBP $^{\mathbf{Riu2}}_{((8,2)+(8,4)+(8,6)+(8,8))}$	25×100 50×200 75×300		All WS	920	100.00%

CA: Classification Accuracy; FD: Feature Dimension; IS: Image Size; SP: Strong Points; WS: Window Size.

4.2 Performances of the ELM classifier

In the machine learning field, confusion matrix and ROC curve are good measurement techniques to compare the performance of supervised learning algorithms (Bashir & Porikli, 2006; Godil, Bostelman, Shackleford, Hong, & Shneier, 2014; Slaby, 2007) .

4.2.1 The confusion matrix

Many detection systems used the confusion matrix to evaluate the system performance under test (SUT) (Kasturi et al., 2009). Using the confusion matrix, all objects are validated to show if there is any matching between the SUT and ground truth (GT). The parameters for this matrix as follows:

- The false positive (FP) where the LP is detected by the system, but it is not in the GT.
- The false negative (FN) where the LP exists in the GT, but the detection system fails to identify it.
- The true positive (TP) means the LP is correctly detected by system and it exists in the GT.
- The true negative (TN) where the LP is not detected by the system and it does not exist in the GT.

From the confusion matrix parameters the accuracy rate (AR) and the detection rate (DR) are calculated as follows:

$$DR = TP/ (TP+ FN) \quad (11)$$

$$AR = (TP+TN)/ (TP+TN+FP+FN) \quad (12)$$

The DR is the proportion of the TP results that are truly predicted as positive results by the detection system. While the AR rate is the proportion of the both TP and TN results for the LP objects in the GT. The detection results based on the confusion matrix are shown in Table 2.

Table 2
The detection results for the proposed method using the ELM classifier.

Method	**No. of Testing Images = 350, TN= 30 Images**					
	FP	**TP**	**FN**	**TN**	**DR**	**AR**
ELM_SURF	18	315	35	23	90.14%	86.44%
ELM_MRELBP	11	336	14	27	96.32%	93.55%
Proposed ELM_SURF_MRELBP	8	349	~2	30	99.71%	97.92%

From the Table 2, it is notable that the combination of MRELBP and SURF descriptors with the ELM classifier improves the detection results. The average of the runtime for the whole detection system per vehicle image was 2.108 seconds. Examples of system detection results are shown in the Fig. 12. The database that was presented by Al-Shemarry et al. (2019) was very challenge database, each image has several news with different conditions. It was observed that all detection results have very low FPs values due to use the mean-shift algorithm with ELM detector. Also, the main reason for increased FP values was when some objects inside a vehicle image were similar to the LP, such as texts or commercial logos.

4.2.2 The receiver operating characteristic (ROC) curves of the classifier

This study uses a ROC curve as a useful tool to evaluate the classifier performance quality (Slaby, 2007). The graphical plot of the ROC curve depends on the true positive rate (TPR) or detection rate against of the false positive rate (FPR). They are defined as follows:

$$TPR= TP/ (TP+ FN) \quad (13)$$

$$FPR = FP/ (FP+TP) \quad (14)$$

The TPR and FPR results from the confusion matrix for the both phases testing and training using the ELM classifier as shown in the Fig. 13. The results are very satisfactory and the TPR and FPR for testing and training are 99.71%, 2.24%, 100%, and 1%, respectively, for complicated images. Also, the high value of the area under ROC curve (AUC) is a good measure of classifier performance (Godil et al., 2014). From Figure. 13 the AUC for the ELM classifier is very close to 1 is about 99.89% for testing phase. Thus, the performance of the ELM is quite satisfactory.

Fig. 12. Successful detection examples for complicated vehicle images with cropped LP images results under low/high light, various viewpoints, with dusk and fog, with distortion, and with low/high contrast problems.

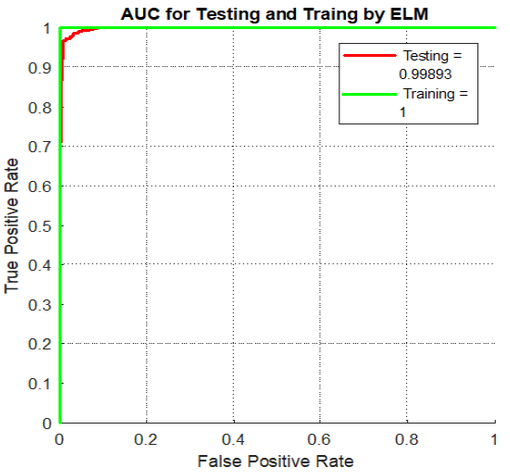


Fig. 13. The ROC curve of the ELM Classifier for training and testing phases.

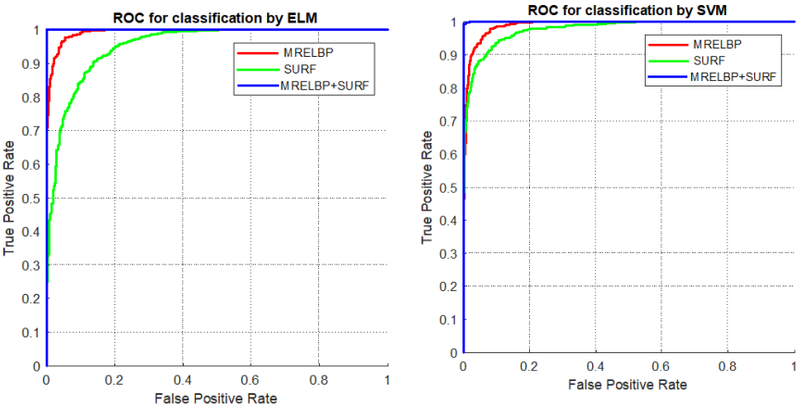
4.3 Comparison with other classifiers

This section illustrates two issues. The first one is the classification accuracy of the LPD system using MRELBP, SURF features, and hybrid features (MRELBP_SURF) for the ELM classifier. The second is the comparative analysis of the proposed LPD system with other machine classifiers, such as SVM, Adaboost, and CNN. We also tried to use the CNN algorithm as classifier and detector, but experiments showed that this algorithm does not detect objects inside an image. The CNN works very well for the recognition task. This means after the LP object is detected at the detection stage, we can use the CNN to recognize the LP as text. Table 3 shows the classification accuracy of the LPD system with MRELBP, SURF, and hybrid features, respectively. The classification accuracy based on MRELBP_SURF with the ELM classifier is outperformed.

Table 3
The average of the classification results for the proposed method with other classifiers.

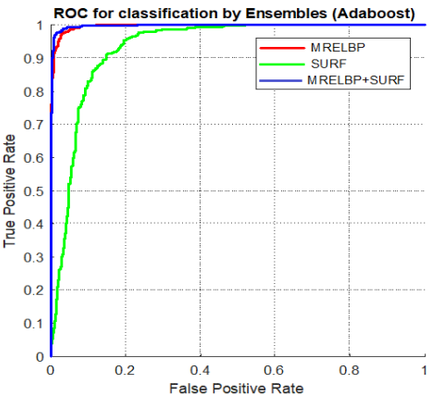
Classifier	Classification Accuracy			Classifier Parameters
	MRELBP	SURF	MRELBP_SURF	
ELM	96.0714%	89.2143%	100.0000%	Sigmoid activation function with 1000 hidden neurons.
SVM	95.8571%	91.9286%	99.5000%	Linear function with 5- fold cross validation model.
Ensembles (Adaboost)	97.7143%	86.5000%	98.2143%	Cascade classifier technique with 10 training iteration stages.

From the Table 3 it is evident that using the hybrid feature descriptors the classification accuracy for all the classifiers has been increased significantly and the ELM outperforms than SVM and Adaboost classifiers. Fig. 14 (a), (b), and (c) represents the ROC curves for the classification accuracy of the proposed system with other classifiers.



(a)

(b)



(c)

Fig. 14. The ROC curves of the proposed system with other classifiers: (a) The ROC curve for ELM classification; (b) The ROC curve for SVM classification; (c) The ROC curve for ensembles (Adaboost) classification;

5. Comparison with other methods

In this section, the proposed framework is compared with the latest LP detection methods that used the same database (Al-Shemarry et al., 2018, 2019; Azam & Gavrilova, 2017; Azam & Islam, 2016; Hasan, 2013; Panahi & Gholampour, 2017; Raghunandan et al., 2017; Silva & Jung, 2018; Wafy & Madbouly, 2016). Also, this method was compared with other methods that have different databases (Anagnostopoulos, Anagnostopoulos, Loumos, & Kayafas, 2006; M. D. A. Asif, Tariq, Baig, & Ahmad, 2014; M. R. Asif, Chun, Hussain, & Fareed, 2016; Y.-N. Chen, Han, Ho, & Fan, 2015; Z.-X. Chen, Liu, Chang, & Wang, 2009; Deb, Chae, & Jo, 2009; Deb & Jo, 2009; Duan, Duc, & Du, 2004; He, Yao, Zhang, Hou, & Han, 2014; Lee, Han, & Ko, 2013; Lee et al., 2010; Lim & Tay, 2010; Y. Luo, Li, Huang, & Han, 2018). The performance of the proposed method and those methods is evaluated using the parameters for ROC curve, the TPR and FPR rates with system runtime. Table 4 shows the performance of the proposed method with some typical methods. Al-Shemarry et al. (2018) proposed detection method for low-quality vehicle images. The method extracted the features using a three preprocessing levels for local binary pattern (3L-LBP) with the AdaBoost algorithm. At the preprocessing stage, a contrast-limited adaptive

histogram equalization method with a high standard derivation was used, which is led to increase the FPR. Azam and Gavrilova (2017) reported a genetic algorithm (GA) depending on the HOG features with a mixture of binary classifiers to classify LP and non-LP images. To improve the classification performance, the genetic algorithm was applied to select the best features subset. It achieved good results under different conditions, but it required a high-contrast preprocessing method to improve images. Hence the FPR increased. Raghunandan et al. (2017) proposed a mathematical model using a Riesz fractional operator to enhance the details of LP edges' information. That method worked well as long as it had a clear LP shape in an image. It was not robust for distorted images and it was time-consuming with a low TPR rate. Also, Al-Shemarry et al. (2019) produced a new descriptor, with multi-level extended local binary patterns (MLELBP) to extract multiple LP features under complicated conditions. The input image was resized to save the processing time. The ELM was used to build the trained model and obtain good detection results. In this method, resizing the image is not always good, especially with distorted images. It lead to the loss of important feature values and tends to reduce the TPR. Azam and Islam (2016) showed good performance techniques using several unsupervised learning methods for each LP problem. However, those techniques leads to increased FPR and are not robust if there is another LP problem that needs to be considered. Hasan (2013) had a TPR less than the proposed method by 7.01% under good conditions. Also, this method used many unsupervised learning methods for detection stage. It does not consider tilted LPs, low/high contrast, and noise problems. Panahi and Gholampour (2017) used an unsupervised learning method to consider the images affected by illumination, weather, and vehicle movement effects. However, the vehicle images are captured using specific devices, which make LPs easy to detect and recognize. Wafy and Madbouly (2016) shows a good TPR which is a little bit less than the proposed method 1.71% and 0.677ms detection time. The method using unsupervised time consuming methods and the TPR time was not reported. Silva and Jung (2018) used deep learning algorithms as good classification methods that are currently used. They just considered tilted LPs with simple conditions. The TPR and FPR are less and higher than the proposed method by 6.19%, respectively. However, the database used in this study (Al-Shemarry et al., 2019) was more complicated compared with existing studies which are used the original English car plates database. This work also compared with other existing methods that used different databases. The reason for selecting different databases, such as Chinese, Korea, Caltech cars and so on, was to compare the performance of our algorithm detecting LPs from a large amount of very distorted and complicated vehicle images. The detection time of Lee et al. (2010) method is much higher than the others due to applying time-consuming methods to detect and extract LP regions. It had a high FPR and low TPR. Anagnostopoulos et al. (2006) depended on thresholding and binarization; they required high-contrast images for achieving good results, which led to increased FPR. The memory complexity of those methods is O (N×M); where N and M is the dimension of the input tested image. Nowadays, the average of memory usage for the proposed detection method is not as big an implementation issue as existing works. From the Table 4, we can see that the proposed method has a very competitive TPR (detection accuracy) and with less system runtime. This method keeps a good balance between TPR and FPR and is more suitable for a real time automatic licence plate detection system.

Table 4
Performance comparisons of the proposed method with others in terms of the TPR, FPR, and runtime.

Ref.	Method	DS	Image condition	FPR	TPR	DT(ms)	PF
Al-Shemarry et al. (2018)	3L-LBP descriptor with AdaBoost algorithm	1030	Illumination, weather effect, low/high lighting effect, dirty	5.56%	98.56%	0.78ms	English
Azam and Gavrilova (2017)	HOG features with a mixture of binary classifiers and GA	540	Texts images background	NR	91.31%	NR	English
Raghunandan et al. (2017)	Riesz fractional operator	114	Complex background, different weather conditions, night light	7.01%	79.4%	NR	English
Hasan (2013)	Canny edge, Horizontal and Vertical edge, three stages Artificial Neural Network (ANN)	69	Simple conditions	NR	92.7%	NR	English
Azam and Islam (2016)	Frequency domain mask, contrast improvement technique, statistical binarization, Radon transform, and entropy vector.	325	Simple tilted LP and low difficult contrast night conditions	6.3%	98.15%	0.450ms	English
Wafy and Madbouly (2016)	Semi-symmetric corner points, morphological feature, linear discriminated analysis (LDA)	405	Simple conditions	NR	98%	1.00ms	English
Panahi and Gholampour (2017)	Vertical Sobel edge operator and Hough transform, ConnectedComponent Algorithm, 2L-SVM	500	Medium quality plates, dirty LPs.	NR	97%	NR	English
Al-Shemarry et al. (2019)	MLELBP_ELM	1500	640×480, various complicated scenes	5%	99.10%	0.735ms	English
Silva and Jung (2018)	Convolutional Neural Network	196	Unconstrained capture scenarios,	5%	93.52%	NR	English
Asif et al. (2016)	YDbDr color space + Otsu method	1511	Simple conditions and background is either yellow or white LPs background with black characters.	6.5%	93.86%	0.33ms	Chinese
Lee et al. (2010)	Local structure patterns, the modified census transform	580	720×486 ,various weather conditions	18%	88.9%	3.293ms	Korea
Lim and Tay (2010)	MSER + Heuristic + CVM	126		~13%	90.47%	NR	Caltech
Chen et al. (2018)	Rectangle shape, texture and color features	1176	640×480, various scenes	4.6	97.3%	0.220ms	Chinese
He et al. (2014)	Blob for candidate detection, filtering affine distortion, saliency detection, post-processing	200	Multi-scale LPs under different inclination directions	19.6%	94.7%	NR	Chinese
Anagnostopoulos et al. (2006)	Sliding concentric windows with thresholding and binarization	1334	Different scene and illumination	9%	96.5%	0.111ms	Greek
Duan et al. (2004)	Hough Transform and contour algorithm	805	800×600, different rotation and lighting conditions	1.6%	98.8%	0.650ms	Vietnamese
Luo et al. (2018)	Single shot multi-box Detector, corner points, character contours based on multi-level thresholding and binarization.	1200	800×800, various scenes, illumination and view angles	27%	96.5%	0.370ms	Taiwanese
Proposed	ELM_MRELBP_SURF	2050	Simple and distorted conditions	2.24%	99.71%	0.323ms	English

DS: Database Size; DT: Detection Time; NR: Not Reported; PF: Plate Format

6. Conclusion and future work

This study illustrates a new detection approach which is invariant to high/low contrast and lighting, rotation, blur, fog, and distortion under difficult conditions due to the use of a robust preprocessing method and strong hybrid-feature descriptors, MRELBP and SURF. A precise LPD system with low FPR is very crucial to contribute more efficiency and safety for transportation systems. In the experiments, this method achieved significant classification and detection results. The confusion matrix and the ROC curve, show that the overall classification accuracy of the ELM classifier is 99.95% and the AUC is close to the 1 for all LPs problems. The accuracy and TPR are 97.92% and 99.71, respectively, with the FPR of 2.24%. The average of the runtime for the whole detection system per vehicle image was 2.108 milliseconds. Also, the ROC curve was used to compare and evaluate the results of the proposed method with other classifiers. The classification results of the ROC curve were very good for the ELM classifier at 100.0000%. The proposed method was also compared with the existing LP detection methods that used either the same database, or a different database. It showed that the proposed method performed better than other methods in terms of the TPR and FPR. This method can improve the work of existing ANPR systems under complicated conditions. In addition, it can be applied to different types of LP data sets, such as Australian LPs and Arabic LPs. Moreover, due to using supervised learning techniques, there is no limitation in our method that relates to objects shape, color, and edge and so on. In future work, for further improvement of the proposed LPD system, we want to take account of more challenges, such as snow, rain conditions and difficult tilted LPs. We are also planning to recognize the LP number under difficult conditions. Therefore, the future target is introducing deep learning for the recognition stage. Also, the proposed system could be applied efficiently to real-time applications.

Acknowledgements

This work was supported by the University of Southern Queensland and Ministry of Higher Education and Scientific Research of Iraq.

References

Al-Shemarry, M. S., Li, Y., & Abdulla, S. (2018). Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images. *Expert Systems With Applications, 92*, 216-235.

Al-Shemarry, M. S., Li, Y., & Abdulla, S. (2019). An Efficient Texture Descriptor for the Detection of License Plates From Vehicle Images in Difficult Conditions. *IEEE Transactions on Intelligent Transportation Systems*.

Anagnostopoulos, C. N. E., Anagnostopoulos, I. E., Loumos, V., & Kayafas, E. (2006). A license plate-recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent Transportation Systems, 7*(3), 377-392.

Arróspide, J., & Salgado, L. (2014). A study of feature combination for vehicle detection based on image processing. *The Scientific World Journal, 2014*.

Ascar Davix, X., Oshin, L., & Shamili, P. (2016). License plate localization by sobel vertical edge detection method. *Int. J. Emerg. Technol. Eng. Res, 4*(6), 48-53.

Ashtari, A. H., Nordin, M. J., & Fathy, M. (2014). An Iranian license plate recognition system based on color features. *IEEE Transactions on Intelligent Transportation Systems, 15*(4), 1690-1705.

Asif, M. D. A., Tariq, U. U., Baig, M. N., & Ahmad, W. (2014). A novel hybrid method for text detection and extraction from news videos. *Middle-East Journal of Scientific Research, 19*(5), 716-722.

Asif, M. R., Chun, Q., Hussain, S., & Fareed, M. S. (2016). Multiple licence plate detection for Chinese vehicles in dense traffic scenarios. *IET Intelligent Transport Systems, 10*(8), 535-544.

Azad, R., Azad, B., & Shayegh, H. R. (2014). Real-time and efficient method for accuracy enhancement of edge based license plate recognition system. *arXiv preprint arXiv:1407.6498*.

Azam, S., & Gavrilova, M. (2017). *License plate image patch filtering using HOG descriptor and bio-inspired optimization.* Paper presented at the Proceedings of the Computer Graphics International Conference.

Azam, S., & Islam, M. M. (2016). Automatic license plate detection in hazardous condition. *Journal of Visual Communication and Image Representation, 36*, 172-186.

Bashir, F., & Porikli, F. (2006). *Performance evaluation of object detection and tracking systems.* Paper presented at the Proceedings 9th IEEE International Workshop on PETS.

Bauer, J., Sünderhauf, N., & Protzel, P. (2007). Comparing several implementations of two recently published feature detectors. *IFAC Proceedings Volumes, 40*(15), 143-148.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006a). Surf: Speeded up robust features. *Computer vision–ECCV 2006*, 404-417.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006b). *Surf: Speeded up robust features.* Paper presented at the European conference on computer vision.

Chen, Y.-N., Han, C.-C., Ho, G.-F., & Fan, K.-C. (2015). Facial/license plate detection using a two-level cascade classifier and a single convolutional feature map. *International Journal of Advanced Robotic Systems, 12*(12), 183.

Chen, Y., Zhao, D., Lv, L., & Zhang, Q. (2018). Multi-task learning for dangerous object detection in autonomous driving. *Information Sciences, 432*, 559-571.

Chen, Z.-X., Liu, C.-Y., Chang, F.-L., & Wang, G.-Y. (2009). Automatic license-plate location and recognition based on feature salience. *IEEE Transactions on Vehicular Technology, 58*(7), 3781-3785.

Deb, K., Chae, H.-U., & Jo, K.-H. (2009). Vehicle License Plate Detection Method Based on Sliding Concentric Windows and Histogram. *JCP, 4*(8), 771-777.

Deb, K., & Jo, K.-H. (2009). A vehicle license plate detection method for intelligent transportation system applications. *Cybernetics and Systems: An International Journal, 40*(8), 689-705.

Duan, T. D., Duc, D. A., & Du, T. L. H. (2004). *Combining Hough transform and contour algorithm for detecting vehicles' license-plates.* Paper presented at the Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.

El Khadiri, I., Kas, M., El Merabet, Y., Ruichek, Y., & Touahni, R. (2018). Repulsive-and-attractive local binary gradient contours: New and efficient feature descriptors for texture classification. *Information Sciences, 467*, 634-653.

EnglishLPDatabase-2001. http://www.zemris.fer.hr/projects/LicensePlates/english/. *accessed July 2016*.

Godil, A., Bostelman, R., Shackleford, W., Hong, T., & Shneier, M. (2014). *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. Retrieved from

Gunn, S. R. (1998). Support vector machines for classification and regression. *ISIS technical report, 14*(1), 5-16.

Guo, Z., Wang, X., Zhou, J., & You, J. (2016). Robust texture image representation by scale selective local binary patterns. *IEEE Transactions on Image Processing, 25*(2), 687-699.

Guo, Z., Zhang, L., & Zhang, D. (2010). A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing, 19*(6), 1657-1663.

Ha, P. S., & Shakeri, M. (2016). *License Plate Automatic Recognition based on edge detection.* Paper presented at the 2016 Artificial Intelligence and Robotics (IRANOPEN).

Han, Y., Virupakshappa, K., & Oruklu, E. (2015). *Robust traffic sign recognition with feature extraction and k-NN classification methods.* Paper presented at the 2015 IEEE International Conference on Electro/Information Technology (EIT).

Hanbay, K., Alpaslan, N., Talu, M. F., & Hanbay, D. (2016). Principal curvatures based rotation invariant algorithms for efficient texture classification. *Neurocomputing, 199*, 77-89.

Hasan, M. (2013). Real Time Detection and Recognition of License Plate in Bengali. In.

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class adaboost. *Statistics and its Interface, 2*(3), 349-360.

He, T., Yao, J., Zhang, K., Hou, Y., & Han, S. (2014). *Accurate multi-scale license plate localization via image saliency.* Paper presented at the Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on.

Hu, Y., Long, Z., & AlRegib, G. (2016). *Completed local derivative pattern for rotation invariant texture classification.* Paper presented at the Image Processing (ICIP), 2016 IEEE International Conference on.

Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: theory and applications. *Neurocomputing, 70*(1-3), 489-501.

Huang, G., Huang, G.-B., Song, S., & You, K. (2015). Trends in extreme learning machines: A review. *Neural Networks, 61*, 32-48.

Kasturi, R., Goldgof, D., Soundararajan, P., Manohar, V., Garofolo, J., Bowers, R., . . . Zhang, J. (2009). Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol. *IEEE Transactions on pattern analysis and machine intelligence, 31*(2), 319-336.

Khan, N. Y., McCane, B., & Wyvill, G. (2011). *SIFT and SURF performance evaluation against various image deformations on benchmark dataset.* Paper presented at the Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on.

Kim, C., Song, D., Kim, C.-S., & Park, S.-K. (2019). Object tracking under large motion: Combining coarse-to-fine search with superpixels. *Information Sciences, 480*, 194-210.

Lalimi, M. A., Ghofrani, S., & McLernon, D. (2013). A vehicle license plate detection method using region and edge based methods. *Computers & Electrical Engineering, 39*(3), 834-845.

Lee, Y., Han, D. K., & Ko, H. (2013). Reinforced adaboost learning for object detection with local pattern representations. *The Scientific World Journal, 2013*.

Lee, Y., Song, T., Ku, B., Jeon, S., Han, D. K., & Ko, H. (2010). *License plate detection using local structure patterns.* Paper presented at the Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on.

Li, D., & Wang, Z. (2016). *A character-based method for license plate detection in complex scenes.* Paper presented at the Chinese Conference on Pattern Recognition.

Lim, H. W., & Tay, Y. H. (2010). *Detection of license plate characters in natural scene with MSER and SIFT unigram classifier.* Paper presented at the Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2010 IEEE Conference on.

Liu, L., Fieguth, P., Pietikäinen, M., & Lao, S. (2015). *Median robust extended local binary pattern for texture classification.* Paper presented at the Image Processing (ICIP), 2015 IEEE International Conference on.

Liu, L., Lao, S., Fieguth, P. W., Guo, Y., Wang, X., & Pietikäinen, M. (2016). Median robust extended local binary pattern for texture classification. *IEEE Transactions on Image Processing, 25*(3), 1368-1381.

Liu, L., Zhao, L., Long, Y., Kuang, G., & Fieguth, P. (2012). Extended local binary patterns for texture classification. *Image and Vision Computing, 30*(2), 86-99.

Luo, Y., Li, Y., Huang, S., & Han, F. (2018). *Multiple Chinese Vehicle License Plate Localization in Complex Scenes.* Paper presented at the 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC).

Luo, Z., Chen, J., Takiguchi, T., & Ariki, Y. (2015). *Rotation-invariant histograms of oriented gradients for local patch robust representation.* Paper presented at the Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific.

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition, 29*(1), 51-59.

Panahi, R., & Gholampour, I. (2017). Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications. *IEEE Transactions on Intelligent Transportation Systems, 18*(4), 767-779.

Panchal, P., Panchal, S., & Shah, S. (2013). A comparison of SIFT and SURF. *International Journal of Innovative Research in Computer and Communication Engineering, 1*(2), 323-327.

Pang, Y., Li, W., Yuan, Y., & Pan, J. (2012). Fully affine invariant SURF for image matching. *Neurocomputing, 85*, 6-10.

Pao, Y.-H., & Takefuji, Y. (1992). Functional-link net computing: theory, system architecture, and functionalities. *Computer, 25*(5), 76-79.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., . . . Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing, 39*(3), 355-368.

Raghunandan, K., Shivakumara, P., Jalab, H. A., Ibrahim, R. W., Kumar, G. H., Pal, U., & Lu, T. (2017). Riesz fractional based model for enhancing license plate detection and recognition. *IEEE Transactions on Circuits and Systems for Video Technology*.

Rajesh, G. K., Kaushik, R., & Jangra, R. (2016). A Comparative Analysis of Object Recognition System Using SIFT, SURF and FAST Algorithms. *IJITR, 4*(3), 3025-3032.

Saravanan, C. (2010). *Color image to grayscale image conversion.* Paper presented at the 2010 Second International Conference on Computer Engineering and Applications.

Scardapane, S., Wang, D., Panella, M., & Uncini, A. (2015). Distributed learning for random vector functional-link networks. *Information Sciences, 301*, 271-284.

Shi, X., Zhao, W., & Shen, Y. (2005). *Automatic license plate recognition system based on color image processing.* Paper presented at the International Conference on Computational Science and Its Applications.

Silva, S. M., & Jung, C. R. (2018). *License Plate Detection and Recognition in Unconstrained Scenarios.* Paper presented at the European Conference on Computer Vision.

Slaby, A. (2007). *ROC analysis with Matlab.* Paper presented at the Information Technology Interfaces, 2007. ITI 2007. 29th International Conference on.

Soora, N. R., & Deshpande, P. S. (2016). Color, scale, and rotation independent multiple license plates detection in videos and still images. *Mathematical Problems in Engineering, 2016*.

Tadic, V., Popovic, M., & Odry, P. (2016). Fuzzified Gabor filter for license plate detection. *Engineering Applications of Artificial Intelligence, 48*, 40-58.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1), 1*(511-518), 3.

Wafy, M., & Madbouly, A. M. (2016). Efficient method for vehicle license plate identification based on learning a morphological feature. *IET Intelligent Transport Systems, 10*(6), 389-395.

Wang, J., Bacic, B., & Yan, W. Q. (2018). An effective method for plate number recognition. *Multimedia Tools and Applications, 77*(2), 1679-1692.

Wang, Y., Zhang, H., Fang, X., & Guo, J. (2009). *Low-resolution Chinese character recognition of vehicle license plate based on ALBP and Gabor filters.* Paper presented at the Advances in Pattern Recognition, 2009. ICAPR'09. Seventh International Conference on.

Xie, L., Ahmad, T., Jin, L., Liu, Y., & Zhang, S. (2018). A New CNN-Based Method for Multi-Directional Car License Plate Detection. *IEEE Transactions on Intelligent Transportation Systems, 19*(2), 507-517.

Zhang, H., Jia, W., He, X., & Wu, Q. (2006). *Learning-based license plate detection using global and local features.* Paper presented at the Pattern Recognition, 2006. ICPR 2006. 18th International Conference on.

Zhang, L., Shi, X., Xia, Y., & Mao, K. (2013). *A multi-filter based license plate localization and recognition framework.* Paper presented at the 2013 Ninth International Conference on Natural Computation (Icnc).

Meeras Salman Al-Shemarry is a Lecturer in Computer Department, Science College, Karbala University, Iraq. She has a Bachelor of Computer Science from Babylon University, Iraq in 2002. She received her Master degree in IT 2010 from University Utara Malaysia (UUM), Malaysia. Currently, she is a PhD Student in the Faculty of Health, Engineering and Sciences, University of Southern Queensland, Australia. Her research interests include system

analysis using UML diagrams, image processing and objects detection, artificial intelligence, database management system.

Yan Li is currently a Professor in Computer Science in the school of sciences, University of Southern Queensland, Australia. Her research interests are in the areas of Artificial Intelligence, Big Data Analytics, Signal and Image Processing, Biomedical Engineering, Artificial Intelligence, Big Data Analytics and Computer Networking Technologies.

7

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Summary and Conclusions of the Thesis

An LPD system is an important application for our daily life to contribute more efficiency and safety in ITSs. It has great potential in helping to monitor road traffic for law enforcement activities. The detection of an LP number from low-quality vehicles images is the main improvement in performance needed for existing detection systems. Many researchers have developed various methods for enhancing ITSs. Identification of various conditions of LPs is a complicated issue, requiring the collection of large data sets. Finding representative multi-features from a large data set plays an important role in identifying LPs. In this thesis, the performance of the proposed LPD system was improved through three main ways:

1. Developing effective methods for detecting LPs under complicated conditions, such as low/high contrast, bad illumination, foggy, dusty, and distorted by high speed and bad weather. They improved the detection system performance with less execution time and good false positive rate.
2. Improving the developed methods by presenting new preprocessing and extraction techniques that can improve the classification accuracy.
3. Investigating which method is better to achieve the main requirements of an LPD system under difficult conditions like distorted vehicle images, low/high contrast, and bad illumination.

To achieve these objectives and answer research questions, four methods were developed, based on different types of texture descriptors: a local binary patterns (LBPs), extended local binary pattern (ELBP) based preprocessing methods, a median robust extended local binary pattern (MRELBP), a median filter histogram of oriented gradient (MHOG), and the speeded-

up a robust feature (SURF) with three supervised learning algorithms, Adaboost, extreme learning machine (ELM), and support vector machine (SVM), were developed in this research. In the following subsections, a summary of those proposed methods is provided.

7.1.1 A three-level features extraction based on LBP descriptor using Adaboost learning algorithm

This method includes two phases: testing and training. At each phase, the same preprocessing and extraction methods were used to capture different types of complicated LPs features (see Chapter 3). Al-Shemarry et al. (2018) applied the concept of an ensemble of cascade Adaboost classifiers to learn the extracted features of the LP due to its discriminative power. The strong cascade classifier contains a large number of weak classifiers to classify three-level extracted LBP (3L-LBP) features which include a LBP grayscale features, a LBP filtered features, and a contrast LBP features. In this study, the texture descriptor LBP is selected to extract key features from low-quality images due to its advantages. To test the effectiveness of this method, it was implemented with 1030 vehicles images, each having 640×480 resolution with difficult conditions, such as low/high contrast, foggy, tilted LP, and distortions. From the experimental results, the overall performance evaluation for detection, precision, and F-measure rates are 98.56%, 95.9%, and 97.19%, respectively, with an FPR of 5.6%. This method was compared against existing LPD methods presented in the literature. It outperforms those methods that used the same databases, in terms of detection accuracy and execution time under difficult conditions. The average detection time for the whole system per vehicle image was 2.001ms. Moreover, the proposed method works without any limitations due to the use of testing and learning phases with a texture descriptor. Many vehicle images in the database include commercial signs and logos which lead to increased FP values and take more processing time.

7.1.2 A multi-preprocessing extraction level using ELBP descriptor based on an ELM classifier

A new extraction technique was developed using multi-preprocessing levels based on ELBP descriptor, MLELBP, to extract different LP features. The preprocessing steps use a Gaussian filter and CLAHE method to improve LP images and capture more complicated features. Those steps improved the classification performance for the LPD system (see Chapter 4). This work successfully reduced the extracted features dimension as well as FP values. The extracted features provide the input data to an ELM classifier to make a decision about the LP regions,

if it is LP or non-LP. At the evaluation stage, this method was tested on further distorted images (unseen data) taken under difficult conditions, such as low/high contrast, foggy, and rotated LPs. The MLELBP_ELM method has three main advantages compared to the 3L-LBP_Adaboost method. The first one is that increasing the size of the training dataset through the preprocessing stage in order to capture more key features from the LP region. The second increases the size of a testing dataset through using an online photo editor application to reflect various difficult conditions for vehicle images. The third is that the detection accuracy and FPR were improved by 0.54% and 0.56%, respectively. The classification and detection rates are 99.78% and 99.10%, respectively, with an FPR of 5%. The average execution time for the whole detection system per vehicle image was 2.4530ms. This method was compared with several existing LPD methods that used the same database. The experimental results showed that the MLELBP_ELM method can produce better results than the 3L-LBP_Adaboost method. The findings indicate that this method is superior in the classification performance over most existing methods under complicated conditions. It can help provide more useful information about complicated LP images to improve an LPD system's performance.

7.1.3 The LBP_MHOG descriptors based on the SVM classifier

In this section, a new preprocessing technique was proposed for improving vehicle images as well as reducing the extraction time. It included the combination of a Gaussian filter and the ECHE technique with the CLAHE algorithm. The MHOG and LBP descriptors were used to extract more difficult representative LP features. Then, the SVM was used to classify the extracted features. The LBP_MHOG_SVM method was introduced to improve an LPD system performance (see Chapter 5). This method tested on an English car LP database, which has three types of LP resolutions, 25×100, 50×200, and 75×300. Therefore, an ensemble of strong detectors or trained models as developed. The performance of this method was evaluated through the 5-fold cross-validation procedure. The LBP_HOG_SVM method was compared in terms of the FPR and running time with the 3L-LBP_Adaboos and MLELBP_ELM methods. It yielded an excellent improvement over existing methods, a 4% improvement for the FPR and 1.50% for accuracy with execution time. Also, this method was compared with other newest existing methods in the literature for the same database using the detection and object localization metrics. The ROC curve also was used to compare and evaluate the results of the proposed method with the ELM classifier. The overall performance evaluation for the object localization metrics of the detection or recall rate is 99.62%, with an FPR of 1.675%. The

average of the runtime for the whole detection system per vehicle image was 2.2187ms. The experimental results demonstrated that the proposed technique could be applied efficiently for real-time applications. Also, this method can help improve the work of existing ANPR systems under complicated conditions.

7.1.4 The MRELBP_SURF features based on an ELM classifier

In order to achieve a good system performance, a new detection approach was developed for detecting distorted LP images. The modified preprocessing version, ECLACHE, of the ECHE_CLAHE method was used in this work. Also, a recently developed texture descriptor, MRELBP with SURF, was used to extract complicated features. Then, the extracted LP features were used as input to the ELM classifier to produce a strong detector (see Chapter 6). Through the experiments, the ELM classifier works very well with different types of texture descriptors. Using the previous method (MHOG_LBP_SVM), the ELM takes more time to classify HOG features. From the confusion matrix and the ROC curve, there is evidence that the overall classification accuracy of the ELM classifier is 99.95% and the AUC are close to 1 for all complicated LPs images. The accuracy and detection rates are 97.92% and 99.71, respectively, with the FPR of 2.24%. The average runtime for the whole detection system per vehicle image was 2.108 seconds. Furthermore, the MRELBP_SURF_ELM based approach can correctly identify the discriminative LP regions correctly and efficiently. The method was superior in the performance and execution time over the most existing algorithms as well as other proposed methods in this research.

Taken together, it can be concluded that the research presented in this thesis has found new robust successful methods for reliable detection of LPs from low-quality vehicles images under difficult conditions. These techniques can improve the performance of the existing ANPR systems under complicated conditions. The outcomes will contribute to increasing the quality of transport systems with better efficiency and safety.

7.2 Future work

The approaches presented in this thesis provide good performance in the LP detection under difficult conditions. The future work will investigate the possibility of using those methods to improve ANPR applications. To facilitate the further development of this work, a few key areas below have been explored.

Concerning the 3L-LBP_Adaboost and MLELBP_ELM algorithms, they can be improved to further reduce the false positive rate and extraction time using the preprocessing techniques for both phases of testing and training. In regards to the dimensionality reduction for the extracted features, a Gaussian filter, median filter, CLAHE, ECHE, and ECLACHE techniques have been used.

One future improvement could be to eliminate those LP objects that look like the LP and have the same characteristics as LP regions, such as texts or commercial signs and logo objects. This step would decrease the processing time as well as the memory required to process the LP detection task.

In addition, using a combination of several supervised machine learning algorithms instead of a single one is very efficient. This is a preferable solution for capturing more information about the LP area and increase the detection system accuracy. Also, in the detection stage using an ensemble of the classifiers could improve the classification accuracy and the efficiency of the trained models compared with using a single classifier.

The study in this thesis has shown that selecting supervised machine learning algorithms to identify and classify the extracted features for the complicated LP is an extremely challenging task. The quality of detection results depends on how the extraction and classification algorithms are selected and developed. The detection algorithms are mostly evaluated using multiple criteria, such as recall (detection) rate or true positive rate, false positive rate, precision rate, f-measure rate, the accuracy rate, and the receiver operating characteristic curve.

Those methods can be applied to different types of LP datasets, such as Australian car LPs, Arabic car LPs, and so on. More generally the proposed methods could be used by other fields that are related to objects detection subjects. Due to using supervised learning techniques, there is no limitation in those methods which are associated with objects shape, color, and edge and so on.

Further study is required to take account of other challenges and to enhance this work for dealing with other difficult conditions, such as licence plates with difficult tilt, rain, and snow in images. The detected LPs are normally stored as images in the memory and used by transportation systems to complete their tasks. This needs more storage devices, therefore, the

LP recognition stage is required. This stage works to recognize the LP number as a text. This is a very easy task to do using deep learning algorithms and template matching techniques with optical character recognition (OCR).

This thesis studied offline detection methods, but it is desirable for this work to be applied to real online LPD systems to see the impact of this research. This will require more work. Therefore, all of the proposed methods need to be employed for online detection. This would be a significant achievement in the field of transport systems for work under difficult conditions.

References

Al-Ghaili, AM, Mashohor, S, Ismail, A & Ramli, AR 2008, 'A new vertical edge detection algorithm and its application', in *2008 International Conference on Computer Engineering & Systems*, IEEE, pp. 204-9.

Al-Ghaili, AM, Mashohor, S, Ramli, AR & Ismail, A 2012, 'Vertical-edge-based car-license-plate detection method', *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 26-38.

Al-Shemarry, MS, Li, Y & Abdulla, S 2018, 'Ensemble of adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images', *Expert Systems With Applications*, vol. 92, pp. 216-35.

Al-Shemarry, MS, Li, Y & Abdulla, S 2019, 'An Efficient Texture Descriptor for the Detection of License Plates From Vehicle Images in Difficult Conditions', *IEEE Transactions on Intelligent Transportation Systems*.

Anagnostopoulos, C-NE 2014, 'License plate recognition: A brief tutorial', *IEEE Intelligent transportation systems magazine*, vol. 6, no. 1, pp. 59-67.

Anagnostopoulos, C-NE, Anagnostopoulos, IE, Psoroulas, ID, Loumos, V & Kayafas, E 2008, 'License plate recognition from still images and video sequences: A survey', *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377-91.

Anagnostopoulos, CNE, Anagnostopoulos, IE, Loumos, V & Kayafas, E 2006, 'A license plate-recognition algorithm for intelligent transportation system applications', *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 377-92.

Angelova, A, Krizhevsky, A, Vanhoucke, V, Ogale, AS & Ferguson, D 2015, 'Real-Time Pedestrian Detection with Deep Network Cascades', in *BMVC*, pp. 32.1-.12.

Arafat, MY, Khairuddin, ASM, Khairuddin, U & Paramesran, R 2019, 'Systematic review on vehicular licence plate recognition framework in intelligent transport systems', *IET Intelligent Transport Systems*.

Asif, MR, Chun, Q, Hussain, S & Fareed, MS 2016, 'Multiple licence plate detection for Chinese vehicles in dense traffic scenarios', *IET Intelligent Transport Systems*, vol. 10, no. 8, pp. 535-44.

Azam, S & Islam, MM 2016, 'Automatic license plate detection in hazardous condition', *Journal of Visual Communication and Image Representation*, vol. 36, pp. 172-86.

Azam, S & Gavrilova, M 2017, 'License plate image patch filtering using HOG descriptor and bio-inspired optimization', in *Proceedings of the Computer Graphics International Conference*, ACM, p. 1.

Baharlou, SM, Hemayat, S, Saberkari, A & Yaghoobi, S 2015, 'Fast and adaptive license plate recognition algorithm for Persian plates', in *Pattern Recognition and Image Analysis (IPRIA), 2015 2nd International Conference on*, IEEE, pp. 1-6.

Baohua, Z, Dahua, Y, Hongmei, H & Lanying, G 2010, 'License plate location algorithm based on histogram equalization', in *2010 International Conference On Computer Design and Applications*, IEEE, pp. V1-517-V1-9.

Barlow, HB 1989, 'Unsupervised learning', *Neural computation*, vol. 1, no. 3, pp. 295-311.

Bashir, F & Porikli, F 2006, 'Performance evaluation of object detection and tracking systems', in *Proceedings 9th IEEE International Workshop on PETS*, pp. 7-14.

Beibut, A, Magzhan, K & Chingiz, K 2014, 'Effective algorithms and methods for automatic number plate recognition', in *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*, IEEE, pp. 1-4.

Bellas, N, Chai, SM, Dwyer, M & Linzmeier, D 2006, 'FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators', in *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, IEEE, p. 8 pp.

Boonsim, N & Prakoonwit, S 2016, 'Car make and model recognition under limited lighting conditions at night', *Pattern Analysis and Applications*, pp. 1-13.

Brunelli, R 2009, *Template matching techniques in computer vision: theory and practice*, John Wiley & Sons.

Caner, H, Gecim, HS & Alkar, AZ 2008, 'Efficient embedded neural-network-based license plate recognition system', *IEEE Transactions on Vehicular Technology*, vol. 57, no. 5, pp. 2675-83.

Castello, P, Coelho, C, Del Ninno, E, Ottaviani, E & Zanini, M 1999, 'Traffic monitoring in motorways by real-time number plate recognition', in *Image Analysis and Processing, 1999. Proceedings. International Conference on*, IEEE, pp. 1128-31.

Chacon, MI & Zimmerman, A 2003, 'License plate location based on a dynamic PCNN scheme', in *Proc. Int. Joint Conf. Neural Netw*, pp. 1195-200.

Chapelle, O, Scholkopf, B & Zien, A 2009, 'Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]', *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542-.

Chen, H-K, Zhao, X-G, Sun, S-Y & Tan, M 2017, 'PLS-CCA heterogeneous features fusion-based low-resolution human detection method for outdoor video surveillance', *International Journal of Automation and Computing*, vol. 14, no. 2, pp. 136-46.

Chen, Y-N, Han, C-C, Ho, G-F & Fan, K-C 2015, 'Facial/license plate detection using a two-level cascade classifier and a single convolutional feature map', *International Journal of Advanced Robotic Systems*, vol. 12, no. 12, p. 183.

Cho, B, Ryu, S, Shin, D & Jung, J 2011, 'License plate extraction method for identification of vehicle violations at a railway level crossing', *International Journal of Automotive Technology*, vol. 12, no. 2, pp. 281-9.

Cortes, C & Vapnik, V 1995, 'Support-vector networks', *Machine learning*, vol. 20, no. 3, pp. 273-97.

Davis, AM, Arunvinodh, C & Np, AM 2015, 'Automatic license plate detection using vertical edge detection method', in *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, IEEE, pp. 1-6.

Deb, K, Chae, H-U & Jo, K-H 2009, 'Vehicle License Plate Detection Method Based on Sliding Concentric Windows and Histogram', *JCP*, vol. 4, no. 8, pp. 771-7.

Dehghan, A, Masood, SZ, Shu, G & Ortiz, E 2017, 'View independent vehicle make, model and color recognition using convolutional neural network', *arXiv preprint arXiv:1702.01721*.

Ding, S, Guo, L & Hou, Y 2017, 'Extreme learning machine with kernel model based on deep

learning', *Neural Computing and Applications*, vol. 28, no. 8, pp. 1975-84.

Du, S, Ibrahim, M, Shehata, M & Badawy, W 2013, '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. 311-25.

Duan, TD, Duc, DA & Du, TLH 2004a, 'Combining Hough transform and contour algorithm for detecting vehicles' license-plates', in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, IEEE, pp. 747-50.

Duan, TD, Duc, DA & Du, TLH 2004b, 'Combining Hough transform and contour algorithm for detecting vehicles' license-plates', in *Intelligent Multimedia, Video and Speech Processing, 2004. Proceedings of 2004 International Symposium on*, IEEE, pp. 747-50.

Duan, TD, Du, TH, Phuoc, TV & Hoang, NV 2005, 'Building an automatic vehicle license plate recognition system', in *Proc. Int. Conf. Comput. Sci. RIVF*, pp. 59-63.

Duda, RO, Hart, PE & Stork, DG 2012, *Pattern classification*, John Wiley & Sons.

EnglishLPDatabase-2001 ' http://www.zemris.fer.hr/projects/LicensePlates/english/', *accessed July 2016*.

Faradji, F, Rezaie, AH & Ziaratban, M 2007, 'A morphological-based license plate location', in *2007 IEEE International Conference on Image Processing*, IEEE, pp. I-57-I-60.

Freund, Y & Schapire, RE 1995, 'A desicion-theoretic generalization of on-line learning and an application to boosting', in *European conference on computational learning theory*, Springer, pp. 23-37.

Friedman, N, Geiger, D & Goldszmidt, M 1997, 'Bayesian network classifiers', *Machine learning*, vol. 29, no. 2-3, pp. 131-63.

Fukunaga, K 2013, *Introduction to statistical pattern recognition*, Elsevier.

Gasser, T & Müller, H-G 1979, 'Kernel estimation of regression functions', in *Smoothing techniques for curve estimation*, Springer, pp. 23-68.

Gerber, C & Chung, M 2016, 'Number Plate Detection with a Multi-Convolutional Neural Network Approach with Optical Character Recognition for Mobile Devices', *Journal of Information Processing Systems*, vol. 12, no. 1.

Gou, C, Wang, K, Yu, Z & Xie, H 2014, 'License plate recognition using MSER and HOG based on ELM', in *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, IEEE, pp. 217-21.

Hartigan, JA & Wong, MA 1979, 'Algorithm AS 136: A k-means clustering algorithm', *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100-8.

Haykin, S 1994, *Neural networks: a comprehensive foundation*, Prentice Hall PTR.

He, Y, Ma, X, Luo, X, Li, J, Zhao, M, An, B & Guan, X 2017, 'Vehicle traffic driven camera placement for better metropolis security surveillance', *arXiv preprint arXiv:1705.08508*.

Ho, WT, Lim, HW & Tay, YH 2009, 'Two-stage license plate detection using gentle Adaboost and SIFT-SVM', in *Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on*, IEEE, pp. 109-14.

Hongliang, B & Changping, L 2004a, 'A hybrid license plate extraction method based on edge

statistics and morphology', in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, IEEE, pp. 831-4.

Hongliang, B & Changping, L 2004b, 'A hybrid license plate extraction method based on edge statistics and morphology', in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, pp. 831-4.

Hontani, H & Koga, T 2001, 'Character extraction method without prior knowledge on size and position information', in *IVEC2001. Proceedings of the IEEE International Vehicle Electronics Conference 2001. IVEC 2001 (Cat. No. 01EX522)*, IEEE, pp. 67-72.

Hosmer Jr, DW, Lemeshow, S & Sturdivant, RX 2013, *Applied logistic regression*, vol. 398, John Wiley & Sons.

Huang, G-B, Zhu, Q-Y & Siew, C-K 2004, 'Extreme learning machine: a new learning scheme of feedforward neural networks', in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, IEEE, pp. 985-90.

Hyvärinen, A & Oja, E 2000, 'Independent component analysis: algorithms and applications', *Neural Networks*, vol. 13, no. 4-5, pp. 411-30.

Johnson, SC 1967, 'Hierarchical clustering schemes', *Psychometrika*, vol. 32, no. 3, pp. 241-54.

Jolliffe, I 2011, *Principal component analysis*, Springer.

Kamat, V & Ganesan, S 1995, 'An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S', in *Proceedings Real-Time Technology and Applications Symposium*, IEEE, pp. 58-9.

Kanayama, K, Fujikawa, Y, Fujimoto, K & Horino, M 1991, 'Development of vehicle-license number recognition system using real-time image processing and its application to travel-time measurement', in *[1991 Proceedings] 41st IEEE Vehicular Technology Conference*, IEEE, pp. 798-804.

Karwal, H & Girdhar, A 2015, 'Vehicle number plate detection system for indian vehicles', in *2015 IEEE International Conference on Computational Intelligence & Communication Technology*, IEEE, pp. 8-12.

Kasaei, SH, Kasaei, SM & Kasaei, SA 2010, 'New Morphology-Based Method for RobustIranian Car Plate Detection and Recognition', *International Journal of Computer Theory and Engineering*, vol. 2, no. 2, p. 264.

Kasturi, R, Goldgof, D, Soundararajan, P, Manohar, V, Garofolo, J, Bowers, R, Boonstra, M, Korzhova, V & Zhang, J 2009, 'Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol', *IEEE Transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 319-36.

Kaur, S & Kaur, S 2014, 'An efficient approach for number plate extraction from vehicles image under image processing', *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 2954-9.

Keller, JM, Gray, MR & Givens, JA 1985, 'A fuzzy k-nearest neighbor algorithm', *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580-5.

Kim, D-S & Chien, S-I 2001, 'Automatic car license plate extraction using modified generalized symmetry transform and image warping', in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*,

IEEE, pp. 2022-7.

Kusakunniran, W, Ngamaschariyakul, K, Chantaraviwat, C, Janvittayanuchit, K & Thongkanchorn, K 2014, 'A Thai license plate localization using SVM', in *Computer Science and Engineering Conference (ICSEC), 2014 International*, IEEE, pp. 163-7.

Le, W & Li, S 2006, 'A hybrid license plate extraction method for complex scenes', in *18th International Conference on Pattern Recognition (ICPR'06)*, IEEE, pp. 324-7.

Lee, ER, Kim, PK & Kim, HJ 1994, 'Automatic recognition of a car license plate using color image processing', in *Proceedings of 1st International Conference on Image Processing*, IEEE, pp. 301-5.

Lee, H-J, Chen, S-Y & Wang, S-Z 2004, 'Extraction and recognition of license plates of motorcycles and vehicles on highways', in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, IEEE, pp. 356-9.

Lee, Y, Han, DK & Ko, H 2013, 'Reinforced adaboost learning for object detection with local pattern representations', *The Scientific World Journal*, vol. 2013.

Li, H & Shen, C 2016, 'Reading car license plates using deep convolutional neural networks and LSTMs', *arXiv preprint arXiv:1601.05610*.

Liu, L, Lao, S, Fieguth, PW, Guo, Y, Wang, X & Pietikäinen, M 2016, 'Median robust extended local binary pattern for texture classification', *IEEE Transactions on Image Processing*, vol. 25, no. 3, pp. 1368-81.

Liu, Y, Huang, H, Cao, J & Huang, T 2017, 'Convolutional neural networks-based intelligent recognition of Chinese license plates', *Soft Computing*, pp. 1-17.

Liu, Y, Huang, H, Cao, J & Huang, T 2018, 'Convolutional neural networks-based intelligent recognition of Chinese license plates', *Soft Computing*, vol. 22, no. 7, pp. 2403-19.

Makinacı, M 2005, 'Support vector machine approach for classification of cancerous prostate regions', *World Academy of Science, Engineering and Technology*, pp. 166-9.

Mao, S, Huang, X & Wang, M 2010, 'An adaptive method for Chinese license plate location', in *2010 8th World Congress on Intelligent Control and Automation*, IEEE, pp. 6173-7.

Masood, SZ, Shu, G, Dehghan, A & Ortiz, EG 2017, 'License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks', *arXiv preprint arXiv:1703.07330*.

Matas, J & Zimmermann, K 2005, 'Unconstrained licence plate and text localization and recognition', in *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, IEEE, pp. 225-30.

MedialabLPRdatabase-2007 'http://www.medialab.ntua.gr/research/LPRdatabase.html', *accessed July 2016*.

Miyamoto, K, Nagano, K, Tamagawa, M, Fujita, I & Yamamoto, M 1991, 'Vehicle license-plate recognition by image analysis', in *Proceedings IECON'91: 1991 International Conference on Industrial Electronics, Control and Instrumentation*, IEEE, pp. 1734-8.

Mohri, M, Rostamizadeh, A & Talwalkar, A 2018, *Foundations of machine learning*, MIT press.

Muhammad, J & Altun, H 2016a, 'Improved license plate detection using HOG-based features

and genetic algorithm', in *2016 24th Signal Processing and Communication Application Conference (SIU)*, IEEE, pp. 1269-72.

Muhammad, J & Altun, H 2016b, 'Improved license plate detection using HOG-based features and genetic algorithm', in *Signal Processing and Communication Application Conference (SIU), 2016 24th*, IEEE, pp. 1269-72.

News, B 2005, 'CCTV network tracks 'getaway' car''. BBC News. 21 November 2005. Retrieved 12 August 2018.

Oberski, D 2016, 'Mixture models: Latent profile and latent class analysis', in *Modern statistical methods for HCI*, Springer, pp. 275-87.

Pan, L & Li, S 2010, 'A new license plate extraction framework based on fast mean shift', in *International Conference on Image Processing and Pattern Recognition in Industrial Engineering*, International Society for Optics and Photonics, p. 782007.

Panahi, R & Gholampour, I 2017, 'Accurate detection and recognition of dirty vehicle plate numbers for high-speed applications', *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 767-79.

Parisi, R, Di Claudio, E, Lucarelli, G & Orlandi, G 1998, 'Car plate recognition by neural networks and image processing', in *ISCAS'98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (Cat. No. 98CH36187)*, IEEE, pp. 195-8.

Park, SH, Kim, KI, Jung, K & Kim, HJ 1999, 'Locating car license plates using neural networks', *Electronics Letters*, vol. 35, no. 17, pp. 1475-7.

Patel, C, Shah, D & Patel, A 2013, 'Automatic number plate recognition system (anpr): A survey', *International Journal of Computer Applications*, vol. 69, no. 9.

Porikli, F & Kocak, T 2006, 'Robust license plate detection using covariance descriptor in a neural network framework', in *Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*, IEEE, pp. 107-.

Prabhakar, P, Anupama, P & Resmi, S 2014, 'Automatic vehicle number plate detection and recognition', in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, IEEE, pp. 185-90.

Puloria, K & Mahajan, S 2015, 'A Review on Automatic Number Plate Recognition System', *International Journal of Software and Hardware Research in Engineering*, vol. 3, no. 1.

Qin, Z, Shi, S, Xu, J & Fu, H 2006, 'Method of license plate location based on corner feature', in *2006 6th World Congress on Intelligent Control and Automation*, IEEE, pp. 8645-9.

Quinlan, JR 1986, 'Induction of decision trees', *Machine learning*, vol. 1, no. 1, pp. 81-106.

Rabiner, LR & Juang, B-H 1986, 'An introduction to hidden Markov models', *ieee assp magazine*, vol. 3, no. 1, pp. 4-16.

Roberts, DJ & Casanova, M 2012, *Automated license plate recognition systems: Policy and operational guidance for law enforcement*.

Samra, GA & Khalefah, F 2013, 'Localization of license plate number using dynamic image processing techniques and genetic algorithms', *IEEE transactions on evolutionary computation*, vol. 18, no. 2, pp. 244-57.

Sanyuan, Z, Mingli, Z & Xiuzi, Y 2004, 'Car plate character extraction under complicated

environment', in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, IEEE, pp. 4722-6.

Sarfraz, M, Ahmed, MJ & Ghazi, SA 2003, 'Saudi Arabian license plate recognition system', in *2003 International Conference on Geometric Modeling and Graphics, 2003. Proceedings*, IEEE, pp. 36-41.

Sarfraz, MS, Shahzad, A, Elahi, MA, Fraz, M, Zafar, I & Edirisinghe, EA 2013, 'Real-time automatic license plate recognition for CCTV forensic applications', *Journal of real-time image processing*, vol. 8, no. 3, pp. 285-95.

Schölkopf, B, Smola, A & Müller, K-R 1997, 'Kernel principal component analysis', in *International conference on artificial neural networks*, Springer, pp. 583-8.

Seber, GA & Lee, AJ 2012, *Linear regression analysis*, vol. 329, John Wiley & Sons.

Sharma, C & Kaur, A 2011, 'Indian vehicle license plate extraction and segmentation', *International Journal of Computer Science and Communication*, vol. 2, no. 2, pp. 593-9.

Sharma, J, Mishra, A, Saxena, K & Kumar, S 2014, 'A hybrid technique for license plate recognition based on feature selection of wavelet transform and artificial neural network', in *2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, IEEE, pp. 347-52.

Sheldon, B 2013, 'Camera surveillance within the UK: Enhancing public safety or a social threat?', in *Developments in Counter-Terrorist Measures and Uses of Technology*, Routledge, pp. 93-104.

Shi, X, Zhao, W & Shen, Y 2005, 'Automatic license plate recognition system based on color image processing', in *International Conference on Computational Science and Its Applications*, Springer, pp. 1159-68.

Silapachote, P, Karuppiah, DR & Hanson, AR 2005, *Feature selection using adaboost for face expression recognition*, DTIC Document.

Silva, SM & Jung, CR 2018, 'License Plate Detection and Recognition in Unconstrained Scenarios', in *European Conference on Computer Vision*, Springer, pp. 593-609.

Song, MK & Sarker, MMK 2014, 'Modeling and implementing two-stage AdaBoost for real-time vehicle license plate detection', *Journal of Applied Mathematics*, vol. 2014.

Sun, D & Watada, J 2015, 'Detecting pedestrians and vehicles in traffic scene based on boosted HOG features and SVM', in *Intelligent Signal Processing (WISP), 2015 IEEE 9th International Symposium on*, IEEE, pp. 1-4.

Taylor, JA 2005, *Forensic person tracking method and apparatus*, Google Patents.

Tsai, W-K, Lo, S-K, Su, C-D & Sheu, M-H 2017, 'Vehicle Detection Algorithm Based on Modified Gradient Oriented Histogram Feature', in *Advances in Intelligent Information Hiding and Multimedia Signal Processing*, Springer, pp. 127-34.

Wan, X, Liu, J & Liu, J 2011, 'A vehicle license plate localization method using color barycenters hexagon model', in *Third International Conference on Digital Image Processing (ICDIP 2011)*, International Society for Optics and Photonics, p. 80092O.

Wang, D, Tian, Y, Geng, W, Zhao, L & Gong, C 2018, 'LPR-Net: Recognizing Chinese license plate in complex environments', *Pattern Recognition Letters*.

Wang, F, Man, L, Wang, B, Xiao, Y, Pan, W & Lu, X 2008, 'Fuzzy-based algorithm for color recognition of license plates', *Pattern Recognition Letters*, vol. 29, no. 7, pp. 1007-20.

Wang, M-L, Liu, Y-H, Liao, B-Y, Lin, Y-S & Horng, M-F 2010, 'A vehicle license plate recognition system based on spatial/frequency domain filtering and neural networks', in *International Conference on Computational Collective Intelligence*, Springer, pp. 63-70.

Wang, S-Z & Lee, H-J 2003, 'Detection and recognition of license plate characters with different appearances', in *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, IEEE, pp. 979-84.

Weng, Y, Shivakumara, P, Lu, T, Meng, LK & Woon, HH 2015, 'A new multi-spectral fusion method for degraded video text frame enhancement', in *Pacific Rim Conference on Multimedia*, Springer, pp. 495-506.

Wu, B-F, Lin, S-P & Chiu, C-C 2007, 'Extracting characters from real vehicle licence plates out-of-doors', *IET computer vision*, vol. 1, no. 1, pp. 2-10.

Wu, M-K, Wei, J-S, Shih, H-C & Ho, CC 2009, 'License plate detection based on 2-level 2D Haar wavelet transform and edge density verification', in *2009 IEEE International Symposium on Industrial Electronics*, IEEE, pp. 1699-704.

Xu, J-F, Li, S-F & Yu, M-S 2004, 'Car license plate extraction using color and edge information', in *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826)*, IEEE, pp. 3904-7.

Y. Han, KVaEO, "Robust traffic sign recognition with feature extraction and k-NN classification methods," 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, 2015, pp. 484-488. & 10.1109/EIT.2015.7293386, d.

Yang, F & Ma, Z 2005, 'Vehicle license plate location based on histogramming and mathematical morphology', in *Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05)*, IEEE, pp. 89-94.

Yongchun, L & Jing, Y 2012, 'Research of license plate character features extraction and recognition', in *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, IEEE, pp. 2154-7.

Yousef, KMA, Al-Tabanjah, M, Hudaib, E & Ikrai, M 2015, 'SIFT based automatic number plate recognition', in *Information and Communication Systems (ICICS), 2015 6th International Conference on*, IEEE, pp. 124-9.

Yu, S, Li, B, Zhang, Q, Liu, C & Meng, MQ-H 2015, 'A novel license plate location method based on wavelet transform and EMD analysis', *Pattern recognition*, vol. 48, no. 1, pp. 114-25.

Yuan, F & Cheu, RL 2003, 'Incident detection using support vector machines', *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3, pp. 309-28.

Zhao, Y, Gu, J, Liu, C, Han, S, Gao, Y & Hu, Q 2010, 'License plate location based on haar-like cascade classifiers and edges', in *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on*, IEEE, pp. 102-5.

Zheng, D, Zhao, Y & Wang, J 2005, 'An efficient method of license plate location', *Pattern Recognition Letters*, vol. 26, no. 15, pp. 2431-8.

Zhu, X & Goldberg, AB 2009, 'Introduction to semi-supervised learning', *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1-130.

APPENDIX A

A

Matlab simulation code for Chapter 3
Ensemble of Adaboost cascades of 3L-LBPs classifiers for license plates detection with low quality images

The simulation codes to detect LPs from low quality vehicle images are presented. The experiment results were obtained using Matlab programming language version R2018a.

```
--------------------------------Adaboost classifier
function [L,hits] = ADABOOST_te(adaboost_model,te_func_handle,test_set,.
                        true_labels)
hypothesis_n = length(adaboost_model.weights);
sample_n = size(test_set,1);
class_n = length(unique(true_labels));
temp_L = zeros(sample_n,class_n,hypothesis_n);      % likelihoods for each weak classifier
 % for each weak classifier, likelihoods of test samples are collected
for i=1:hypothesis_n
   [temp_L(:,:,i),hits,error_rate] = te_func_handle(adaboost_model.parameters{i},...
                                        test_set,ones(sample_n,1),true_labels);
   temp_L(:,:,i) = temp_L(:,:,i)*adaboost_model.weights(i);
end
 L = sum(temp_L,3);
hits = sum(likelihood2class(L)==true_labels);
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function adaboost_model = ADABOOST_tr(tr_func_handle, te_func_handle, train_set,
labels, no_of_hypothesis)
adaboost_model = struct('weights',zeros(1,no_of_hypothesis),...
                  'parameters',[]); %cell(1,no_of_hypothesis));
 sample_n = size(train_set,1);
samples_weight = ones(sample_n,1)/sample_n;
 for turn=1:no_of_hypothesis
   adaboost_model.parameters{turn} = tr_func_handle(train_set,samples_weight,labels);
   [L,hits,error_rate] = te_func_handle(adaboost_model.parameters{turn},...
                               train_set,samples_weight,labels);
   if(error_rate==1)
      error_rate=1-eps;
   elseif(error_rate==0)
      error_rate=eps;
   end
    % The weight of the turn-th weak classifier
   adaboost_model.weights(turn) = log10((1-error_rate)/error_rate);
   C=likelihood2class(L);
   t_labeled=(C==labels);  % true labeled samples
    % Importance of the true classified samples is decreased for the next weak classifier
   samples_weight(t_labeled) = samples_weight(t_labeled)*...
                ((error_rate)/(1-error_rate));
    % Normalization
   samples_weight = samples_weight/sum(samples_weight);
end
 % Normalization
adaboost_model.weights=adaboost_model.weights/sum(adaboost_model.weights);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


--------------------------------LBP descriptor
function BasicLBP = ComputeRotationInvariance(RotateIndex, NeighborPoints,
tempLBPpre, tempLBPcur, tempLBPpos, tempLBPpreC, tempLBPposC, binCount,
BasicLBP)
```

```
minLBP = BasicLBP;
if RotateIndex == 1
    for p = 1 : NeighborPoints - 1
        tempLBPpreT = bitor(bitshift(tempLBPpre, -1 * p), bitshift(bitand(tempLBPpre,
(uint8(2^p) - 1)), (NeighborPoints - p)));
        tempLBPcurT = bitor(bitshift(tempLBPcur, -1 * p), bitshift(bitand(tempLBPcur,
(uint8(2^p) - 1)), (NeighborPoints - p)));
        tempLBPposT = bitor(bitshift(tempLBPpos, -1 * p), bitshift(bitand(tempLBPpos,
(uint8(2^p) - 1)), (NeighborPoints - p)));
          temp = (tempLBPpreC + bitshift(double(tempLBPpreT), 1)) +
bitshift(double(tempLBPcurT), (NeighborPoints + 1)) + bitshift(double(tempLBPposT),
(NeighborPoints * 2 + 1)) + tempLBPposC * 2 ^ (binCount - 1);
            if temp < minLBP
            minLBP = temp;
        end
    end
    BasicLBP = minLBP;
else

    tempLBPpreT = RotLBP(tempLBPpre, NeighborPoints);
    tempLBPcurT = RotLBP(tempLBPcur, NeighborPoints);
    tempLBPposT = RotLBP(tempLBPpos, NeighborPoints);
    temp = tempLBPpreC + bitshift(double(tempLBPpreT), 1) +
bitshift(double(tempLBPcurT), (NeighborPoints + 1)) + bitshift(double(tempLBPposT),
(NeighborPoints * 2 + 1)) + tempLBPposC * 2 ^ (binCount - 1);
     BasicLBP = temp;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function LBP= efficientLBP(inImg, varargin)
%% efficientLBP
% The function implements LBP (Local Binary Pattern analysis).
%% Deafult params
isRotInv=false;
isChanWiseRot=false;
filtR=generateRadialFilterLBP(8, 1);
 %% Get user inputs overriding default values
funcParamsNames={'filtR', 'isRotInv', 'isChanWiseRot'};
assignUserInputs(funcParamsNames, varargin(Y. Han & 10.1109/EIT.2015.7293386));
 if ischar(inImg) && exist(inImg, 'file')==2 % In case of file name input- read graphical file
    inImg=imread(inImg);
end
 nClrChans=size(inImg, 3);
 inImgType=class(inImg);
calcClass='single';
isCalcClassInput=strcmpi(inImgType, calcClass);
if ~isCalcClassInput
    inImg=cast(inImg, calcClass);
end
imgSize=size(inImg);
 nNeigh=size(filtR, 3);
```

```
 if nNeigh<=8
    outClass='uint8';
elseif nNeigh>8 && nNeigh<=16
    outClass='uint16';
elseif nNeigh>16 && nNeigh<=32
    outClass='uint32';
elseif nNeigh>32 && nNeigh<=64
    outClass='uint64';
else
    outClass=calcClass;
end
if isRotInv
    nRotLBP=nNeigh;
    nPixelsSingleChan=imgSize(1)*imgSize(2);
    iSingleChan=reshape( 1:nPixelsSingleChan, imgSize(1), imgSize(2) );
else
    nRotLBP=1;
end
 nEps=-3;
weigthVec=reshape(2.^( (1:nNeigh) -1), 1, 1, nNeigh);
weigthMat=repmat( weigthVec, imgSize([1, 2]) );
binaryWord=zeros(imgSize(1), imgSize(2), nNeigh, calcClass);
LBP=zeros(imgSize, outClass);
possibleLBP=zeros(imgSize(1), imgSize(2), nRotLBP);
for iChan=1:nClrChans
    % Initiate neighbours relation filter and LBP's matrix
    for iFiltElem=1:nNeigh
        % Rotate filter- to compare center to next neigbour
        filtNeight=filtR(:, :, iFiltElem);
              % calculate relevant LBP elements via filtering
        binaryWord(:, :, iFiltElem)=cast( ...
            roundnS(filter2( filtNeight, inImg(:, :, iChan), 'same' ), nEps) >= 0,...
            calcClass );
        % Without rounding sometimes inaqulity happens in some pixels
        % compared to pixelwiseLBP
    end % for iFiltElem=1:nNeigh
     for iRot=1:nRotLBP
        % find all relevant LBP candidates
        possibleLBP(:, :, iRot)=sum(binaryWord.*weigthMat, 3);
        if iRot < nRotLBP
            binaryWord=circshift(binaryWord, [0, 0, 1]); % shift binaryWord elements
        end
    end
        if isRotInv
        if iChan==1 || isChanWiseRot
            % Find minimal LBP, and the rotation applied to first color channel
            [minColroInvLBP, iMin]=min(possibleLBP, [], 3);
                    % calculte 3D matrix index
            iCircShiftMinLBP=iSingleChan+(iMin-1)*nPixelsSingleChan;
        else
```

```
         % the above rotation of the first channel, holds to rest of the channels
         minColroInvLBP=possibleLBP(iCircShiftMinLBP);
      end % if iChan==1 || isChanWiseRot
   else
      minColroInvLBP=possibleLBP;
   end % if isRotInv

   if strcmpi(outClass, calcClass)
      LBP(:, :, iChan)=minColroInvLBP;
   else
      LBP(:, :, iChan)=cast(minColroInvLBP, outClass);
   end
end % for iChan=1:nClrChans
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ Boxes ] = funcSol3( Im, ~ )
 Im = imgaussfilt(Im,0.25);
Im = adapthisteq(Im,'clipLimit',0.25,'Distribution','rayleigh');
 I= Im;
 [~, mserConnComp] = detectMSERFeatures(I, ...
   'RegionAreaRange',[140 9000],'ThresholdDelta',3);
mserStats = regionprops(mserConnComp, 'BoundingBox', 'Eccentricity', ...
   'Solidity', 'Extent', 'Euler', 'Image');
% Compute the aspect ratio using bounding box data.
bbox = vertcat(mserStats.BoundingBox);
w = bbox(:,3);
h = bbox(:,4);
aspectRatio = w./h;
% Threshold the data to determine which regions to remove. These thresholds
% may need to be tuned for other images.
filterIdx = aspectRatio' > 3;
filterIdx = filterIdx | [mserStats.Eccentricity] > .990 ;
filterIdx = filterIdx | [mserStats.Solidity] < .3;
filterIdx = filterIdx | [mserStats.Extent] < 0.2 | [mserStats.Extent] > 0.9;
filterIdx = filterIdx | [mserStats.EulerNumber] < -4;
% Remove OR Filter out regions
mserStats(filterIdx) = [];
regionImage = mserStats(6).Image;
regionImage = padarray(regionImage, [1 1]);
 % Compute the stroke width image.
distanceImage = bwdist(~regionImage);
skeletonImage = bwmorph(regionImage, 'thin', inf);
strokeWidthValues = distanceImage(skeletonImage);
strokeWidthMetric = std(strokeWidthValues)/mean(strokeWidthValues);
 % Threshold the stroke width variation metric
strokeWidthThreshold = 0.4;
strokeWidthFilterIdx = strokeWidthMetric > strokeWidthThreshold;
 % Process the remaining regions
for j = 1:numel(mserStats)
    regionImage = mserStats(j).Image;
   regionImage = padarray(regionImage, [1 1], 0);
```

```
    distanceImage = bwdist(~regionImage);
   skeletonImage = bwmorph(regionImage, 'thin', inf);
    strokeWidthValues = distanceImage(skeletonImage);
    strokeWidthMetric = std(strokeWidthValues)/mean(strokeWidthValues);
    strokeWidthFilterIdx(j) = strokeWidthMetric > strokeWidthThreshold;
 end
mserStats(strokeWidthFilterIdx) = [];
bboxes = vertcat(mserStats.BoundingBox);
if isempty(bboxes)
   Boxes = [];
   return
end
 xmin = bboxes(:,1);
ymin = bboxes(:,2);
xmax = xmin + bboxes(:,3) - 1;
ymax = ymin + bboxes(:,4) - 1;
 % Expand the bounding boxes by a small amount.
expansionAmount = 0.02;
xmin = (1-expansionAmount*7) * xmin;
ymin = (1-expansionAmount) * ymin;
xmax = (1+expansionAmount*4) * xmax;
ymax = (1+expansionAmount) * ymax;
 % Clip the bounding boxes to be within the image bounds
xmin = max(xmin, 1);
ymin = max(ymin, 1);
xmax = min(xmax, size(I,2));
ymax = min(ymax, size(I,1));
 % Show the expanded bounding boxes
expandedBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
% Compute the overlap ratio
overlapRatio = bboxOverlapRatio(expandedBBoxes, expandedBBoxes);
 % Set the overlap ratio between a bounding box and itself to zero to
% simplify the graph representation.
n = size(overlapRatio,1);
overlapRatio(1:n+1:n^2) = 0;
 % Create the graph
g = graph(overlapRatio);
 % Find the connected text regions within the graph
componentIndices = conncomp(g);
 % Merge the boxes based on the minimum and maximum dimensions.
xmin = accumarray(componentIndices', xmin, [], @min);
ymin = accumarray(componentIndices', ymin, [], @min);
xmax = accumarray(componentIndices', xmax, [], @max);
ymax = accumarray(componentIndices', ymax, [], @max);
 % Compose the merged bounding boxes using the [x y width height] format.
Boxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
  % Remove bounding boxes that only contain one text region
numRegionsInGroup = histcounts(componentIndices);
Boxes(numRegionsInGroup == 1, :) = [];
 end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Boxes = funcSol5( Im,l )
 l=l+1;
[y,x]=size(Im);
Im2 = imadjust( Im,[0.1;0.5] );
BW = edge(Im2,'Sobel');
 reg = regionprops(BW,'Area','Centroid','BoundingBox');
 reg([reg.Area]<100)=[];
regBad = [];
for i = 1:length(reg)
    % filter some boxes
    if reg(i).BoundingBox(4) < 15
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(3) < 10
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(3) < 70 && reg(i).BoundingBox(4) < 70
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(3) < 70 && reg(i).BoundingBox(4) > 70
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(3) > 150 && reg(i).BoundingBox(4) > 150
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(4) < 20 && reg(i).BoundingBox(3) > 80
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(3) > 300
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(4) > 100
        regBad = [regBad,i];
    end
    %remove regions that are at the top
    if reg(i).BoundingBox(1) < 50 ||  reg(i).BoundingBox(1) > x - 50
        regBad = [regBad,i];
    elseif reg(i).BoundingBox(2) < 150
        regBad = [regBad,i];
    end
end
reg(unique(regBad))=[];
%if multiple candidates - leave with max corners
iMax=[];
for i = 1:length(reg)
    corners = detectFASTFeatures(imcrop(Im,(reg(i).BoundingBox)));
    iMax(i) = length(corners);
end
[~,chosen]=max(iMax);
 if ~isempty(chosen)
    Boxes = reg(chosen).BoundingBox;
else
    %just to avoid error
    Boxes = [1,1,1,1];
end
 end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [radInterpFilt]=generateRadialFilterLBP(p, r)
%% Default params
if nargin<2
   r=1;
   if nargin<1
      p=8;
   end
end
 %% verify params leget values
r=max(1, r);    % radius below 1 is illegal
p=round(p);     % non integer number of neighbours sound oucward
p=max(1, p);    % number of neighbours below 1 is illegal
%% find elements angles, aranged counter clocwise starting from "X axis"
% See http://www.ee.oulu.fi/mvg/files/pdf/pdf_6.pdf for illustration
theta=linspace(0, 2*pi, p+1)+pi/2;
theta=theta(1:end-1);          % remove obsolite last element (0=2*pi)
 %% Find relevant coordinates
[rowsFilt, colsFilt] = pol2cart(theta, repmat(r, size(theta) )); % convert to cartesian
nEps=-3;
rowsFilt=roundnS(rowsFilt, nEps);
colsFilt=roundnS(colsFilt, nEps);
 % Matrix indexes should be integers
rowsFloor=floor(rowsFilt);
rowsCeil=ceil(rowsFilt);
 colsFloor=floor(colsFilt);
colsCeil=ceil(colsFilt);
 rowsDistFloor=1-abs( rowsFloor-rowsFilt );
rowsDistCeil=1-abs( rowsCeil-rowsFilt );
colsDistFloor=1-abs( colsFloor-colsFilt );
colsDistCeil=1-abs( colsCeil-colsFilt );
 %Find minimal filter dimentions, based on indexes
filtDims=[ceil( max(rowsFilt) )-floor( min(rowsFilt) ),...
ceil( max(colsFilt) )-floor( min(colsFilt) ) ];
filtDims=filtDims+mod(filtDims+1, 2); % verify filter dimentions are odd
 filtCenter=(filtDims+1)/2;
 %% Convert cotersian coordinates to matrix elements coordinates via simple shift
rowsFloor=rowsFloor+filtCenter(1);
rowsCeil=rowsCeil+filtCenter(1);
colsFloor=colsFloor+filtCenter(2);
colsCeil=colsCeil+filtCenter(2);

%% Generate the filter- each 2D slice for filter element
radInterpFilt=zeros( [filtDims,  p], 'single'); % initate filter with zeros
for iP=1:p
   radInterpFilt( rowsFloor(iP), colsFloor(iP), iP )=...
      radInterpFilt( rowsFloor(iP), colsFloor(iP), iP )+rowsDistFloor(iP)+colsDistFloor(iP);
   radInterpFilt( rowsFloor(iP), colsCeil(iP), iP )=...
   radInterpFilt( rowsFloor(iP), colsCeil(iP), iP )+rowsDistFloor(iP)+colsDistCeil(iP);
  radInterpFilt( rowsCeil(iP), colsFloor(iP), iP )=...
```

```
   radInterpFilt( rowsCeil(iP), colsFloor(iP), iP )+rowsDistCeil(iP)+colsDistFloor(iP);
      radInterpFilt( rowsCeil(iP), colsCeil(iP), iP )=...
       radInterpFilt( rowsCeil(iP), colsCeil(iP), iP )+rowsDistCeil(iP)+colsDistCeil(iP);
       radInterpFilt( :, :, iP )=radInterpFilt( :, :, iP )/sum(sum(radInterpFilt( :, :, iP )));
end
radInterpFilt( filtCenter(1), filtCenter(2), : )=...
radInterpFilt( filtCenter(1), filtCenter(2), : )-1;
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Histogram = LBPTOP(VolData, FxRadius, FyRadius, TInterval, NeighborPoints,
TimeLength, BorderLength, bBilinearInterpolation, Bincount, Code)
[height width Length] = size(VolData);
 XYNeighborPoints = NeighborPoints(1);
XTNeighborPoints = NeighborPoints(2);
YTNeighborPoints = NeighborPoints(3);
 if (Bincount == 0)
   % normal code
   nDim = 2^(YTNeighborPoints);
   Histogram = zeros(3, nDim);
else
   % uniform code
   Histogram = zeros(3, Bincount); % Bincount = 59;
end
 if (bBilinearInterpolation == 0)
      for i = TimeLength + 1 : Length - TimeLength
            for yc = BorderLength + 1 : height - BorderLength
                  for xc = BorderLength + 1 : width - BorderLength
                      CenterVal = VolData(yc, xc, i);
            %% In XY plane
            BasicLBP = 0;
            FeaBin = 0;
                for p = 0 : XYNeighborPoints - 1
               X = floor(xc + FxRadius * cos((2 * pi * p) / XYNeighborPoints) + 0.5);
               Y = floor(yc - FyRadius * sin((2 * pi * p) / XYNeighborPoints) + 0.5);
                CurrentVal = VolData(Y, X, i);
               if CurrentVal >= CenterVal
                  BasicLBP = BasicLBP + 2 ^ FeaBin;
               end
               FeaBin = FeaBin + 1;
            end
            %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
               Histogram(1, BasicLBP + 1) = Histogram(1, BasicLBP + 1) + 1;
            else
               Histogram(1, Code(BasicLBP + 1, 2) + 1) = Histogram(1, Code(BasicLBP + 1,
2) + 1) + 1;
            end
```

```
            %% In XT plane
            BasicLBP = 0;
            FeaBin = 0;
            for p = 0 : XTNeighborPoints - 1
               X = floor(xc + FxRadius * cos((2 * pi * p) / XTNeighborPoints) + 0.5);
               Z = floor(i + TInterval * sin((2 * pi * p) / XTNeighborPoints) + 0.5);
                  CurrentVal = VolData(yc, X, Z);
                 if CurrentVal >= CenterVal
                  BasicLBP = BasicLBP + 2 ^ FeaBin;
               end
               FeaBin = FeaBin + 1;
            end
             %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
               Histogram(2, BasicLBP + 1) = Histogram(2, BasicLBP + 1) + 1;
            else % uniform patterns
               Histogram(2, Code(BasicLBP + 1, 2) + 1) = Histogram(2, Code(BasicLBP + 1,
2) + 1) + 1;
            end
              BasicLBP = 0;
            FeaBin = 0;
            for p = 0 : YTNeighborPoints - 1
               Y = floor(yc - FyRadius * sin((2 * pi * p) / YTNeighborPoints) + 0.5);
               Z = floor(i + TInterval * cos((2 * pi * p) / YTNeighborPoints) + 0.5);
                  CurrentVal = VolData(Y, xc, Z);
                if CurrentVal >= CenterVal
                  BasicLBP = BasicLBP + 2 ^ FeaBin;
               end
               FeaBin = FeaBin + 1;
            end
            %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
               Histogram(3, BasicLBP + 1) = Histogram(3, BasicLBP + 1) + 1;
            else
               Histogram(3, Code(BasicLBP + 1, 2) + 1) = Histogram(3, Code(BasicLBP + 1,
2) + 1) + 1;
            end
            end
      end
   end
else % bilinear interpolation
   for i = TimeLength + 1 : Length - TimeLength
```

```
            for yc = BorderLength + 1 : height - BorderLength
           for xc = BorderLength + 1 : width - BorderLength
            CenterVal = VolData(yc, xc, i);
            %% In XY plane
            BasicLBP = 0;
            FeaBin = 0;
            for p = 0 : XYNeighborPoints - 1
                % bilinear interpolation
               x1 = single(xc + FxRadius * cos((2 * pi * p) /
XYNeighborPoints));%%"float" are called "single" in Matlab
               y1 = single(yc - FyRadius * sin((2 * pi * p) / XYNeighborPoints));
                u = x1 - floor(x1);
               v = y1 - floor(y1);
               ltx = floor(x1);
               lty = floor(y1);
               lbx = floor(x1);
               lby = ceil(y1);
               rtx = ceil(x1);
               rty = floor(y1);
               rbx = ceil(x1);
               rby = ceil(y1);
               % the values of neighbors that do not fall exactly on
               % pixels are estimated by bilinear interpolation of
               % four corner points near to it.
               CurrentVal = floor(VolData(lty, ltx, i) * (1 - u) * (1 - v) + VolData(lby, lbx, i) *
(1 - u) * v + VolData(rty, rtx, i) * u * (1 - v) + VolData(rby, rbx, i) * u * v);
                  if CurrentVal >= CenterVal
                  BasicLBP = BasicLBP + 2 ^ FeaBin;
               end
               FeaBin = FeaBin + 1;
            end
            %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
               Histogram(1, BasicLBP + 1) = Histogram(1, BasicLBP + 1) + 1;
            else
               Histogram(1, Code(BasicLBP + 1, 2) + 1) = Histogram(1, Code(BasicLBP + 1,
2) + 1) + 1;
            end
          %% In XT plane
            BasicLBP = 0;
            FeaBin = 0;
            for p = 0 : XTNeighborPoints - 1
               % bilinear interpolation
               x1 = single(xc + FxRadius * cos((2 * pi * p) / XTNeighborPoints));
               z1 = single(i + TInterval * sin((2 * pi * p) / XTNeighborPoints));
                u = x1 - floor(x1);
```

```
                v = z1 - floor(z1);
                ltx = floor(x1);
                lty = floor(z1);
                lbx = floor(x1);
                lby = ceil(z1);
                rtx = ceil(x1);
                rty = floor(z1);
                rbx = ceil(x1);
                rby = ceil(z1);
                % the values of neighbors that do not fall exactly on
                % pixels are estimated by bilinear interpolation of
                % four corner points near to it.
                CurrentVal = floor(VolData(yc, ltx, lty) * (1 - u) * (1 - v) + VolData(yc, lbx, lby)
* (1 - u) * v + VolData(yc, rtx, rty) * u * (1 - v) + VolData(yc, rbx, rby) * u * v);
                  if CurrentVal >= CenterVal
                   BasicLBP = BasicLBP + 2 ^ FeaBin;
                end
                FeaBin = FeaBin + 1;
            end
            %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
                Histogram(2, BasicLBP + 1) = Histogram(2, BasicLBP + 1) + 1;
            else
                Histogram(2, Code(BasicLBP + 1, 2) + 1) = Histogram(2, Code(BasicLBP + 1,
2) + 1) + 1;
            end
          %% In YT plane
            BasicLBP = 0;
            FeaBin = 0;
            for p = 0 : YTNeighborPoints - 1
                % bilinear interpolation
                y1 = single(yc - FyRadius * sin((2 * pi * p) / YTNeighborPoints));
                z1 = single(i + TInterval * cos((2 * pi * p) / YTNeighborPoints));
                u = y1 - floor(y1);
                v = z1 - floor(z1);
                ltx = floor(y1);
                lty = floor(z1);
                lbx = floor(y1);
                lby = ceil(z1);
                rtx = ceil(y1);
                rty = floor(z1);
                rbx = ceil(y1);
                rby = ceil(z1);
                % the values of neighbors that do not fall exactly on
                % pixels are estimated by bilinear interpolation of
                % four corner points near to it.
```

```
                CurrentVal = floor(VolData(ltx, xc, lty) * (1 - u) * (1 - v) +      VolData(lbx, xc,
lby) * (1 - u) * v + VolData(rtx, xc, rty) * u * (1 - v) + VolData(rbx, xc, rby) * u * v);
                if CurrentVal >= CenterVal
                    BasicLBP = BasicLBP + 2 ^ FeaBin;
                end
                FeaBin = FeaBin + 1;
            end
            %% if Bincount is "0", it means basic LBP-TOP will be
            %% computed and uniform patterns does not work in this case
            %%. Otherwide it should be the number of the uniform
            %%patterns, then "Code" keeps the lookup-table of the basic
            %%LBP and uniform LBP
            if Bincount == 0
                Histogram(3, BasicLBP + 1) = Histogram(3, BasicLBP + 1) + 1;
            else
                Histogram(3, Code(BasicLBP + 1, 2) + 1) = Histogram(3, Code(BasicLBP + 1,
2) + 1) + 1;
            end
        end %%
    end %%
  end %%
end
%% normalization
for j = 1 : 3
   Histogram(j, :) = Histogram(j, :)./sum(Histogram(j, :));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function classes = likelihood2class(likelihoods)
[sample_n,class_n] = size(likelihoods);
maxs = (likelihoods==repmat(max(likelihoods,[],2),[1,class_n]));
 classes=zeros(sample_n,1);
for i=1:sample_n
   classes(i) = find(maxs(i,:),1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function LBP= pixelwiseLBP(inImg, varargin) % isRotInv, isChanWiseRot, filtR
%% Deafult params
isRotInv=false;
isChanWiseRot=false;
filtR=generateRadialFilterLBP(8, 1);
%% Get user inputs overriding default values
funcParamsNames={'filtR', 'isRotInv', 'isChanWiseRot'};
assignUserInputs(funcParamsNames, varargin(Y. Han & 10.1109/EIT.2015.7293386));
if ischar(inImg) && exist(inImg, 'file')==2 % In case of file name input- read graphical file
   inImg=imread(inImg);
end
 nClrChans=size(inImg, 3);
 inImgType=class(inImg);
calcClass='single';
```

```
 isCalcClassInput=strcmpi(inImgType, calcClass);
if ~isCalcClassInput
    inImg=cast(inImg, calcClass);
end
imgSize=size(inImg);
filtDims=size(filtR);
nNeigh=filtDims(3);
if nNeigh<=8
    outClass='uint8';
elseif nNeigh>8 && nNeigh<=16
    outClass='uint16';
elseif nNeigh>16 && nNeigh<=32
    outClass='uint32';
elseif nNeigh>32 && nNeigh<=64
    outClass='uint64';
else
    outClass=calcClass;
end
 LBP=zeros(imgSize, outClass);
nEps=-3;
weigthVec=reshape(2.^( (1:nNeigh) -1), 1, nNeigh);
%% Primitive pixelwise solution
filtDimsR=floor(filtDims([1, 2])/2); % Filter Radius
% update index values, so it will be from 1 to N-1, where N is number of pixels in
% support area, including the central pixel
 % Padding image with zeroes, to deal with the edges
chanImgPad=zeros(imgSize(1)+2*filtDimsR(1), imgSize(2)+2*filtDimsR(2), calcClass);
padImgSize=size(chanImgPad);
currChanLBP=zeros(padImgSize, outClass);
if isRotInv
    if verLessThan('matlab', '7.14') % due to some issue with circshift and non dounle inputs
        iCircShiftMinLBP=zeros(padImgSize, 'double');
    else
        iCircShiftMinLBP=zeros(padImgSize, 'int8'); % outClass % Limits number fo color
channels to 127
    end
end
hWaitbar=waitbar(0, 'Calculating LBP in pixel-wise manner',...
    'Name', 'pixel-wise LBP!');
hTicPixelwiseLBP=tic;
is2dFilter = filtDims(1)>1 && filtDims(2)>1;
for iChan=1:nClrChans
    chanImgPad(( 1+filtDimsR(1) ):( end-filtDimsR(1) ),...
        ( 1+filtDimsR(2) ): (end-filtDimsR(2) ))=inImg(:, :, iChan);
    nRows=padImgSize(1)-2*filtDimsR(1);
    for iRow=( filtDimsR(1)+1 ):( padImgSize(1)-filtDimsR(1) )
        for iCol=( filtDimsR(2)+1 ):( padImgSize(2)-filtDimsR(2) )
            subImg=chanImgPad(iRow+( -filtDimsR(1):filtDimsR(1) ),...
                iCol+( -filtDimsR(2):filtDimsR(2) ));
            % find differences between current pixel, and it's neighours
```

```
            diffVec=sum( bsxfun(@times, filtR, subImg) );
            if is2dFilter
               diffVec = sum(diffVec);
            end
            diffVec=roundnS(diffVec, nEps);
            binaryWord=( diffVec(:)>=0 );
            if isRotInv
               if iChan==1 || isChanWiseRot % go through all posible binary
                  % word combination, finding minimal LBP
                  [minLBP, iCircShiftMinLBP(iRow, iCol)]=...
                     sortNeighbours(binaryWord, weigthVec);
               else % if iChan==1 || isChanWiseRot
                  [minLBP, ~]=sortNeighbours( binaryWord, weigthVec,...
                     iCircShiftMinLBP(iRow, iCol) );
               end % if iChan==1 || isChanWiseRot
            else
               minLBP=weigthVec*binaryWord;
            end % if isRotInv
            currChanLBP(iRow, iCol)=cast( minLBP,  outClass);   % convert to decimal.
         end % for iCol=(1+filtDimsR(2)):(imgSize(2)-filtDimsR(2))
           % Present waitbar- a bar with progress, time passed and time remaining
         waitbarTimeRemaining(hWaitbar, hTicPixelwiseLBP,...
            (( iRow-filtDimsR(1) )+nRows*(iChan-1))/(nClrChans*nRows));
      end % for iRow=(1+filtDimsR(1)):(imgSize(1)-filtDimsR(1))
         % crop the margins resulting from zero padding
      LBP(:, :, iChan)=currChanLBP(( filtDimsR(1)+1 ):( end-filtDimsR(1) ),...
         ( filtDimsR(2)+1 ):( end-filtDimsR(2) ));
      if iChan==nClrChans
         close(hWaitbar); % close the waitbar
      end
   end % for iChan=1:nClrChans
   function [minLBP, iCircShift]=sortNeighbours( origBinWord, weigthVec, iShift)
    nElems=numel(origBinWord);
   if size(origBinWord, 1)~=nElems
      origBinWord=origBinWord(:);
   end
   if size(weigthVec, 2)~=nElems
      weigthVec=reshape(weigthVec, 1, nElems);
   end
   if nargin < 3 || isempty(iShift)
      % initial values- current LBP, zero shift
      iCircShift=0;
      minLBP=weigthVec*origBinWord;
      % go through all posible binary word combination, finding minimal LBP
      nShifts=numel(origBinWord)-1;
         for iCurrShift=1:nShifts
         origBinWord=circshift(origBinWord, 1);
         currLBP=weigthVec*origBinWord;
         if currLBP < minLBP
            minLBP=currLBP;
```

```
            iCircShift=iCurrShift;
        end % if currLBP < minLBP
    end % for iCurrShift=iShift
else
    iCircShift=iShift(1);
    minLBP=weigthVec*circshift(origBinWord, iCircShift);
end % if nargin < 3 || isempty(iShift)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Histogram = RIVLBP(VolData, TInterval, FRadius, NeighborPoints, BorderLength,
TimeLength, RotateIndex, bBilinearInterpolation)
% This function is to compute the Basic VLBP and two kinds of rotation invariant VLBP
features for a video sequence
 [height width Length] = size(VolData);
binCount = (NeighborPoints + 1) * 2 + NeighborPoints;
nDim = 2 ^ binCount;
Histogram = zeros(nDim, 1);
 if bBilinearInterpolation == 0
     for i = TimeLength + 1 : Length - TimeLength
            for yc = BorderLength + 1 :  height - BorderLength
                for xc = BorderLength + 1 : width - BorderLength
                        CenterVal = VolData(yc, xc, i);
                BasicLBP = 0;
                FeaBin = 0;
                        %% In previous frame
                CurrentVal = VolData(yc, xc, i - TInterval);

                if CurrentVal >= CenterVal
                    BasicLBP = BasicLBP + 2 ^ FeaBin;
                end
                tempLBPpreC = BasicLBP;
                FeaBin = FeaBin + 1;
                        tempLBPpre = 0;
                for p = 0 : NeighborPoints - 1
                    X = floor(xc + FRadius * cos((2 * pi * p) / NeighborPoints) + 0.5);
                    Y = floor(yc - FRadius * sin((2 * pi * p) / NeighborPoints) + 0.5);
                                CurrentVal = VolData(Y, X, i - TInterval);
                                if CurrentVal >= CenterVal
                        BasicLBP = BasicLBP + 2 ^ FeaBin;
                        tempLBPpre = tempLBPpre + 2 ^ p;
                    end
                    FeaBin = FeaBin + 1;
                end
                        %% In current frame
                tempLBPcur = 0;
                for p = 0 : NeighborPoints - 1
                    X = floor(xc + FRadius * cos((2 * pi * p) / NeighborPoints) + 0.5);
                    Y = floor(yc - FRadius * sin((2 * pi * p) / NeighborPoints) + 0.5);
                    CurrentVal = VolData(Y, X, i);
                    if CurrentVal >= CenterVal
                        BasicLBP = BasicLBP + 2 ^ FeaBin;
```

```
                tempLBPcur = tempLBPcur + 2 ^ p;
            end
            FeaBin = FeaBin + 1;
        end
                %% In post frame
        tempLBPpos = 0;
        for p = 0 : NeighborPoints - 1
            X = floor(xc + FRadius * cos((2 * pi * p) / NeighborPoints) + 0.5);
            Y = floor(yc - FRadius * sin((2 * pi * p) / NeighborPoints) + 0.5);
                    CurrentVal = VolData(Y, X, i + TInterval);
                    if CurrentVal >= CenterVal
                BasicLBP = BasicLBP + 2 ^ FeaBin;
                tempLBPpos = tempLBPpos + 2 ^ p;
            end
            FeaBin = FeaBin + 1;
        end
                tempLBPposC = 0;

        CurrentVal = VolData(yc, xc, i + TInterval);

        if CurrentVal >= CenterVal
            BasicLBP = BasicLBP + 2 ^ FeaBin;
            tempLBPposC = 1;
        end
        %% Rotation invariance (if RotateIndex = 1/2)
        if (RotateIndex == 1)||(RotateIndex == 2)
            % if RotateIndex == 0, basic VLBP is computed
            % else for rotation invariance code
            BasicLBP = ComputeRotationInvariance(RotateIndex, NeighborPoints,
tempLBPpre, tempLBPcur, tempLBPpos, tempLBPpreC, tempLBPposC, binCount,
BasicLBP);
        end
        % the index under matlab start from 1 in the vector and matrix
        Histogram(BasicLBP + 1) = Histogram(BasicLBP + 1) + 1;
      end
    end
  end
else
  for i = TimeLength + 1: Length - TimeLength
    for yc = BorderLength + 1 :  height - BorderLength
      for xc = BorderLength + 1 : width - BorderLength
        CenterVal = VolData(yc, xc, i);
        BasicLBP = 0;
        FeaBin = 0;
        %% In previous frame
        CurrentVal = VolData(yc, xc, i - TInterval);
        if CurrentVal >= CenterVal
            BasicLBP = BasicLBP + 2 ^ FeaBin;
        end
        tempLBPpreC = BasicLBP;
```

```
            FeaBin = FeaBin + 1;
            tempLBPpre = 0;
            for p = 0 : NeighborPoints - 1
               % bilinear interpolation
               x1 = single(xc + FRadius * cos((2 * pi * p)/NeighborPoints));
               y1 = single(yc - FRadius * sin((2 * pi * p)/NeighborPoints));
                u = x1 - floor(x1);
               v = y1 - floor(y1);
               ltx = (floor(x1));
               lty = (floor(y1));
               lbx = (floor(x1));
               lby = (ceil(y1));
               rtx = (ceil(x1));
               rty = (floor(y1));
               rbx = (ceil(x1));
               rby = (ceil(y1));
               % values of neighbors that do not fall exactly on
               % pixels are estimated by bilinear interpolation of
               % four corner points near to it
               CurrentVal = floor(VolData(lty, ltx, i - TInterval) * (1 - u) * (1 - v) +
VolData(lby, lbx, i - TInterval) * (1 - u) * v + VolData(rty, rtx, i - TInterval) * u * (1 - v) +
VolData(rby, rbx, i - TInterval) * u * v);
                  if CurrentVal >= CenterVal
                  BasicLBP = BasicLBP + 2 ^ FeaBin;
                  tempLBPpre = tempLBPpre + 2 ^ p;
               end
               FeaBin = FeaBin + 1;
            end
              %% In current frame
            tempLBPcur = 0;
            for p = 0 : NeighborPoints - 1
               % bilinear interpolation
               x1 = single(xc + FRadius * cos((2 * pi * p)/NeighborPoints));
               y1 = single(yc - FRadius * sin((2 * pi * p)/NeighborPoints));
               u = x1 - floor(x1);
               v = y1 - floor(y1);
               ltx = (floor(x1));
               lty = (floor(y1));
               lbx = (floor(x1));
               lby = (ceil(y1));
               rtx = (ceil(x1));
               rty = (floor(y1));
               rbx = (ceil(x1));
               rby = (ceil(y1));
               % values of neighbors that do not fall exactly on
               % pixels are estimated by bilinear interpolation of
               % four corner points near to it
               CurrentVal = floor(VolData(lty, ltx, i) * (1 - u) * (1 - v) + VolData(lby, lbx, i) *
(1 - u) * v + VolData(rty, rtx, i) * u * (1 - v) + VolData(rby, rbx, i) * u * v);
               if CurrentVal >= CenterVal
```

```
                BasicLBP = BasicLBP + 2 ^ FeaBin;
                tempLBPcur = tempLBPcur + 2 ^ p;
            end
            FeaBin = FeaBin + 1;
        end
        %% In post frame
        tempLBPpos = 0;
         for p = 0 : NeighborPoints - 1
            % bilinear interpolation
            x1 = single(xc + FRadius * cos((2 * pi * p)/NeighborPoints));
            y1 = single(yc - FRadius * sin((2 * pi * p)/NeighborPoints));
            u = x1 - floor(x1);
            v = y1 - floor(y1);
            ltx = (floor(x1));
            lty = (floor(y1));
            lbx = (floor(x1));
            lby = (ceil(y1));
            rtx = (ceil(x1));
            rty = (floor(y1));
            rbx = (ceil(x1));
            rby = (ceil(y1));
            % values of neighbors that do not fall exactly on
            % pixels are estimated by bilinear interpolation of
            % four corner points near to it
            CurrentVal = floor(VolData(lty, ltx, i + TInterval) * (1 - u) * (1 - v) +
VolData(lby, lbx, i + TInterval) * (1 - u) * v + VolData(rty, rtx, i + TInterval) * u * (1 - v) +
VolData(rby, rbx, i + TInterval) * u * v);
            if CurrentVal >= CenterVal
                BasicLBP = BasicLBP + 2 ^ FeaBin;
                tempLBPpos = tempLBPpos + 2 ^ p;
            end
            FeaBin  = FeaBin + 1;
        end
        tempLBPposC = 0;
        CurrentVal = VolData(yc, xc, i + TInterval);
        if (CurrentVal >= CenterVal)
            BasicLBP = BasicLBP + 2 ^ FeaBin;
            tempLBPposC = 1;
        end
        %% rotation invariance (if RotateIndex = 1/2)
        if (RotateIndex == 1) || (RotateIndex == 2)
            %for roataion invariance code
            BasicLBP = ComputeRotationInvariance(RotateIndex, NeighborPoints,
tempLBPpre, tempLBPcur, tempLBPpos, tempLBPpreC, tempLBPposC, binCount,
BasicLBP);
        end
        % the index under matlab start from 1 in the vector and
        % matrix
        Histogram(BasicLBP + 1) = Histogram(BasicLBP + 1) + 1;
    end
```

```
        end
    end
end
%% Normalization
Total = 0;
for i = 1 : nDim
    Total = Total + Histogram(i);
end
Histogram = Histogram./Total;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function minLBP = RotLBP(LBPCode, NeighborPoints)
%% For a basic LBP code, this function is to get its rotation invariance
%  corresponding code
 minLBP = LBPCode;
for p = 1 : NeighborPoints - 1
    tempCode = bitor(bitshift(LBPCode, -1 * p), bitshift(bitand(LBPCode, (uint8(2 ^ p) - 1)),
(NeighborPoints - p)));
    if tempCode < minLBP
        minLBP = tempCode;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function outData=roundnS(inData, nEps)
quantVal=10^nEps;
outData=round(inData/quantVal)*quantVal;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
 % DEMONSTRATION OF ADABOOST_tr and ADABOOST_te
% Using adaboost with linear threshold classifier
% for a two class classification problem.
% Bug Reporting: Please contact the author for bug reporting and comments.
 tic
%% Read all images from location
ImgFolder = '040603';
dirLoc = ['baza_slika/',ImgFolder,'/'];
 imagefiles = dir([dirLoc, '*.jpg']);
nfiles = length(imagefiles);    % files found
 for ii=1:nfiles
   currentfilename = imagefiles(ii).name;
   currentimage = imread([dirLoc,currentfilename]);
   images(Rabiner & Juang) = currentimage;
end
    %% if cropped folder exists - delete it. If not create.
toSave = ['baza_slika/',ImgFolder,'_cropped'];
if exist(toSave)==7
    rmdir (toSave, 's');
end
mkdir(toSave);
 %% run one by one
 k=0;
```

```
 l=0;
for ii=1:nfiles
    Im = rgb2gray(images(Rabiner & Juang));
    Im = adapthisteq(Im,'clipLimit',0.01,'Distribution','rayleigh');
       %L2:Gaussian filtering image
   Im = imgaussfilt(Im,0.01);
    %L3:Constract image

   % Im=imhistmatch(Im,Im);
   edgeThreshold = 0.10;
amount = 0.05;
Im = localcontrast(Im, edgeThreshold, amount);
 %get candidate bounding boxes
    boxes = funcSol3(Im,k);
    k=k+1;
    chosen=1;
    iMax=[];
    %**********************LBP*****************
          %if there is more than one box output
    if size(boxes,1)>1
       for i = 1:size(boxes,1)
          corners = detectFASTFeatures(imcrop(Im,(boxes(i,:))));
          iMax(i) = length(corners);
       end
       [~,chosen]=max(iMax);
    end
       %if there are no boxes - choose alternative method
    if size(boxes,1) == 0 || length(detectFASTFeatures(imcrop(Im,(boxes(chosen,:)))))<50
       k=k-1;
       boxes = funcSol5(Im,l);
        l=l+1;
       chosen=1;
         end
  ITextRegion = insertShape(images(Rabiner & Juang), 'Rectangle',
boxes(chosen,:),'Color','green','LineWidth',2);
  imshow(ITextRegion);
   pause(1)
     % cropped LP images and saved it for recogntion stage
      imwrite(imcrop(images(Rabiner &
Juang),(boxes(chosen,:))),[toSave,'/',imagefiles(ii).name(1:end-4),'_cropped.jpg'])
end
%*************************************************
 cd ('C:\Users\U1069157\Desktop\LP-DETECTION\Tr-PositiveLP\'); % please replace "..."
by your images path
a = dir('*.jpg'); % directory of images, ".jpg" can be changed, for example, ".bmp" if you use
for i = 1 : length(a)
    ImgName = getfield(a, {i}, 'name');
    Imgdat = imread(ImgName);
    if size(Imgdat, 3) == 4 % if color images, convert it to gray
       Imgdat = rgb2gray(Imgdat);
```

```
    end
    [height width] = size(Imgdat);
    if i == 1
        VolData = zeros(height, width, length(a));
    end
   % VolData(:, :, i) = Imgdat;
end
RotateIndex = 1;
 % parameter set
% 1. the radii parameter in space and Time axis; They could be 1, 2 or 3 or 4
FRadius = 1;
TInterval = 2;
 % 2. the number of the neighboring points; It can be 2 and 4.
NeighborPoints = 4;
 TimeLength = 2;
BorderLength = 1;
bBilinearInterpolation = 1;
fHistogram = RIVLBP(VolData, TInterval, FRadius, NeighborPoints, BorderLength,
TimeLength, RotateIndex, bBilinearInterpolation);
FxRadius = 1;
FyRadius = 1;
TInterval = 2;
 TimeLength = 2;
BorderLength = 1;
 bBilinearInterpolation = 1;  % 0: not / 1: bilinear interpolation
Bincount = 59; %59 / 0
NeighborPoints = [8 8 8]; % XY, XT, and YT planes, respectively
if Bincount == 0
    Code = 0;
    nDim = 2 ^ (NeighborPoints(1));  %dimensionality of basic LBP
else
    % uniform patterns for neighboring points with 8
    U8File = importdata('UniformLBP8.txt');
    BinNum = U8File(1, 1);
    Code = U8File(2 : end, :);
    nDim = Bincount; %dimensionality of uniform patterns
    clear U8File;
end
% call LBPTOP
Histogram = LBPTOP(VolData, FxRadius, FyRadius, TInterval, NeighborPoints,
TimeLength, BorderLength, bBilinearInterpolation, Bincount, Code);
%**************************************************************
% Creating the training and testing sets
tr_n = 200;
te_n = 200;
weak_learner_n = 20;
tr_set = abs(rand(tr_n,2))*100;
te_set = abs(rand(te_n,2))*100;
 tr_labels = (tr_set(:,1)-tr_set(:,2) > 0) + 1;
te_labels = (te_set(:,1)-te_set(:,2) > 0) + 1;
```

```
 % Displaying the training and testing sets
 % Training and testing error rates
tr_error = zeros(1,weak_learner_n);
te_error = zeros(1,weak_learner_n);
 for i=1:weak_learner_n
    adaboost_model = ADABOOST_tr(@threshold_tr,@threshold_te,tr_set,tr_labels,i);
    [L_tr,hits_tr] = ADABOOST_te(adaboost_model,@threshold_te,tr_set,tr_labels);
    tr_error(i) = (tr_n-hits_tr)/tr_n;
    [L_te,hits_te] = ADABOOST_te(adaboost_model,@threshold_te,te_set,te_labels);
    te_error(i) = (te_n-hits_te)/te_n;
end
fprintf('The total No. of vechiles images is: ');
disp(ii);
fprintf('The Number of vheciles images detect by system(TP) is: ');
disp(k);
 fprintf('The Number of vheciles images not detect by system (FN) is: ');
disp(ii-k);
y= toc;
fprintf('The Processing Time per detected image is: %0f\n ', round(y/ii));
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% read images
clc
clear all
cd ('C:\Users\meeras\Desktop\codes\lbp\LBP_Matlab\Tr-PositiveLP\'); % please replace "..."
by your images path
a = dir('*.jpg'); % directory of images, ".jpg" can be changed, for example, ".bmp" if you use
for i = 1 : length(a)
    ImgName = getfield(a, {i}, 'name');
    Imgdat = imread(ImgName);
    if size(Imgdat, 3) == 3 % if color images, convert it to gray
        Imgdat = rgb2gray(Imgdat);
    end
    [height width] = size(Imgdat);
    if i == 1
        VolData = zeros(height, width, length(a));
    end
    VolData(:, :, i) = Imgdat;
end
cd ..
 RotateIndex = 1;
 % parameter set
% 1. the radii parameter in space and Time axis; They could be 1, 2 or 3 or 4
FRadius = 1;
TInterval = 2;
 % 2. the number of the neighboring points; It can be 2 and 4.
NeighborPoints = 4;
 % 3. "TimeLength" and "BorderLength" are the parameters for bordering parts in time and
% space which would not be computed for features. Usually they are same to TInterval and
% the bigger one of "FRadius";
TimeLength = 2;
```

```
BorderLength = 1;
% 4. "bBilinearInterpolation" : if use bilinear interpolation for computing a
% neighbor point in a circle: 1 (yes), 0 (not)
bBilinearInterpolation = 1;
 % call VLBP
fHistogram = RIVLBP(VolData, TInterval, FRadius, NeighborPoints, BorderLength,
TimeLength, RotateIndex, bBilinearInterpolation);
%% LBP-TOP
% parameter set
 % 1. "FxRadius", "FyRadius" and "TInterval" are the radii parameter along X, Y and T axis;
They can be 1, 2, 3 and 4. "1" and "3" are recommended.
%  Pay attention to "TInterval". "TInterval * 2 + 1" should be smaller than the length of the
input sequence "Length".
% For example, if one sequence includes seven frames, and you set TInterval
% to three, only the pixels in the frame 4 would be considered as central
% pixel and computed to get the LBP-TOP feature.
FxRadius = 1;
FyRadius = 1;
TInterval = 2;
 % 2. "TimeLength" and "BoderLength" are the parameters for bodering parts in time and
space which would not
% be computed for features. Usually they are same to TInterval and the
% bigger one of "FxRadius" and "FyRadius";
TimeLength = 2;
BorderLength = 1;

% 3. "bBilinearInterpolation" : if use bilinear interpolation for computing a
% neighbor point in a circle: 1 (yes), 0 (not)
bBilinearInterpolation = 1;  % 0: not / 1: bilinear interpolation
%% 59 is only for neighboring points with 8. If won't compute uniform
%% patterns, please set it to 0, then basic LBP will be computed
Bincount = 59; %59 / 0
NeighborPoints = [8 8 8]; % XY, XT, and YT planes, respectively
if Bincount == 0
   Code = 0;
   nDim = 2 ^ (NeighborPoints(1));  %dimensionality of basic LBP
else
   % uniform patterns for neighboring points with 8
   U8File = importdata('UniformLBP8.txt');
   BinNum = U8File(1, 1);
   Code = U8File(2 : end, :);
   nDim = Bincount; %dimensionality of uniform patterns
   clear U8File;
end
% call LBPTOP
Histogram = LBPTOP(VolData, FxRadius, FyRadius, TInterval, NeighborPoints,
TimeLength, BorderLength, bBilinearInterpolation, Bincount, Code);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
function [L,hits,error_rate] = threshold_te(model,test_set,sample_weights,true_labels)
```

```
% TESTING THRESHOLD CLASSIFIER
feat = test_set(:,model.dim);
if(strcmp(model.pos_neg,'pos'))
   ind = (feat>model.min_error_thr)+1;
else
   ind = (feat<model.min_error_thr)+1;
end
 hits = sum(ind==true_labels);
error_rate = sum(sample_weights(ind~=true_labels));
 L = zeros(length(feat),2);
L(ind==1,1) = 1;
L(ind==2,2) = 1;
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function model = adaboostBin(X, t, M)
% Adaboost for binary classification (weak learner: kmeans)
% Input:
%   X: d x n data matrix
%   t: 1 x n label (0/1)
% Output:
%   model: trained model structure
% Written by Mo Chen (sth4nth@gmail.com).
t = t+1;
k = 2;
[d,n] = size(X);
w = ones(1,n)/n;
%M = 100;
Alpha = zeros(1,M);
Theta = zeros(d,k,M);
T = sparse(1:n,t,1,n,k,n);  % transform label into indicator matrix
for m = 1:M
   % weak learner
   E = spdiags(w',0,n,n)*T;
   E = E*spdiags(1./sum(E,1)',0,k,k);
   c = X*E;
   [~,y] = min(sqdist(c,X),[],1);
   Theta(:,:,m) = c;
   % adaboost
   I = y~=t;
   e = dot(w,I);
   alpha = log((1-e)/e);
   w = w.*exp(alpha*I);
   w = w/sum(w);
   Alpha(m) = alpha;
end
model.alpha = Alpha;
model.theta = Theta;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function t = adaboostBinPred(model, X)
% Prediction of binary Adaboost
```

```
% input:
%   model: trained model structure
%   X: d x n data matrix
% output:
%   t: 1 x n prediction
Alpha = model.alpha;
Theta = model.theta;
M = size(Alpha,2);
t = zeros(1,size(X,2));
for m = 1:M
    c = Theta(:,:,m);
    [~,y] = min(sqdist(c,X),[],1);
    y(y==1) = -1;
    y(y==2) = 1;
    t = t+Alpha(m)*y;
end
t = sign(t);
t(t==-1) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function blocks = cirInterpSingleRadius(img)
 global lbpPoints;
global lbpRadius;
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
imgNewW = imgW - 2*lbpRadius;
 % the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
 radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
 % Determine the dimensions of the input img.
[ysize,xsize] = size(img);
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
    spoints(i,1) = -radius * sin((i-1)*angleStep);
    spoints(i,2) = radius * cos((i-1)*angleStep);
end
 miny = min(spoints(:,1));
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
```

```
if(xsize < bsizex || ysize < bsizey)
   error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
 % Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
 % Compute the LBP code img
for i = 1 : neighbors
   y = spoints(i,1) + origy;
   x = spoints(i,2) + origx;
   % Calculate floors, ceils and rounds for the x and y.
   fy = floor(y);
   cy = ceil(y);
   ry = round(y);
      fx = floor(x);
   cx = ceil(x);
   rx = round(x);
      % Check if interpolation is needed.
   if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
      % Interpolation is not needed, use original datatypes
      imgNew = img(ry:ry+dy,rx:rx+dx);
      blocks(i,:) = imgNew(:)';
   else
      % Interpolation needed, use double type images
      ty = y - fy;
      tx = x - fx;
            % Calculate the interpolation weights.
      w1 = (1 - tx) * (1 - ty);
      w2 =      tx  * (1 - ty);
      w3 = (1 - tx) *      ty ;
      w4 =      tx  *      ty ;
      % Compute interpolated pixel values
      imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
         w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
      blocks(i,:) = imgNew(:)';
   end
end % loop neighbors
 end % end of the function
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function blocks = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius)
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
imgNewW = imgW - 2*lbpRadius;
 % the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
 radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
 % Determine the dimensions of the input img.
[ysize,xsize] = size(img);
```

```
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
   spoints(i,1) = -radius * sin((i-1)*angleStep);
   spoints(i,2) = radius * cos((i-1)*angleStep);
end
 miny = min(spoints(:,1));
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
   error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
 % Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
 % Compute the LBP code img
for i = 1 : neighbors
   y = spoints(i,1) + origy;
   x = spoints(i,2) + origx;
   % Calculate floors, ceils and rounds for the x and y.
   fy = floor(y);
   cy = ceil(y);
   ry = round(y);
      fx = floor(x);
   cx = ceil(x);
   rx = round(x);
      % Check if interpolation is needed.
   if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
      % Interpolation is not needed, use original datatypes
      imgNew = img(ry:ry+dy,rx:rx+dx);
      blocks(i,:) = imgNew(:)';
   else
      % Interpolation needed, use double type images
      ty = y - fy;
      tx = x - fx;
            % Calculate the interpolation weights.
      w1 = (1 - tx) * (1 - ty);
      w2 =      tx  * (1 - ty);
      w3 = (1 - tx) *      ty ;
      w4 =      tx  *      ty ;
      % Compute interpolated pixel values
```

```
        imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
            w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
        blocks(i,:) = imgNew(:)';
    end
end % loop neighbors
 end % end of the function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mapping = get_mapping(samples)
 numAllLBPs = 2^samples;
table = 0 : numAllLBPs-1;
newMax = samples + 2; % number of patterns in the resulting LBP code

for i = 0:2^samples - 1
    j = bitset(bitshift(i,1),1,bitget(i,samples)); % rotate left
    numt = sum(bitget(bitxor(i,j),1:samples));
    if numt <= 2
        table(i+1) = sum(bitget(i,1:samples));
    else
        table(i+1) = samples+1;
    end
end

mapping.table = table;
mapping.samples = samples;
mapping.num = newMax;
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--------------------------------Training Stage
global nump1 %number of positive samples 1
global nump2 %number of positive samples 2
global nump3 %number of positive samples 3
global numn  %number of negative samples
global path1_1
global path2_1
global path3_1
global path4_1
tic
%% preprocessing of train images
path1 = 'train/rect';   %rectangle training lps
path2 = 'train/slope';  %slope rectangle training lps
path3 = 'train/square'; %square training lps
path4 = 'train/nonlp';  %trainging nonlps

path1_1 = 'realtrain/rect/';   %rectangle training lps
path2_1 = 'realtrain/slope/';  %slope rectangle training lps
path3_1 = 'realtrain/square/'; %square training lps
path4_1 = 'realtrain/nonlp/';  %trainging nonlps

%preprocessed training lps will be saved in 'realtrain' directory
if ~isdir('realtrain')
```

```
    mkdir('realtrain');
    mkdir('realtrain/rect');
    mkdir('realtrain/slope');
    mkdir('realtrain/square');
    mkdir('realtrain/nonlp')
end
 %image files in directories
files1 = dir(fullfile(path1,'*.jpg'));
files2 = dir(fullfile(path2,'*.jpg'));
files3 = dir(fullfile(path3,'*.jpg'));
files4 = dir(fullfile(path4,'*.jpg'));
 %number of positive and negative trainging samples
nump1 = numel(files1); %number of rectangle training lps = positive 1
nump2 = numel(files2); %number of slope training lps = positive 2
nump3 = numel(files3); %number of square training lps = positive 3
numn = numel(files4);  %number of nonlps = negative
 %preprocessing of rectangle training lps
for samples = 1 : nump1
    file = fullfile(path1, files1(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);

    M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
    [m n] = size(M);
    p = 20;
    q = 80;
       if p >= m && q >= n
       M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q >= n
       m = p;
       M_pad = imresize(M, [m, n]);
       M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p >= m && q < n
       n = q;
       M_pad = imresize(M, [m, n]);
       M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q < n
       M_pad = imresize(M, [p, q]);
    end
       M = M_pad;
      numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
```

```
    fin_lc=zeros(256,1);
       for i=1:size(M,1)
       for j=1:size(M,2)
           value=M(i,j);
           cnts(value+1)=cnts(value+1)+1;
           probf(value+1)=cnts(value+1)/numofpixels;
       end
    end
    sum=0;
    n=255;
       for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
            Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
     imwrite(Y, strcat(path1_1, num2str(samples), '.jpg'));
end
 %preprocessing of slope rectangle training lps
for samples = 1 : nump2
    file = fullfile(path2, files2(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
       M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
       [m n] = size(M);
    p = 25;
    q = 70;
       if p >= m && q >= n
       M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q >= n
       m = p;
       M_pad = imresize(M, [m, n]);
       M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p >= m && q < n
       n = q;
       M_pad = imresize(M, [m, n]);
       M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q < n
       M_pad = imresize(M, [p, q]);
    end
```

```
    M = M_pad;
    numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
       for i=1:size(M,1)
       for j=1:size(M,2)
          value=M(i,j);
          cnts(value+1)=cnts(value+1)+1;
          probf(value+1)=cnts(value+1)/numofpixels;
       end
    end
   sum=0;
    n=255;
       for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
            Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
     imwrite(Y, strcat(path2_1, num2str(samples), '.jpg'));
end
 %preprocessing of square training lps
for samples = 1 : nump3
    file = fullfile(path3, files3(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
       M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
      [m n] = size(M);
    p = 32;
    q = 45;
       if p >= m && q >= n
       M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q >= n
       m = p;
       M_pad = imresize(M, [m, n]);
       M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
       M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p >= m && q < n
```

```
        n = q;
        M_pad = imresize(M, [m, n]);
        M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q < n
        M_pad = imresize(M, [p, q]);
    end
        M = M_pad;
     numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
        for i=1:size(M,1)
        for j=1:size(M,2)
            value=M(i,j);
            cnts(value+1)=cnts(value+1)+1;
            probf(value+1)=cnts(value+1)/numofpixels;
        end
    end
        sum=0;
    n=255;
        for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
        for i=1:size(M,1)
          for j=1:size(M,2)
              Y(i,j)=fin_lc(M(i,j)+1);
          end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
        imwrite(Y, strcat(path3_1, num2str(samples), '.jpg'));
end
 %preprocessing of training nonlps
for samples = 1 : numn
    file = fullfile(path4, files4(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
     M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
        numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
```

```
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
       for i=1:size(M,1)
       for j=1:size(M,2)
           value=M(i,j);
           cnts(value+1)=cnts(value+1)+1;
           probf(value+1)=cnts(value+1)/numofpixels;
       end
    end
       sum=0;
    n=255;
       for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
             Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
    imwrite(Y, strcat(path4_1, num2str(samples), '.jpg'));
end
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
--------------------------------Testing stage
global Y1
tic
%import trained model
model1 = load('model1.mat');
model2 = load('model2.mat');
model3 = load('model3.mat');
 if ~isdir('result')
    mkdir('result');
end
if ~isdir('detected')
       mkdir('detected');
       rmdir('detected');
end
if ~isdir('detected')
       mkdir('detected');
end
%% testing
files = dir(fullfile('test','*.jpg'));
for id = 1 : numel(files)
    file = fullfile('test/', files(id).name);
      %preprocessing
    X=imread(file);
```

```
M = imgaussfilt(X,0.25);
M = imresize(M,[240, 320]);
M=rgb2gray(M);
numofpixels=size(M,1)*size(M,2);
Y=uint8(zeros(size(M,1),size(M,2)));
cnts=zeros(256,1);
probf=zeros(256,1);
prbc=zeros(256,1);
cum=zeros(256,1);
fin_lc=zeros(256,1);
 for i=1:size(M,1)
    for j=1:size(M,2)
        value=M(i,j);
        cnts(value+1)=cnts(value+1)+1;
        probf(value+1)=cnts(value+1)/numofpixels;
    end
end
 sum=0;
n=255;
 for i=1:size(probf)
   sum=sum+cnts(i);
   cum(i)=sum;
   prbc(i)=cum(i)/numofpixels;
   fin_lc(i)=round(prbc(i)*n);
end
 for i=1:size(M,1)
   for j=1:size(M,2)
      Y(i,j)=fin_lc(M(i,j)+1);
   end
end
 Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
W = imresize(Y,[480, 640]);
 lbpRadiusSet = [2 4 6 8];
lbpPointsSet = [8 8 8 8];
 % selecting mrelbp features from preprocessed test image
xsum=0;
ysum=0;
num = 0;
px = [];
py = [];
px1 = [];
py1 = [];
px2 = [];
py2 = [];
for i = 95 : 4 : size(Y,1) - 40
   for j = 30 : 24 : size(Y,2) - 120
       Y1 = Y(i:i+19,j:j+79);
        testfeatures1 = [];
        testfeatures2 = [];
       testfeatures3 = [];
```

```
        for idxLbpRadius = 1 : length(lbpRadiusSet)
            lbpRadius = lbpRadiusSet(idxLbpRadius);
            lbpPoints = lbpPointsSet(idxLbpRadius);
             mapping = get_mapping(lbpPoints);
            blockSize = lbpRadius*2+1;
             if idxLbpRadius > 1
                lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
            else
                lbpRadiusPre = 0;
            end
             cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,1,1);
             testfeatures1 = [testfeatures1 cfmsWithLabels_LBP];
            cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,1,2);
             testfeatures2 = [testfeatures2 cfmsWithLabels_LBP];
            cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,1,3);
             testfeatures3 = [testfeatures3 cfmsWithLabels_LBP];
        end
        testclass1=adaboostBinPred(model1,testfeatures1');
       testclass2=adaboostBinPred(model2,testfeatures2');
       testclass3=adaboostBinPred(model3,testfeatures3');
        if testclass1 == 1 ||   testclass2 == 1 || testclass3 == 1
            num = num + 1;
            px = [px i * 2];
            py = [py j * 2];
            result = imcrop(X,[i*2 j*2 250 45]);
            imwrite(result, strcat('detected/', num2str(id), '_cropped_', num2str(num), '.jpg'));
        end
    end
  end
   fh = figure('Name','LP detection','NumberTitle','off');
  subplot(2,1,1);
  imshow( W, 'border', 'tight' ); %//show your image
  subplot(2,1,2);
  imshow( X, 'border', 'tight' ); %//show improve image
  hold on;
  for i = 1 : length(px)
      rectangle('Position', [py(i) px(i) 250 45], 'EdgeColor', 'g'); %// draw rectangle on image
      frm = getframe( fh ); %// get the image+rectangle
      imwrite( frm.cdata, strcat('result/',num2str(id),'.jpg') ); %// save to file
  end
  pause(3);
  close(fh);
  disp(strcat('test image ', num2str(id), ' done' ));
  disp(strcat('The number of ensemble classifiers is: ') );
 num
end
 toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global nump1 %number of positive samples 1
global nump2 %number of positive samples 2
```

```
global nump3 %number of positive samples 3
global numn  %number of negative samples
 global lbpRadius
global lbpPoints
tic
%% selecting mrelbp features from preprocessed train images
lbpRadiusSet = [2 4 6 8];
lbpPointsSet = [8 8 8 8];
 trainfeatures1 = [];
trainfeatures2 = [];
trainfeatures3 = [];
trainfeatures4 = [];
for idxLbpRadius = 1 : length(lbpRadiusSet)
    lbpRadius = lbpRadiusSet(idxLbpRadius);
    lbpPoints = lbpPointsSet(idxLbpRadius);
    mapping = get_mapping(lbpPoints);
    blockSize = lbpRadius*2+1;
    if idxLbpRadius > 1
        lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
    else
        lbpRadiusPre = 0;
    end
    cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,1);
    trainfeatures1 = [trainfeatures1 cfmsWithLabels_LBP];
    clear cfmsWithLabels_LBP
    cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,2);
    trainfeatures2 = [trainfeatures2 cfmsWithLabels_LBP];
    clear cfmsWithLabels_LBP
    cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,3);
    trainfeatures3 = [trainfeatures3 cfmsWithLabels_LBP];
    clear cfmsWithLabels_LBP
    cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,4);
    trainfeatures4 = [trainfeatures4 cfmsWithLabels_LBP];
end
 trainfeatures1 = [trainfeatures1; trainfeatures4];
trainfeatures2 = [trainfeatures2; trainfeatures4];
trainfeatures3 = [trainfeatures3; trainfeatures4];
classes1 = zeros(1, 2030);
for i = 1 : 500
    classes1(i) = 1;
end
classes2 = zeros(1, 1607);
for i = 1 : 77
    classes2(i) = 1;
end
classes3 = zeros(1, 1574);
for i = 1 : 44
    classes3(i) = 1;
end
```

```
--------------------------------Trainned models
model1=adaboostBin(trainfeatures1',classes1,100);
model2=adaboostBin(trainfeatures2',classes2,100);
model3=adaboostBin(trainfeatures3',classes3,100);
 save('model1.mat', '-struct', 'model1');
save('model2.mat', '-struct', 'model2');
save('model3.mat', '-struct', 'model3');
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

APPENDIX B

B

Matlab simulation code for Chapter 4

An efficient texture descriptor for the detection of license plates from vehicle images in difficult conditions

The simulation codes to detect LPs from vehicles images having difficult conditions are presented. The experiment results were obtained using Matlab programming language version R2018a.

```
---------------------------------ELM CLASSIFIER----------------
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
ActivationFunction)
 REGRESSION=0;
CLASSIFIER=1;
 %%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                   %   Release raw training data array
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                    %   Release raw testing data array
NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
NumberofInputNeurons=size(P,1);
 if Elm_Type~=REGRESSION
    %%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1,1);                               %   Find and save in 'label' class label from training
and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberofTrainingData+NumberofTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;
        %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
     %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
```

```
            break;
        end
    end
    temp_TV_T(j,i)=1;
  end
  TV.T=temp_TV_T*2-1;
end                                 %   end if of Elm_Type
%%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
%%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                            %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);          %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
%%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
  case {'sig','sigmoid'}
    %%%%%%%% Sigmoid
    H = 1 ./ (1 + exp(-tempH));
  case {'sin','sine'}
    %%%%%%%% Sine
    H = sin(tempH);
  case {'hardlim'}
    %%%%%%%% Hard Limit
    H = double(hardlim(tempH));
  case {'tribas'}
    %%%%%%%% Triangular basis function
    H = tribas(tempH);
  case {'radbas'}
    %%%%%%%% Radial basis function
    H = radbas(tempH);
    %%%%%%%% More activation functions can be added here
end
clear tempH;                %   Release the temparary array for calculation of hidden neuron
output matrix H
%%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T'

end_time_train=cputime;
TrainingTime=end_time_train-start_time_train       %   Calculate CPU time (seconds) spent
for training ELM

%%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                     %   Y: the actual output of the training data
if Elm_Type == REGRESSION
```

```
    TrainingAccuracy=sqrt(mse(T - Y))             %   Calculate training accuracy (RMSE) for
regression case
end
clear H;
 %%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
    case {'tribas'}
        %%%%%%%% Triangular basis function
        H_test = tribas(tempH_test);
    case {'radbas'}
        %%%%%%%% Radial basis function
        H_test = radbas(tempH_test);
        %%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';                       %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test           %   Calculate CPU time (seconds) spent by
ELM predicting the whole testing data
 if Elm_Type == REGRESSION
    TestingAccuracy=sqrt(mse(TV.T - TY))            %   Calculate testing accuracy (RMSE) for
regression case
end
 if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;
     for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
```

```
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy,TY] =
elm_kernel(TrainingData_File, TestingData_File, Elm_Type, Regularization_coefficient,
Kernel_type, Kernel_para)
%%%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;
%%%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                          %   Release raw training data array
 %%%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                           %   Release raw testing data array
C = Regularization_coefficient;
NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
if Elm_Type~=REGRESSION
    %%%%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(cat(2,T,TV.T),2);
    label=zeros(1,1);        %   Find and save in 'label' class label from training and testing data
sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:(NumberofTrainingData+NumberofTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;
        %%%%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
```

```
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
     %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;
                                %   end if of Elm_Type
end
 %%%%%%%%%% Training Phase %
tic;
n = size(T,2);
Omega_train = kernel_matrix(P',Kernel_type, Kernel_para);
OutputWeight=((Omega_train+speye(n)/C)\(T'));
TrainingTime=toc
 %%%%%%%%%% Calculate the training output
Y=(Omega_train * OutputWeight)';        %   Y: the actual output of the training data
%%%%%%%%%% Calculate the output of testing input
tic;
Omega_test = kernel_matrix(P',Kernel_type, Kernel_para,TV.P');
TY=(Omega_test' * OutputWeight)';                       %   TY: the actual output of the testing
data
TestingTime=toc
 %%%%%%%%%% Calculate training & testing classification accuracy
 if Elm_Type == REGRESSION
%%%%%%%%%% Calculate training & testing accuracy (RMSE) for regression case
    TrainingAccuracy=sqrt(mse(T - Y))
    TestingAccuracy=sqrt(mse(TV.T - TY))
end
 if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
    MissClassificationRate_Testing=0;
     for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
    end
    TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
```

```
    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end
%%%%%%%%%%%%%%%%%% Kernel Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function omega = kernel_matrix(Xtrain,kernel_type, kernel_pars,Xt)
 nb_data = size(Xtrain,1);
if strcmp(kernel_type,'RBF_kernel'),
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        omega = exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*Xtrain*Xt';
        omega = exp(-omega./kernel_pars(1));
    end
elseif strcmp(kernel_type,'lin_kernel')
    if nargin<4,
        omega = Xtrain*Xtrain';
    else
        omega = Xtrain*Xt';
    end
elseif strcmp(kernel_type,'poly_kernel')
    if nargin<4,
        omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
    else
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end
elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
        omega1 = XXh1-XXh1';
        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1));
    else
        XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
        XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
        omega = XXh1+XXh2' - 2*(Xtrain*Xt');

        XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
        XXh22 = sum(Xt,2)*ones(1,nb_data);
```

```
        omega1 = XXh11-XXh22';
        omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1));
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
elm_MultiOutputRegression(TrainingData_File, TestingData_File, No_of_Output,
NumberofHiddenNeurons, ActivationFunction)
%%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1:No_of_Output)';
P=train_data(:,No_of_Output+1:size(train_data,2))';
clear train_data;                                %   Release raw training data array
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1:No_of_Output)';
TV.P=test_data(:,No_of_Output+1:size(test_data,2))';
clear test_data;                                 %   Release raw testing data array
 NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
NumberofInputNeurons=size(P,1);
 %%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
 %%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                                         %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
%%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = hardlim(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                                     %   Release the temparary array for calculation of
hidden neuron output matrix H
 %%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';
```

```
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train       %   Calculate CPU time (seconds) spent
for training ELM
 %%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                          %   Y: the actual output of the training data
TrainingAccuracy=sqrt(mse(T - Y))              %   Calculate training accuracy (RMSE) for
regression case
clear H;
 %%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%%% Sigmoid
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
        %%%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';                       %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test          %   Calculate CPU time (seconds) spent by
ELM predicting the whole testing data

TestingAccuracy=sqrt(mse(TV.T - TY))           %   Calculate testing accuracy (RMSE) for
regression case
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [TestingTime, TestingAccuracy, output] = elm_predict(TestingData_File)
 %%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                %   Release raw testing data array
NumberofTestingData=size(TV.P,2);
load elm_model.mat;
 if Elm_Type~=REGRESSION
    %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
```

```
    for i = 1:NumberofTestingData
        for j = 1:size(label,2)
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;

end                                     %   end if of Elm_Type
%%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
        %%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';                       %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test;          %   Calculate CPU time (seconds) spent
by ELM predicting the whole testing data

if Elm_Type == REGRESSION
    TestingAccuracy=sqrt(mse(TV.T - TY));           %   Calculate testing accuracy (RMSE) for
regression case
    output=TY;
end

if Elm_Type == CLASSIFIER
%%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Testing=0;

    for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        output(i)=label(label_index_actual);
```

```
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/NumberofTestingData;
end
 save('elm_output','output');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [TrainingTime,TrainingAccuracy] = elm_train(TrainingData_File, Elm_Type,
NumberofHiddenNeurons, ActivationFunction)
 %%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;
 %%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                    %   Release raw training data array
NumberofTrainingData=size(P,2);
NumberofInputNeurons=size(P,1);
if Elm_Type~=REGRESSION
    %%%%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(T,2);
    label=zeros(1,1);                    %   Find and save in 'label' class label from training
and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:NumberofTrainingData
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;
        %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
end                                  %   end if of Elm_Type
 %%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
```

```
 %%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                                %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);          %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
 %%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = hardlim(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                                %   Release the temparary array for calculation of
hidden neuron output matrix H
 %%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train        %   Calculate CPU time (seconds) spent
for training ELM

%%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                         %   Y: the actual output of the training data
if Elm_Type == REGRESSION
    TrainingAccuracy=sqrt(mse(T - Y))               %   Calculate training accuracy (RMSE) for
regression case
    output=Y;
end
clear H;

if Elm_Type == CLASSIFIER
%%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Training=0;
     for i = 1 : size(T, 2)
        [x, label_index_expected]=max(T(:,i));
        [x, label_index_actual]=max(Y(:,i));
        output(i)=label(label_index_actual);
        if label_index_actual~=label_index_expected
            MissClassificationRate_Training=MissClassificationRate_Training+1;
        end
```

```
    end
    TrainingAccuracy=1-MissClassificationRate_Training/NumberofTrainingData
end

if Elm_Type~=REGRESSION
    save('elm_model', 'NumberofInputNeurons', 'NumberofOutputNeurons', 'InputWeight',
'BiasofHiddenNeurons', 'OutputWeight', 'ActivationFunction', 'label', 'Elm_Type');
else
    save('elm_model', 'InputWeight', 'BiasofHiddenNeurons', 'OutputWeight',
'ActivationFunction', 'Elm_Type');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function blocks = cirInterpSingleRadius(img)
 global lbpPoints;
global lbpRadius;
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
imgNewW = imgW - 2*lbpRadius;
% the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
 radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
 % Determine the dimensions of the input img.
[ysize,xsize] = size(img);
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
    spoints(i,1) = -radius * sin((i-1)*angleStep);
    spoints(i,2) = radius * cos((i-1)*angleStep);
end
miny = min(spoints(:,1));
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
    error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
% Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
 % Compute the LBP code img
```

```
for i = 1 : neighbors
   y = spoints(i,1) + origy;
   x = spoints(i,2) + origx;
   % Calculate floors, ceils and rounds for the x and y.
   fy = floor(y);
   cy = ceil(y);
   ry = round(y);
   fx = floor(x);
   cx = ceil(x);
   rx = round(x);
      % Check if interpolation is needed.
   if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
      % Interpolation is not needed, use original datatypes
      imgNew = img(ry:ry+dy,rx:rx+dx);
      blocks(i,:) = imgNew(:)';
   else
      % Interpolation needed, use double type images
      ty = y - fy;
      tx = x - fx;
      % Calculate the interpolation weights.
      w1 = (1 - tx) * (1 - ty);
      w2 =      tx  * (1 - ty);
      w3 = (1 - tx) *      ty ;
      w4 =      tx  *      ty ;
      % Compute interpolated pixel values
      imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
         w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
      blocks(i,:) = imgNew(:)';
   end
end % loop neighbors
end % end of the function
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function blocks = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius)
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
imgNewW = imgW - 2*lbpRadius;
% the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
% Determine the dimensions of the input img.
[ysize,xsize] = size(img);
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
   spoints(i,1) = -radius * sin((i-1)*angleStep);
   spoints(i,2) = radius * cos((i-1)*angleStep);
end
 miny = min(spoints(:,1));
```

```
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
   error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
 % Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
% Compute the LBP code img
for i = 1 : neighbors
   y = spoints(i,1) + origy;
   x = spoints(i,2) + origx;
   % Calculate floors, ceils and rounds for the x and y.
   fy = floor(y);
   cy = ceil(y);
   ry = round(y);
    fx = floor(x);
   cx = ceil(x);
   rx = round(x);
   % Check if interpolation is needed.
   if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
      % Interpolation is not needed, use original datatypes
      imgNew = img(ry:ry+dy,rx:rx+dx);
      blocks(i,:) = imgNew(:)';
   else
      % Interpolation needed, use double type images
      ty = y - fy;
      tx = x - fx;
      % Calculate the interpolation weights.
      w1 = (1 - tx) * (1 - ty);
      w2 =      tx  * (1 - ty);
      w3 = (1 - tx) *      ty ;
      w4 =      tx  *      ty ;
      % Compute interpolated pixel values
      imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
         w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
      blocks(i,:) = imgNew(:)';
   end
end % loop neighbors
 end % end of the function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function mapping = get_mapping(samples)
numAllLBPs = 2^samples;
table = 0 : numAllLBPs-1;
newMax = samples + 2; % number of patterns in the resulting LBP code
for i = 0:2^samples - 1
    j = bitset(bitshift(i,1),1,bitget(i,samples)); % rotate left
    numt = sum(bitget(bitxor(i,j),1:samples));
    if numt <= 2
        table(i+1) = sum(bitget(i,1:samples));
    else
        table(i+1) = samples+1;
    end
end
mapping.table = table;
mapping.samples = samples;
mapping.num = newMax;
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,check,setnum)
 global nump  %number of positive samples
global numn  %number of negative samples
 global pathp
global pathn
 global lbpRadius
global lbpPoints
global Y1
 numLBPbins = mapping.num;
 if check == 0
    if setnum == 1
        samplenum = nump;
        path = pathp;
    elseif setnum == 2
        samplenum = numn;
        path = pathn;
    end
    cfmsWithLabels_MRELBP_CINIRD =
zeros(samplenum,(numLBPbins*numLBPbins*2));
    for idxSample = 1 : samplenum
        Joint_CINIRD = zeros(numLBPbins,numLBPbins,2);
        img = imread(strcat(path, num2str(idxSample), '.jpg'));
         img = samp_prepro(img);
         imgExt = padarray(img,[1 1],'symmetric','both');
        imgblks = im2col(imgExt,[3 3],'sliding');
        a = median(imgblks);
        b = reshape(a,size(img));
        CImg = b(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
        CImg = CImg(:) - mean(CImg(:));
        CImg(CImg >= 0) = 2;
        CImg(CImg < 0) = 1;
        if lbpRadius == 2
```

```
        filWin = 3;
        halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgCurr = reshape(imgMedian,size(img));
        NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
        NILBPImage = NILBPImage(:);
        histNI = hist(NILBPImage,0:(numLBPbins-1));
        NILBPImage = NILBPImage + 1;
         RDLBPImage =
RDLBP_Image_SmallestRadiusOnly(b,imgCurr,lbpRadius,lbpPoints,mapping,'image');
        RDLBPImage = RDLBPImage(:);
        histRD = hist(RDLBPImage,0:(numLBPbins-1));
        RDLBPImage = RDLBPImage + 1;
    else
        if mod(lbpRadius,2) == 0
            filWin = lbpRadius + 1;
        else
            filWin = lbpRadius;
        end
        halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgCurr = reshape(imgMedian,size(img));
        NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
        NILBPImage = NILBPImage(:);
        histNI = hist(NILBPImage,0:(numLBPbins-1));
        NILBPImage = NILBPImage + 1;
         if mod(lbpRadiusPre,2) == 0
            filWin = lbpRadiusPre + 1;
        else
            filWin = lbpRadiusPre;
        end
         halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgPre = reshape(imgMedian,size(img));
         RDLBPImage =
NewRDLBP_Image(imgCurr,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,'image');
        RDLBPImage = RDLBPImage(:);
        histRD = hist(RDLBPImage,0:(numLBPbins-1));
        RDLBPImage = RDLBPImage + 1;
    end
        for i = 1 : length(NILBPImage)
        Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) =
Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) + 1;
    end
```

```
        cfmsWithLabels_MRELBP_CINIRD(idxSample,:) = Joint_CINIRD(:)';
    end
else
    cfmsWithLabels_MRELBP_CINIRD = zeros(1,(numLBPbins*numLBPbins*2));
    Joint_CINIRD = zeros(numLBPbins,numLBPbins,2);
     img = samp_prepro(Y1);
    imgExt = padarray(img,[1 1],'symmetric','both');
    imgblks = im2col(imgExt,[3 3],'sliding');
    a = median(imgblks);
    b = reshape(a,size(img));
    CImg = b(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
    CImg = CImg(:) - mean(CImg(:));
    CImg(CImg >= 0) = 2;
    CImg(CImg < 0) = 1;
    if lbpRadius == 2
        filWin = 3;
        halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgCurr = reshape(imgMedian,size(img));
        NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
        NILBPImage = NILBPImage(:);
        histNI = hist(NILBPImage,0:(numLBPbins-1));
        NILBPImage = NILBPImage + 1;
         RDLBPImage =
RDLBP_Image_SmallestRadiusOnly(b,imgCurr,lbpRadius,lbpPoints,mapping,'image');
        RDLBPImage = RDLBPImage(:);
        histRD = hist(RDLBPImage,0:(numLBPbins-1));
        RDLBPImage = RDLBPImage + 1;
    else
        if mod(lbpRadius,2) == 0
            filWin = lbpRadius + 1;
        else
            filWin = lbpRadius;
        end
        halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        % each column of imgblks represents a feature vector
        imgMedian = median(imgblks);
        imgCurr = reshape(imgMedian,size(img));
        NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
        NILBPImage = NILBPImage(:);
        histNI = hist(NILBPImage,0:(numLBPbins-1));
        NILBPImage = NILBPImage + 1;
         if mod(lbpRadiusPre,2) == 0
            filWin = lbpRadiusPre + 1;
        else
            filWin = lbpRadiusPre;
```

```
        end
         halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgPre = reshape(imgMedian,size(img));
         RDLBPImage =
NewRDLBP_Image(imgCurr,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,'image');
        RDLBPImage = RDLBPImage(:);
        histRD = hist(RDLBPImage,0:(numLBPbins-1));
        RDLBPImage = RDLBPImage + 1;
    end
       for i = 1 : length(NILBPImage)
        Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) =
Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) + 1;
    end
    cfmsWithLabels_MRELBP_CINIRD = Joint_CINIRD(:)';
end
cfmsWithLabels_LBP = cfmsWithLabels_MRELBP_CINIRD;
clear cfmsWithLabels_MRELBP_CINIRD;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result =
NewRDLBP_Image(img,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,mode)
blocks1 = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius);
blocks1 = blocks1';
 imgPre = imgPre(lbpRadius-lbpRadiusPre+1:end-(lbpRadius-lbpRadiusPre),lbpRadius-
lbpRadiusPre+1:end-(lbpRadius-lbpRadiusPre));
blocks2 = cirInterpSingleRadiusNew(imgPre,lbpPoints,lbpRadiusPre);
blocks2 = blocks2';
 radialDiff = blocks1 - blocks2;
radialDiff(radialDiff >= 0) = 1;
radialDiff(radialDiff < 0) = 0;
 bins = 2^lbpPoints;
weight = 2.^(0:lbpPoints-1);
radialDiff = radialDiff .* repmat(weight,size(radialDiff,1),1);
% mapping = getmapping(lbpPoints,'riu2');
 radialDiff = sum(radialDiff,2);
result = radialDiff;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end
 if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
```

```
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
else
    % Otherwise return a matrix of unsigned integers
    if ((bins-1) <= intmax('uint8'))
        result = uint8(result);
    elseif ((bins-1) <= intmax('uint16'))
        result = uint16(result);
    else
        result = uint32(result);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = NILBP_Image(img,lbpPoints,mapping,mode)
blocks = cirInterpSingleRadius(img);
blocks = blocks';
blocks = blocks - repmat(mean(blocks,2),1,size(blocks,2));
blocks(blocks >= 0) = 1;
blocks(blocks < 0) = 0;
 weight = 2.^(0:lbpPoints-1);
blocks = blocks .* repmat(weight,size(blocks,1),1);
blocks = sum(blocks,2);
result = blocks;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end

if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
else
    % Otherwise return a matrix of unsigned integers
    % result = reshape(result,size(imgTemp));
    if ((bins-1) <= intmax('uint8'))
        result = uint8(result);
    elseif ((bins-1) <= intmax('uint16'))
        result = uint16(result);
    else
        result = uint32(result);
```

```
    end
end

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result =
RDLBP_Image_SmallestRadiusOnly(imgCenSmooth,img,lbpRadius,lbpPoints,mapping,mod
e)
blocks1 = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius);
blocks1 = blocks1';
imgTemp = imgCenSmooth(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
blocks2 = repmat(imgTemp(:),1,lbpPoints);
 radialDiff = blocks1 - blocks2;
 radialDiff(radialDiff >= 0) = 1;
radialDiff(radialDiff < 0) = 0;
 bins = 2^lbpPoints;
weight = 2.^(0:lbpPoints-1);
radialDiff = radialDiff .* repmat(weight,size(radialDiff,1),1);
% mapping = getmapping(lbpPoints,'riu2');
 radialDiff = sum(radialDiff,2);
result = radialDiff;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end
 if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
else
    % Otherwise return a matrix of unsigned integers
    if ((bins-1) <= intmax('uint8'))
        result = uint8(result);
    elseif ((bins-1) <= intmax('uint16'))
        result = uint16(result);
    else
        result = uint32(result);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sampleIn = samp_prepro(sampleIn)
% image sample preprocessing
sampleIn = double(sampleIn);
sampleIn = sampleIn - mean(sampleIn(:));
```

```
% sampleIn = sampleIn / sqrt(mean(mean(sampleIn .^ 2)));
sampleIn = sampleIn / std(sampleIn(:));
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function D = sqdist(X1, X2)
D = bsxfun(@plus,dot(X2,X2,1),dot(X1,X1,1)')-2*(X1'*X2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
global nump  %number of positive samples
global numn  %number of negative samples
 global pathp
global pathn
 %% preprocessing of train images
tic
path1 = 'train/lp';   %training lps
path2 = 'train/nonlp';  %trainging nonlps
 pathp = 'realtrain/lp/';   %rectangle training lps
pathn = 'realtrain/nonlp/';  %trainging nonlps
 %preprocessed training lps will be saved in 'realtrain' directory
if ~isdir('realtrain')
    mkdir('realtrain');
    mkdir('realtrain/lp');
    mkdir('realtrain/nonlp')
end
 %image files in directories
files1 = dir(fullfile(path1,'*.jpg'));
files2 = dir(fullfile(path2,'*.jpg'));

%number of positive and negative trainging samples
nump = numel(files1); %number of training lps = positive
numn = numel(files2);  %number of nonlps = negative
 %preprocessing of training lps
for samples = 1 : nump
    file = fullfile(path1, files1(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
       M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
    [m n] = size(M);
    p = 25;
    q = 100;
    if p >= m && q >= n
        M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q >= n
        m = p;
        M_pad = imresize(M, [m, n]);
        M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p >= m && q < n
        n = q;
```

```
        M_pad = imresize(M, [m, n]);
        M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q < n
        M_pad = imresize(M, [p, q]);
    end
    M = M_pad;
    numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
        for i=1:size(M,1)
        for j=1:size(M,2)
            value=M(i,j);
            cnts(value+1)=cnts(value+1)+1;
            probf(value+1)=cnts(value+1)/numofpixels;
        end
    end
    sum=0;
    n=255;
size(probf)
        sum=sum+cnts(i);
        cum(i)=sum;
        prbc(i)=cum(i)/numofpixels;
        fin_lc(i)=round(prbc(i)*n);
    end
        for i=1:size(M,1)
          for j=1:size(M,2)
              Y(i,j)=fin_lc(M(i,j)+1);
          end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
     imwrite(Y, strcat(pathp, num2str(samples), '.jpg'));
end
 %preprocessing of training nonlps
for samples = 1 : numn
    file = fullfile(path2, files2(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
    M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
    numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
```

```
    fin_lc=zeros(256,1);
        for i=1:size(M,1)
        for j=1:size(M,2)
            value=M(i,j);
            cnts(value+1)=cnts(value+1)+1;
            probf(value+1)=cnts(value+1)/numofpixels;
        end
    end
    sum=0;
    n=255;
     for i=1:size(probf)
       sum=sum+cnts(i);
       cum(i)=sum;
       prbc(i)=cum(i)/numofpixels;
       fin_lc(i)=round(prbc(i)*n);
    end
    for i=1:size(M,1)
           for j=1:size(M,2)
               Y(i,j)=fin_lc(M(i,j)+1);
           end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
    imwrite(Y, strcat(pathn, num2str(samples), '.jpg'));
end
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global Y1
tic
 if ~isdir('result')
    mkdir('result');
end
if ~isdir('detected')
    mkdir('detected');
end
 %% testing
files = dir(fullfile('test','*.jpg'));
for id = 1 : numel(files)
    file = fullfile('test/', files(id).name);
        %preprocessing
    X=imread(file);
     M = imgaussfilt(X,0.25);
    M = imresize(M,[240, 320]);
    M=rgb2gray(M);
    numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
```

```
 for i=1:size(M,1)
   for j=1:size(M,2)
      value=M(i,j);
      cnts(value+1)=cnts(value+1)+1;
      probf(value+1)=cnts(value+1)/numofpixels;
   end
end
 sum=0;
n=255;

for i=1:size(probf)
  sum=sum+cnts(i);
  cum(i)=sum;
  prbc(i)=cum(i)/numofpixels;
  fin_lc(i)=round(prbc(i)*n);
end
 for i=1:size(M,1)
  for j=1:size(M,2)
     Y(i,j)=fin_lc(M(i,j)+1);
  end
end
Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
 lbpRadiusSet = [2 4 6 8];
lbpPointsSet = [8 8 8 8];
 % selecting mrelbp features from preprocessed test image
xsum=0;
ysum=0;
num = 0;
px = [];
py = [];
px1 = [];
py1 = [];
px2 = [];
py2 = [];
for i = 100 : 5 : size(Y,1) - 70
  for j = 50 : 20 : size(Y,2) - 160
      Y1 = Y(i:i+24,j:j+99);
      testfeatures1 = [];
      for idxLbpRadius = 1 : length(lbpRadiusSet)
         lbpRadius = lbpRadiusSet(idxLbpRadius);
         lbpPoints = lbpPointsSet(idxLbpRadius);
          mapping = get_mapping(lbpPoints);
         blockSize = lbpRadius*2+1;
          if idxLbpRadius > 1
            lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
         else
            lbpRadiusPre = 0;
         end
          cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,1,1);
          testfeatures1 = [testfeatures1 cfmsWithLabels_LBP];
```

```
            end
            feature = testfeatures1 / max(testfeatures1);
            dlmwrite('testdata',[1 feature], ' ');
            [TestingTime, TestingAccuracy, output]= elm_predict('testdata');

            if output == 1
                num = num + 1;
                px = [px i * 2];
                py = [py j * 2];
%                 result = X(i*2:i*2+44,j*2:j*2+199);
%                 result = imresize(result, [45,200]);
                result = imcrop(X,[i*2 j*2 200 50]);
                %imwrite(result, strcat('detected/', num2str(id), '_cropped_', num2str(num), '.jpg'));
            end
        end
    end

    fh = figure;
    imshow( X, 'border', 'tight' ); %//show your image
    hold on;

    for i = 1 : length(px)
        rectangle('Position', [py(i) px(i) 200 50], 'EdgeColor', 'g'); %// draw rectangle on image
        frm = getframe( fh ); %// get the image+rectangle
        imwrite( frm.cdata, strcat('result/',num2str(id),'.jpg') ); %// save to file
    end
    pause(3);
    close(fh);
    disp(strcat('test image ', num2str(id), ' done' ));
end
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addpath('mrelbp');
addpath('elmbase');
 global nump  %number of positive samples
global numn  %number of negative samples
 global lbpRadius
global lbpPoints
%% selecting mlelbp features from preprocessed train images
tic
lbpRadiusSet = [1 2.5 4];
lbpPointsSet = [8 12 16];
 trainfeatures1 = [];
trainfeatures2 = [];

for idxLbpRadius = 1 : length(lbpRadiusSet)
    lbpRadius = lbpRadiusSet(idxLbpRadius);
    lbpPoints = lbpPointsSet(idxLbpRadius);

    mapping = get_mapping(lbpPoints);
```

```
   blockSize = lbpRadius*2+1;

   if idxLbpRadius > 1
      lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
   else
      lbpRadiusPre = 0;
   end
   cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,1);
   trainfeatures1 = [trainfeatures1 cfmsWithLabels_LBP];

   clear cfmsWithLabels_LBP
   cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,2);
   trainfeatures2 = [trainfeatures2 cfmsWithLabels_LBP];
   clear cfmsWithLabels_LBP
end
 trainfeatures1 = [trainfeatures1; trainfeatures2];
shape = size(trainfeatures1);
feature = [];
for i = 1 : shape(1)
   feature = [feature; trainfeatures1(i,:)/max(trainfeatures1(i,:))];
end
classes = zeros(nump + numn, 1);
for i = 1 : nump
   classes(i) = 1;
end
 M = [classes feature];
dlmwrite('traindata',M, ' ');
 elm_train('traindata', 1, 550, 'sig');
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C

A Matlab simulation code for Chapter 5
Developing learning-based preprocessing methods for detecting complicated licence plates

The simulation codes to detect LPs from complicated vehicles images are presented. The experiment results were obtained using Matlab programming language version R2018a.

```
%Image descriptor based on Histogram of Orientated Gradients and local binary pattern
 function H=HOG(Im)
nwin_x=3;%set here the number of HOG windows per bound box
nwin_y=3;
B=9;%set here the number of histogram bins
[L,C]=size(Im); % L num of lines ; C num of columns
H=zeros(nwin_x*nwin_y*B,1); % column vector with zeros
m=sqrt(L/2);
if C==1 % if num of columns==1
   Im=im_recover(Im,m,2*m);%verify the size of image, e.g. 25x50
   L=2*m;
   C=m;
end
Im=double(Im);
step_x=floor(C/(nwin_x+1));
step_y=floor(L/(nwin_y+1));
cont=0;
hx = [-1,0,1];
hy = -hx';
grad_xr = imfilter(double(Im),hx);
grad_yu = imfilter(double(Im),hy);
angles=atan2(grad_yu,grad_xr);
magnit=((grad_yu.^2)+(grad_xr.^2)).^.5;
for n=0:nwin_y-1
   for m=0:nwin_x-1
      cont=cont+1;
      angles2=angles(n*step_y+1:(n+2)*step_y,m*step_x+1:(m+2)*step_x);
      magnit2=magnit(n*step_y+1:(n+2)*step_y,m*step_x+1:(m+2)*step_x);
      v_angles=angles2(:);
      v_magnit=magnit2(:);
      K=max(size(v_angles));
      %assembling the histogram with 9 bins (range of 20 degrees per bin)
      bin=0;
      H2=zeros(B,1);
      for ang_lim=-pi+2*pi/B:2*pi/B:pi
         bin=bin+1;
         for k=1:K
            if v_angles(k)<ang_lim
               v_angles(k)=100;
               H2(bin)=H2(bin)+v_magnit(k);
            end
         end
      end

      H2=H2/(norm(H2)+0.01);
      H((cont-1)*B+1:cont*B,1)=H2;
   end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [feature] = hog_feature_vector(im)

% Convert RGB iamge to grayscale
if size(im,3)==3
   im=rgb2gray(im);
end
im=double(im);
 rows=size(im,1);
cols=size(im,2);
Ix=im; %Basic Matrix assignment
Iy=im; %Basic Matrix assignment
 % Gradients in X and Y direction. Iy is the gradient in X direction and Iy
% is the gradient in Y direction
for i=1:rows-2
   Iy(i,:)=(im(i,:)-im(i+2,:));
end
for i=1:cols-2
   Ix(:,i)=(im(:,i)-im(:,i+2));
end
gauss=fspecial('gaussian',8); %% Initialized a gaussian filter with sigma=0.5 * block width.
 angle=atand(Ix./Iy); % Matrix containing the angles of each edge gradient
angle=imadd(angle,180); %Angles in range (0,180)
magnitude=sqrt(Ix.^2 + Iy.^2);
angle(isnan(angle))=0;
magnitude(isnan(magnitude))=0;
feature=[]; %initialized the feature vector
 % Iterations for Blocks
for i = 0: rows/8 - 2
   for j= 0: cols/8 -2
      mag_patch = magnitude(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);
      %mag_patch = imfilter(mag_patch,gauss);
      ang_patch = angle(8*i+1 : 8*i+16 , 8*j+1 : 8*j+16);
      block_feature=[];
      %Iterations for cells in a block
      for x= 0:1
         for y= 0:1
            angleA =ang_patch(8*x+1:8*x+8, 8*y+1:8*y+8);
            magA   =mag_patch(8*x+1:8*x+8, 8*y+1:8*y+8);
            histr  =zeros(1,9);
            %Iterations for pixels in one cell
            for p=1:8
               for q=1:8
                  alpha= angleA(p,q);
                                    % Binning Process (Bi-Linear Interpolation)
                  if alpha>10 && alpha<=30
                     histr(1)=histr(1)+ magA(p,q)*(30-alpha)/20;
                     histr(2)=histr(2)+ magA(p,q)*(alpha-10)/20;
                  elseif alpha>30 && alpha<=50
                     histr(2)=histr(2)+ magA(p,q)*(50-alpha)/20;
```

```
                    histr(3)=histr(3)+ magA(p,q)*(alpha-30)/20;
                elseif alpha>50 && alpha<=70
                    histr(3)=histr(3)+ magA(p,q)*(70-alpha)/20;
                    histr(4)=histr(4)+ magA(p,q)*(alpha-50)/20;
                elseif alpha>70 && alpha<=90
                    histr(4)=histr(4)+ magA(p,q)*(90-alpha)/20;
                    histr(5)=histr(5)+ magA(p,q)*(alpha-70)/20;
                elseif alpha>90 && alpha<=110
                    histr(5)=histr(5)+ magA(p,q)*(110-alpha)/20;
                    histr(6)=histr(6)+ magA(p,q)*(alpha-90)/20;
                elseif alpha>110 && alpha<=130
                    histr(6)=histr(6)+ magA(p,q)*(130-alpha)/20;
                    histr(7)=histr(7)+ magA(p,q)*(alpha-110)/20;
                elseif alpha>130 && alpha<=150
                    histr(7)=histr(7)+ magA(p,q)*(150-alpha)/20;
                    histr(8)=histr(8)+ magA(p,q)*(alpha-130)/20;
                elseif alpha>150 && alpha<=170
                    histr(8)=histr(8)+ magA(p,q)*(170-alpha)/20;
                    histr(9)=histr(9)+ magA(p,q)*(alpha-150)/20;
                elseif alpha>=0 && alpha<=10
                    histr(1)=histr(1)+ magA(p,q)*(alpha+10)/20;
                    histr(9)=histr(9)+ magA(p,q)*(10-alpha)/20;
                elseif alpha>170 && alpha<=180
                    histr(9)=histr(9)+ magA(p,q)*(190-alpha)/20;
                    histr(1)=histr(1)+ magA(p,q)*(alpha-170)/20;
                end
            end
        end
        block_feature=[block_feature histr]; % Concatenation of Four histograms to form
one block feature
        end
    end
    % Normalize the values in the block using L1-Norm
    block_feature=block_feature/sqrt(norm(block_feature)^2+.01);
    feature=[feature block_feature]; %Features concatenation
  end
end
 feature(isnan(feature))=0; %Removing Infinitiy values
 % Normalization of the feature vector using L2-Norm
feature=feature/sqrt(norm(feature)^2+.001);
for z=1:length(feature)
  if feature(z)>0.2
     feature(z)=0.2;
  end
end
feature=feature/sqrt(norm(feature)^2+.001);
 % toc;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic
global nump  %number of positive samples
```

```
global numn  %number of negative samples
global pathp
global pathn
global flags
%% preprocessing of train images
path1 = 'train/lp';   %training lps
path2 = 'train/nonlp';  %trainging nonlps
pathp = 'realtrain/lp/';   %rectangle training lps
pathn = 'realtrain/nonlp/';  %trainging nonlps
%preprocessed training lps will be saved in 'realtrain' directory
if ~isdir('realtrain')
   mkdir('realtrain');
   mkdir('realtrain/lp');
   mkdir('realtrain/nonlp')
end
%image files in directories
files1 = dir(fullfile(path1,'*.jpg'));
files2 = dir(fullfile(path2,'*.jpg'));
%number of positive and negative trainging samples
nump = numel(files1); %number of training lps = positive
numn = numel(files2);  %number of nonlps = negative
flags = zeros(nump,1);
%preprocessing of training lps
for samples = 1 : nump
   file = fullfile(path1, files1(samples).name);
   X = imread(file);
%     X = imresize(X, 0.5);
   M = imgaussfilt(X,0.25);
%     M = imresize(M, [60 250]);
   M = rgb2gray(M);
   [m n] = size(M);
      if m < 37.5
      p = 25;
      q = 100;
      flags(samples) = 1;
   elseif m >= 37.5 && m < 62.5
      p = 50;
      q = 200;
      flags(samples) = 2;
   elseif m >= 62.5
      p = 75;
      q = 300;
      flags(samples) = 3;
   end
     %M = imresize(M, [p, q]);
   if p >= m && q >= n
      M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
      M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
   elseif p < m && q >= n
      m = p;
```

```
        M_pad = imresize(M, [m, n]);
        M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p >= m && q < n
        n = q;
        M_pad = imresize(M, [m, n]);
        M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
        M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
    elseif p < m && q < n
        M_pad = imresize(M, [p, q]);
    end
     M = M_pad;
        numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
        for i=1:size(M,1)
        for j=1:size(M,2)
            value=M(i,j);
            cnts(value+1)=cnts(value+1)+1;
            probf(value+1)=cnts(value+1)/numofpixels;
        end
    end
        sum=0;
    n=255;
      for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
        for i=1:size(M,1)
          for j=1:size(M,2)
              Y(i,j)=fin_lc(M(i,j)+1);
          end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
    imwrite(Y, strcat(pathp, num2str(samples), '.jpg'));
end
 %preprocessing of training nonlps
for samples = 1 : numn
    file = fullfile(path2, files2(samples).name);
    X = imread(file);
    M = imgaussfilt(X,0.25);
    %M = imresize(M, [60 250]);
    M = rgb2gray(M);
      numofpixels=size(M,1)*size(M,2);
```

```
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
      for i=1:size(M,1)
       for j=1:size(M,2)
          value=M(i,j);
          cnts(value+1)=cnts(value+1)+1;
          probf(value+1)=cnts(value+1)/numofpixels;
       end
    end
     sum=0;
    n=255;
      for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
            Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
    Y = adapthisteq(Y,'clipLimit',0.01,'Distribution','rayleigh');
       imwrite(Y, strcat(pathn, num2str(samples), '.jpg'));
end
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if ~isdir('result')
    mkdir('result');
end
if ~isdir('detected')
    mkdir('detected');
end
 % load model
model1 = loadCompactModel('model1');
model2 = loadCompactModel('model2');
model3 = loadCompactModel('model3');
%% testing
files = dir(fullfile('test','*.png'));
for id = 1 : numel(files)
    file = fullfile('test/', files(id).name);
%preprocessing
      X=imread(file);
    M = imgaussfilt(X,0.25);
    M=rgb2gray(M);
```

```
 numofpixels=size(M,1)*size(M,2);
Y=uint8(zeros(size(M,1),size(M,2)));
cnts=zeros(256,1);
probf=zeros(256,1);
prbc=zeros(256,1);
cum=zeros(256,1);
fin_lc=zeros(256,1);
 for i=1:size(M,1)
    for j=1:size(M,2)
        value=M(i,j);
        cnts(value+1)=cnts(value+1)+1;
        probf(value+1)=cnts(value+1)/numofpixels;
    end
end
 sum=0;
n=255;
 for i=1:size(probf)
   sum=sum+cnts(i);
   cum(i)=sum;
   prbc(i)=cum(i)/numofpixels;
   fin_lc(i)=round(prbc(i)*n);
end
 for i=1:size(M,1)
   for j=1:size(M,2)
      Y(i,j)=fin_lc(M(i,j)+1);
   end
end
Y = adapthisteq(Y,'clipLimit',0.05,'Distribution','rayleigh');
 % selecting hog features from preprocessed test image
num = 0;
px1 = [];
py1 = [];
px2 = [];
py2 = [];
px3 = [];
py3 = [];
    for i = 200 : 10 : size(Y,1) - 100
   for j = 50 : 10 : size(Y,2) - 350
      tic
        Y2 = Y(i:i+ 74,j:j+299);
         Y1 = Y(i:i+ 49,j:j+199);
          Y3 = Y(i:i+24,j:j+99);
        label1 = predict(model2,[hog_feature_vector(Y1) extractLBPFeatures(Y1)]);
        label2 = predict(model3,[hog_feature_vector(Y2) extractLBPFeatures(Y2)]);
         label3 = predict(model1,[hog_feature_vector(Y3) extractLBPFeatures(Y3)]);
        %label2 = predict(model1,hog_feature_vector(Y1));
         testfeatures1 = [hog_feature_vector(Y1) extractLBPFeatures(Y1)];
         testfeatures2 = [hog_feature_vector(Y2) extractLBPFeatures(Y2)];
         testfeatures23 = [hog_feature_vector(Y3) extractLBPFeatures(Y3)];
         a = toc;
```

```
          w = w + a;
                if label2 == 1 || label1==1 || label3==1
                px2 = [px2 i];
                py2 = [py2 j];
                continue;
            end
        end
    end
     fh = figure;
    imshow( X, 'border', 'tight' ); %//show your image
    hold on;
    px21 = unique(px2);
    py21 = [];
    for i = 1 : length(px21)
        temp = [];
        for j = 1 : length(px2)
            if px21(i) == px2(j)
                temp = [temp; py2(j)];
            end
        end
        py21 = [py21; round(mean(temp))];
    end

    for i = 1 : length(px21)
        if i > 1
            if abs(px21(i)-px21(i-1)) > 30
                rectangle('Position', [py21(i)-10 px21(i) 250 50], 'EdgeColor', 'g'); %// draw
rectangle on image
                frm = getframe( fh ); %// get the image+rectangle
                imwrite( frm.cdata, strcat('result/',num2str(id),'.jpg') ); %// save to file
            end
        else
            rectangle('Position', [py21(i)-10 px21(i) 250 50], 'EdgeColor', 'g'); %// draw rectangle
on image
            frm = getframe( fh ); %// get the image+rectangle
            imwrite( frm.cdata, strcat('result/',num2str(id),'.jpg') ); %// save to file
        end
    end
     pause(3);
    close(fh);
    disp(strcat('test image ', num2str(id), ' done' ));
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%tic

global nump  %number of positive samples
global numn  %number of negative samples
 global pathp
global pathn
 global flags
 clc
 %image files in directories
```

```
files1 = dir(fullfile(pathp,'*.jpg'));
files2 = dir(fullfile(pathn,'*.jpg'));
 trainfeatures1 = [];
trainfeatures2 = [];
trainfeatures3 = [];
trainfeatures4_1 = [];
trainfeatures4_2 = [];
trainfeatures4_3 = [];
 n1 = 0;
n2 = 0;
n3 = 0;
for i = 1 : nump
   %file = fullfile(pathp, files1(i).name);
   file = fullfile(pathp, strcat(num2str(i), '.jpg'));
   X = imread(file);
   if flags(i) == 1
      trainfeatures1 =[trainfeatures1; hog_feature_vector(X) extractLBPFeatures(X)];
      n1 = n1 + 1;
   elseif flags(i) == 2
      trainfeatures2 =[trainfeatures2; hog_feature_vector(X) extractLBPFeatures(X)];
      n2 = n2 + 1;
   elseif flags(i) == 3
      trainfeatures3 =[trainfeatures3; hog_feature_vector(X) extractLBPFeatures(X)];
      n3 = n3 + 1;
   end
end
for i = 1 : numn
   file = fullfile(pathn, files2(i).name);
   X = imread(file);
   X = imresize(X, [25, 100]);
   trainfeatures4_1 = [trainfeatures4_1; hog_feature_vector(X) extractLBPFeatures(X)];
end
for i = 1 : numn
   file = fullfile(pathn, files2(i).name);
   X = imread(file);
   X = imresize(X, [50, 200]);
   trainfeatures4_2 = [trainfeatures4_2; hog_feature_vector(X) extractLBPFeatures(X)];
end
for i = 1 : numn
   file = fullfile(pathn, files2(i).name);
   X = imread(file);
   X = imresize(X, [75, 300]);
   trainfeatures4_3 = [trainfeatures4_3; hog_feature_vector(X) extractLBPFeatures(X)];
end

trainfeatures1 = [trainfeatures1; trainfeatures4_1];
trainfeatures2 = [trainfeatures2; trainfeatures4_2];
trainfeatures3 = [trainfeatures3; trainfeatures4_3];
classes1 = zeros(n1 + numn, 1);
for i = 1 : n1
```

```
    classes1(i) = 1;
end
classes2 = zeros(n2 + numn, 1);
for i = 1 : n2
    classes2(i) = 1;
end
 classes3 = zeros(n3 + numn, 1);
for i = 1 : n3
    classes3(i) = 1;
end
Mdl1 = fitcsvm(trainfeatures1,classes1,'Crossval','on','KFold',5);
saveCompactModel(Mdl1.Trained(Y. Han & 10.1109/EIT.2015.7293386),'model1');
Mdl2 = fitcsvm(trainfeatures2,classes2,'Crossval','on','KFold',5);
saveCompactModel(Mdl2.Trained(Y. Han & 10.1109/EIT.2015.7293386),'model2');
Mdl3 = fitcsvm(trainfeatures3,classes3,'Crossval','on','KFold',5);
saveCompactModel(Mdl3.Trained(Y. Han & 10.1109/EIT.2015.7293386),'model3');
toc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D

Matlab simulation code for Chapter 6
Distorted vehicle licence plates detection using hybrid feature descriptors

The simulation codes for detecting LPs from distorted vehicles images are presented. The experiment results were obtained using Matlab programming language version R2018a

---------- Convert training and testing vehicles images from ".jpg" to ".png"---------

```
global nump
folder = 'train\lp or test\';
vet_files=dir(fullfile(folder, '*.jpg'));
nump = numel(vet_files);
for i=1:nump
   inputFullFileName = fullfile(folder, vet_files(i).name);
   outputFullFileName = strrep(inputFullFileName, '.jpg', '.png');
   thisImage = imread(inputFullFileName);
   imwrite(thisImage, outputFullFileName);
end
```

----------Exterme learning machine classifier----------

```
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
elm(TrainingData_File, TestingData_File, Elm_Type, NumberofHiddenNeurons,
ActivationFunction)
 %%%%%%%%%%% Macro definition
REGRESSION=0;
CLASSIFIER=1;
 %%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                   %   Release raw training data array
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                    %   Release raw testing data array
 NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
NumberofInputNeurons=size(P,1);
 if Elm_Type~=REGRESSION
   %%%%%%%%%%%% Pre-processing the data of classification
   sorted_target=sort(cat(2,T,TV.T),2);
   label=zeros(1,1);       %Find and save in 'label' class label from training and testing data
sets
   label(1,1)=sorted_target(1,1);
   j=1;
   for i = 2:(NumberofTrainingData+NumberofTestingData)
      if sorted_target(1,i) ~= label(1,j)
         j=j+1;
         label(1,j) = sorted_target(1,i);
      end
   end
   number_class=j;
   NumberofOutputNeurons=number_class;
   %%%%%%%%%% Processing the targets of training
   temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
```

```
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
     %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;

end                 %   end if of Elm_Type
%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                                    %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = double(hardlim(tempH));
    case {'tribas'}
        %%%%%%%% Triangular basis function
        H = tribas(tempH);
    case {'radbas'}
        %%%%%%%% Radial basis function
```

```
        H = radbas(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                    %   Release the temparary array for calculation of hidden neuron
output matrix H
%%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';                        % implementation without regularization
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train       %   Calculate CPU time (seconds) spent
for training ELM
%%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                            %   Y: the actual output of the training data
if Elm_Type == REGRESSION
   TrainingAccuracy=sqrt(mse(T - Y))               %   Calculate training accuracy (RMSE) for
regression case
end
clear H;
%%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);             %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
   case {'sig','sigmoid'}
      %%%%%%%% Sigmoid
      H_test = 1 ./ (1 + exp(-tempH_test));
   case {'sin','sine'}
      %%%%%%%% Sine
      H_test = sin(tempH_test);
   case {'hardlim'}
      %%%%%%%% Hard Limit
      H_test = hardlim(tempH_test);
   case {'tribas'}
      %%%%%%%% Triangular basis function
      H_test = tribas(tempH_test);
   case {'radbas'}
      %%%%%%%% Radial basis function
      H_test = radbas(tempH_test);
      %%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';                      %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test          %   Calculate CPU time (seconds) spent by
ELM predicting the whole testing data

if Elm_Type == REGRESSION
```

```
   TestingAccuracy=sqrt(mse(TV.T - TY))          %   Calculate testing accuracy (RMSE) for
regression case
end
 if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
   MissClassificationRate_Training=0;
   MissClassificationRate_Testing=0;
    for i = 1 : size(T, 2)
       [x, label_index_expected]=max(T(:,i));
       [x, label_index_actual]=max(Y(:,i));
       if label_index_actual~=label_index_expected
          MissClassificationRate_Training=MissClassificationRate_Training+1;
       end
    end
   TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
   for i = 1 : size(TV.T, 2)
       [x, label_index_expected]=max(TV.T(:,i));
       [x, label_index_actual]=max(TY(:,i));
       if label_index_actual~=label_index_expected
          MissClassificationRate_Testing=MissClassificationRate_Testing+1;
       end
    end
   TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end
---------------------------------------------------------------------------------------
----------------------Kernal_ELM classifier-----------------------------------------
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy,TY] =
elm_kernel(TrainingData_File, TestingData_File, Elm_Type, Regularization_coefficient,
Kernel_type, Kernel_para)
REGRESSION=0;
CLASSIFIER=1;
%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                             %   Release raw training data array
 %%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                              %   Release raw testing data array
 C = Regularization_coefficient;
NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
 if Elm_Type~=REGRESSION
   %%%%%%%%%%%% Preprocessing the data of classification
   sorted_target=sort(cat(2,T,TV.T),2);
   label=zeros(1,1);               %   Find and save in 'label' class label from training and testing
data sets
   label(1,1)=sorted_target(1,1);
```

```
    j=1;
    for i = 2:(NumberofTrainingData+NumberofTestingData)
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;
    %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
     %%%%%%%%%% Processing the targets of testing
    temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
    for i = 1:NumberofTestingData
        for j = 1:number_class
            if label(1,j) == TV.T(1,i)
                break;
            end
        end
        temp_TV_T(j,i)=1;
    end
    TV.T=temp_TV_T*2-1;
                                        %   end if of Elm_Type
end
%%%%%%%%%%% Training Phase
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tic;
n = size(T,2);
Omega_train = kernel_matrix(P',Kernel_type, Kernel_para);
OutputWeight=((Omega_train+speye(n)/C)\(T'));
TrainingTime=toc
 %%%%%%%%%%% Calculate the training output
Y=(Omega_train * OutputWeight)';                    %   Y: the actual output of the training
data
%%%%%%%%%%% Calculate the output of testing input
tic;
Omega_test = kernel_matrix(P',Kernel_type, Kernel_para,TV.P');
TY=(Omega_test' * OutputWeight)';                    %   TY: the actual output of the testing
data
TestingTime=toc
 %%%%%%%%%% Calculate training & testing classification accuracy
```

```
 if Elm_Type == REGRESSION
%%%%%%%%%% Calculate training & testing accuracy (RMSE) for regression case
   TrainingAccuracy=sqrt(mse(T - Y))
   TestingAccuracy=sqrt(mse(TV.T - TY))
end
if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
   MissClassificationRate_Training=0;
   MissClassificationRate_Testing=0;
    for i = 1 : size(T, 2)
       [x, label_index_expected]=max(T(:,i));
       [x, label_index_actual]=max(Y(:,i));
       if label_index_actual~=label_index_expected
          MissClassificationRate_Training=MissClassificationRate_Training+1;
       end
   end
   TrainingAccuracy=1-MissClassificationRate_Training/size(T,2)
   for i = 1 : size(TV.T, 2)
       [x, label_index_expected]=max(TV.T(:,i));
       [x, label_index_actual]=max(TY(:,i));
       if label_index_actual~=label_index_expected
          MissClassificationRate_Testing=MissClassificationRate_Testing+1;
       end
   end
   TestingAccuracy=1-MissClassificationRate_Testing/size(TV.T,2)
end
  %%%%%%%%%%%%%%%%%%% Kernel Matrix
   function omega = kernel_matrix(Xtrain,kernel_type, kernel_pars,Xt)
 nb_data = size(Xtrain,1);
 if strcmp(kernel_type,'RBF_kernel'),
   if nargin<4,
       XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
       omega = XXh+XXh'-2*(Xtrain*Xtrain');
       omega = exp(-omega./kernel_pars(1));
   else
       XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
       XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
       omega = XXh1+XXh2' - 2*Xtrain*Xt';
       omega = exp(-omega./kernel_pars(1));
   end
   elseif strcmp(kernel_type,'lin_kernel')
   if nargin<4,
       omega = Xtrain*Xtrain';
   else
       omega = Xtrain*Xt';
   end
   elseif strcmp(kernel_type,'poly_kernel')
   if nargin<4,
       omega = (Xtrain*Xtrain'+kernel_pars(1)).^kernel_pars(2);
   else
```

```
        omega = (Xtrain*Xt'+kernel_pars(1)).^kernel_pars(2);
    end
    elseif strcmp(kernel_type,'wav_kernel')
    if nargin<4,
        XXh = sum(Xtrain.^2,2)*ones(1,nb_data);
        omega = XXh+XXh'-2*(Xtrain*Xtrain');
        XXh1 = sum(Xtrain,2)*ones(1,nb_data);
omega1 = XXh1-XXh1';
omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1));
          else
XXh1 = sum(Xtrain.^2,2)*ones(1,size(Xt,1));
XXh2 = sum(Xt.^2,2)*ones(1,nb_data);
omega = XXh1+XXh2' - 2*(Xtrain*Xt');
XXh11 = sum(Xtrain,2)*ones(1,size(Xt,1));
XXh22 = sum(Xt,2)*ones(1,nb_data);
omega1 = XXh11-XXh22';
omega = cos(kernel_pars(3)*omega1./kernel_pars(2)).*exp(-omega./kernel_pars(1));
    end
end
-------------------------------------------------------------------------------------------
----------------------------------ELM_Multi-output-Regression----------------------
function [TrainingTime, TestingTime, TrainingAccuracy, TestingAccuracy] =
elm_MultiOutputRegression(TrainingData_File, TestingData_File, No_of_Output,
NumberofHiddenNeurons, ActivationFunction)
 %%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1:No_of_Output)';
P=train_data(:,No_of_Output+1:size(train_data,2))';
clear train_data;                                   %   Release raw training data array
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1:No_of_Output)';
TV.P=test_data(:,No_of_Output+1:size(test_data,2))';
clear test_data;                                    %   Release raw testing data array
 NumberofTrainingData=size(P,2);
NumberofTestingData=size(TV.P,2);
NumberofInputNeurons=size(P,1);
 %%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
 %%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                                    %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
 %%%%%%%%%%% Calculate hidden neuron output matrix H
```

```
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = hardlim(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                                    %   Release the temparary array for calculation of
hidden neuron output matrix H
 %%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train        %   Calculate CPU time (seconds) spent
for training ELM
 %%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                             %   Y: the actual output of the training data
TrainingAccuracy=sqrt(mse(T - Y))               %   Calculate training accuracy (RMSE) for
regression case
clear H;
%%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H_test = 1 ./ (1 + exp(-tempH_test));
    case {'sin','sine'}
        %%%%%%%% Sine
        H_test = sin(tempH_test);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H_test = hardlim(tempH_test);
        %%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';                       %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test           %   Calculate CPU time (seconds) spent by
ELM predicting the whole testing data
TestingAccuracy=sqrt(mse(TV.T - TY))            %   Calculate testing accuracy (RMSE) for
regression case
```

```
---------------------------------------------------------------------------------------------
-----------------------------ELM_Predict procedure----------------------------------
function [TestingTime, TestingAccuracy, output] = elm_predict(TestingData_File)
 REGRESSION=0;
CLASSIFIER=1;
 %%%%%%%%%%% Load testing dataset
test_data=load(TestingData_File);
TV.T=test_data(:,1)';
TV.P=test_data(:,2:size(test_data,2))';
clear test_data;                                    %   Release raw testing data array
 NumberofTestingData=size(TV.P,2);
 load elm_model.mat;
 if Elm_Type~=REGRESSION
    %%%%%%%%%%% Processing the targets of testing
   temp_TV_T=zeros(NumberofOutputNeurons, NumberofTestingData);
   for i = 1:NumberofTestingData
      for j = 1:size(label,2)
         if label(1,j) == TV.T(1,i)
            break;
         end
      end
      temp_TV_T(j,i)=1;
   end
   TV.T=temp_TV_T*2-1;

end                                                 %   end if of Elm_Type
%%%%%%%%%%% Calculate the output of testing input
start_time_test=cputime;
tempH_test=InputWeight*TV.P;
clear TV.P;             %   Release input of testing data
ind=ones(1,NumberofTestingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);              %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH_test=tempH_test + BiasMatrix;
switch lower(ActivationFunction)
   case {'sig','sigmoid'}
      %%%%%%%%% Sigmoid
      H_test = 1 ./ (1 + exp(-tempH_test));
   case {'sin','sine'}
      %%%%%%%%% Sine
      H_test = sin(tempH_test);
   case {'hardlim'}
      %%%%%%%%% Hard Limit
      H_test = hardlim(tempH_test);
      %%%%%%%%% More activation functions can be added here
end
TY=(H_test' * OutputWeight)';       %   TY: the actual output of the testing data
end_time_test=cputime;
TestingTime=end_time_test-start_time_test;          %   Calculate CPU time (seconds) spent
by ELM predicting the whole testing data
```

```
 if Elm_Type == REGRESSION
    TestingAccuracy=sqrt(mse(TV.T - TY));           %   Calculate testing accuracy (RMSE) for
regression case
    output=TY;
end
 if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
    MissClassificationRate_Testing=0;
     for i = 1 : size(TV.T, 2)
        [x, label_index_expected]=max(TV.T(:,i));
        [x, label_index_actual]=max(TY(:,i));
        output(i)=label(label_index_actual);
        if label_index_actual~=label_index_expected
            MissClassificationRate_Testing=MissClassificationRate_Testing+1;
        end
    end
    TestingAccuracy=1-MissClassificationRate_Testing/NumberofTestingData;
end

save('elm_output','output');
------------------------------------------------------------------------------------------------------
------------ ELM_Train procedure----------------------------------------------------

function [TrainingTime,TrainingAccuracy] = elm_train(TrainingData_File, Elm_Type,
NumberofHiddenNeurons, ActivationFunction)
 REGRESSION=0;
CLASSIFIER=1;

%%%%%%%%%% Load training dataset
train_data=load(TrainingData_File);
T=train_data(:,1)';
P=train_data(:,2:size(train_data,2))';
clear train_data;                                   %   Release raw training data array
 NumberofTrainingData=size(P,2);
NumberofInputNeurons=size(P,1);
 if Elm_Type~=REGRESSION
    %%%%%%%%%% Preprocessing the data of classification
    sorted_target=sort(T,2);
    label=zeros(1,1);                               %   Find and save in 'label' class label from training
and testing data sets
    label(1,1)=sorted_target(1,1);
    j=1;
    for i = 2:NumberofTrainingData
        if sorted_target(1,i) ~= label(1,j)
            j=j+1;
            label(1,j) = sorted_target(1,i);
        end
    end
    number_class=j;
    NumberofOutputNeurons=number_class;
```

```
    %%%%%%%%%% Processing the targets of training
    temp_T=zeros(NumberofOutputNeurons, NumberofTrainingData);
    for i = 1:NumberofTrainingData
        for j = 1:number_class
            if label(1,j) == T(1,i)
                break;
            end
        end
        temp_T(j,i)=1;
    end
    T=temp_T*2-1;
end                                 %   end if of Elm_Type
%%%%%%%%%% Calculate weights & biases
start_time_train=cputime;
%%%%%%%%%% Random generate input weights InputWeight (w_i) and biases
BiasofHiddenNeurons (b_i) of hidden neurons
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;                            %   Release input of training data
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);          %   Extend the bias matrix
BiasofHiddenNeurons to match the demention of H
tempH=tempH+BiasMatrix;
%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        %%%%%%%% Sigmoid
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        %%%%%%%% Sine
        H = sin(tempH);
    case {'hardlim'}
        %%%%%%%% Hard Limit
        H = hardlim(tempH);
        %%%%%%%% More activation functions can be added here
end
clear tempH;                        %   Release the temparary array for calculation of
hidden neuron output matrix H

%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';
end_time_train=cputime;
TrainingTime=end_time_train-start_time_train       %   Calculate CPU time (seconds) spent
for training ELM

%%%%%%%%%% Calculate the training accuracy
Y=(H' * OutputWeight)';                     %   Y: the actual output of the training data
if Elm_Type == REGRESSION
```

```
   TrainingAccuracy=sqrt(mse(T - Y))             %   Calculate training accuracy (RMSE) for
regression case
   output=Y;
end
clear H;
 if Elm_Type == CLASSIFIER
%%%%%%%%%% Calculate training & testing classification accuracy
   MissClassificationRate_Training=0;
    for i = 1 : size(T, 2)
       [x, label_index_expected]=max(T(:,i));
       [x, label_index_actual]=max(Y(:,i));
       output(i)=label(label_index_actual);
       if label_index_actual~=label_index_expected
          MissClassificationRate_Training=MissClassificationRate_Training+1;
       end
   end
   TrainingAccuracy=1-MissClassificationRate_Training/NumberofTrainingData
end
if Elm_Type~=REGRESSION
   save('elm_model', 'NumberofInputNeurons', 'NumberofOutputNeurons', 'InputWeight',
'BiasofHiddenNeurons', 'OutputWeight', 'ActivationFunction', 'label', 'Elm_Type');
else
   save('elm_model', 'InputWeight', 'BiasofHiddenNeurons', 'OutputWeight',
'ActivationFunction', 'Elm_Type');
end
-------------------------------------------------------------------------------------------------------------
----------Codes for MRELBP descriptor-----------------------------------
function blocks = cirInterpSingleRadius(img)
 global lbpPoints;
global lbpRadius;
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
imgNewW = imgW - 2*lbpRadius;
 % the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
 radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
 % Determine the dimensions of the input img.
[ysize,xsize] = size(img);
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
   spoints(i,1) = -radius * sin((i-1)*angleStep);
   spoints(i,2) = radius * cos((i-1)*angleStep);
end
 miny = min(spoints(:,1));
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
```

```
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
    error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
 % Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
 % Compute the LBP code img
for i = 1 : neighbors
    y = spoints(i,1) + origy;
    x = spoints(i,2) + origx;
    % Calculate floors, ceils and rounds for the x and y.
    fy = floor(y);
    cy = ceil(y);
    ry = round(y);
    fx = floor(x);
    cx = ceil(x);
    rx = round(x);
    % Check if interpolation is needed.
    if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
        % Interpolation is not needed, use original datatypes
        imgNew = img(ry:ry+dy,rx:rx+dx);
        blocks(i,:) = imgNew(:)';
    else
        % Interpolation needed, use double type images
        ty = y - fy;
        tx = x - fx;
        % Calculate the interpolation weights.
        w1 = (1 - tx) * (1 - ty);
        w2 =      tx  * (1 - ty);
        w3 = (1 - tx) *      ty ;
        w4 =      tx  *      ty ;
        % Compute interpolated pixel values
        imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
            w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
        blocks(i,:) = imgNew(:)';
    end
end % loop neighbors
 end % end of the function
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function blocks = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius)
 [imgH,imgW] = size(img);
 imgNewH = imgH - 2*lbpRadius;
```

```
imgNewW = imgW - 2*lbpRadius;
 % the interpolated img
blocks = zeros(lbpPoints,imgNewH*imgNewW);
 radius = lbpRadius;
neighbors = lbpPoints;
spoints = zeros(neighbors,2);
 % Determine the dimensions of the input img.
[ysize,xsize] = size(img);
 % Angle step
angleStep = 2 * pi / neighbors;
for i = 1 : neighbors
    spoints(i,1) = -radius * sin((i-1)*angleStep);
    spoints(i,2) = radius * cos((i-1)*angleStep);
end
 miny = min(spoints(:,1));
maxy = max(spoints(:,1));
minx = min(spoints(:,2));
maxx = max(spoints(:,2));
 % Block size, each LBP code is computed within angleStep block of size bsizey*bsizex
bsizey = ceil(max(maxy,0)) - floor(min(miny,0))+1;
bsizex = ceil(max(maxx,0)) - floor(min(minx,0))+1;
 % Coordinates of origin (0,0) in the block
origy = 1 - floor(min(miny,0));
origx = 1 - floor(min(minx,0));
 % Minimum allowed size for the input img depends
% on the radius of the used LBP operator.
if(xsize < bsizex || ysize < bsizey)
    error('Too small input img. Should be at least (2*radius+1) x (2*radius+1)');
end
% Calculate dx and dy;
dx = xsize - bsizex;
dy = ysize - bsizey;
% Compute the LBP code img
for i = 1 : neighbors
    y = spoints(i,1) + origy;
    x = spoints(i,2) + origx;
    % Calculate floors, ceils and rounds for the x and y.
    fy = floor(y);
    cy = ceil(y);
    ry = round(y);
        fx = floor(x);
    cx = ceil(x);
    rx = round(x);
    % Check if interpolation is needed.
    if (abs(x - rx) < 1e-6) && (abs(y - ry) < 1e-6)
        % Interpolation is not needed, use original datatypes
        imgNew = img(ry:ry+dy,rx:rx+dx);
        blocks(i,:) = imgNew(:)';
    else
        % Interpolation needed, use double type images
```

```
        ty = y - fy;
        tx = x - fx;
        % Calculate the interpolation weights.
        w1 = (1 - tx) * (1 - ty);
        w2 =      tx  * (1 - ty);
        w3 = (1 - tx) *      ty ;
        w4 =      tx  *      ty ;
        % Compute interpolated pixel values
        imgNew = w1*img(fy:fy+dy,fx:fx+dx) + w2*img(fy:fy+dy,cx:cx+dx) + ...
           w3*img(cy:cy+dy,fx:fx+dx) + w4*img(cy:cy+dy,cx:cx+dx);
        blocks(i,:) = imgNew(:)';
    end
end % loop neighbors
end % end of the function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
function mapping = get_mapping(samples)
 numAllLBPs = 2^samples;
table = 0 : numAllLBPs-1;
newMax = samples + 2; % number of patterns in the resulting LBP code
 for i = 0:2^samples - 1
    j = bitset(bitshift(i,1),1,bitget(i,samples)); % rotate left
    numt = sum(bitget(bitxor(i,j),1:samples));
    if numt <= 2
        table(i+1) = sum(bitget(i,1:samples));
    else
        table(i+1) = samples+1;
    end
end
 mapping.table = table;
mapping.samples = samples;
mapping.num = newMax;
 end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,check,setnum)
 global nump  %number of positive samples
global numn  %number of negative samples
 global pathp
global pathn
 global lbpRadius
global lbpPoints
global Y1
 numLBPbins = mapping.num;
 if check == 0
    if setnum == 1
        samplenum = nump;
        path = pathp;
    elseif setnum == 2
        samplenum = numn;
        path = pathn;
```

```
    end
    cfmsWithLabels_MRELBP_CINIRD =
zeros(samplenum,(numLBPbins*numLBPbins*2));
    for idxSample = 1 : samplenum
        Joint_CINIRD = zeros(numLBPbins,numLBPbins,2);
        img = imread(strcat(path, num2str(idxSample), '.png'));
       img = samp_prepro(img);
        imgExt = padarray(img,[1 1],'symmetric','both');
        imgblks = im2col(imgExt,[3 3],'sliding');
        a = median(imgblks);
        b = reshape(a,size(img));
        CImg = b(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
        CImg = CImg(:) - mean(CImg(:));
        CImg(CImg >= 0) = 2;
        CImg(CImg < 0) = 1;
        if lbpRadius == 2
            filWin = 3;
            halfWin = (filWin-1)/2;
            imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
            imgblks = im2col(imgExt,[filWin filWin],'sliding');
            imgMedian = median(imgblks);
            imgCurr = reshape(imgMedian,size(img));
            NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
            NILBPImage = NILBPImage(:);
            histNI = hist(NILBPImage,0:(numLBPbins-1));
            NILBPImage = NILBPImage + 1;
             RDLBPImage =
RDLBP_Image_SmallestRadiusOnly(b,imgCurr,lbpRadius,lbpPoints,mapping,'image');
            RDLBPImage = RDLBPImage(:);
            histRD = hist(RDLBPImage,0:(numLBPbins-1));
            RDLBPImage = RDLBPImage + 1;
        else
            if mod(lbpRadius,2) == 0
                filWin = lbpRadius + 1;
            else
                filWin = lbpRadius;
            end
            halfWin = (filWin-1)/2;
            imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
            imgblks = im2col(imgExt,[filWin filWin],'sliding');
            imgMedian = median(imgblks);
            imgCurr = reshape(imgMedian,size(img));
            NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
            NILBPImage = NILBPImage(:);
            histNI = hist(NILBPImage,0:(numLBPbins-1));
            NILBPImage = NILBPImage + 1;
             if mod(lbpRadiusPre,2) == 0
                filWin = lbpRadiusPre + 1;
            else
                filWin = lbpRadiusPre;
```

```
            end
             halfWin = (filWin-1)/2;
            imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
            imgblks = im2col(imgExt,[filWin filWin],'sliding');
            imgMedian = median(imgblks);
            imgPre = reshape(imgMedian,size(img));
             RDLBPImage =
NewRDLBP_Image(imgCurr,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,'image');
            RDLBPImage = RDLBPImage(:);
            histRD = hist(RDLBPImage,0:(numLBPbins-1));
            RDLBPImage = RDLBPImage + 1;
        end
            for i = 1 : length(NILBPImage)
            Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) =
Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) + 1;
        end
                cfmsWithLabels_MRELBP_CINIRD(idxSample,:) = Joint_CINIRD(:)';
    end
else
    cfmsWithLabels_MRELBP_CINIRD = zeros(1,(numLBPbins*numLBPbins*2));
    Joint_CINIRD = zeros(numLBPbins,numLBPbins,2);
    img = samp_prepro(Y1);
        imgExt = padarray(img,[1 1],'symmetric','both');
    imgblks = im2col(imgExt,[3 3],'sliding');
    a = median(imgblks);
    b = reshape(a,size(img));
    CImg = b(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
    CImg = CImg(:) - mean(CImg(:));
    CImg(CImg >= 0) = 2;
    CImg(CImg < 0) = 1;
    if lbpRadius == 2
        filWin = 3;
        halfWin = (filWin-1)/2;
        imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
        imgblks = im2col(imgExt,[filWin filWin],'sliding');
        imgMedian = median(imgblks);
        imgCurr = reshape(imgMedian,size(img));
        NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
        NILBPImage = NILBPImage(:);
        histNI = hist(NILBPImage,0:(numLBPbins-1));
        NILBPImage = NILBPImage + 1;
        RDLBPImage =
RDLBP_Image_SmallestRadiusOnly(b,imgCurr,lbpRadius,lbpPoints,mapping,'image');
        RDLBPImage = RDLBPImage(:);
        histRD = hist(RDLBPImage,0:(numLBPbins-1));
        RDLBPImage = RDLBPImage + 1;
    else
        if mod(lbpRadius,2) == 0
            filWin = lbpRadius + 1;
        else
```

```
        filWin = lbpRadius;
    end
    halfWin = (filWin-1)/2;
    imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
    imgblks = im2col(imgExt,[filWin filWin],'sliding');
    % each column of imgblks represents a feature vector
    imgMedian = median(imgblks);
    imgCurr = reshape(imgMedian,size(img));
    NILBPImage = NILBP_Image(imgCurr,lbpPoints,mapping,'image');
    NILBPImage = NILBPImage(:);
    histNI = hist(NILBPImage,0:(numLBPbins-1));
    NILBPImage = NILBPImage + 1;
     if mod(lbpRadiusPre,2) == 0
        filWin = lbpRadiusPre + 1;
    else
        filWin = lbpRadiusPre;
    end
     halfWin = (filWin-1)/2;
    imgExt = padarray(img,[halfWin halfWin],'symmetric','both');
    imgblks = im2col(imgExt,[filWin filWin],'sliding');
    imgMedian = median(imgblks);
    imgPre = reshape(imgMedian,size(img));
     RDLBPImage =
NewRDLBP_Image(imgCurr,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,'image');
    RDLBPImage = RDLBPImage(:);
    histRD = hist(RDLBPImage,0:(numLBPbins-1));
    RDLBPImage = RDLBPImage + 1;
  end
    for i = 1 : length(NILBPImage)
     Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) =
Joint_CINIRD(NILBPImage(i),RDLBPImage(i),CImg(i)) + 1;
  end
  cfmsWithLabels_MRELBP_CINIRD = Joint_CINIRD(:)';
end
 cfmsWithLabels_LBP = cfmsWithLabels_MRELBP_CINIRD;
clear cfmsWithLabels_MRELBP_CINIRD;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result =
NewRDLBP_Image(img,imgPre,lbpRadius,lbpRadiusPre,lbpPoints,mapping,mode)
blocks1 = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius);
blocks1 = blocks1';
 imgPre = imgPre(lbpRadius-lbpRadiusPre+1:end-(lbpRadius-lbpRadiusPre),lbpRadius-
lbpRadiusPre+1:end-(lbpRadius-lbpRadiusPre));
blocks2 = cirInterpSingleRadiusNew(imgPre,lbpPoints,lbpRadiusPre);
blocks2 = blocks2';
 radialDiff = blocks1 - blocks2;
 radialDiff(radialDiff >= 0) = 1;
radialDiff(radialDiff < 0) = 0;
 bins = 2^lbpPoints;
```

```
weight = 2.^(0:lbpPoints-1);
radialDiff = radialDiff .* repmat(weight,size(radialDiff,1),1);
% mapping = getmapping(lbpPoints,'riu2');
 radialDiff = sum(radialDiff,2);
result = radialDiff;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end
  if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
else
    % Otherwise return a matrix of unsigned integers
    if ((bins-1) <= intmax('uint8'))
        result = uint8(result);
    elseif ((bins-1) <= intmax('uint16'))
        result = uint16(result);
    else
        result = uint32(result);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
function result = NILBP_Image(img,lbpPoints,mapping,mode)
blocks = cirInterpSingleRadius(img);
blocks = blocks';
blocks = blocks - repmat(mean(blocks,2),1,size(blocks,2));
 blocks(blocks >= 0) = 1;
blocks(blocks < 0) = 0;
 weight = 2.^(0:lbpPoints-1);
blocks = blocks .* repmat(weight,size(blocks,1),1);
 blocks = sum(blocks,2);
result = blocks;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
```

```
end
 if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
else
    % Otherwise return a matrix of unsigned integers
    % result = reshape(result,size(imgTemp));
    if ((bins-1) <= intmax('uint8'))
        result = uint8(result);
    elseif ((bins-1) <= intmax('uint16'))
        result = uint16(result);
    else
        result = uint32(result);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result =
RDLBP_Image_SmallestRadiusOnly(imgCenSmooth,img,lbpRadius,lbpPoints,mapping,mod
e)
blocks1 = cirInterpSingleRadiusNew(img,lbpPoints,lbpRadius);
blocks1 = blocks1';
imgTemp = imgCenSmooth(lbpRadius+1:end-lbpRadius,lbpRadius+1:end-lbpRadius);
blocks2 = repmat(imgTemp(:),1,lbpPoints);
 radialDiff = blocks1 - blocks2;
 radialDiff(radialDiff >= 0) = 1;
radialDiff(radialDiff < 0) = 0;
 bins = 2^lbpPoints;
weight = 2.^(0:lbpPoints-1);
radialDiff = radialDiff .* repmat(weight,size(radialDiff,1),1);
% mapping = getmapping(lbpPoints,'riu2');
 radialDiff = sum(radialDiff,2);
result = radialDiff;
% Apply mapping if it is defined
if isstruct(mapping)
    bins = mapping.num;
    for i = 1:size(result,1)
        for j = 1:size(result,2)
            result(i,j) = mapping.table(result(i,j)+1);
        end
    end
end
  if (strcmp(mode,'h') || strcmp(mode,'hist') || strcmp(mode,'nh'))
    % Return with LBP histogram if mode equals 'hist'.
    result = hist(result(:),0:(bins-1));
    if (strcmp(mode,'nh'))
        result = result/sum(result);
    end
```

```
else
   % Otherwise return a matrix of unsigned integers
   if ((bins-1) <= intmax('uint8'))
      result = uint8(result);
   elseif ((bins-1) <= intmax('uint16'))
      result = uint16(result);
   else
      result = uint32(result);
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sampleIn = samp_prepro(sampleIn)
% image sample preprocessing
sampleIn = double(sampleIn);
sampleIn = sampleIn - mean(sampleIn(:));
% sampleIn = sampleIn / sqrt(mean(mean(sampleIn .^ 2)));
sampleIn = sampleIn / std(sampleIn(:));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
function D = sqdist(X1, X2)
% Pairwise square Euclidean distance between two sample sets
% Input:
%   X1, X2: dxn1 dxn2 sample matrices
% Output:
%   D: n1 x n2 square Euclidean distance matrix
% Written by Mo Chen (sth4nth@gmail.com).
D = bsxfun(@plus,dot(X2,X2,1),dot(X1,X1,1)')-2*(X1'*X2);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
----------------------Preprocessing stage-----------
global nump  %number of positive samples
global numn  %number of negative samples
 global pathp
global pathn
 %% preprocessing of train images
tic
path1 = 'train/lp';   %training lps
path2 = 'train/nonlp';  %tranging nonlps
 pathp = 'realtrain/lp/';   %rectangle training lps
pathn = 'realtrain/nonlp/';  %tranging nonlps
 %preprocessed training lps will be saved in 'realtrain' directory
if ~isdir('realtrain')
   mkdir('realtrain');
   mkdir('realtrain/lp');
   mkdir('realtrain/nonlp')
end
 %image files in directories
 files1 = dir(fullfile(path1,'*.png'));
files2 = dir(fullfile(path2,'*.png'));
%number of positive and negative tranging samples
```

```
nump = numel(files1); %number of training lps = positive
numn = numel(files2);  %number of nonlps = negative

%preprocessing of training lps
for samples = 1 : nump
   file = fullfile(path1, files1(samples).name);
      X = imread(file);
      M = rgb2gray(X);
       [m n] = size(M);
   p = 50;
   q = 200;
   if p >= m && q >= n
      M_pad = padarray(M, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
      M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
   elseif p < m && q >= n
      m = p;
      M_pad = imresize(M, [m, n]);
      M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
      M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
   elseif p >= m && q < n
      n = q;
      M_pad = imresize(M, [m, n]);
      M_pad = padarray(M_pad, [floor((p-m)/2) floor((q-n)/2)], 'replicate','post');
      M_pad = padarray(M_pad, [ceil((p-m)/2) ceil((q-n)/2)], 'replicate','pre');
   elseif p < m && q < n
      M_pad = imresize(M, [p, q]);
   end
      M = M_pad;
      numofpixels=size(M,1)*size(M,2);
   Y=uint8(zeros(size(M,1),size(M,2)));
   cnts=zeros(256,1);
   probf=zeros(256,1);
   prbc=zeros(256,1);
   cum=zeros(256,1);
   fin_lc=zeros(256,1);
      for i=1:size(M,1)
      for j=1:size(M,2)
         value=M(i,j);
         cnts(value+1)=cnts(value+1)+1;
         probf(value+1)=cnts(value+1)/numofpixels;
      end
   end
      sum=0;
   n=255;

   for i=1:size(probf)
     sum=sum+cnts(i);
     cum(i)=sum;
     prbc(i)=cum(i)/numofpixels;
     fin_lc(i)=round(prbc(i)*n);
```

```
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
            Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
    imwrite(Y, strcat(pathp, num2str(samples), '.png'));
end
 %preprocessing of training nonlps
for samples = 1 : numn
    file = fullfile(path2, files2(samples).name);
    X = imread(file);
    X = imresize(X, 0.5);
   M = imgaussfilt(X,0.25);
    M = rgb2gray(M);
     numofpixels=size(M,1)*size(M,2);
    Y=uint8(zeros(size(M,1),size(M,2)));
    cnts=zeros(256,1);
    probf=zeros(256,1);
    prbc=zeros(256,1);
    cum=zeros(256,1);
    fin_lc=zeros(256,1);
       for i=1:size(M,1)
       for j=1:size(M,2)
          value=M(i,j);
          cnts(value+1)=cnts(value+1)+1;
          probf(value+1)=cnts(value+1)/numofpixels;
       end
    end
       sum=0;
    n=255;
       for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
    end
       for i=1:size(M,1)
         for j=1:size(M,2)
            Y(i,j)=fin_lc(M(i,j)+1);
         end
    end
        imwrite(Y, strcat(pathn, num2str(samples), '.png'));
end
toc
-------------------------------------Training\Extraction Stages--------------------
addpath('mrelbp');
addpath('elmbase');
 global nump  %number of positive samples
global numn  %number of negative samples
```

```
 global lbpRadius
global lbpPoints
 %% selecting mrelbp features from preprocessed train images
tic
lbpRadiusSet = [2 4 6 8];
lbpPointsSet = [8 8 8 8];
trainfeatures1 = [];
trainfeatures2 = [];
trainfeatures4 = [];
trainfeatures5 = [];
for idxLbpRadius = 1 : length(lbpRadiusSet)
    lbpRadius = lbpRadiusSet(idxLbpRadius);
    lbpPoints = lbpPointsSet(idxLbpRadius);
       mapping = get_mapping(lbpPoints);
    blockSize = lbpRadius*2+1;
       if idxLbpRadius > 1
       lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
    else
       lbpRadiusPre = 0;
    end
    %--------------------SURF--------------------
    Z=  detectSURFFeatures(Y);
    [SURF_features, valid_points] = extractFeatures(Y,Z);
    strongestPoints = valid_points.selectStrongest(60);
   trainfeatures4= [trainfeatures4 SURF_features ];
    trainfeatures5= [trainfeatures5 SURF_features];
       cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,1);
    trainfeatures1 = [trainfeatures1 cfmsWithLabels_LBP  ];
    clear cfmsWithLabels_LBP
    cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,0,2);
    trainfeatures2 = [trainfeatures2 cfmsWithLabels_LBP  ];
    clear cfmsWithLabels_LBP
end
 trainfeatures1 = [trainfeatures1; trainfeatures2];
trainfeatures4 = [trainfeatures4; trainfeatures5];
 shape = size(trainfeatures1);
shape2= size(trainfeatures4);
feature = [];
feature4= [];
for i = 1 : shape(1)
    feature = [feature; trainfeatures1(i,:)/max(trainfeatures1(i,:))];
end
for i = 1 : shape2(1)
    feature4 = [feature4; trainfeatures4(i,:)/max(trainfeatures4(i,:))];
end
 classes = zeros(nump + numn, 1);
for i = 1 : nump
    classes(i) = 1;
end
 M = [classes feature];
```

```
M4 = [feature4];
dlmwrite('traindata',M, ' ');
%dlmwrite('traindata',M4, ' ');
 elm_train('traindata', 1, 1000, 'sig');
toc
----------------------------------------Testing or Detection Stage----------
global Y1
tic
if ~isdir('result')
   mkdir('result');
end
if ~isdir('detected')
   mkdir('detected');
end
%% testing
files = dir(fullfile('test','*.png'));
for id = 1 : numel(files)
   file = fullfile('test/', files(id).name);
      %preprocessing
   X=imread(file);
% M = imresize(X,[240, 320]);
   M=rgb2gray(X);
    numofpixels=size(M,1)*size(M,2);
   Y=uint8(zeros(size(M,1),size(M,2)));
   cnts=zeros(256,1);
   probf=zeros(256,1);
   prbc=zeros(256,1);
   cum=zeros(256,1);
   fin_lc=zeros(256,1);
    for i=1:size(M,1)
       for j=1:size(M,2)
          value=M(i,j);
          cnts(value+1)=cnts(value+1)+1;
          probf(value+1)=cnts(value+1)/numofpixels;
       end
   end
    sum=0;
   n=255;
    for i=1:size(probf)
      sum=sum+cnts(i);
      cum(i)=sum;
      prbc(i)=cum(i)/numofpixels;
      fin_lc(i)=round(prbc(i)*n);
   end
    for i=1:size(M,1)
      for j=1:size(M,2)
         Y(i,j)=fin_lc(M(i,j)+1);
      end
   end
      lbpRadiusSet = [2 4 6 8];
```

```
lbpPointsSet = [8 8 8 8];
 % selecting mrelbp features from preprocessed test image
xsum=0;
ysum=0;
num = 0;
px = [];
py = [];
px1 = [];
py1 = [];
px2 = [];
py2 = [];
px3 = [];
 for i = 126 :100: size(Y,1) - 165
  for j = 50 :18: size(Y,2) - 350
      Y1 = Y(i:i+49,j:j+199);
      testfeatures1 = [];
      for idxLbpRadius = 1 : length(lbpRadiusSet)
         lbpRadius = lbpRadiusSet(idxLbpRadius);
         lbpPoints = lbpPointsSet(idxLbpRadius);
          mapping = get_mapping(lbpPoints);
         blockSize = lbpRadius*2+1;
          if idxLbpRadius > 1
            lbpRadiusPre = lbpRadiusSet(idxLbpRadius-1);
         else
            lbpRadiusPre = 0;
         end
          cfmsWithLabels_LBP = MRELBP(mapping,lbpRadiusPre,1,1);
          testfeatures1 = [testfeatures1 cfmsWithLabels_LBP];
      end
      feature = testfeatures1 / max(testfeatures1);
      dlmwrite('testdata',[1 feature], ' ');
      [TestingTime, TestingAccuracy, output]= elm_predict('testdata');
      if output == 1
         %num = num + 1;
         px2 = [px2 i ];
         py2 = [py2 j ];
          continue;
      end
   end
end
 fh = figure;
imshow( X, 'border', 'loose' ); %//show your image
hold on;
 px21 = unique(px2);
py21 = [];
for i = 1 : length(px21)
   temp = [];
   for j = 1 : length(px2)
      if px21(i) == px2(j)
         temp = [temp; py2(j)];
```

```
            end
        end
        py21 = [py21; round(mean(temp))];
    end
    for i = 1 : length(px21)
        if i > 1
            if abs(px21(i)-px21(i-1)) > 30
                rectangle('Position', [py21(i)-10 px21(i) 300 100], 'EdgeColor', 'y'); %// draw
rectangle on image
                frm = getframe( fh ); %// get the image+rectangle
                imwrite( frm.cdata, strcat('result/',num2str(id),'.png') ); %// save to file
            end
        end
    end
        pause(3);
    close(fh);
    disp(strcat('test image ', num2str(id), ' done' ));
 end
toc
------------------------------ROC_curve-----------------------
%             + AROC:     Area Under ROC Curve.                    %
%             + Accuracy: Maximum accuracy obtained.                 %
%             + Sensi:    Optimum threshold sensitivity.             %
%             + Speci:    Optimum threshold specificity.             %
%             + PPV:      Positive predicted value.                %
%             + NPV:      Negative predicted value.                  %
%         - curve:    Matrix which contains the specificity and specifi-%
%                     city of each threshold point in columns.         %
function ROC_data = roc_curve(classes1,classes2, dispp, dispt)
    % Setting default parameters and detecting errors
    if(nargin<4), dispt = 1;    end
    if(nargin<3), dispp = 1;    end
   % if(nargin<2), error('Class_1 or class_2 are not indicated.'); end
   model1 = loadCompactModel('model1');
        y=load('classes1.mat');
    class_1 = 0.13* randn(700,1);
    class_2 = 0.5 + 0.1* randn(700,1);
      % Calculating the threshold values between the data points
    s_data = unique(sort([class_1; class_2]));         % Sorted data points
    s_data(isnan(s_data)) = [];                % Delete NaN values
    d_data = diff(s_data);                    % Difference between consecutive points
    if(isempty(d_data)), error('Both class data are the same!'); end
    d_data(length(d_data)+1,1) = d_data(length(d_data));% Last point
    thres(1,1) = s_data(1) - d_data(1);                % First point
    thres(2:length(s_data)+1,1) = s_data + d_data./2;   % Threshold values
     % Sorting each class
    if(nanmean(class_1)>nanmean(class_2))
    end         % Calculating the sensibility and specificity of each threshold
    curve = zeros(size(thres,1),2);
    distance = zeros(size(thres,1),1);
```

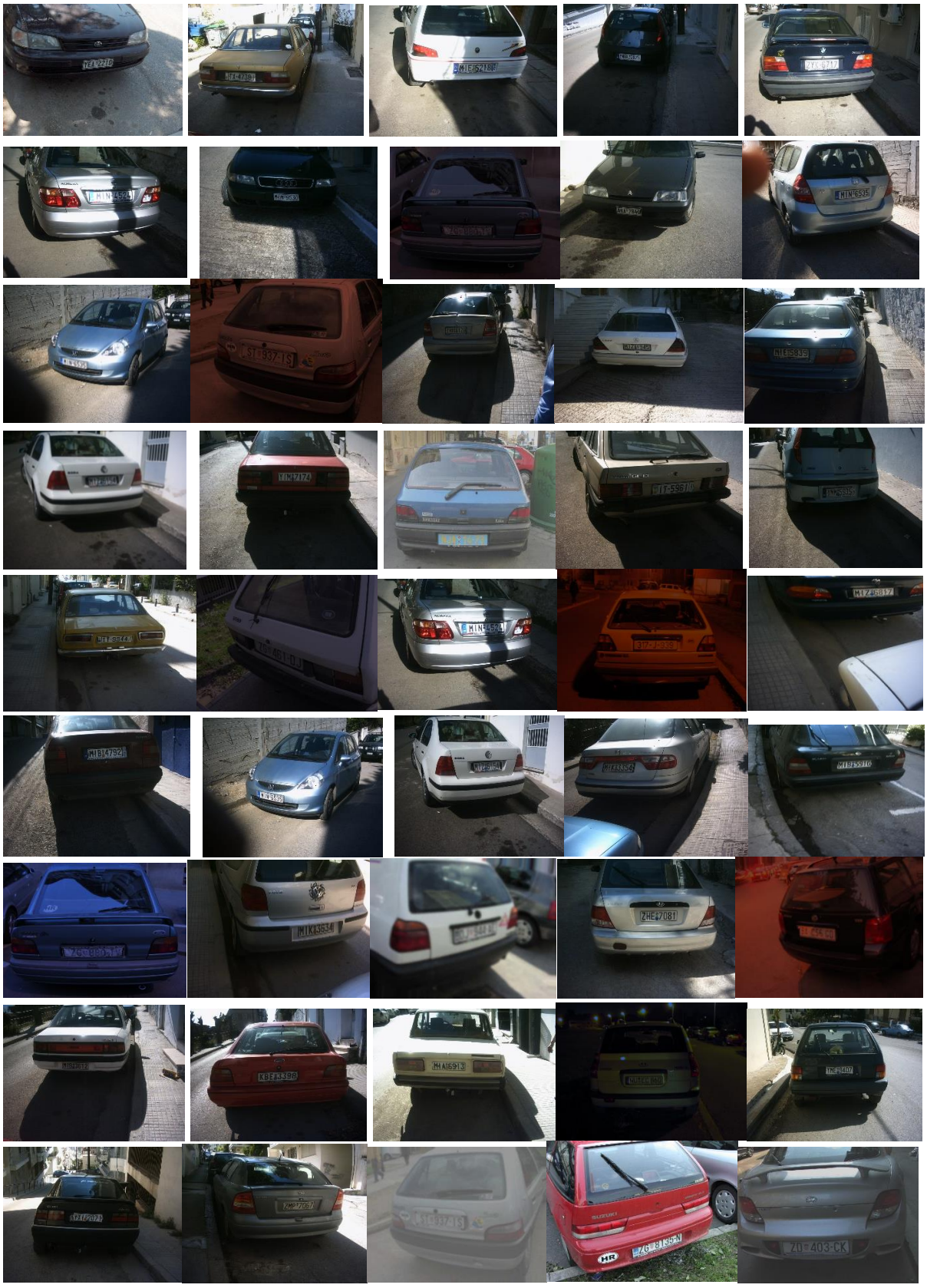
```
    for id_t = 1:1:length(thres)
        TP = length(find(class_2 >= thres(id_t)));    % True positives
        FP = length(find(class_1 >= thres(id_t)));    % False positives
        FN = length(find(class_2 < thres(id_t)));     % False negatives
        TN = length(find(class_1 < thres(id_t)));     % True negatives
              curve(id_t,1) = TP/(TP + FN);   % Sensitivity
        curve(id_t,2) = TN/(TN + FP);   % Specificity
              % Distance between each point and the optimum point (0,1)
        distance(id_t)= sqrt((1-curve(id_t,1))^2+(curve(id_t,2)-1)^2);
    end         % Optimum threshold and parameters
    [~, opt] = min(distance);
    TP = length(find(class_2 >= thres(opt)));
    FP = length(find(class_1 >= thres(opt)));
    FN = length(find(class_2 < thres(opt)));
    TN = length(find(class_1 < thres(opt)));
    param.Threshold = thres(opt);       % Optimum threshold position
    param.Sensi = curve(opt,1);         % Optimum threshold's sensitivity
    param.Speci = curve(opt,2);         % Optimum threshold's specificity
    param.AROC  = abs(trapz(1-curve(:,2), curve(:,1))); % Area under curve
    param.Accuracy = (TP+TN)/(TP+TN+FP+FN);           % Maximum accuracy
    param.PPV   = TP/(TP+FP);           % Positive predictive value
    param.NPV   = TN/(TN+FN);           % Negative predictive value
     % Plotting if required
    if(dispp == 1)
        fill_color = [11/255, 208/255, 217/255];
        fill([1-curve(:,2); 1], [curve(:,1); 0], fill_color,'FaceAlpha',0.10);
        hold on; plot(1-curve(:,2), curve(:,1), '-r', 'LineWidth', 3);
        hold on; plot(1-curve(opt,2), curve(opt,1), 'ob', 'MarkerSize', 5);
        hold on; plot(1-curve(opt,2), curve(opt,1), 'xb', 'MarkerSize', 12);
        hold off; axis square; grid on; xlabel('False Positive Rate'); ylabel('True Positive Rate');
        legend('Threshold =  0.2140','Location','SE');
        title(['AUC for Testing by ELM = ' num2str(param.AROC)]);
    end         % Log screen parameters if required
    if(dispt == 1)
        fprintf('\n ROC CURVE PARAMETERS\n');
        fprintf(' ----------------------------\n');
        fprintf('  - Distance:     %.4f\n', distance(opt));
        fprintf('  - Threshold:    %.4f\n', param.Threshold);
        fprintf('  - Sensitivity:  %.4f\n', param.Sensi);
        fprintf('  - Specificity:  %.4f\n', param.Speci);
        fprintf('  - AROC:         %.4f\n', param.AROC);
        fprintf('  - Accuracy:     %.4f%%\n', param.Accuracy*100);
        fprintf('  - PPV:          %.4f%%\n', param.PPV*100);
        fprintf('  - NPV:          %.4f%%\n', param.NPV*100);
        fprintf(' \n');
    end         % Assinging parameters and curve data
    ROC_data.param = param;
    ROC_data.curve = curve;
end
-------------------------------------THE END
```

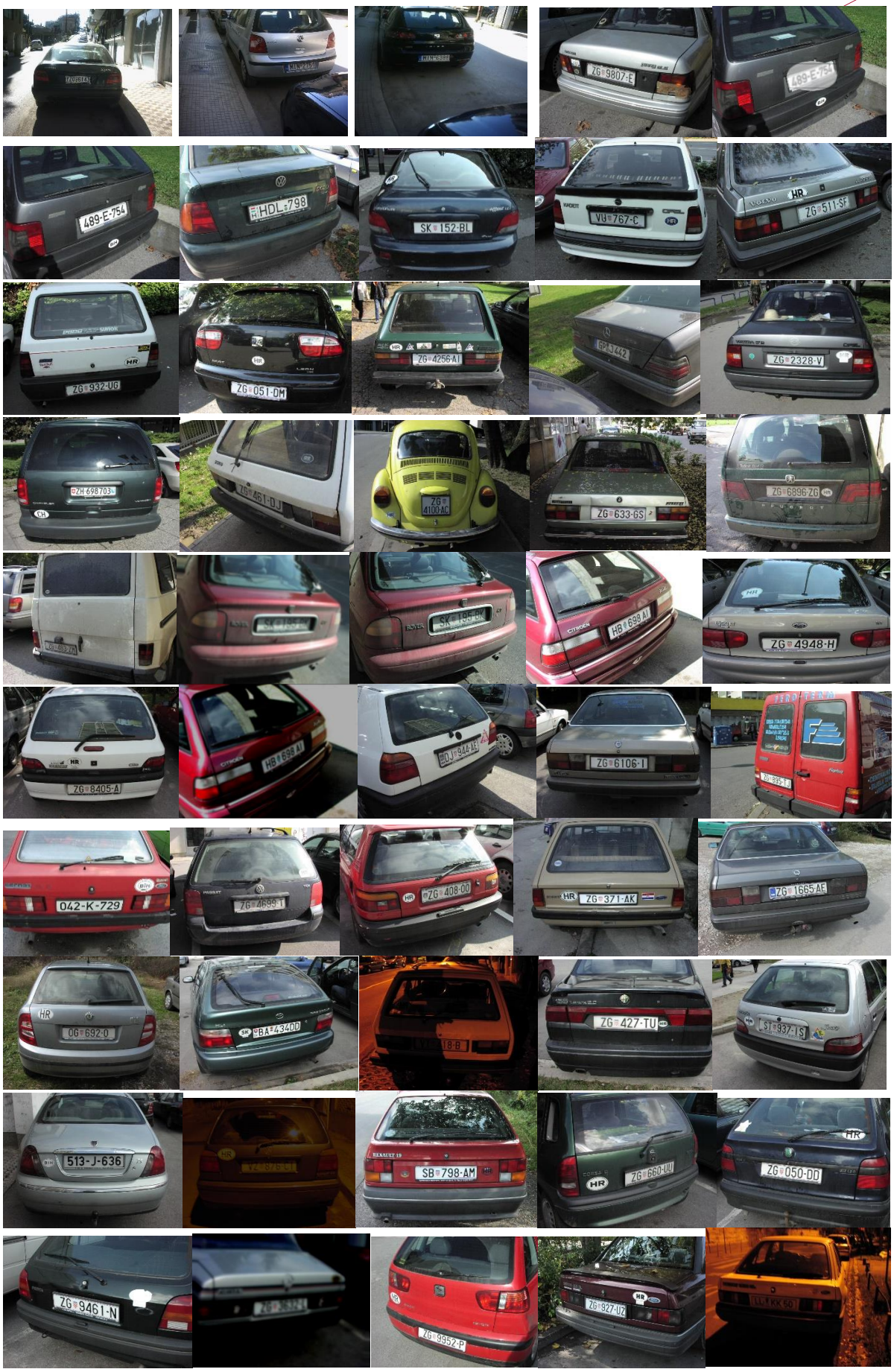
E

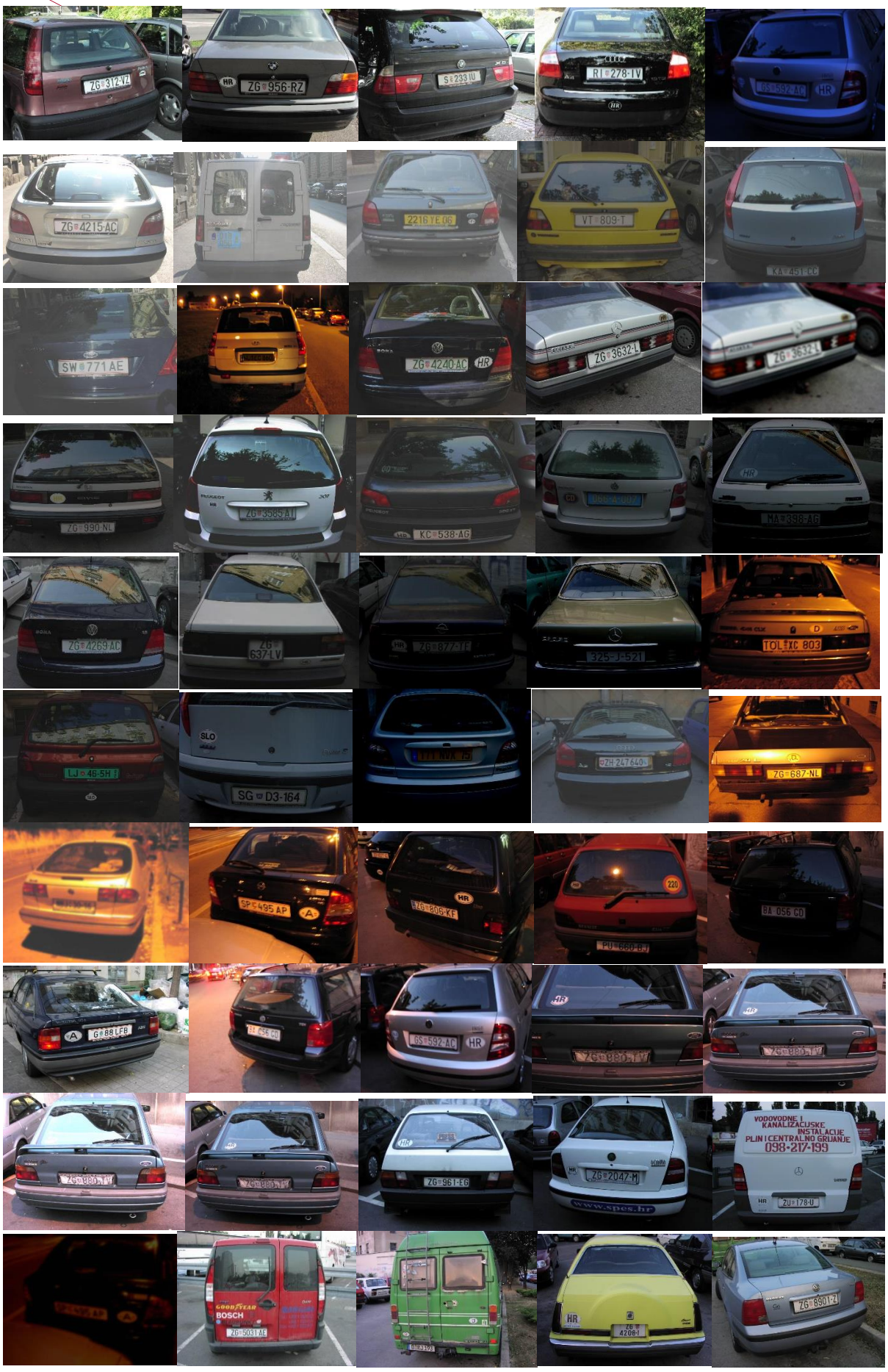
English cars database for research project

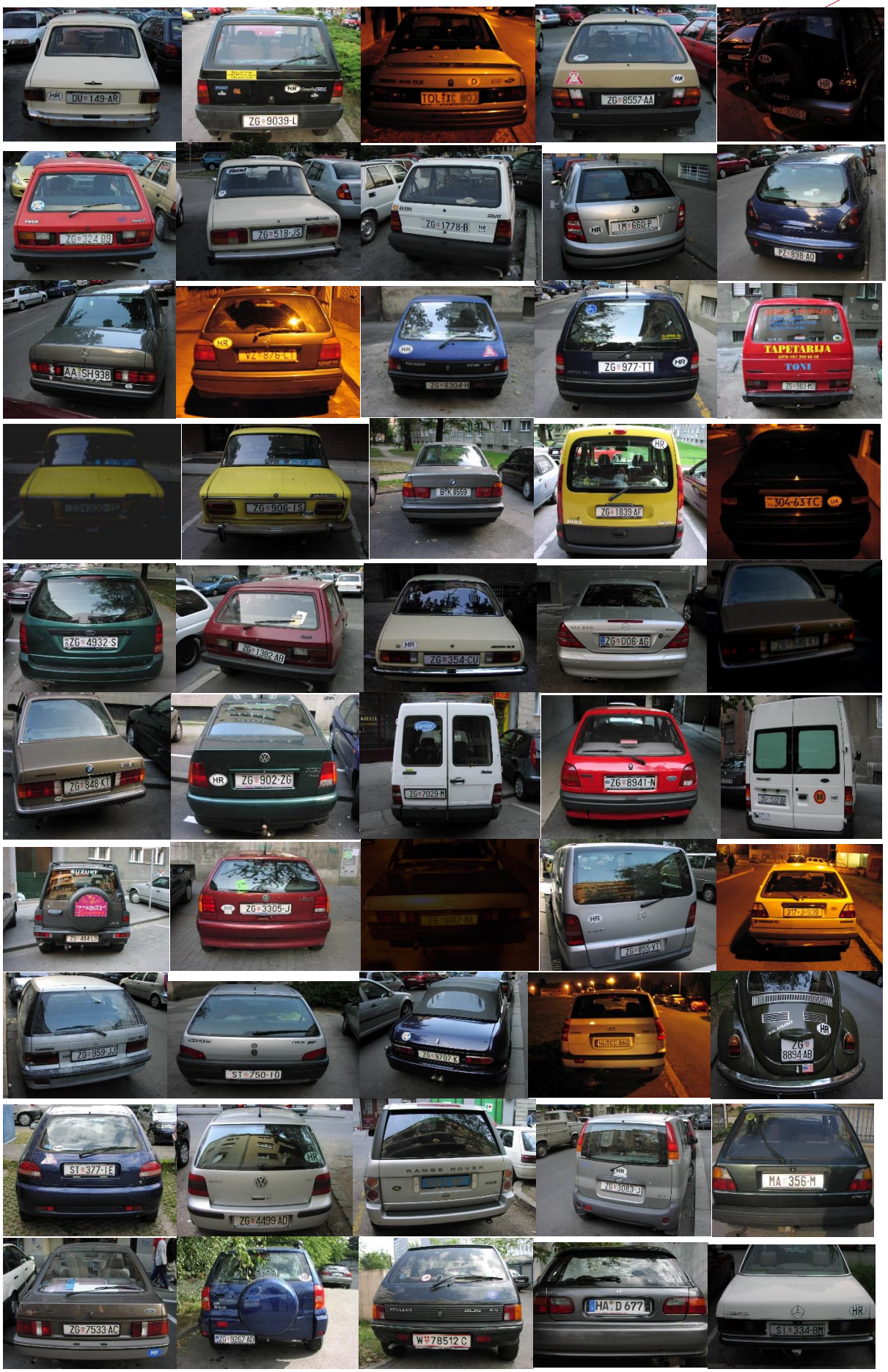
This appendix includes some samples for testing vehicle images and real training licences plates images for English cars database.

Some samples for vehicles images



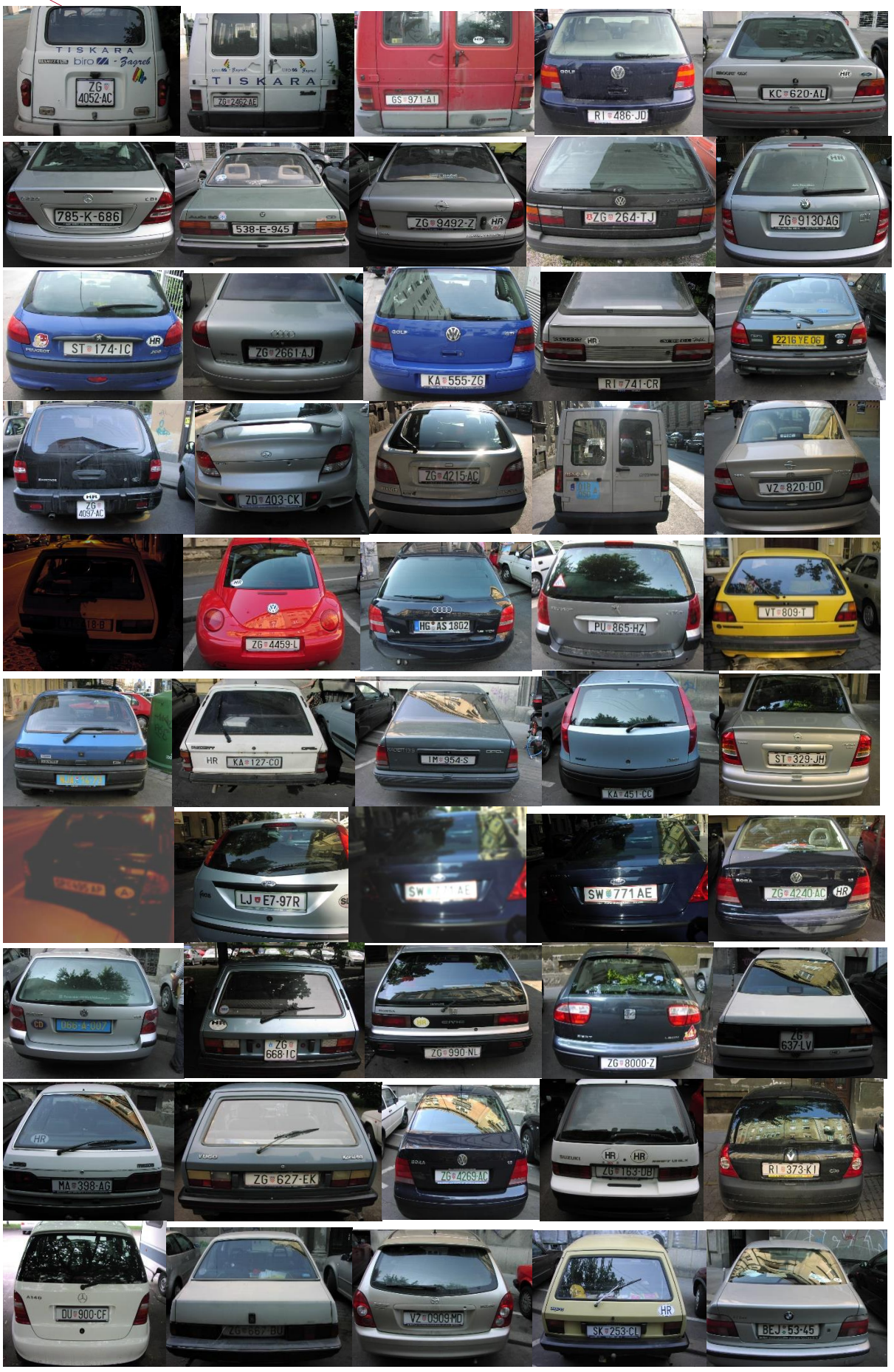

ZG 9807-E
489-E-754
HDL 798
SK 152-BL
VU 767-C
ZG 511-SF
ZG 932-UG
ZG 051-DM
ZG 4256-AI
ZG 2328-V
ZH 698703
ZG 461-DJ
ZG 4100-AC
ZG 633-GS
ZG 6896-ZG
ZG 4948-H
ZG 8405-A
ZG 6106-I
042-K-729
ZG 4699-T
ZG 408-00
ZG 371-AK
ZG 1665-AE
OG 692-O
SB 798-AM
ZG 427-TU
ZG 660-UU
ZG 050-DD
513-J-636
ZG 9461-N
ZG 3952-P
ZG 927-UZ

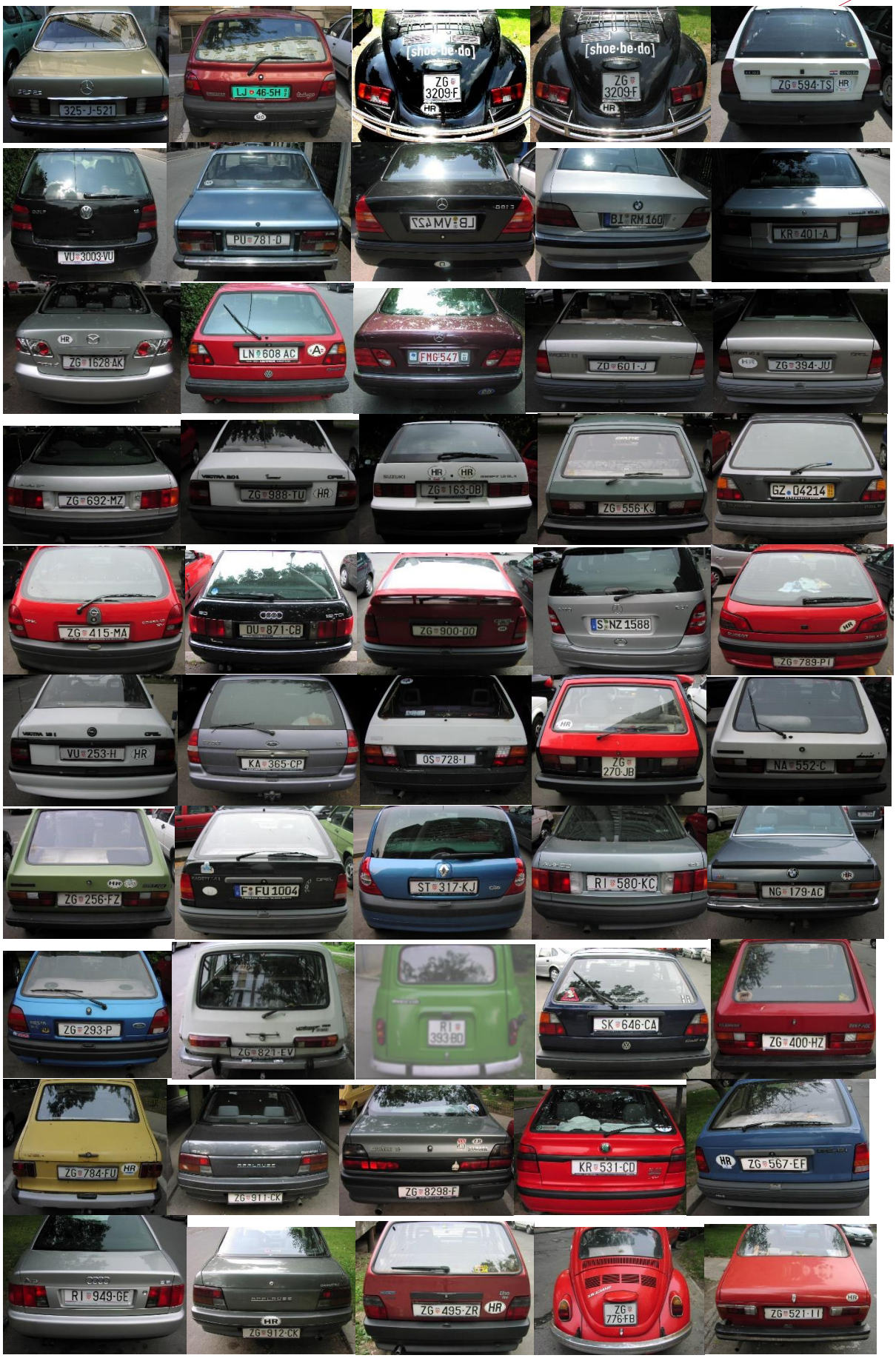

ZG 312-VZ
ZG 956-RZ
S 233 IU
RI 278-IV
ZG 4215-AC
2216 YE 06
VT 809-T
KA 451 EC
SW 771 AE
ZG 4240-AC
ZG 3632-L
ZG 990-NL
ZG 3585-AI
KC 538-AG
MA 398-AG
ZG 4269-AC
ZG 637-LV
325-J-521
TOL XC 803
LJ 46-5H
SG D3-164
ZG 687-NL
SP 495 AP
ZG 8066 KF
BA 056 CD
G 88 LFB
ZG 961-EG
ZG 2047-M
VODOVODNE I KANALIZACIJSKE INSTALACIJE PLIN I CENTRALNO GRIJANJE 098-217-199
ZU 178-U
BOSCH
ZG 5031 AE
ZG 890-TZ


DU 149-AR
ZG 9039-L
TOLIX 803
ZG 8557-AA
ZG 324-BB
ZG 518-JS
ZG 1778-B
IM 660-P
PZ 898-AO
AA SH938
VZ 876-CI
ZG 6304-H
ZG 977-TT
TAPETARIJA
TONI
ZG 906-TS
BHK 8659
ZG 1839-AF
304-63TC
ZG 4932-S
ZG 1382-AB
ZG 354-CU
ZG 006-AG
ZG 848-KT
ZG 902-ZG
ZG 7029-M
ZG 8941-N
ZG 3305-J
ZG 687-NL
ZG 855-YT
ZG 959-JJ
ST 750-TO
ZG 9707-X
ZG 8834-AB
ST 377-IE
ZG 4499-AD
ZG 3083-J
MA 356-M
ZG 7533-AC
W 78512 C
HA D 677
SI 334-BM

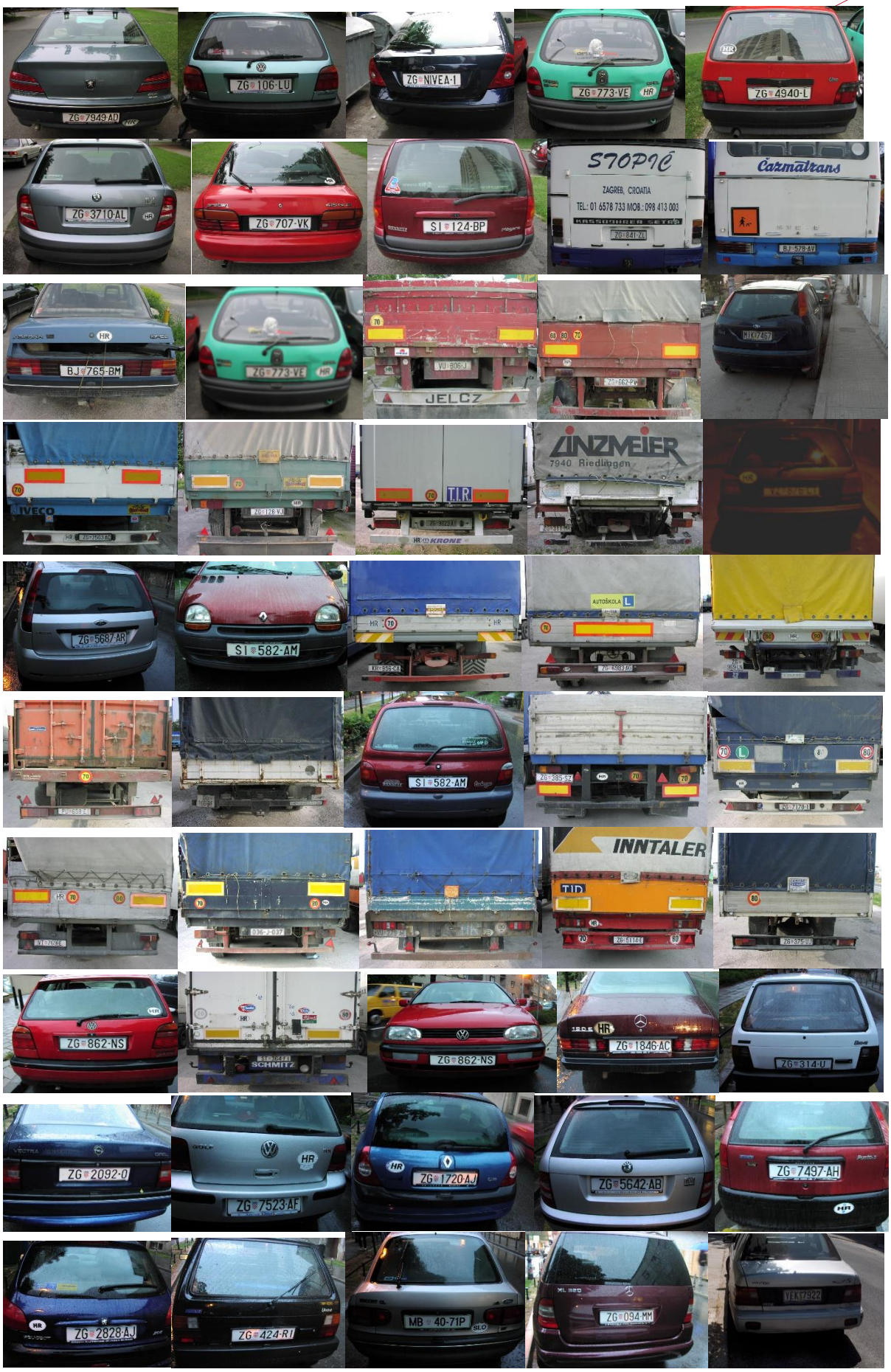

SK 553-AL
SK 314-CK
ZG 414-BV
SP GO 65
G 22 SKD
ZD 986-CU
HONDA
ZG 3428-J
CK 425-AB
ZG 1041-P
ZG 624-K
K 8948 M
AKO 02-81
ZG 5023-B
ZG 8635-T
VZ 539-AP
RTV- SERVIS
SERVIS - PRODAJA - MONTAŽA
MOB: 098 221-018
ZG 2751-B
ZG 887-K
ZG 754-VV
ZG 687-NL
ZG 8240-AF
ZG 4847-S
SPORTAGE
ZG 4309-AG
ZG 2877-J
F GH 1662
ZG 424-GH
ZG 2808-AA
KA 570-CD
ZG 9353-AA
ZG 542-VB
ZG 216-TK
ZG 9830-AE
ZG 7761-M
ZG 348-CR
ZG 416-PZ
ZG 2273-E
R1 238-JL
ZG 4497-P
DU 986-BM
NG 619-AA
ZG 8524-AF
CA 890 BC
ZG 593-ZH

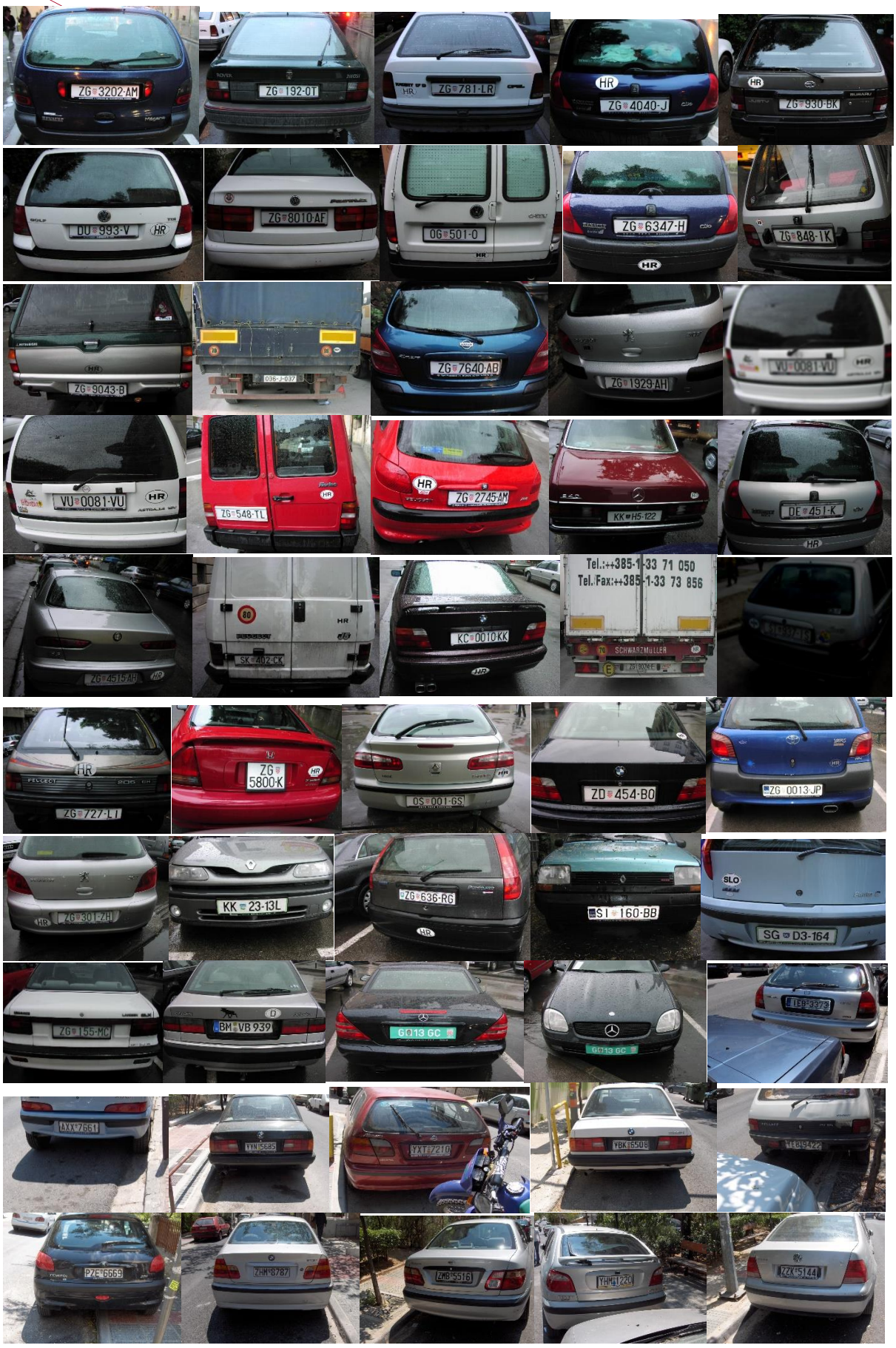

317-J-939
ZG 977-SI
ZG 3254-D
W 51620 L
KT 451-AN
ZG 9572-J
KA AC 1689
KA AC 1689
ZG 200-ID
PF-S 3682
BJ 749-G
ZG 155-UB
BHA 19-31
ZG 1379-N
ZG 316-GB
DE 359-M
VU 369-AN
PU 659-GB
KK 97-34H
ZG 4094-O
812-T-475
ZG 4154-AC
SB 268-BZ
KZ 425-V
DU 486-AD
VK 750-AI
ZG 3227-N
ZG 7499-AJ
ST 102-EU
PU 938-CE
OG 392-A
VK 605-AN
RI 982-JF
ZG 8152-G
RI 545-IC
ZD 448-AK
SI 450-AB
DA 545-AI
OS 269-DT
DU 711-CH
PZ 807-AO
AKO 04-60
ST 133-EV
ST 747-EA
AL 111 AT
304-E-726

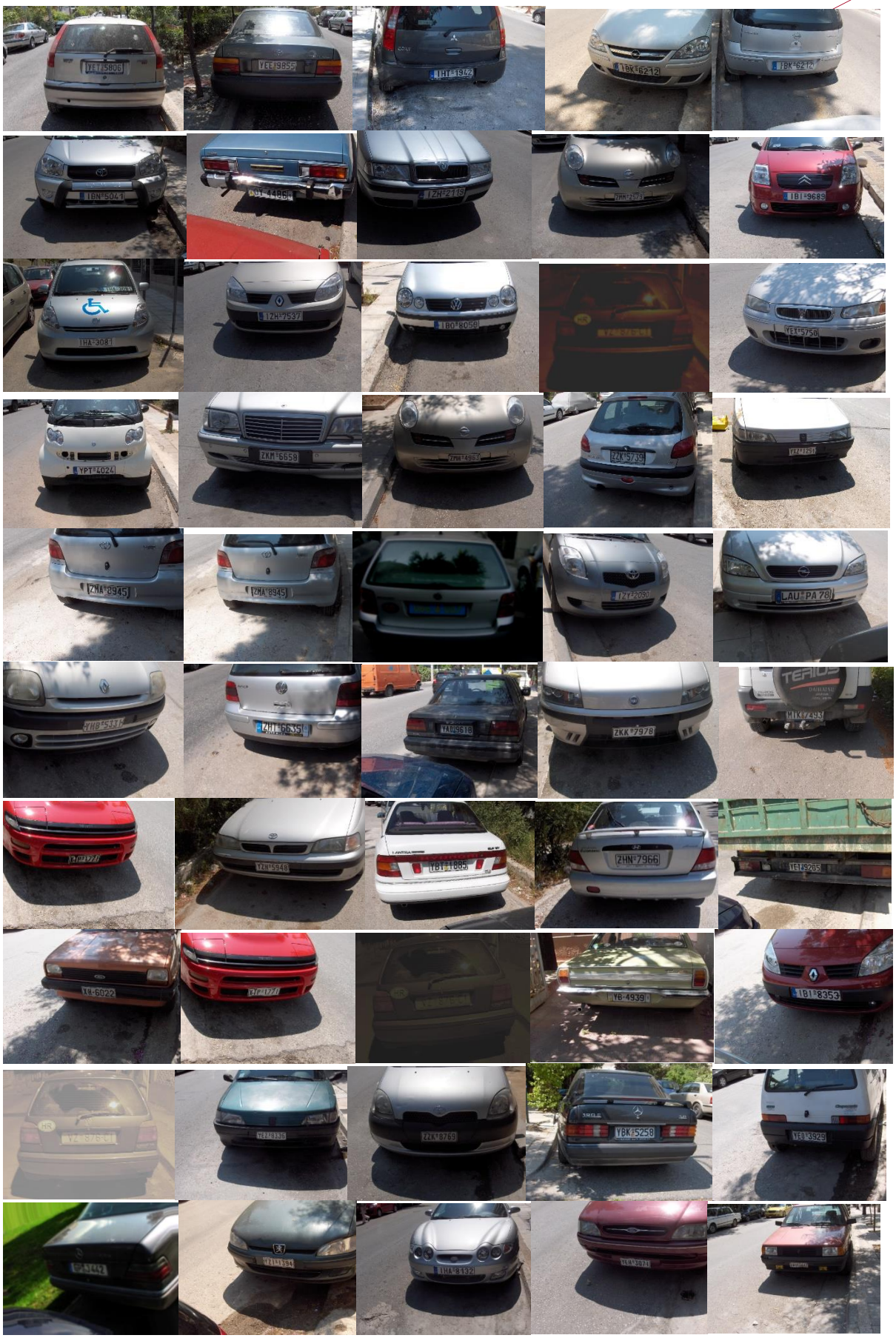

TISKARA
ZG 4052-AC
TISKARA
ZG 2462-AE
GS 971-AI
RI 486-JD
KC 620-AL
785-K-686
538-E-945
ZG 9492-Z
ZG 264-TJ
ZG 9130-AG
ST 174-IC
ZG 2661-AJ
KA 555-ZG
RI 741-CR
221G YE 06
ZG 4057-AC
ZD 403-CK
ZG 4215-AC
VZ 820-DD
ZG 4459-L
HG AS 1802
PU 865-HZ
VT 809-T
HR KA 127-CO
IM 954-S
KA 451-CC
ST 329-JH
LJ E7-97R
SW 771 AE
ZG 4240-AC
ZG 668-IC
ZG 990-NL
ZG 8000-Z
ZG 637-LV
MA 398-AG
ZG 627-EK
ZG 4269-AC
ZG 163-DB
RI 373-KI
DU 900-CF
VZ 0909-MD
SK 253-CL
BEJ 53-45

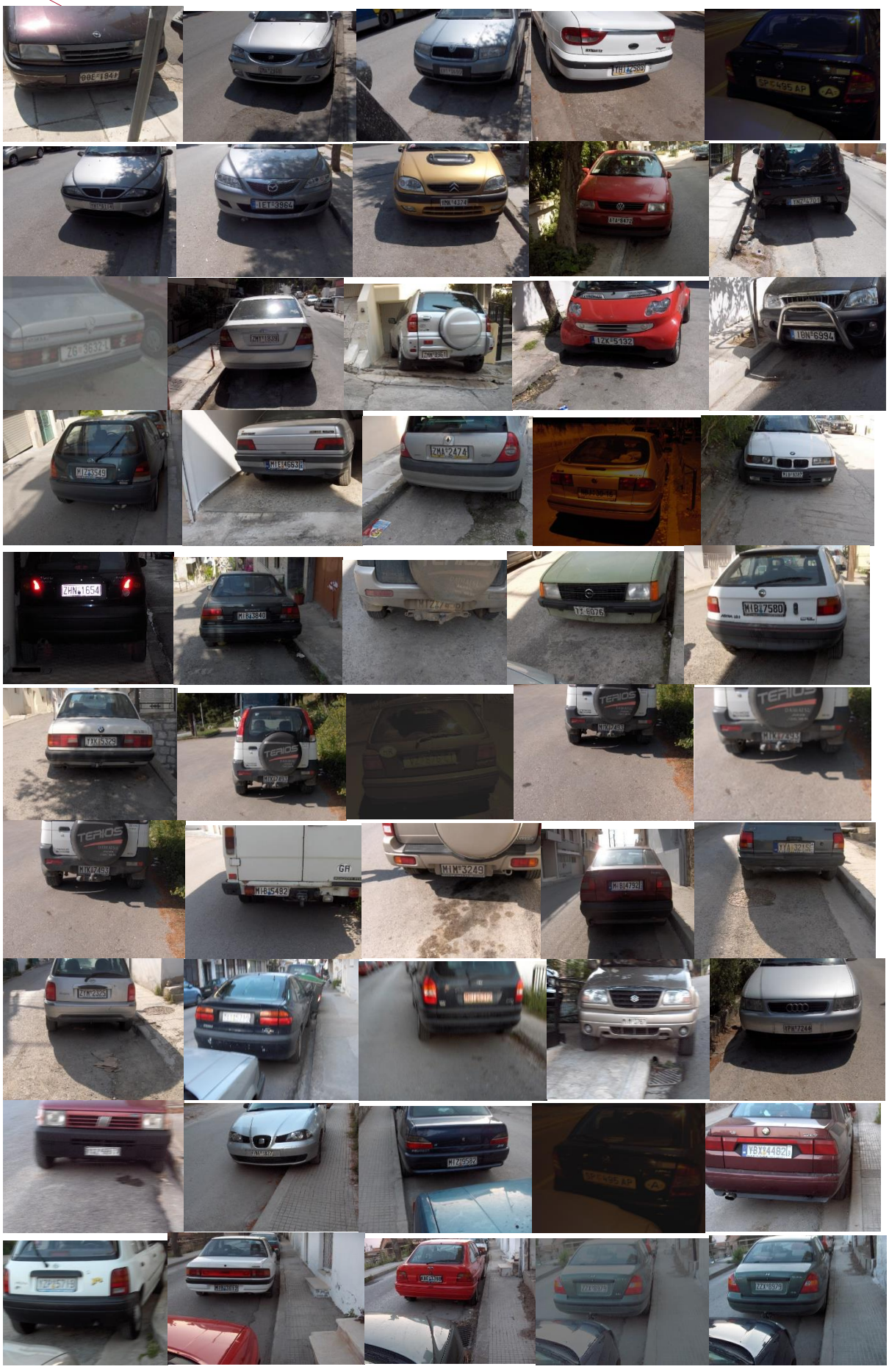

325-J-521
LJ 46-5H
[shoe·be·do]
ZG 3209-F
[shoe·be·do]
ZG 3209-F
ZG 594-TS
VU 3003-VU
PU 781-D
BI RM 160
KR 401-A
ZG 1628-AK
LN 608 AC
FMG 547
ZD 601-J
ZG 394-JU
ZG 692-MZ
ZG 988-TU
ZG 163-DB
ZG 556-KJ
GZ 04214
ZG 415-MA
DU 871-CB
ZG 900-DD
S NZ 1588
ZG 789-PI
VU 253-H
KA 365-CP
OS 728-I
ZG 270-JB
NA 552-C
ZG 256-FZ
F FU 1004
ST 317-KJ
RI 580-KC
NG 179-AC
ZG 293-P
ZG 821-EV
RI 393 BD
SK 646-CA
ZG 400-HZ
ZG 784-FU
ZG 911-CK
ZG 8298-F
KR 531-CD
ZG 567-EF
RI 949-GE
ZG 912-CK
ZG 495-ZR
ZG 776-FB
ZG 521-II

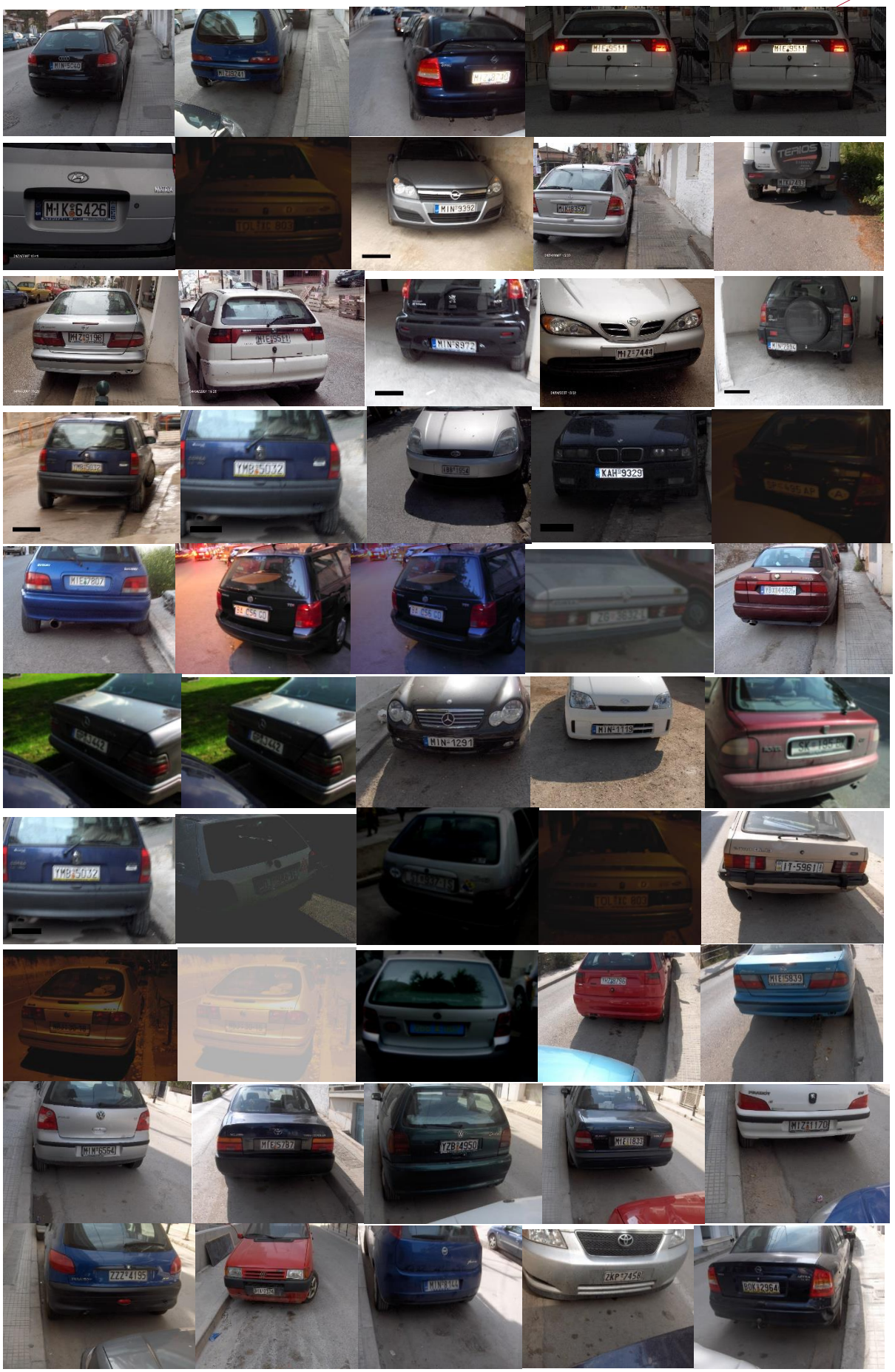

ZG 5670-I
RI 1201-AR
RI 393-BD
ZG 4134-D
ZG 568-TM
ZG 1001-DJ
ZG 120-EL
ZG 505-CE
ZG 4134-D
ZG 119-FH
SI 610-AK
ZG 966-ZT
ZG 210-EH
ZG 655-SC
ZG 367-SO
ZG 248-EB
ZG 5572-AB
ZG 2424-B
ZG 710-KS
ZG 1908-AA
ZG 679-VF
ZG 443-TP
ZG 145-BD
ZG 995-NN
KA 596-CJ
ZG 4949-S
ZG 636-TT
ZG 093-PZ
ZG 1455-AL
ZG 4851-T
OG 788-P
ZG 441-AJ
ZG 5942-G
ZG 120-JL
ZG 8258-H
ZG 790-DO
ZG 160-DU
ZG 759-RR
VU 135-F
ZG 2101-O
ZG 401-CC

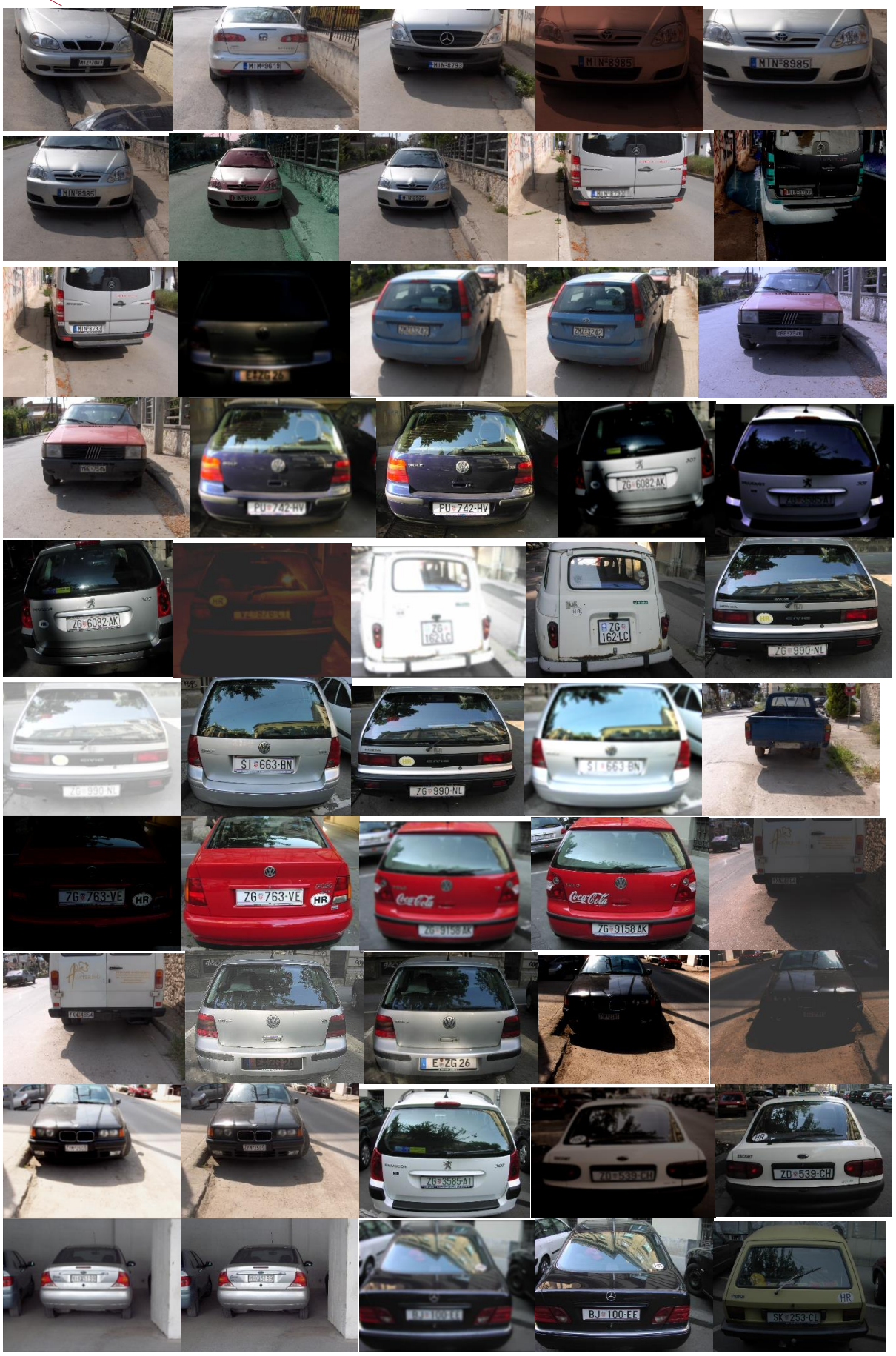

ZG 7949-AD
ZG 106-LU
ZG NIVEA-1
ZG 773-VE
ZG 4940-L
ZG 3710-AL
ZG 707-VK
SI 124-BP
STOPIĆ
ZAGREB, CROATIA
TEL.: 01 6578 733 MOB.: 098 413 003
Čazmatrans
BJ 765-BM
ZG 773-VE
JELCZ
LINZMEIER
7940 Riedlingen
IVECO
T.I.R
KRONE
ZG 5687-AR
SI 582-AM
AUTOŠKOLA
SI 582-AM
INNTALER
TID
ZG 862-NS
SCHMITZ
ZG 862-NS
ZG 1846-AC
ZG 314-U
ZG 2092-O
ZG 7523-AF
ZG 1720-AJ
ZG 5642-AB
ZG 7497-AH
ZG 2828-AJ
ZG 424-RI
MB 40-71P
ZG 094-MM


ZG 3202-AM
ZG 192-OT
ZG 781-LR
ZG 4040-J
ZG 930-BK
DU 993-V
ZG 8010-AF
OG 501-O
ZG 6347-H
ZG 848-IK
ZG 9043-B
ZG 7640-AB
ZG 1929-AH
VU 0081 VU
VU 0081-VU
ZG 548-TL
ZG 2745-AM
KK H5-122
DE 451-K
ZG 4515-AH
SK 402-CK
KC 0010 KK
Tel.:++385-1-33 71 050
Tel./Fax:++385-1-33 73 856
SCHWANZMÜLLER
ZG 727-LI
ZG 5800-K
OS 001-GS
ZD 454-BO
ZG 0013-JP
ZG 301-ZH
KK 23-13L
ZG 636-RG
SI 160-BB
SLO
SG D3-164
ZG 155-MC
BM VB 939
GO13 GC
LXX 7661
YXT 7210
YBK 6508
PZE 6669
ZHM 8787
ZMB 5516
ZZK 5144



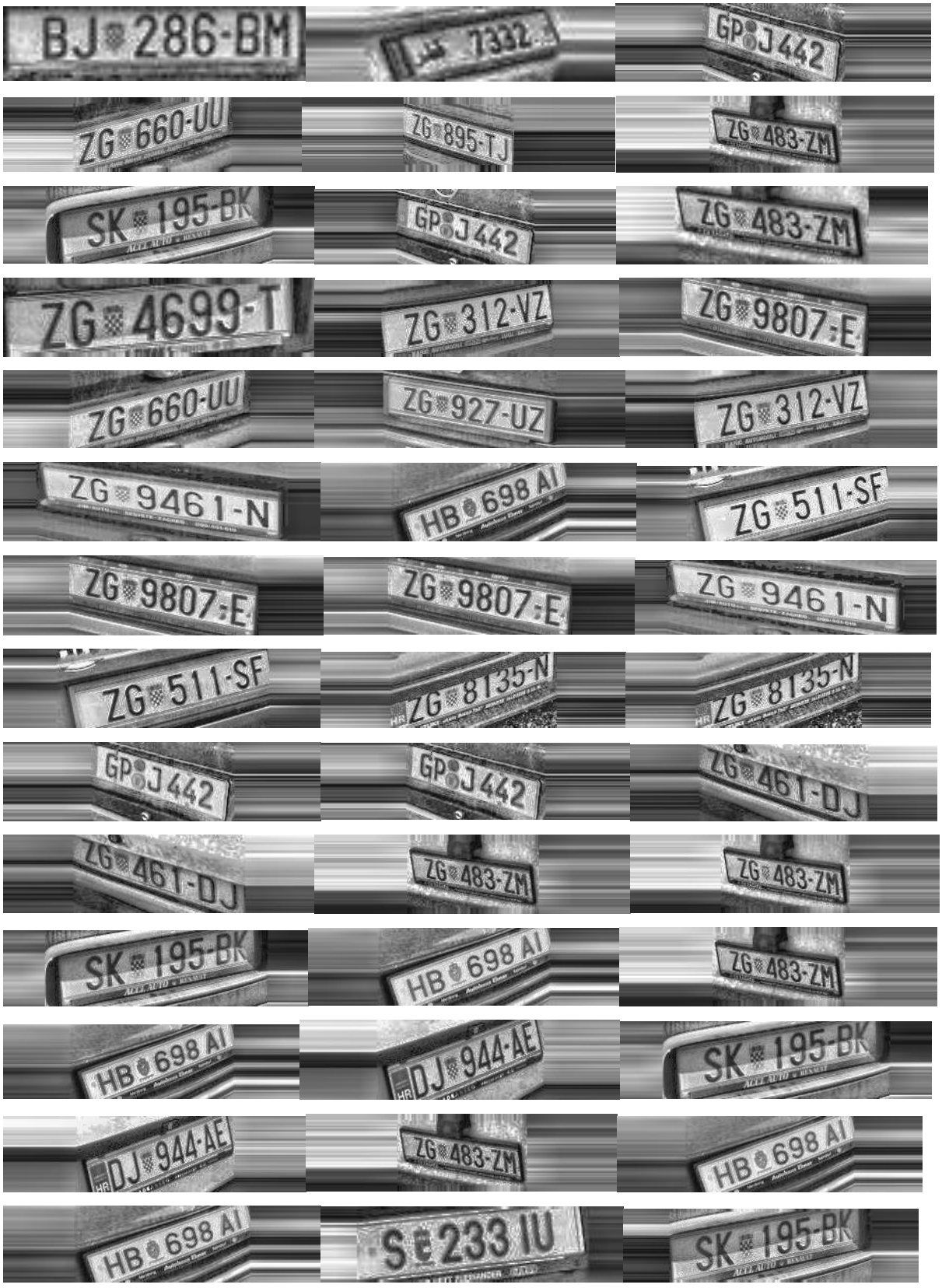





MIN 8985
ZG 6082 AK
PU 742-HV
ZG 162-LC
ZG 990-NL
SI 663-BN
ZG 763-VE
HR
ZG 9158 AK
E ZG 26
ZG 3585 AI
ZD 539-CH
BJ 100-EE
SK 253-CL


ZG 4269 AC
PE 59 FT
PE 59 FT
ZG 637-LV
ZG 877-TF
ZG 877-TF
RI 373-KI
RI 373-KI
RI 681-GN
ZG 884-AM
ZG 884-AM
VZ 0909-MD
SG D3-164
VZ 0909-MD
ZG 655-UR
ZG 637-LV
ZG 655-UR
SK 253-CL
BEJ 53-45
BEJ 53-45
LJ 46-5H
LJ 46-5H
KR 401-A
KR 401-A
SG D3-164
819 KXH 75
819 KXH 75
171 NVX 75
171 NVX 75
ZG 763-VE

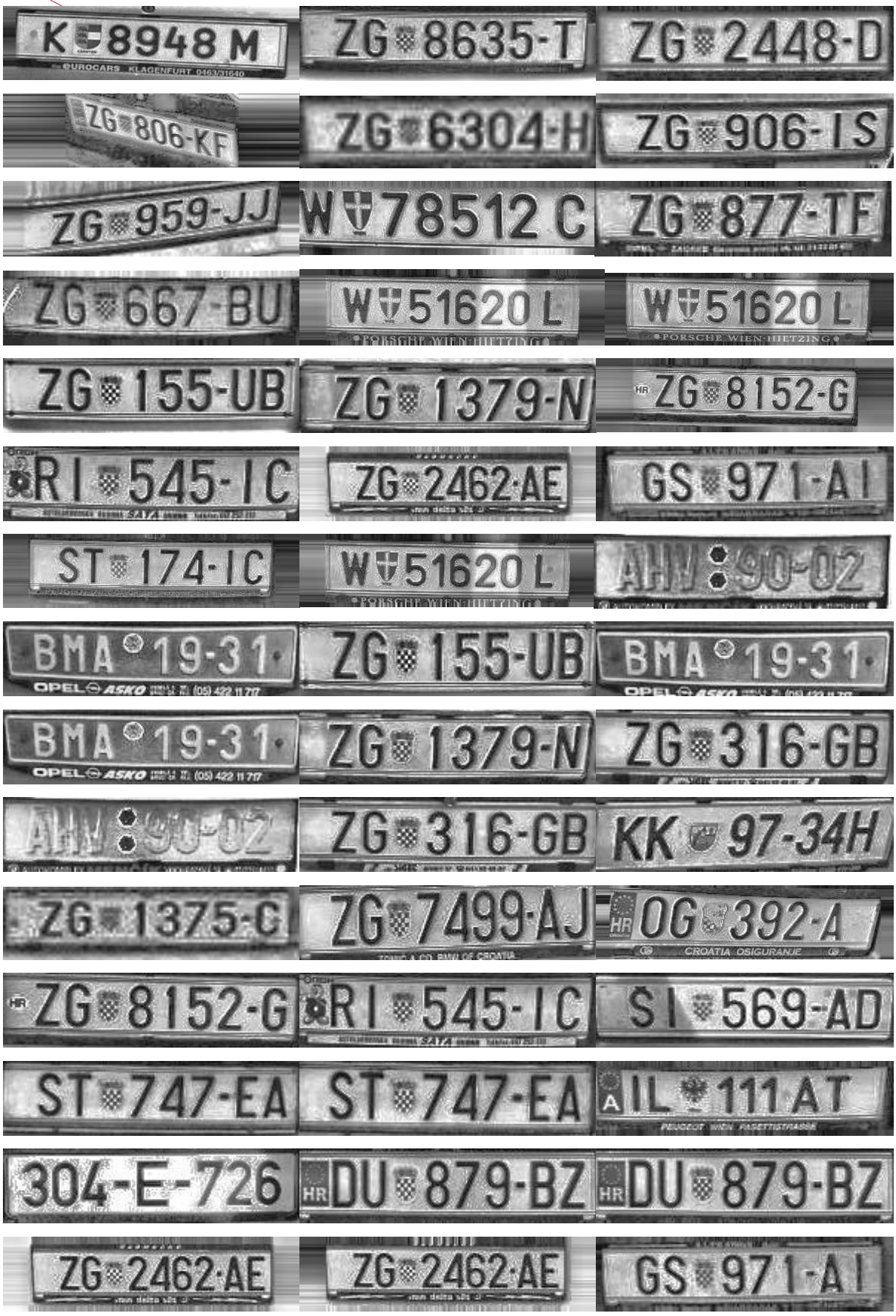
Some samples for real training licences plates images

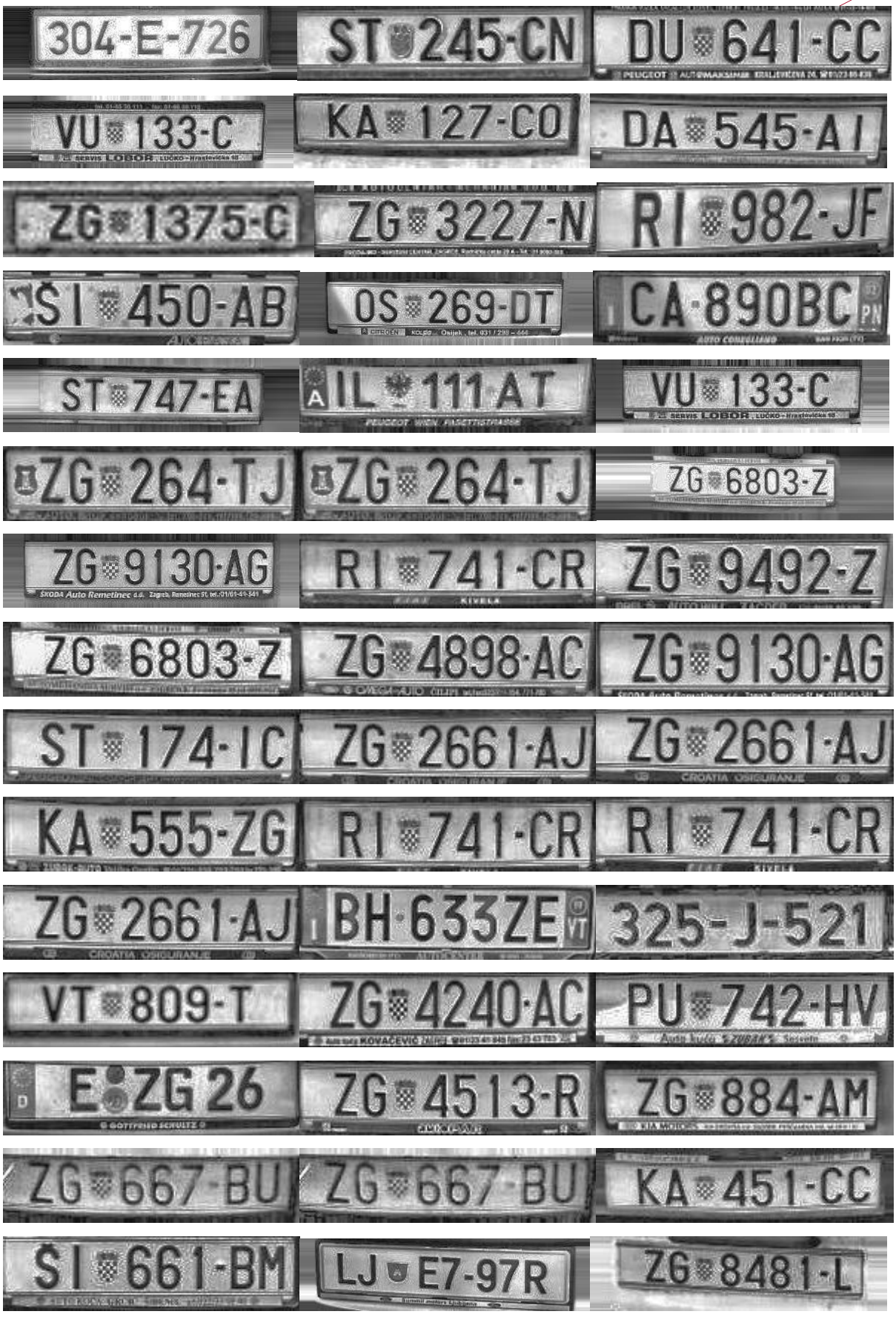


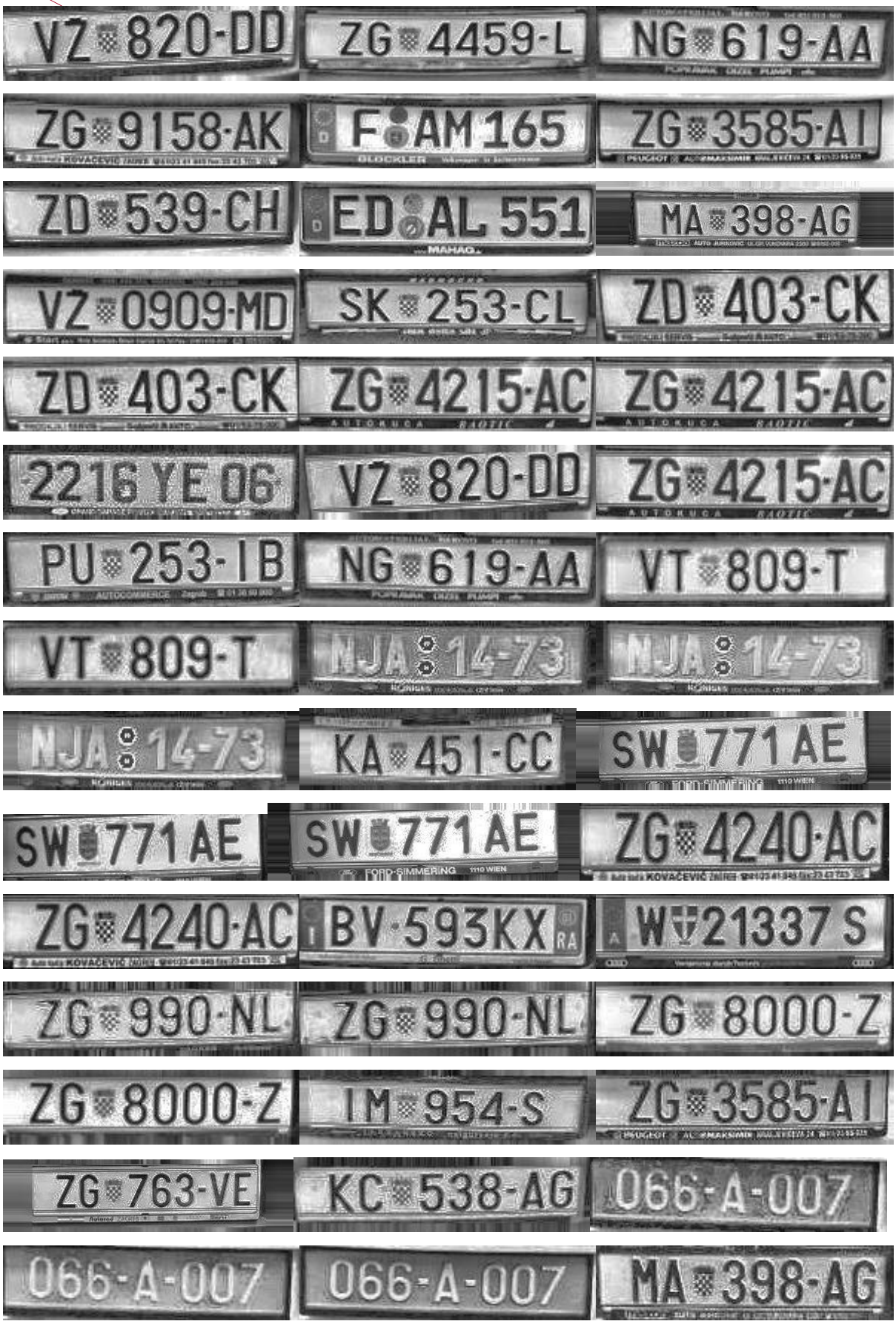

ZG 895-TJ
ZG 895-TJ
ZG 6896-ZG
ZG 4699-T
SK 195-BK
513-J-636
ST 937-IS
ZG 9461-N
ZG 9461-N
ZG 9461-N
ZG 660-UU
ZG 3632-L
ZG 660-UU
ZG 3632-L
ZG 927-UZ
ZG 927-UZ
ZG 956-RZ
SK 195-BK
S E 233 IU
ZG 3632-L
ZG 956-RZ
ZG 483-ZM
ZG 927-UZ
GS 592-AC
NBJ 30-16
LJ V5-44S
LJ V5-44S
AKO 02-81
LL KK 50
NBJ 30-16
TÖL XC 803
LJ V5-44S
VT 418-B
ZG 687-NL
304-63 TC
TÖL XC 803
VT 418-B
LJ V5-44S
AKO 02-81
ZG 687-NL
ZG 687-NL
VZ 876-C1
317-J-939
317-J-939
LL KK 50


NU EC 860
NBJ 30-16
NBJ 30-16
NBJ 30-16
VZ 876-CI
VZ 876-CI
NBJ 30-16
ZG 9005-L
PU 660-BJ
ZG 1382-AB
PU 660-BJ
GS 592-AC
ZG 903-MD
ZG 855-VT
ZG 806-KF
ZG 806-KF
ZG 806-KF
PU 660-BJ
B RJ 193
530-C-001
ZG 9005-L
ZU 178-U
ZG 9005-L
PU 660-BJ
BA 056 CD
B RJ 193
ZG 4208-I
ZG 8901-Z
ZG 484-LD
PZ 898-AO
SP 495 AP
PU 660-BJ
BA 056 CD
ZG 9005-L
530-C-001
ZG 9005-L
BA 056 CD
ZG 4208-I
SP 495 AP
SP 495 AP
AKO 04-60
ZG 806-KF
ZG 806-KF
SP 495 AP
K 8948 M

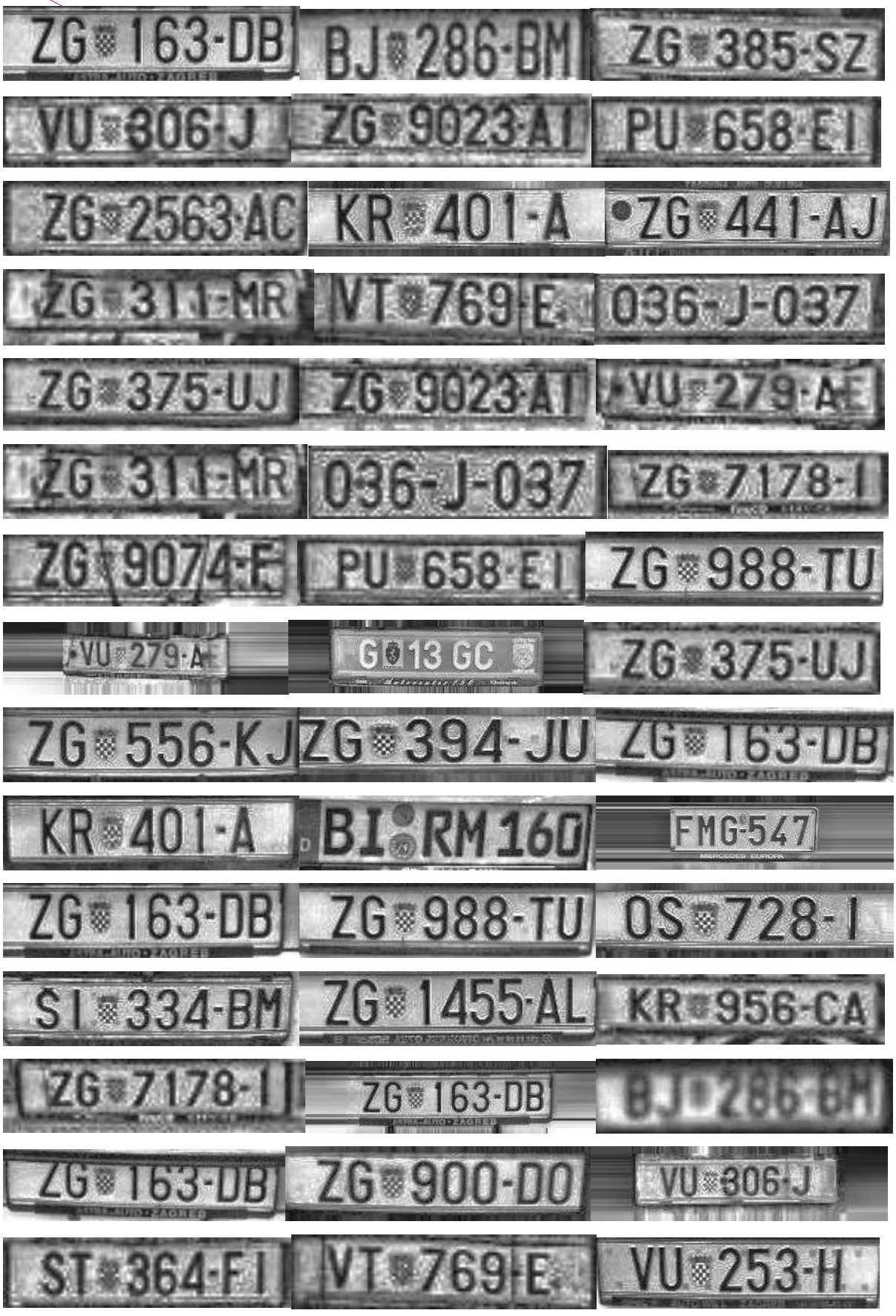

ZG 9005-L
ZG 9005-L
BA 056 GD
BA 056 GD
BA 056 GD
GS 592-AC
ZG 880-TV
ZG 880-TV
ZG 2047-M
ZG 880-TV
ZG 2047-M
ZG 5031-AE
B RJ 193
B RJ 193
ZG 324-BB
ZG 1382-AB
DU 149-AR
ZG 1382-AB
ZG 1382-AB
ZG 1382-AB
ZG 1382-AB
ZG 1382-AB
ZG 1532-AB
AKO 04-60
PU 660-BJ
BA 056 GD
ZG 2047-M
ZU 178-U
AA SH 938
ZG 354-CU
ZG 848-KT
ZG 3083-J
ZG 4499-AD
530-C-001
530-C-001
K 8948 M
ZG 2751-B
ZG 887-K
ZG 5023-B
LJ T7 31F
ZG 2751-B
ZG 887-K
ZG 424-GH
ZG 624-K
VZ 539-AP

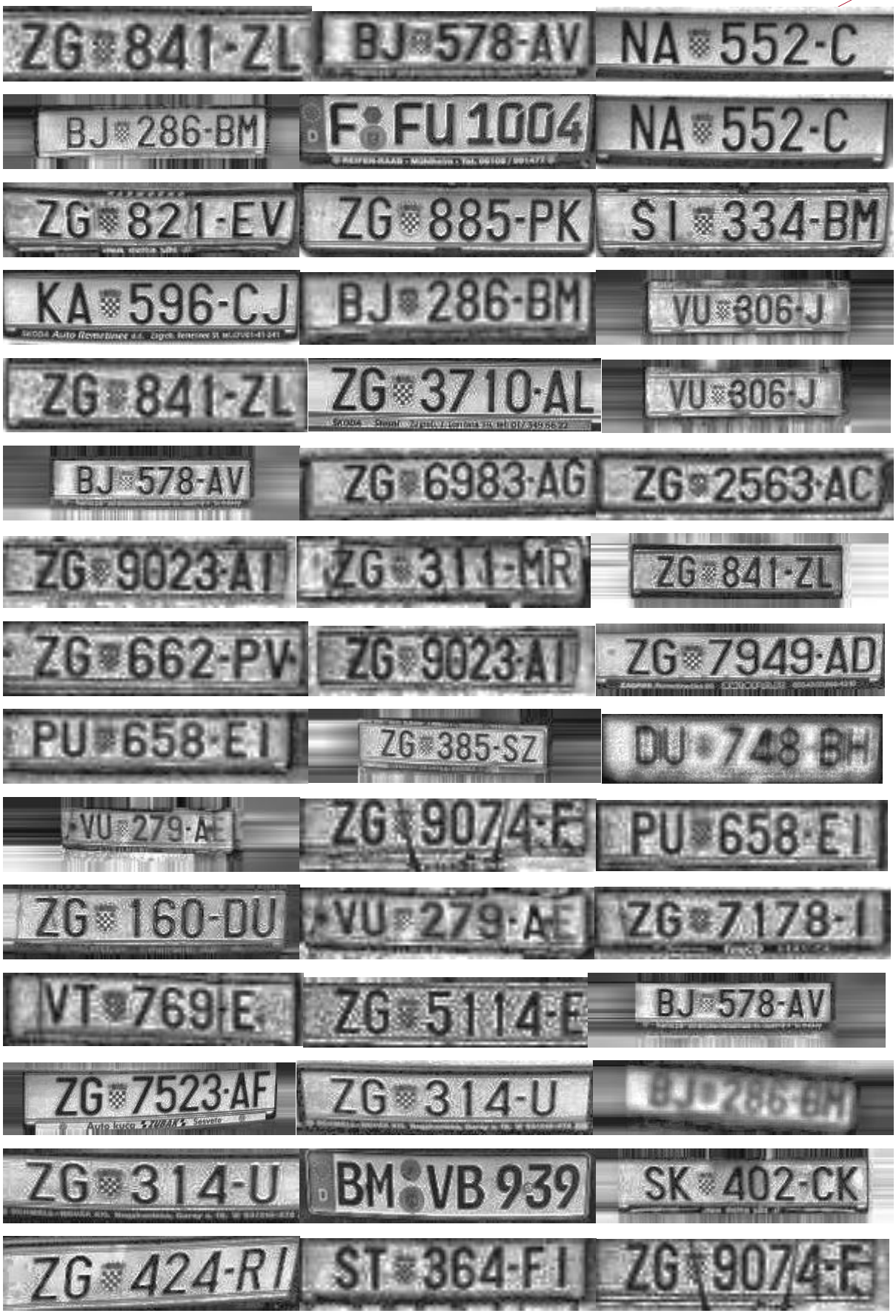

K 8948 M
ZG 8635-T
ZG 2448-D
ZG 806-KF
ZG 6304-H
ZG 906-IS
ZG 959-JJ
W 78512 C
ZG 877-TF
ZG 667-BU
W 51620 L
W 51620 L
ZG 155-UB
ZG 1379-N
HR ZG 8152-G
RI 545-IC
ZG 2462-AE
GS 971-AI
ST 174-IC
W 51620 L
BMA 19-31
ZG 155-UB
BMA 19-31
BMA 19-31
ZG 1379-N
ZG 316-GB
ZG 316-GB
KK 97-34H
ZG 1375-C
ZG 7499-AJ
HR OG 392-A
HR ZG 8152-G
RI 545-IC
SI 569-AD
ST 747-EA
ST 747-EA
A IL 111 AT
304-E-726
HR DU 879-BZ
HR DU 879-BZ
ZG 2462-AE
ZG 2462-AE
GS 971-AI

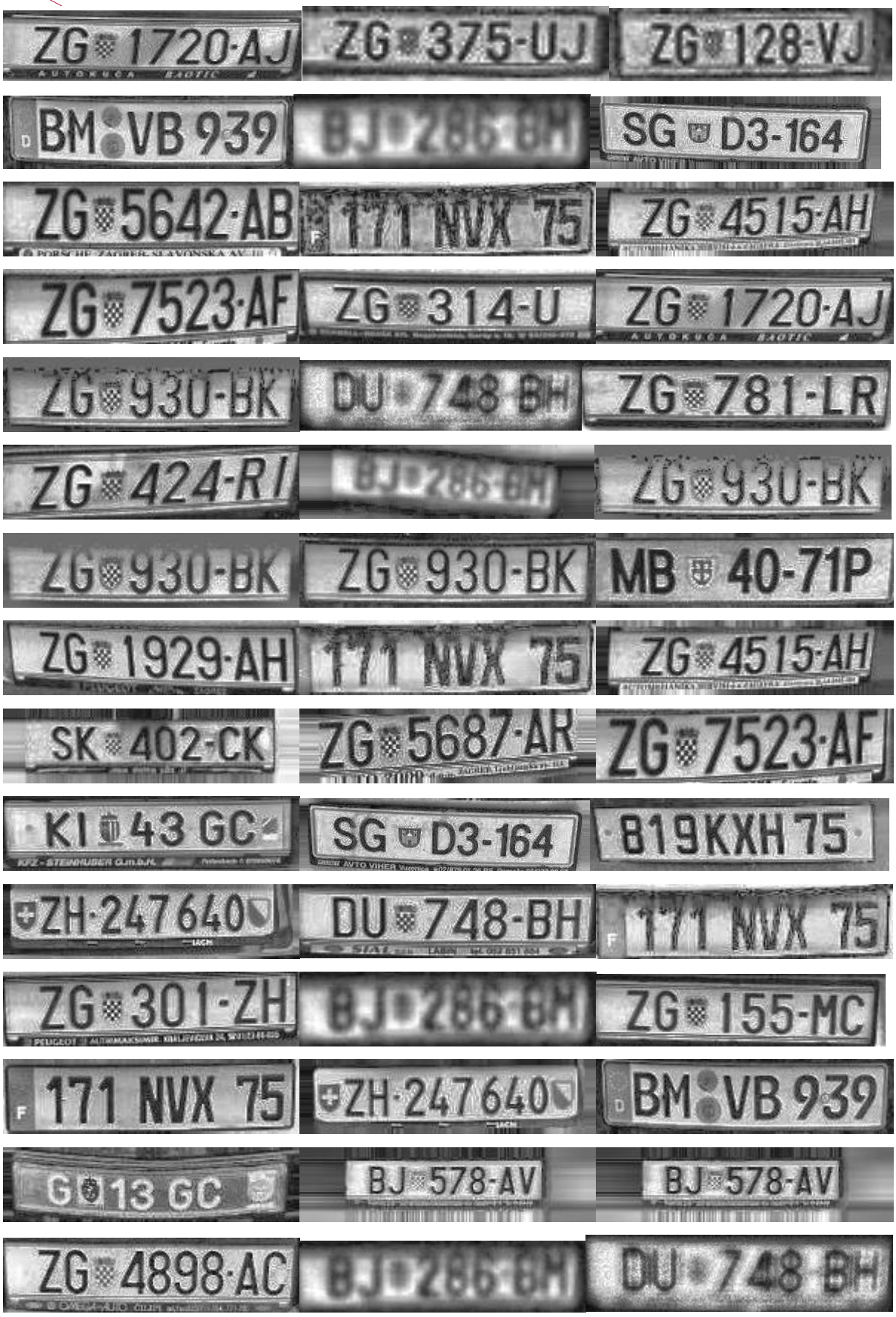

304-E-726
ST 245-CN
DU 641-CC
VU 133-C
KA 127-CO
DA 545-AI
ZG 1375-C
ZG 3227-N
RI 982-JF
ŠI 450-AB
OS 269-DT
CA 890BC
ST 747-EA
IL 111 AT
VU 133-C
ZG 264-TJ
ZG 264-TJ
ZG 6803-Z
ZG 9130-AG
RI 741-CR
ZG 9492-Z
ZG 6803-Z
ZG 4898-AC
ZG 9130-AG
ST 174-IC
ZG 2661-AJ
ZG 2661-AJ
KA 555-ZG
RI 741-CR
RI 741-CR
ZG 2661-AJ
BH 633ZE
325-J-521
VT 809-T
ZG 4240-AC
PU 742-HV
E ZG 26
ZG 4513-R
ZG 884-AM
ZG 667-BU
ZG 667-BU
KA 451-CC
ŠI 661-BM
LJ E7-97R
ZG 8481-L


VZ 820-DD
ZG 4459-L
NG 619-AA
ZG 9158-AK
F AM 165
ZG 3585-AI
ZD 539-CH
ED AL 551
MA 398-AG
VZ 0909-MD
SK 253-CL
ZD 403-CK
ZD 403-CK
ZG 4215-AC
ZG 4215-AC
2216 YE 06
VZ 820-DD
ZG 4215-AC
PU 253-IB
NG 619-AA
VT 809-T
VT 809-T
NJA 14-73
NJA 14-73
NJA 14-73
KA 451-CC
SW 771 AE
SW 771 AE
SW 771 AE
FORD-SIMMERING
ZG 4240-AC
ZG 4240-AC
BV 593KX
W 21337 S
ZG 990-NL
ZG 990-NL
ZG 8000-Z
ZG 8000-Z
IM 954-S
ZG 3585-AI
ZG 763-VE
KC 538-AG
066-A-007
066-A-007
066-A-007
MA 398-AG


MA 398-AG
ZG 4269-AC
ZG 4269-AC
ZG 8000-Z
066-A-007
PE 59 FT
RI 373-KI
ZG 877-TF
RI 373-KI
RI 373-KI
ZG 667-BU
ZG 667-BU
ZG 667-BU
ZG 667-BU
ZG 667-BU
VZ 0909-MD
SK 253-CL
ZD 403-CK
2216 YE 06
HG AS 1802
PU 865-HZ
KZ 386-L
NJA 14-73
ST 329-JH
SW 771 AE
ZG 6082-AK
ZG 990-NL
SI 663-BN
SK 180-BM
ZG 4898-AC
KC 538-AG
BJ 100-EE
ZG 4269-AC
ZG 5715-AK
ZG 877-TF
ZG 8297-I
ZG 7871-S
DU 748 BH
ZG 655-UR
SK 180-BM
BEJ 53-45
325-J-521
LJ 46-5H 02
LJ 46-5H 02
LJ 46-5H 02


ZG 163-DB
BJ 286-BM
ZG 385-SZ
VU 306-J
ZG 9023-AI
PU 658-EI
ZG 2563-AC
KR 401-A
ZG 441-AJ
ZG 311-MR
VT 769-E
036-J-037
ZG 375-UJ
ZG 9023-AI
VU 279-AE
ZG 311-MR
036-J-037
ZG 7178-I
ZG 9074-F
PU 658-EI
ZG 988-TU
VU 279-A
G 13 GC
ZG 375-UJ
ZG 556-KJ
ZG 394-JU
ZG 163-DB
KR 401-A
BI RM 160
FMG 547
ZG 163-DB
ZG 988-TU
OS 728-I
SI 334-BM
ZG 1455-AL
KR 956-CA
ZG 7178-I
ZG 163-DB
BJ 286-BM
ZG 163-DB
ZG 900-DO
VU 306-J
ST 364-FI
VT 769-E
VU 253-H


ZG 841-ZL
BJ 578-AV
NA 552-C
BJ 286-BM
F FU 1004
NA 552-C
ZG 821-EV
ZG 885-PK
SI 334-BM
KA 596-CJ
BJ 286-BM
VU 306-J
ZG 841-ZL
ZG 3710-AL
VU 306-J
BJ 578-AV
ZG 6983-AG
ZG 2563-AC
ZG 9023-AI
ZG 311-MR
ZG 841-ZL
ZG 662-PV
ZG 9023-AI
ZG 7949-AD
PU 658-EI
ZG 385-SZ
DU 748 BH
VU 279-A
ZG 9074-F
PU 658-EI
ZG 160-DU
VU 279-A
ZG 7178-I
VT 769-E
ZG 5114-E
BJ 578-AV
ZG 7523-AF
ZG 314-U
ZG 314-U
BM VB 939
SK 402-CK
ZG 424-RI
ST 364-FI
ZG 9074-F


ZG 1720-AJ
ZG 375-UJ
ZG 128-VJ
BM VB 939
SG D3-164
ZG 5642-AB
171 NVX 75
ZG 4515-AH
ZG 7523-AF
ZG 314-U
ZG 1720-AJ
ZG 930-BK
DU 748 BH
ZG 781-LR
ZG 424-RI
ZG 930-BK
ZG 930-BK
ZG 930-BK
MB 40-71P
ZG 1929-AH
171 NVX 75
ZG 4515-AH
SK 402-CK
ZG 5687-AR
ZG 7523-AF
KI 43 GC
SG D3-164
B19KXH 75
ZH 247640
DU 748-BH
171 NVX 75
ZG 301-ZH
ZG 155-MC
171 NVX 75
ZH 247640
BM VB 939
G 13 GC
BJ 578-AV
BJ 578-AV
ZG 4898-AC
DU 748 BH