Origin-Destination Matrix Estimation of Traffic Flow on Highway Network

Sergio Varela Ramirez Kristian Kovačić

Supervisor: Asst. Prof. Edouard Ivanjko

Technical report: UNIZG-FTTS-ZITS-01-13

July 2013

Department of Intelligent Transportation Systems
Faculty of Transport and Traffic Sciences
University of Zagreb
Zagreb, Croatia

Abstract

Today's road traffic control problems include solving a traffic situation with many congestions, increased traffic demand in peak hours, need for high mobility and fast response in case of an incident. Video sensor or camera combined with state of the art image processing algorithm is more and more becoming the approach to road traffic monitor. Advantage of obtained traffic video footage is that many high level traffic information can be extracted. High level traffic information includes incident detection, vehicle classification, origin-destination matrix estimation, etc. This reports deals with the possibility to track vehicles in a traffic network using license plate recognition. The final results include developed image processing system, vehicle detection and license plate recognition to create an application that fulfills all mentioned requirements.

Extraction of traffic data using computer vision technology is nowadays related to creation of traffic statistics and to improvement of the traffic conditions for drivers. The first development direction has been carried out on two different videos and the data extracted were used to compile traffic statistics on a real world traffic highway and establish the principles of the automatic highway network origin destination estimation that will be developed in the future.

This reports includes presentation of experimental results obtained from real world highway traffic video footage and conclude with a summary of the tasks done and future work proposals.

Contents

1 Introduction

This report is written in the scope of the first authors IAESTE internship at the Department of Intelligent Transportation Systems of the Faculty of Traffic and Transport Sciences University of Zagreb and its volunteer work on the project Computer Vision Innovations for Safe Traffic (VISTA). Project VISTA has been carried out since March 2013 with a planned duration of 24 months. It aims to improve traffic surround view parking assistance, automatic headlight detection, traffic sign detection and recognition, among others with computer technology.

Project participants are the Faculty of Electrical Engineering and Computing (UNIZG-FER) as leading institution and Faculty of Transport and Traffic Sciences (UNIZG-FTTS) as partner institution, both from University in Zagreb.

Most of the research and development work described in this report has been carried out during the internship of the first author of this technical report. Internship was sponsored from the student organization IAESTE and from UNIZG-FTTS. Work equipment was provided from the Department of Intelligent Transportation Systems from UNIZG-FTTS.

The area of research which is investigated is origin-destination (OD) matrix estimation of traffic flow in a highway traffic network, vehicle detection in a highway traffic video footage, license plate recognition and database for manipulation of extracted vehicle data. In order to make the described research, Microsoft Visual Studio 2012 (programming of the needed application in C++), openCV library for image processing and CARMEN® ANPR Engine for plate recognition will be used.

Aim of this research and development is to advance in the project VISTA, investigate procedures of optimal moving vehicle recognition, test accuracy of CARMEN® ANPR Engine in plate recognition process, analyze execution time and finally evaluate implemented application using a real world road traffic video example.

This report is organized as follows. Section 2 describes the problem that has to be solved, and following section 3 describes state of the art approaches used for OD matrix estimation. Section 4 describes applied approach for vehicle detection from road traffic video footage. Section 5 describes the architecture of the developed application. Section 6 shows the obtained experimental results of several test videos and finally section 7 ends this report with conclusion and future work proposals.

2 Problem description

For an effective traffic situation monitoring, first step is to get traffic network a few basic parameters that are needed: (i) distance between vehicles; (ii) length of vehicles; (iii) velocity of vehicles; and (iv) vehicle trajectories. These parameters need to be measured and calculated with great accuracy in order to get useful statistical data. From this basic parameters, more complex parameters like traffic flow, traffic structure regarding vehicle types and road link load can be obtained. In order to estimate this data the developed system needs to use computed features like average speed of each vehicle that has been detected in its travel through the traffic network. Nodes of the monitored traffic where a vehicle has been detected need to be logged. In this case nodes denotes points in the road network where the traffic is being monitored using a video camera. Mostly such points are entry or exit points of a highway network, crossroads or toll plazas.

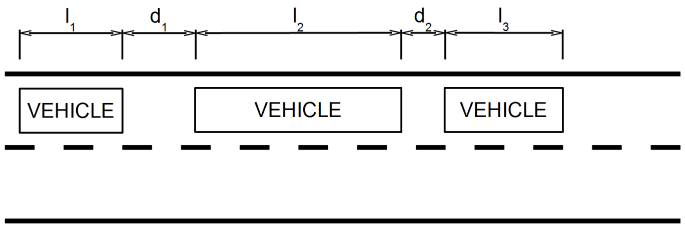

Figure 1: Basic parameters used for traffic monitoring

In Fig. 1, monitored basic parameters, vehicle lenght and distance between vehicles are given. Parameter l_i presents vehicle length in [m] and d_i presents distance between vehicles in [m]. These parameters can be measured with image processing techniques. To calculate dynamic parameters such as the average velocity of vehicles, additional parameter - time needs to be known. In Eq. 1, n is the total number of monitored vehicles and t_i represents a time interval in which vehicle travels distance s_i.

$$avg\ v = \frac{1}{n}\sum_{i=0}^{n} v_i = \sum_{i=0}^{n} \frac{s_i}{t_i} \tag{1}$$

	$Node_1$	$Node_2$	$Node_3$	$Node_4$	
$Node_1$	m_{11}	m_{12}	m_{13}	m_{14}	$\sum_{j=0}^{4} m_{1,j}$
$Node_2$	m_{21}	m_{22}	m_{23}	m_{24}	$\sum_{j=0}^{4} m_{2,j}$
$Node_3$	m_{31}	m_{32}	m_{33}	m_{34}	$\sum_{j=0}^{4} m_{3,j}$
$Node_4$	m_{41}	m_{42}	m_{43}	m_{44}	$\sum_{j=0}^{4} m_{4j}$
	$\sum_{i=0}^{4} m_{i,1}$	$\sum_{i=0}^{4} m_{i,2}$	$\sum_{i=0}^{4} m_{i,3}$	$\sum_{i=0}^{4} m_{i,4}$	

Table 1: OD matrix

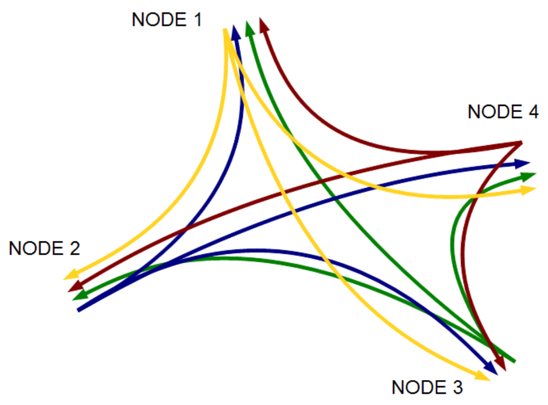

Figure 2: Nodes and connection between nodes

Further processing consists of computing an OD matrix. In this matrix, rows presents input node and matrix column exit node of the monitored highway network. Each cell in the OD matrix represents link between two nodes, where corresponding cell value is the number of vehicles traveled over this link. Nodes are specified by rows and columns. In Fig. 2, traffic network with four nodes is used as an example with the associated OD matrix in Table 1. It has to be noticed that in this case every node is simultaneously an input and an exit node. From this reason the OD matrix given in Table 1. is not symmetrical. Total number of vehicles crossed from or to a node is calculated with summarizing values in the specific row or column. Mentioned parameters represent good foundation for calculation of further statistical information.

3 State of the art approaches

Computer vision gives many approaches in solving of the above mentioned problems. Modern computer's architecture allows fast computation using multi-core CPUs or fast parallel graphical processing units (GPUs). Main classification between today's algorithms used in computer vision can be made based on the platform that they are made for. For image processing, algorithms are executed in most cases on the GPU, as GPU is considered to have a good architecture for sequential processing large amounts of data. In cases where image processing algorithms consists of many branching mechanisms, CPU architecture should be considered for executing algorithms rather then GPU architecture. GPU architecture gives much lower performance when it is executing an algorithm with many branching mechanisms.

On aspect of traffic network monitoring, one of the main processes in computer vision software is object detection and recognition. It can be done by various methods such as foreground/background (FG/BG) image segmentation, edge detection, high level object recognition, etc. Using these methods, more complex parameters can be computed. Extraction of different information from an image (average velocity of a vehicle, distance between vehicles, etc.) can be achieved with algorithms that process and "understand" vehicle physics model.

In Fig. 3, basic principle of FG/BG image segmentation is shown. It is based on separating a foreground image (moving objects) from a background image (static objects). Background image is created by combining a collection of consecutive images. After the background image is created, foreground image is calculated by subtracting the latest image from a collection of previously images stored and the background image [1].

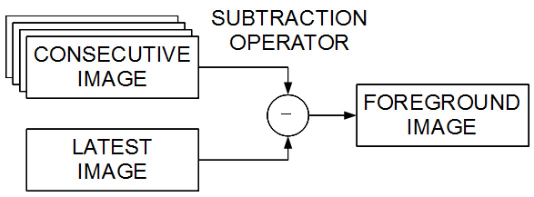

Figure 3: Work principle of an FG/BG image segmentation method.

Unlike FG/BG image segmentation, edge detection methods are based on processing a single image. From an image, edges are extracted based on an algorithm that compares differences in pixel features. Segment of the image where a difference in pixel features is high can be defined as edge of an object [2]. Object recognition methods are useful in cases where a specific object needs to be recognized and separated from other objects (e.g. to separate a specific vehicle type) [3].

4 Vehicle detection

First requirement to detect a vehicle from an image is to separate the vehicle from the rest of the image. After the vehicle is separated, it is possible to extract its additional high level parameters such as position and velocity. In the field of computer vision there are a few basic principles on which algorithms for object detection are based:

1. Separation of dynamic (foreground objects - vehicles and pedestrians) and static (background) segment of an image [1];
2. Calculating optical flow of a specific segments of an image (detection of moving objects) [4];
3. Object detection with algorithms based on Hough method [5];
4. Detection of a vehicle based on algorithms that use modified Haar methods.

From mentioned methods, first and second require to perform calculations on a multiple consecutive images simultaneously. Contrary to this, third and fourth need to perform calculations only on one image. In scope of this work the first method is used i.e. foreground/background image segmentation method.

After foreground/background segmentation is performed, contours on foreground image with moving objects have to be found. Area of contours for each object is calculated. If area is larger than specific threshold value, then it is considered that moving object is a vehicle and further processing is done to extract additional information such as the license plate registration number.

5 Application architecture

The application architecture of the developed application consist of two parts. First the image processing part that is executed in each single node. Figure 4 shows the structure of the image processing performed in order to detect a vehicle and extract its license plate data. The video input is first resized to a smaller size to speed up further processing since HD video footage with corresponding high resolution images are used. Next step is background/foreground segmentation to ensure that the system is able to identify the contours.

Then from all the contours that have been identified, the ones that fit between a minimum and maximum area boundaries are sent to license plate recognition. The developed *Recognize()* method uses CARMEN® SDK to extract vehicle data (license plate number, confidence, license plate coordinates, country and time stamp) and finally store the data in the node database.

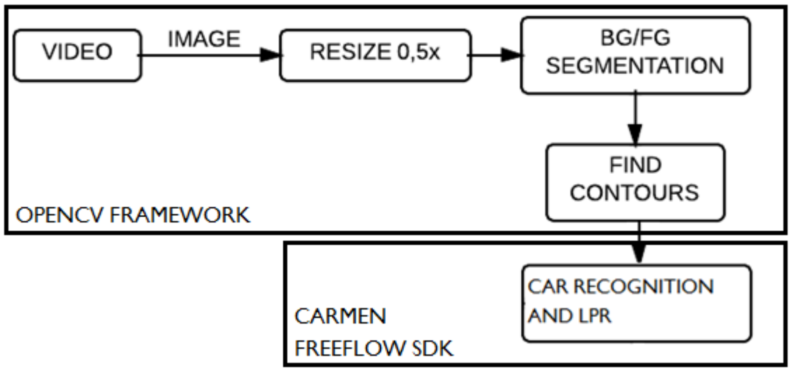

Figure 4: Application architecture

Second part consist of a database which structure depends on the monitored highway traffic network architecture. The vehicle data extracted is stored in a local database that is been setup for each node. In parallel, data related to vehicle plate number, node id and time stamp when the vehicle was registered, is stored in a central database that is been setup for shown in Fig. 5. Regarding the node ID the system can differentiate between input and exit nodes in the the highway traffic network.

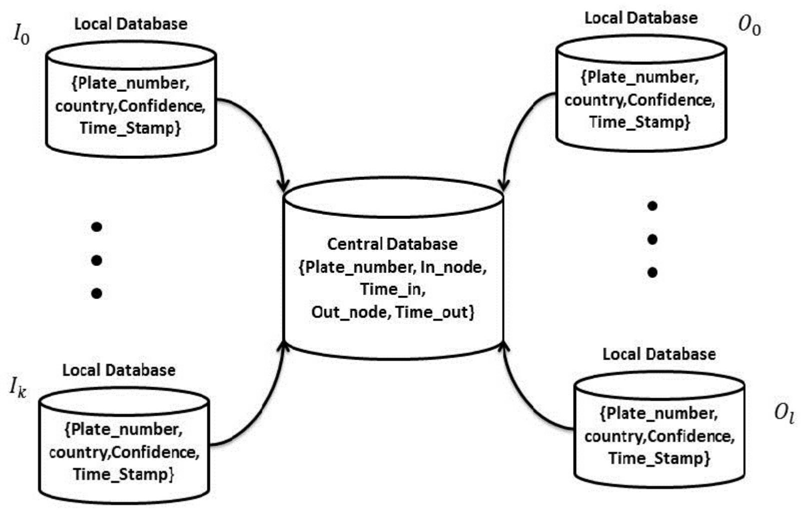
Local Database
I_0
{Plate_number, country,Confidence, Time_Stamp}
Local Database
O_0
{Plate_number, country,Confidence, Time_Stamp}
Central Database {Plate_number, In_node, Time_in, Out_node, Time_out}
Local Database
I_k
{Plate_number, country,Confidence, Time_Stamp}
Local Database
O_l
{Plate_number, country,Confidence, Time_Stamp}

Figure 5: Highway network structure seen from database point of view

6 Experimental results

For the experimental test, two different real world highway traffic videos are used. First video is a traffic node with low traffic flow, used to make a first estimation of how the program should work in a controlled environment. Second video is a node with high and variable traffic flow. With the first estimation set, the second video is used to increase the accuracy in detection and recognition using a real world example highway traffic video footage.

6.1 Highway network

As mentioned above the purpose of a video analysis is to monitor a highway traffic network and to record incoming cars information in the input nodes and outcoming cars information in the output nodes. This car information (plate, country, node, incoming/outcoming time) will be stored in local databases in each node to have redundant information in case to need verification, simultaneously this information will be stored in a central database where a register of the car will be created with plate information, input node, incoming time, output node, outcoming time. This information is necessary to set the coefficients of the highway network matrix. The network will be taken as a black box as shown in Fig. 6.

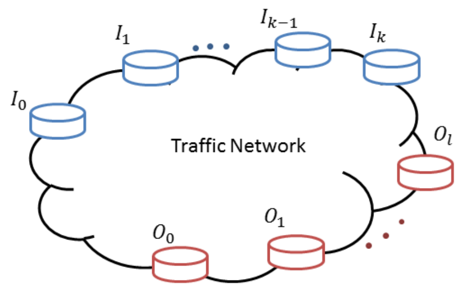

Figure 6: Traffic network with input and exit nodes

First Matlab simulation to test the OD matrix estimation was carried out with 512 car registration plates, four inputs and four output traffic nodes has been made. Discrete incoming times and a random lapse was added to each incoming time as outcoming time and the OD resulting matrix is shown in the Eq. 2, the database is shown in Table 2 and each node plate record is shown in input record in Table 3 and output record in Table 4.

$$T = \begin{bmatrix} 27 & 28 & 40 & 33 \\ 31 & 33 & 35 & 29 \\ 38 & 27 & 26 & 37 \\ 32 & 40 & 27 & 29 \end{bmatrix} \tag{2}$$

Table 2: Example of vehicle registration plate data in central database.

Plate No	T in	T out	Node in	Node out
'OSDA373'	'1'	'1.7598'	'3'	'4'
'ABGMR94'	'1'	'1.2909'	'2'	'1'
'4P75013'	'1'	'1.2774'	'4'	'3'
'1P57957'	'1'	'1.0061'	'1'	'2'
'DEGF1208'	'2'	'2.2425'	'1'	'4'
'CA012BF'	'2'	'2.9367'	'2'	'3'
'DW7487X'	'2'	'2.8602'	'4'	'1'

Table 3: License plates recorded in each input node

InNode 1	InNode 2	InNode 3	InNode 4
'1P57957'	'ABGMR94'	'OSDA373'	'4P75013'
'DEGF1208'	'CA012BF'	'KTKR49'	'DW7487X'
'2E77135'	'MMES222'	'MILMA23'	'BILP272'
'ERE2'	'W85156U'	'LAUAF22'	'EL544CN'
'CB3000T'	'O189TO190'	'CE4727L'	'MJA6034'

6.2 Video footage processing

In the first version of the program *contours* method has been used to select an area of the whole image and forward it to the plate recognition part, to not process all of the possible contours in the image and the detected object must have a minimum area. Then only a smaller part of the whole image is processed in the *Recognize* method, which shows the car plate once this is recognized using the CARMEN® software. After this step also several other features of the car plate as ID, country, confidence, coordinates of recognition, etc. are available. Once the plate is recognized, every five frames the video is evaluated to check the car plate again.

With this base, the aim was to improve the code and keep the compromise between processing speed and accuracy, keeping in mind that the application will run in real time. The starting point was to identify the plate one time and make sure that the plate recognized is correct and matches with the image. To solve these problems in

Table 4: License plates recorded in each output node

OutNode 1	OutNode 2	OutNode 3	OutNode 4
'ABGMR94'	'1P57957'	'4P75013'	'OSDA373'
'DW7487X'	'KTKR49'	'CA012BF'	'DEGF1208'
'BILP272'	'2E77135'	'MMES222'	'MILMA23'
'EL544CN'	'LAUAF22'	'W85156U'	'ERE2'

a first approximation, the plates between two different frames are compared, if there is difference a new car is registered, with the order of entrance, time stamp, plate and country. For the second part CARMEN® software feature *confidence* is used. It gives an extracted license plate accuracy value in a range between 0-100 denoting how much the program is sure that the plate is correct. Taking advantage of this feature a trigger of $40[\%]$ was set as threshold to accept a plate as a correct. Lower confidence values denote an error in the LPR process. For further analysis the confidence value is also registered with car data.

The first approximation explained above worked in the first video example with low traffic flow. In the second example with variable traffic flow, this methodology was not efficient. Due to real world aspects like speed, environmental effect, special cases as car proximity, etc. that intervene in the plate detection and recognition. These aspects have to be taken into account for the program improvement.

To reduce the recognition error due to the speed and car proximity used algorithm has been optimized with the most probable cases obtained during a video analysis. The resulting cases are, two different car plates in two consecutive frames (either correct/correct, error/correct, correct/error) or an error found according to CARMEN® confidence and after a time lapse the same plate is recognized as a correct one which is shown in the Fig. 8.

Regarding to the environment aspects in the video, such as sunlight, shines, camera movements, etc. we thought about two different filters to mitigate the effect of the environment in the plate recognition process. The first filter thought to manipulate brightness and contrast and the second filter to sharpen the image. Influence of the sharpener filter is presented in Fig. 7.

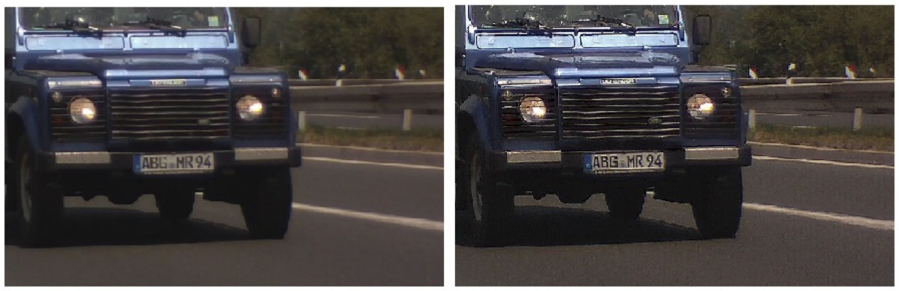

Figure 7: Right image without filter, left image with sharpener filter

Second approach was to use a fix image cut instead of the cut given by the *contours* method. Execution times are within a range in the fix image cut and with the variable image execution times can have high range of values. For this task the video was analyzed to store the coordinates when a plate is recognized Fig. 9. Then the data was processed in Matlab. The criteria to select the fix coordinates (p1(red), p2(blue), p3(green), p4(black)) was to calculate the average (points in yellow) of each point coordinates and then apply it in the algorithm, finally after a experimental test the coordinates were readjusted (squares in cyan) for a optimal recognition. Processing a fix image also means that absolute coordinates(p1(red), p2(blue), p3(green), p4(black)) where the plate was recognized in the Fig. 10 with the average points in yellow. This feature can be exploited to further improvement of the analysis.

Finally a video test with the sharpener filter has been done. Due to contrast and brightness filters are currently inefficient implemented in C++ and without filter and a fix image and anaylising every three frames instead of five, the results in the Table 5 and the excecution times Table 6 represented in *clock ticks*. *Clock ticks* are used as an absolute measure unit, due to known processor speed (3.3 [Ghz]). With 526 total number of cars that are counted and these statistics, the configuration without filters is the most optimal to execute. Time analysis in Fig. 11 shows a correct recognition of correct cars and the interval where wrong cars are detected, between minute 15 and 16 three cars are detected as wrong, in the rest of the time segments one or two cars are detected as wrong or the plate is corrected by the algorithm.

Table 5: Statistics car count results with and without sharpener filter

	Total time evaluated	Total cars count	Correct cars	Mean confidence	Wrong cars	Corrected cars
Analysis with sharpener filter	29m 20s	534	507	70.9633	27	94
Analysis without sharpener filter	29m 20s	532	515	74.7599	17	80

6.3 Estimated traffic data

From the video example, the traffic data shown in Table 7 regarding to number of $\frac{cars}{country}$. Traffic flow can be also estimated in 1088 $\frac{cars}{hour}$.

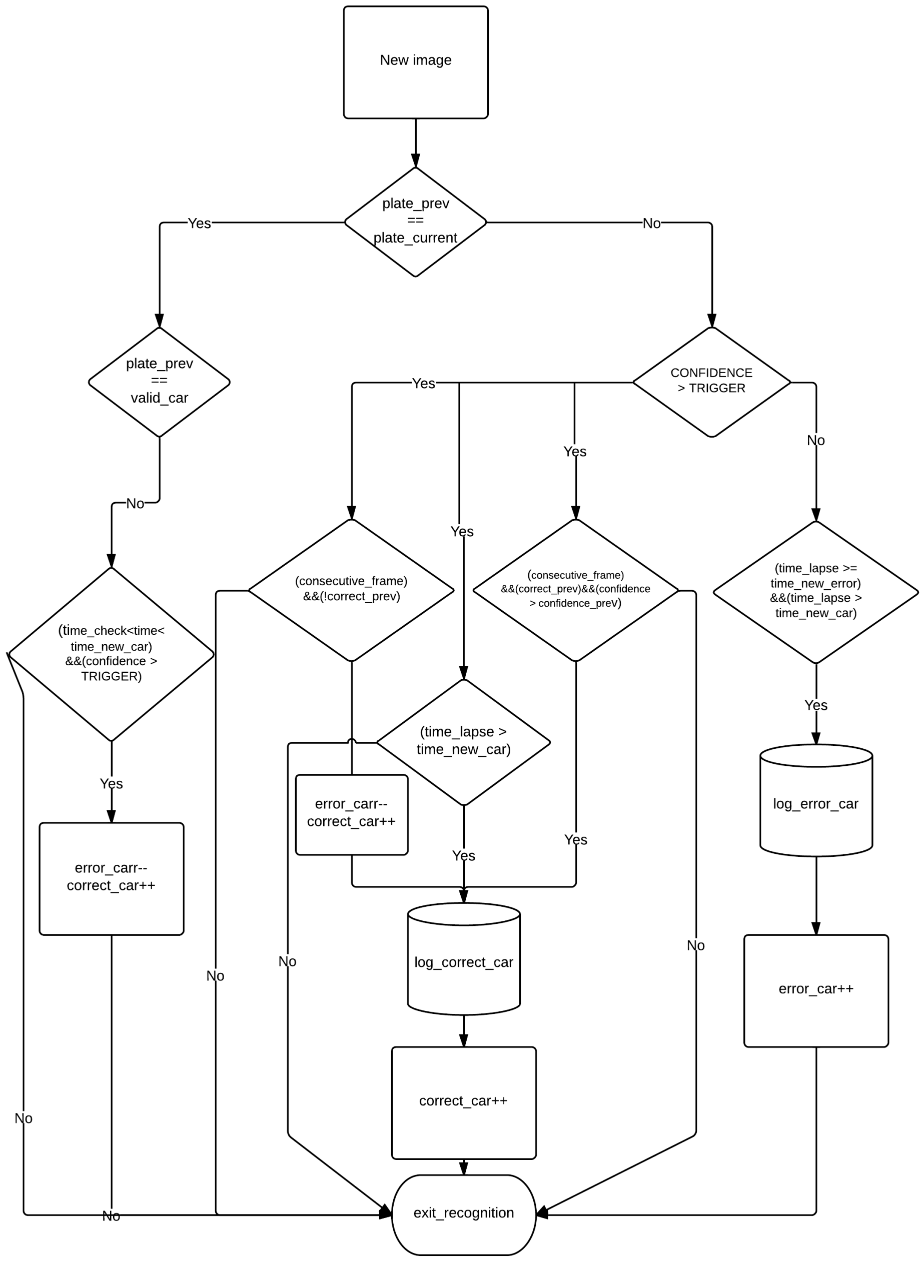
New image
plate_prev == plate_current
Yes
No
plate_prev == valid_car
CONFIDENCE > TRIGGER
Yes
No
(time_check<time< time_new_car) &&(confidence > TRIGGER)
(consecutive_frame) &&(!correct_prev)
(time_lapse > time_new_car)
(consecutive_frame) &&(correct_prev)&&(confidence > confidence_prev)
(time_lapse >= time_new_error) &&(time_lapse > time_new_car)
error_carr-- correct_car++
log_correct_car
correct_car++
log_error_car
error_car++
exit_recognition

Figure 8: Flow chart of the recognize method

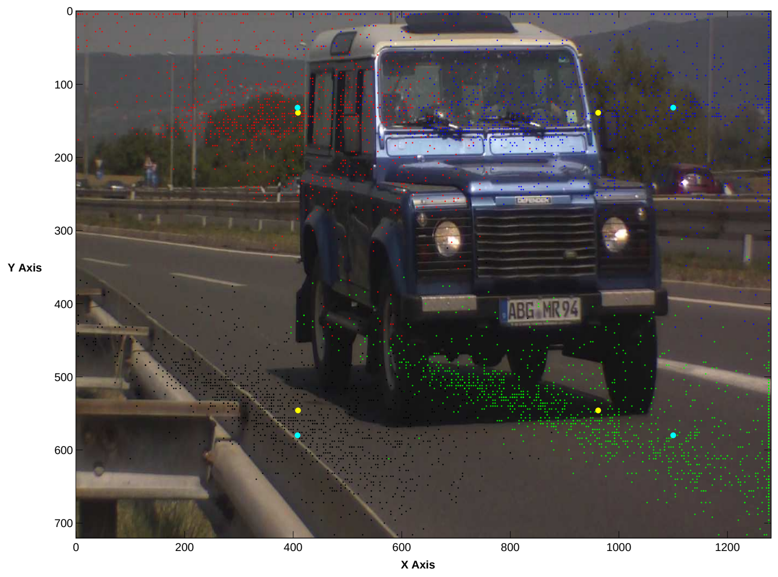

Figure 9: Example of coordinates scatter where images are extracted to recognize the license plate by *CARMEN*.

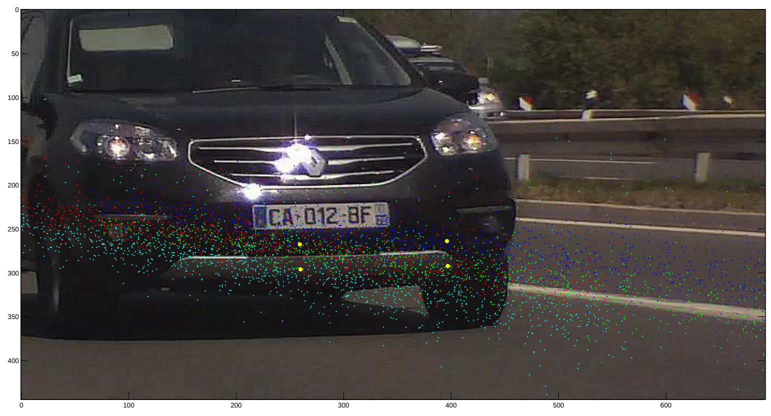

Figure 10: Example of coordinates scatter where the plates are mostly recognized by *CARMEN*.

Table 6: Excecution times with and without sharpener filter

	Contours for loop		Processing Image with Car		Recognize method	
	Max time	Min time	Max time	Min time	Max time	Min time
Analysis with sharpener filter	1718	100	1709	100	1650	50
Analysis without sharpener filter	1218	106	1206	90	1138	34

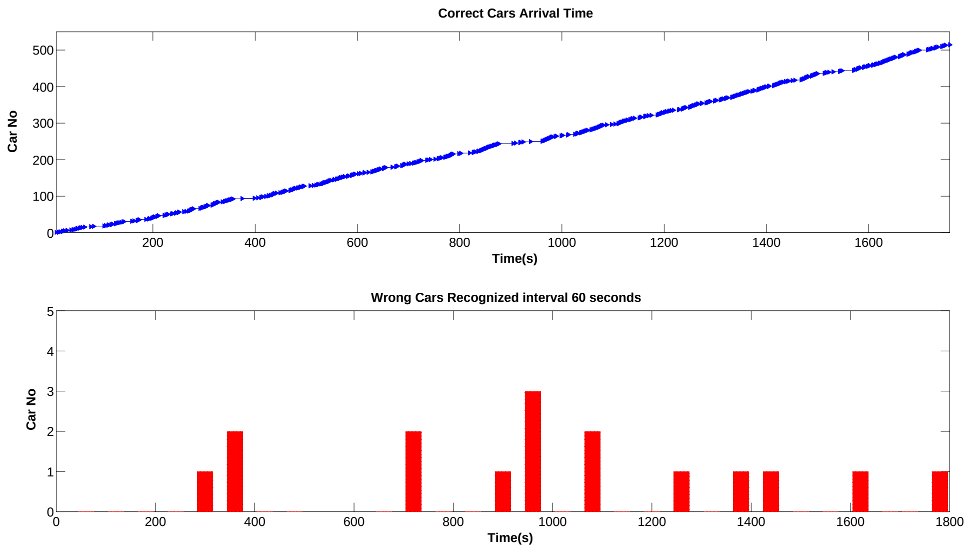

Figure 11: Upper image: Time of correct cars arrival. Lower image: Time statistics of wrong recognition in 60 seconds interval.

Table 7: Traffic data compiled from the video example

Country	Number of Cars	$Percentage[\%]$
Austria	83	15.6
Belgium	6	1.1
Bosnia i Herzegovina	1	0.1
Bulgary	1	0.1
Croatia	47	8.8
Czech Republic	72	13.5
Finland	2	0.3
France	5	0.9
Lithuania	1	0.1
Germany	166	31.2
Hungary	5	0.9
Italy	3	0.5
Netherlands	1	0.1
Poland	88	16.5
Slovakia	11	2.0
Slovenia	17	3.2
Switzerland	2	0.3
Turkey	13	2.4
Unknown	8	1.5
Total Cars	532	100

7 Conclusion and Future work

First to sum up the work done. The traffic network bases and architecture were set. For this purpose a video example of a highway with real world traffic flow was analyzed. From the investigation the data necessary to improve execution times, algorithms and to set constrains to perform the program efficiently were extracted, analyzed and applied in the program.

Execution time was improved, with the hypothesis that the vehicle plates can be recognized in a specific area of the image. Video example was analyzed and the information necessary was extracted to implement the fix image extraction, this reduced the execution time compared with the dynamic image extraction. Second the images that are sent to recognize method at first were saved on a hard disk with jpeg format and then loaded, then they were saved in cache. Performance efficiency was improved from the development of the algorithm, the goal is to register the most correct vehicle plates without repeating or correcting supposed wrong plates. With this objective a traffic plates filter was implemented with the most frequent cases on the real world traffic node. As a result after the internship of the first author, the execution time and the code were optimized with the resources available for the investigation.

For further work various new technologies should be examined. Such technologies include SIMD capability like ones on GPU (CUDA, vertex and pixel shaders, etc.) and CPU (instruction sets of SSE type). Algorithms should support a multicore CPU where they need to work on multiple threads simulatenously to achieve a higher performance.

However the research was done over few samples and it's matter of further investigation to cover more cases, generalize the method to make the fix image size adaptation automatic to all kind of video footages that can be processed. So these cases should be investigated in depth in the future. On the other hand, the line of investigation should go in the direction to create the network with several input and output nodes, manage additional vehicle data (time, positions, distances) to enable computation of additional statistics for the OD estimation and finally perform the described OD estimation procedure in a real time system.

8 Acknowledgment

This project is co-financed by the European Union from the European Regional Development Fund.

The contents of this report is the sole responsibility of the University of Zagreb, Faculty of Electrical Engineering and Computing, Faculty of Transport and Traffic Sciences and do not necessary reflect the views of the European Union.

References

[1] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.

[2] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, 1986.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[4] Pushmeet Kohli Michael W. Tao, Jiamin Bai and Sylvain Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. In *Computer Graphics Forum (Eurographics 2012)*, 200.

[5] J. Matas, C. Galambos, and J.V. Kittler. Robust detection of lines using the progressive probabilistic hough transform. In *CVIU 78 1*, 200.