Performance Tuning Of A Smartphone-Based Overtaking Assistant

Subhadeep Patra, Carlos T. Calafate, Juan-Carlos Cano, and Pietro Manzoni [1]

Abstract— **ITS solutions suffer from the slow pace of adoption by manufacturers despite the interest shown by both consumers and industry. Our goal is to develop ITS applications using already available technologies to make them affordable, quick to deploy, and easy to adopt. In this paper we introduce EYES, an overtaking assistance solution that provides drivers with a real-time video feed from the vehicle located just in front. Our application thus provides a better view of the road ahead, and of any vehicles travelling in the opposite direction, being especially useful when the front view of the driver is blocked by large vehicles. We evaluated our application using the MJPEG video encoding format, and have determined the most effective resolution and JPEG quality choice for our case. Experimental results from the tests performed with the application in both indoor and outdoor scenarios, allow us to be optimistic about the effectiveness and applicability of smartphones in providing overtaking assistance based on video streaming in vehicular networks.**

Keywords— **Android application, real implementation, video transmission, live streaming, vehicular network, ITS.**

I. Introduction

INTELLIGENT Transportation Systems (ITS) are advanced solutions that make use of vehicular and infrastructured networks to provide innovative services related to both traffic and mobility management, and that interface with other models of transport. ITS aims at using the already available transport networks in a smarter manner, resulting in significant coordination and safety improvements. Our goal here is to *integrate smartphones into vehicular networks* to develop ITS applications that can reach out to the masses in a short period of time. The choice of smartphones is not only justified by their wide availability and use, but also because they are evolving towards high performance terminals with multi-core microprocessors packed with sufficiently accurate onboard sensors.

The architecture and application has been developed for the Android platform, and has been named EYES. The minimum requirement of EYES is having Android devices equipped with at least a GPS and a back camera. The application makes use of the camera to record video and transmit it over the vehicular network, thus providing an enhanced multimedia information aid for overtaking. The location information of the vehicles gathered from the GPS is useful since the transmission of the video feed only occurs between cars travelling in the same direction, and always occurs from the vehicle in front to the vehicle travelling behind. The Android devices are to be placed on the vehicle dashboard with the camera facing the windshield, so that a clear view of the road in front and cars coming from the opposite direction can be captured. Once started, the application requires no further user interaction to operate. EYES can be specially useful in scenarios where the view of the driver is blocked by a larger vehicle, or when a long queue of cars is located ahead and the driver wishes to overtake. In this case, it will automatically receive the video stream from the leading vehicle, and play the received feed on screen, thus aiding the driver in deciding the safest moment to overtake.

We have found many different drive safety applications in the literature that are targeted for smartphones, but only a handful aimed at providing visual aids to the drivers, namely SignalGuru [1], CarSafe [2]and iOnRoad [3]. However, none of these smartphone based applications actually provides real-time visual overtaking aids provided by other cars taking advantage of vehicular networks, even though the idea of video-based overtaking assistance systems is not new. Works like the See-Through System [4], which was later improved in [5], although not being targeted for smartphones, are focused on the issue of video-based overtaking assistance. Other related works worth mentioning are [6] and [7], which demonstrate the feasibility of such video-based assistance systems. In [6] authors proposed performance improvements to a video-based overtaking assistant by focusing on codec-channel adaptation issues, whereas [7] focuses on the reallocation of wireless channel resources to enhance the visual quality. Thus, in order to fulfill the need for a visual overtaking assistance application targeted at consumers, we decided to develop an application which, if combined with an existing vehicular network, would require no additional hardware besides a smartphone to operate. The proposed EYES application is targeted at smartphones since we aim at achieving rapid acceptance, and to promote the close integration of smartphones into vehicular networks.

We have evaluated our application in both indoor and outdoor scenarios. The indoor tests consisted of comparing the performance of EYES using different resolutions and quality settings for an MJPEG[2] video stream. MJPEG involves compressing the video stream separately as JPEG [8] images. The encoding format was tested focusing mainly on the delay between capture and playback of the video

[1]Department of Computer Engineering, Universitat Politècnica de València, Camino de Vera S/N 46022, Valencia, Spain. e-mails: subpat@doctor.upv.es, {calafate, jucano, pmanzoni}@disca.upv.es.

[2]http://en.wikipedia.org/wiki/Motion_JPEG

stream. Then, choosing the best resolution and JPEG quality based on the indoor experiments, we have performed outdoor tests using real cars.

The rest of this paper is organized as follows: In section II, we will present an overview of the developed application. Later, in section III, we will present the application modules and some implementation details. The setup used to deploy and validate EYES will be described in detail in section IV. In section V, we will present the preliminary results achieved with the application in both a real testbed and a laboratory environment. Finally, section VI concludes this paper summarizing our contributions.

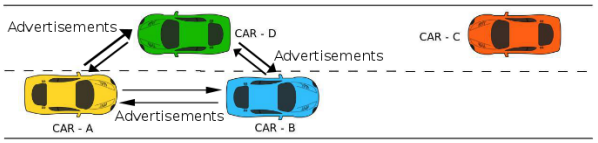
II. Overview of the proposed architecture

The goal of the EYES application is providing assistance during overtaking by streaming real-time video coming from one vehicle to another. The minimum requirements for our application is the availability of a smartphone with GPS and a back camera, along with a vehicular network for video transmission.

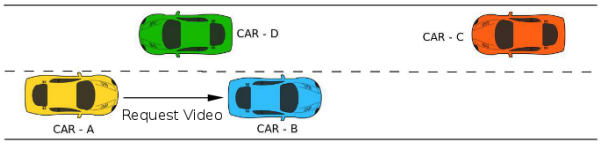
The functionality of EYES can be split in three simple steps for easy understanding. *Step one*, involves *electing the sender and the receiver* of the video which is subject to some special tests and validation conditions. In *step two* the actual *video transmission* occurs between the sender and receiver chosen in phase one. Finally, *step three* is where the application decides to *terminate* the video transmission and playback. This step also involves testing a special condition to stop the video streaming.

In the first step, each device equipped with a back camera and running EYES, *broadcasts* an advertisement containing its location and direction, while they are simultaneously listening for incoming broadcast messages coming from other devices. Whenever a device receives broadcast messages from other devices, it first verifies whether the source of the message is valid. This *validity check* is based on tests which basically involve checking if the source and destination vehicles are traveling one ahead of the other, and in the same direction. For a more detailed description of these validation conditions refer to Section III. If several valid sources are found, the device *requests* video from the best source, which is selected based on the distance between sender and receiver vehicles. The source vehicle, upon receiving the request to send video from the destination vehicle, starts *streaming* the video signal over the vehicular network, in step two. However, before sending the video, the source double-checks the validation conditions used in step one. The destination vehicle starts playing the video onscreen as soon as it starts receiving it. The streaming and playback process is *stopped* in step three, only when the vehicle behind successfully overtakes, or when it stops following the vehicle in-front.

Fig. 1 provides more details about step one. In this example, we have four cars, all of them using our application. CAR-A and CAR-B are travelling



(a) The vehicles exchange *advertisements*.



(b) The client *requests* video from the server.

Fig. 1: Functional overview of EYES - *Step one*.

in one direction, while CAR-C and CAR-D travel in the opposite direction. First, the cars broadcast the advertisement to each other as shown in Fig. 1a. Since CAR-C is not within the range of any other car, nobody is able to communicate with it. Each car, upon receiving the advertisement, performs the validity checks to see if the sender of the advertisement is travelling ahead in the same direction and lane. In this case, only CAR-A finds the advertisement message from CAR-B to be valid, and thus requests video from it, as depicted in Fig 1b.

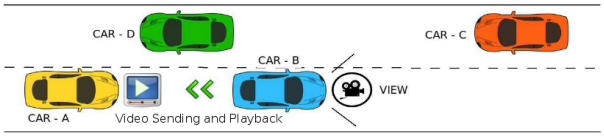


Fig. 2: Functional overview of EYES - *Step two*.

Similarly, Fig. 2 shows that CAR-B, upon receiving the video request from CAR-A, rechecks the validity conditions and starts streaming the video. CAR-A starts receiving the video stream and plays it onscreen for its driver. It may be noted here that a device can act both as video source and destination. This is because, while a device is receiving video from another device, it may also be streaming its own video capture to a completely different device.

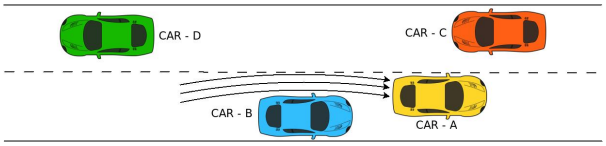


Fig. 3: Functional overview of EYES - *Step three*.

Fig. 3, shows that CAR-A has overtaken CAR-B, and this causes the video transmission to stop. Now, CAR-B may request the video feed from CAR-A which is now travelling ahead, and all the steps above would be repeated in that case.

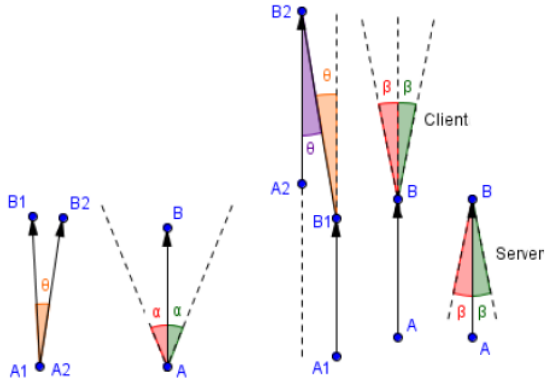
III. Implementation details

From the previous section we already know that the functionality of EYES can be split into three

Message Type	From → To	Message Contents
Hello	Server → Client	Location and Direction
Request	Client → Server	Location and Direction
Ready	Server → Client	Video sender port
Reject	Server → Client	-
Data	Server → Client	Location, Direction and Speed
Data-Ack	Client → Server	-
End	Client → Server	-

TABLE I: Messages exchanged between the Server and Client.

steps. Also, a device running it can act as both server and client at the same time, receiving video from another device while streaming video to a completely different device. In this section we consider two devices out of which one will be streaming and the other just receiving. The device sending the video is considered as the server, while the receiver acts as a client. Although the server and client roles are not established at the beginning of *step one*, we will use the words server and client to refer to the devices that will be attaining the respective role in the future for the sake of clarity.



(a) Same direction test. (b) Same lane test.

Fig. 4: Validation conditions used to initiate video streaming.

In *step one*, the server starts advertising the availability of the video feed by broadcasting a *hello* message which contains its location information. The client, upon receiving it, can determine if the server is ahead and travelling in the same direction with the help of some validity checks. Fig. 4 shows the proposed validity tests that take place during *step one*. It includes the *same direction test* and the *same lane test* conditions. The *same direction test* is used to detect whether two vehicles are travelling in the same direction. For understanding the *same direction test*, let us assume we have two cars, one travelling from point A1 to B1, and the other one from A2 to B2, as shown in Fig. 4a. Notice that, even if two cars are travelling in the same direction at a same speed, its it hard for them to have an overlapping displacement vector; in other words, the angle between the two vectors is not 0. This can happen due to different driving styles and GPS errors. Thus, we measure the angle θ between these two vectors and compare it to a predefined threshold α. If θ is less than α, we can safely assume that the two vehicles are travelling in the same direction. Now, even if two vehicles are travelling in the same direction, it does not necessarily mean that one is ahead of the other, both vehicles may be travelling on different lanes or parallel roads altogether. To check if one is following the other one on the same lane, we perform the *same lane test*, and for this purpose we draw an imaginary line joining the current locations of the two vehicles, as shown in fig. 4b, where B1 and B2 are the current locations. Then, we measure the angle of intersection of this line joining the points B1 and B2 with the displacement vectors of the vehicles. When the measured angle of intersection θ is less than a predefined angle β, then the vehicles are considered to be travelling on the same lane. Being on different lanes will result in a higher value of the measured angle θ, and the *same lane test* will fail. If these two conditions are satisfied, then the two vehicles are assumed to be travelling in the same direction, one following the other.

The client, on receipt of valid advertisements from various servers, chooses the best one based on its distance to the server, and then tries to connect to the chosen server by sending a *request* message. The server, upon receiving the *request* from the client, rechecks the validity of the client by performing the *same direction* and *same lane* tests once again before sending a *ready* or *reject* message, which denotes whether it is ready to send the video being captured by its camera. Table I, details the packet types exchanged between the server and the client.

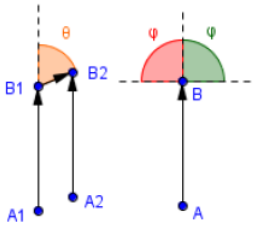


Fig. 5: Overtake test to terminate video streaming.

If the reply from the server was a *reject* message, the client tries to contact a different server. Otherwise *step two* is initiated, which involves starting video streaming and playback, at the server and client ends, respectively. Beside streaming video, the

server, during this period, keeps sending a *data* message every second. The *data* message contains the location, direction and speed information of the vehicle where the server is located. This way, its corresponding client can check whether an overtake has occurred. To find out if an overtake was successful, the *overtake test* is evaluated, as shown in fig. 5. This test is similar to the *same lane test*, the only difference being that the angle θ measured here is the other linear pair of the angle of intersection between the displacement vector and the line formed by joining the current location of the two vehicles. Also, the threshold φ used here is usually a much larger value. Upon receiving the *data* message from the server, the client, if it still has not overtaken as suggested by the *overtake test*, replies to the server with a *data-ack* message to keep the video connection alive.

If the *overtake test* detects that overtaking has occurred, *step three* takes place, and so the client can request the server to terminate the video stream by sending an *end* message. The server can choose to stop video streaming upon receipt of the *end* message from the client, or if the waiting time for a *data-ack* from the client expires. This waiting time is used to detect cases of eventual disconnections. In our implementation we have fixed this waiting time to 3 seconds, which is adequate to detect disconnections, especially when considering that all communications occur between two cars, one just ahead of the other.

IV. Creating the vehicular network

For proper operation, the developed application assumes the availability of a vehicular network, although the vehicles we use on a daily basis still lack the capability to communicate with one another. So, for testing EYES, we equipped cars with GRCBoxes [9] inside them. GRCBox is a low cost connectivity device based on a *Raspberry Pi*[3] which enables the integration of smartphones into vehicular networks. It was developed mainly due to the difficulty in creating an adhoc network using smartphones. Another important feature provided by GRCBox is the support for V2X communications. The different networks supported by the GRCBox include adhoc, cellular and Wifi access points, among others. Thus, we use the adhoc network support of the GRCBoxes to create the required network for EYES.

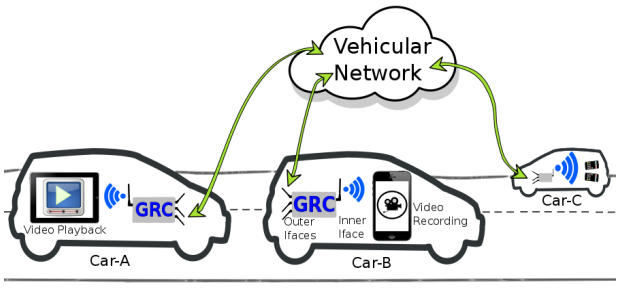


Fig. 6: EYES working together with GRCBox.

[3]https://www.raspberrypi.org

Fig. 6 shows how the application works when combined with GRCBox. Each car within the vehicular network has a GRCBox mounted. The smartphones of the passengers within the car are connected to the GRCBox, which is equipped with Wifi-enabled USB interfaces to communicate in adhoc mode, creating a vehicular network. Even though GRCBox is supposed to be equipped with 802.11p for vehicular communication, we used 802.11a devices instead as 802.11p-enabled hardware was not available while setting up the GRCBox to perform the tests. In future experiments we intend to use 802.11p compatible hardware to take advantage of the WAVE standard.

As shown in fig. 6, Car-B is ahead of Car-A, and both of them are travelling in the same direction and running EYES, so the smartphone in Car-B starts recording the video autonomously and sending it to Car-A, relying on the vehicular network created using the GRCBoxes available within the cars. Concerning the video, it is played on the device in Car-A as soon as video reception starts.

V. Application Validation

For validating EYES, we performed tests in both indoor and outdoor scenarios. The indoor tests consisted of various analysis to choose the best resolution and JPEG quality settings for our application. The outdoor tests, on the other hand, involved testing our application and evaluating the various conditions for initiating or terminating video streaming, using real cars driven around the Universitat Politècnica de València. Each car was equipped with a GRCBox to create the required vehicular network, and the Android devices used were a Nexus 7 and a Samsung Galaxy Note 10.1 (2014 Edition). The Nexus 7 from Google was powered by a quad-core 1.2 GHz processor, ULP GeForce GPU, 1 GB RAM and 1.2 MP camera. The Samsung Galaxy Note 10.1, on the other hand, was equipped with a quad-core 1.9 GHz plus quad-core 1.3 GHz processors, 3 GB ram, 8 MP primary camera and 2 MP secondary camera.

A. Delay requirements

The most important factor to determine the proper functioning of a driving assistance application based on streaming real-time video, is the *delay* between video capture and playback. To calculate an admissible value of delay between video capture and playback, let us assume two cars travelling in the opposite direction on a road located in a densely populated area where the possibility of accidents is much higher because roads tend to be more crowded. We know that the maximum speed limit on such roads is usually about 50 km/h. Assuming the worst case where both cars are travelling at maximum speed limit, the *relative velocity* (V_R) can be calculated using the formula:

$$V_R = V_A + V_B$$

where V_A and V_B are the velocities of the two cars travelling in the opposite direction, V_R is found to be 100 km/h or 27.778 m/s.

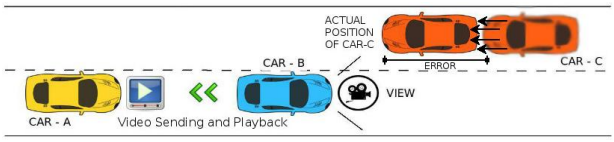


Fig. 7: Error due to delay.

Since there is a delay involved between video capture and playback, the car coming from opposite direction will be in fact closer than the position shown by the application. Fig. 7 demonstrates such a situation, where CAR-A is receiving video feed from CAR-B which shows the position of CAR-C. However, due to the delay involved, CAR-C is located at a position much closer than shown in the video feed. Now, if the allowable error in the position of the vehicle coming from the opposite direction is limited to 10 meters, as displayed by the application, then the *maximum allowable delay* would be *0.36 seconds* in accordance with the equation: $time = distance/speed$. So, in the results that follow, we must make sure that such maximum delay requirement is met.

B. Indoor tests

MJPEG is a simple video compression format since the video stream is compressed separately as JPEG images. Thus, when talking about compression-ratios, the performance of MJPEG is limited. So, our decision to choose a particular resolution and JPEG quality cannot depend solely on delay as throughput is also an important criteria when considering MJPEG. Hence, we first start by calculating the throughput of MJPEG video for different resolutions.

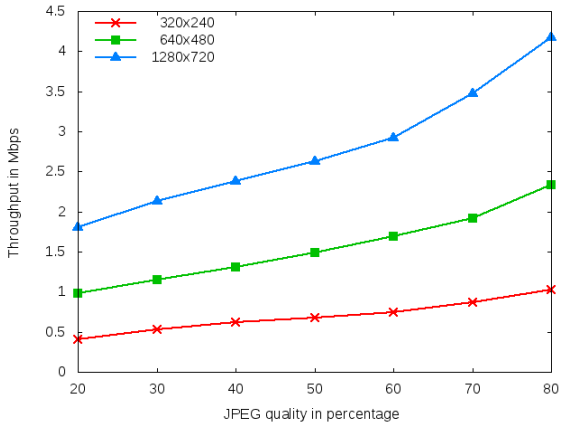


Fig. 8: Variation of throughput with JPEG quality for a 10fps MJPEG video.

In Fig. 8, the number of frames per second for the MJPEG video stream was fixed at 10 because we believe that a 10fps video is sufficient for our application. Also, we adjusted the quality of the JPEG in the video stream in the range from 20 to 80 percent, since for lower values the video quality was too low, whereas a JPEG quality of more than 80 percent did not show any significant improvements in the perceived quality. From the figure we can observe that, for a resolution of 320x240, the average throughput varies from 0.405 to 1.029 Mbps. For 640x480, it lies between 0.976 to 2.336 Mbps, and in case of a 1280x720 resolution, it ranges between 1.805 to 4.177 Mbps.

The proper functioning of the EYES application is largely dependent on the availability of a vehicular network, which in our case has been created using GRCBoxes. Thus, the selection of the resolution and JPEG quality to be used by the application depends on the bandwidth provided by the vehicular network. From our experiments with GRCBox we found that it is capable of providing a mean bandwidth of 10.5 Mbps for TCP traffic, and 15.5 Mbps for UDP traffic, although the worst value for both TCP and UDP was close to 5.5 Mbps. Since, an Android device with our application installed can simultaneously act as video source and destination, the effective bandwidth available for one-way video transmission in a worst case scenario is 2.75 Mbps. At a data rate of 2.75 Mbps, all the different combinations of resolution up to HD with JPEG quality up to 50 percent, as suggested by the Fig. 8, can be supported by the vehicular network created using GRCBoxes.

Now that we have identified the resolutions and JPEG quality pairs supported by the vehicular network that we created using GRCBoxes, the next step is to find out the delay between video capture and playback for each of these combinations to select the best settings for our application.

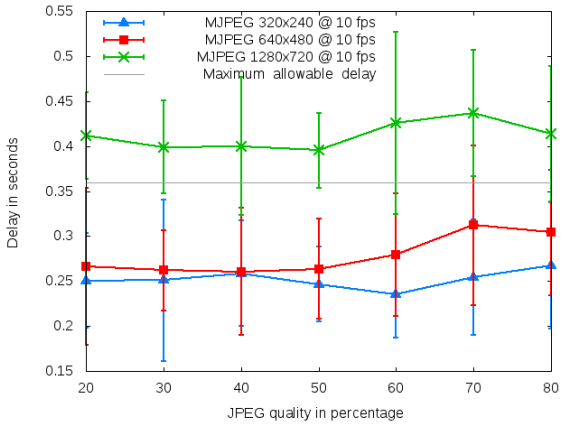


Fig. 9: Delay comparison of different resolutions for MJPEG video.

Fig. 9 shows the delay comparison of different resolutions for MJPEG video stream. We observe that, for a resolution of 320x240, the average delay suffer minimal variations (from 0.24 to 0.27 seconds), whereas for 640x480 it ranges from 0.26 to 0.31 seconds. Eventually, for a resolution of 1280x720, we see that the mean delay lies between 0.4 and 0.44 seconds, becoming excessive for our purposes.

Next, we want to select the most appropriate res-

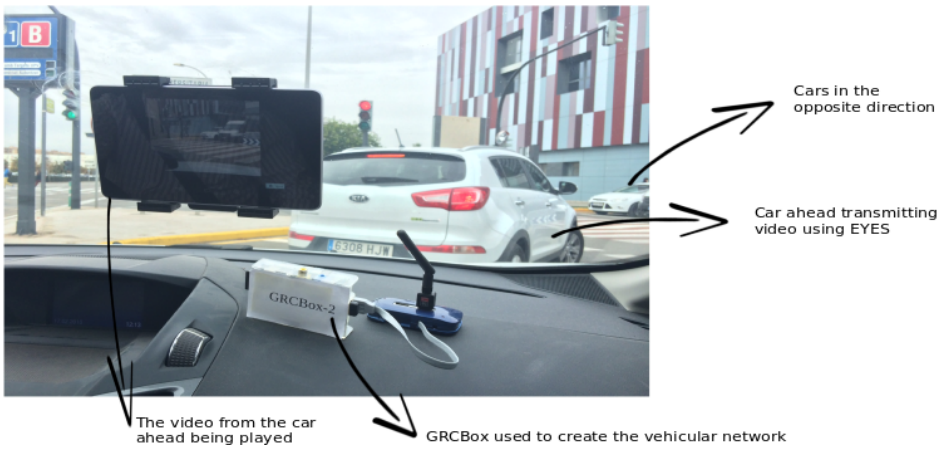


Fig. 10: The experiments with the application in a real scenario.

olution and JPEG quality for the MJPEG video stream for use in the scope of our application. However, we have previously seen that a delay of more than 360 ms is unacceptable for our real-time visual overtaking aid. Consequently, despite considering that vehicular networks created using GRCBoxes would allow us to support resolutions up to HD with a JPEG quality up to 50 percent, we chose to use the MJPEG compression scheme for the resolution of 640x480 at 10fps with JPEG quality set to 80 percent for the video stream, a choice mainly associated to the delay restriction.

C. Outdoor tests

In our developed architecture, the three important conditions on which the EYES application is dependent were described in Section III, and each of these conditions, namely *same direction test*, *same lane test* and *overtake test*, were dependent on threshold values. We have performed a wide set of tests in a real scenario, and our aim was to evaluate reasonable values of the threshold angles α, β and φ, for two cars where one follows the other throughout the whole experiment while travelling along a particular route, so that there is non-stop streaming of video between them. More details regarding the outdoor tests can be found in [10].

Fig. 10 shows a photo taken during one of the *outdoor tests* [4]. In this picture, we can see that the front car is trying to take a right turn, and the back car is receiving the video from the car ahead and playing it onscreen. While doing our outdoor tests with the application, we collected the various angles used in the three different validation tests. Below we can see the graphical representation of the data obtained during the experiment.

Fig. 11 shows the density plot of the angles measured by the *same direction test* at the client side. Most observations for the *same direction test* lies within 20 degrees, which is satisfactory. It is also noticeable that many peaks occur due to GPS errors, also because the route followed had a lot of turns and

[4]https://www.youtube.com/watch?v=jrIWbFjN3Hw

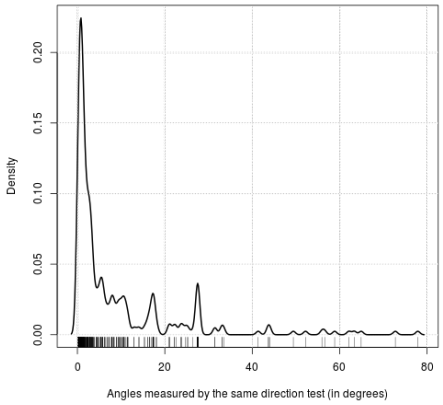


Fig. 11: Results of the *same direction test*.

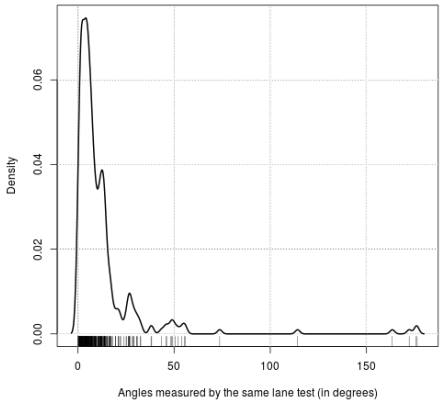


Fig. 12: Results of the *same lane test*.

curves, and so the two cars were not always moving along a straight path.

Fig. 12 shows the density graph of the *same lane test* for the client. From this particular plot, we can see that most observations for the *same lane test* also lie within 20 degrees. Notice that this value is too high considering that this test is very sensitive, and used to detect if cars are travelling on the same lane, and so we find that this condition may not be

too useful when considering the accuracy of current technology. Recall that the *same direction test* and *same lane test* are relevant when starting the video streaming, and are evaluated by both the sender and the receiver, but only data from the receiver (i.e. the client) has been plotted in Figs. 11 and 12 as similar values have also been obtained at the server end.

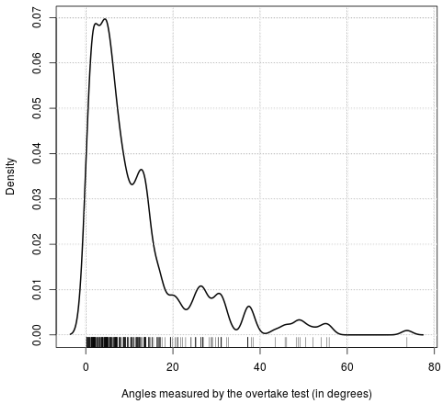


Fig. 13: Results of the *overtake test*.

Finally, fig. 13 shows the density plot for the observations of the *overtake test*. Note that, in order to simplify the graph analysis, the values used are: $180° - \theta$, where θ represents the measured angles in the *overtake test*. The *overtake test* is only evaluated by the client, and we have used its data to produce the graph. We find that the results from this test were pretty much what we expected since all plotted values are below 90 degrees.

VI. Conclusions

In this paper we have presented EYES, a driving safety application that is able to help drivers in safe overtaking. The system provides a real-time video feed captured by the smartphone installed in the vehicle ahead, and which is streamed to the smartphone of the driver seated in the car behind, where the video is displayed without user intervention. Thus, it provides drivers with important information and helps them to decide whether it is safe to overtake. The developed application works correctly and was tested using the MJPEG video encoding format, and best results have been achieved for a resolution of 640x480 with JPEG quality set to 80 percent. We have also evaluated the different test conditions used for starting and stopping autonomous video capture, and found that thresholds of 20 degrees for the *same direction test*, and 90 degrees for the *overtake test* are reasonable. We acknowledge the fact that combining smartphones with vehicular networks indeed opens a new horizon for ITS applications. In the future, we will focus our attention on improving our application by evaluating different alternatives for the *same lane* test, which includes, among others, incorporating image processing techniques for license plate recognition, to assist the client in choosing the correct video server.

Acknowledgments

This work was partially supported by the *European Commission* under *Svāgata.eu*, the Erasmus Mundus Programme, Action 2 (EMA2) and the *Ministerio de Economía y Competitividad, Programa Estatal de Investigación, Desarrollo e Innovación Orientada a los Retos de la Sociedad, Proyectos I+D+I 2014*, Spain, under Grant TEC2014-52690-R.

References

[1] Emmanouil Koukoumidis, Margaret Martonosi, and Li-Shiuan Peh, "Leveraging smartphone cameras for collaborative road advisories," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 5, pp. 707–723, 2012.

[2] Chuang-Wen You, Nicholas D Lane, Fanglin Chen, Rui Wang, Zhenyu Chen, Thomas J Bao, Martha Montes-de Oca, Yuting Cheng, Mu Lin, Lorenzo Torresani, et al., "Carsafe app: Alerting drowsy and distracted drivers using dual cameras on smartphones," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 13–26.

[3] "iOnRoad official website," `http://www.ionroad.com/`, Accessed: 2015-02-8.

[4] Cristina Olaverri-Monreal, Pedro Gomes, Ricardo Fernandes, Fausto Vieira, and Michel Ferreira, "The See-Through system: A VANET-enabled assistant for overtaking maneuvers," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 123–128.

[5] Pedro Gomes, Cristina Olaverri-Monreal, and Michel Ferreira, "Making vehicles transparent through V2V video streaming," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 2, pp. 930–938, 2012.

[6] Alexey Vinel, Evgeny Belyaev, Karen Egiazarian, and Yevgeni Koucheryavy, "An overtaking assistance system based on joint beaconing and real-time video transmission," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 5, pp. 2319–2329, 2012.

[7] Evgeniy Belyaev, Pavlo Molchanov, Alexey Vinel, and Yevgeni Koucheryavy, "The use of automotive radars in video-based overtaking assistance applications," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 3, pp. 1035–1042, 2013.

[8] Gregory K Wallace, "The JPEG still picture compression standard," *Communications of the ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[9] Sergio M Tornell, Subhadeep Patra, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni, "GRCBox: Extending Smartphone Connectivity in Vehicular Networks," *International Journal of Distributed Sensor Networks*, 2014.

[10] Subhadeep Patra, Javier H Arnanz, Carlos T Calafate, Juan-Carlos Cano, and Pietro Manzoni, "Eyes: A novel overtaking assistance system for vehicular networks," in *Ad-hoc, Mobile, and Wireless Networks*. Springer, 2015.