5G-CAGE: A Context and Situational Awareness System for City Public Safety with Video Processing at a Virtualized Ecosystem

Pedro E. Lopez-de-Teruel
Universidad de Murcia
pedroe@um.es

Manuel Gil Perez
Universidad de Murcia
mgilperez@um.es

Felix J. Garcia Clemente
Universidad de Murcia
fgarcia@um.es

Alberto Ruiz Garcia
Universidad de Murcia
aruiz@um.es

Gregorio Martinez Perez
Universidad de Murcia
gregorio@um.es

Abstract

In this article we present 5G-CAGE, an ongoing project aimed to deploy a city safety solution that enables monitoring and analytics of video streams collected from distributed sources of a Smart City. Unlike current proposals based on inflexible architectures or limited networks, 5G-CAGE leverages 5G's high throughput and low latency, as well as its enhanced dynamism and adaptability with advanced virtualization-based technologies. In this context, 5G-CAGE defines a virtualization-enabled solution called City Object Detection (CODet), which allows recognizing interest objects in safety related situations, such as vehicles (e.g. license plates or brands), obstacles in emergency settings, or human faces recognition, to name a few. It can process multiple streams collected from fixed and moving cameras used as a distributed visual sensing system, adequately combining image processing and computer vision algorithms in a virtualized ecosystem. This paper presents initial tests in the specific task of locating and recognizing vehicle license plates, where the CODet virtualized solution has been successfully integrated and tested in the 5GINFIRE platform, an EU-funded project which provides a playground wherein new components, architectures, and APIs may be tried and proposed before being ported to 5G networks.

1. Introduction

Existing computer vision solutions in public safety scenarios are currently implemented to operate with TCP/IP protocols running on dedicated wired or wireless networks. Nevertheless, porting these solutions to take advantage of the upcoming 5G technologies is an open issue currently under investigation, whose relevant proposed approaches are still being tested in lab environments [12]. The evolved, faster, and more robust 5G networks can enable more connected devices and autonomous systems and applications, thus opening a whole new set of opportunities for the Public Safety industry. Video surveillance and new generation Public Safety Networks (PSN) are intrinsically tied together, as the number of distributed devices such as static or moving cameras, recorders, and wearable visual sensors keeps growing in the Smart Cities, while technologies for automatic (non-human driven) scene interpretation constantly progress. Thus, with 5G the amount of data potentially available for public safety applications can tremendously increase, mostly in terms of video and audio streaming, location information, and context-awareness. This huge mass of data produced from heterogeneous sources opens clear opportunities to be used in a more coordinated and autonomous way. Computer vision, big data analytics, and machine learning algorithms can then be used to generate machine-based knowledge which can drastically speed up response times for public safety interventions (e.g. fires and crimes).

In a Smart City context, leveraging on the expected large number of 5G connected citizens (with multiple personal devices) and IoT sensors, PSN and applications need to evolve towards more flexible and scalable approaches by following the Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) principles [16]. In fact, SDN/NFV platforms for 5G networks allow dynamic and fully virtualized deployment, automated orchestration, and configuration of specialized virtual functions implementing scene monitoring, video/audio transcoding, data analysis, and object detection. Existing approaches on the public safety management still lack the flexibility and dynamism to efficiently manage PSN in highly dynamic scenarios, as the ones related to Smart Cities [6].

Our main paper contribution in this context is the 5G-

2473-9944/19/$31.00 ©2019 IEEE
DOI 10.1109/ICCVW.2019.00336

CAGE[1] proposal, which is able to deploy and test a PSN solution that, analyzing video streams collected from heterogeneous and distributed sources in a Smart City, aims to provide advanced features for early detection of specific objects related to city safety. To this end, 5G-CAGE offers a new Virtual Network Function (VNF), called City Object Detection (CODet), which accesses live video streams available from all sorts of moving cameras and visual sensors connected to the network edge, either in city or automotive environments, and provides monitoring and analysis of features of such video streams. This virtualization-enabled city safety solution is able to recognize and detect given objects such as vehicle license plates and brands, obstacles in emergency settings, and human faces recognition, to name a few. As a first testing standpoint, the CODet VNF has been integrated and tested successfully in the 5GINFIRE virtualized platform [1], which is initially devoted to detect vehicle license plates for city safety purposes, e.g. police staff interested in certain vehicle license plates of interest.

The remainder of the paper is structured as follows. Section 2 references previous works on 5G computing and computer vision techniques relevant to the targeted PSN scenario. Section 3 describes in thorough detail the 5G-CAGE deployment, while Section 4 details the experimental results. Finally, conclusions and future works are drawn in Section 5.

2. Related work

It is a well-known fact that the number of visual sensors connected to the network has grown exponentially in the last years [2]. With 5G, the amount of available data can increase tremendously, mostly in terms of video and audio streaming, location information, and context-awareness [14]. Big data analytics and machine learning algorithms can be used to exploit these large amounts of data but, in order to scale the computation adequately, advanced techniques for distributing the processing work must be adopted to avoid overload and achieve maximum efficiency, following the so called *edge computing* paradigm [15].

In the more specific field of computer vision tasks, which can be considered of interest in public safety, one of the key applications is Automatic License Plate Recognition (ALPR). This problem is essentially considered as solved if the input image has adequate resolution and lighting, and good results can be obtained with a classical approach of three stages: i) performing some kind of feature extraction in the input image using edges, morphological operations, contours, local binary patterns, etc.; ii) using a binary classifier to detect candidate plate positions in the input image; and iii) using some well-trained Optical Character Recognition (OCR) on the set of countries plate styles [4].

Not only plates [8], but also cars themselves [17] can also be detected and located with end-to-end trained Convolutional Neural Networks (CNN) in more challenging scenarios under less constraints, though these kinds of techniques need more computational power. Long Short-Term Memory (LSTM) networks –a kind of Recurrent Neural Network (RNN)– have also been used for this purpose [13]. Other works have also been dedicated to even classify vehicle brands and models from images [5, 9]. More traditional approaches based on ad-hoc feature extraction (e.g. Histograms of Gradients, HoG) have also been successfully applied to efficient vehicle detection and location [7]. Most of these techniques can also be trained and adapted to detect and classify other type of objects which can be of interest in a PSN scenario, such as bikes, people, signals, road obstacles, and so on.

Solutions based on computer vision and machine learning algorithms such as the ones outlined above can be enforced and exercised in multiple and diverse application areas. For example, in Smart City scenarios a plethora of solutions for vehicle classification, human behavior, vehicular traffic flow, crowd analysis, and emergency situations, among others, have been widely proposed [10, 11]. However, few of these works benefit from the advantages of upcoming 5G networks, such as high throughput and computational power resources that can enable advanced deep learning techniques (e.g. CNNs and RNNs) with high accurate classification results. It should be noted that a human behavior recognition solution for Smart Cities was recently presented in [3], where the problems and challenges for future Smart City applications that want to leverage 5G capabilities were also discussed. In spite of these new works, none of them, to the best of our knowledge, leverage the new advanced virtualization-based technologies such as NFV as a main key driver of 5G.

Despite the advances made by the works discussed above, the application of computer vision and machine learning techniques on virtualization platforms is still a great challenge to explore in more detail, in order to decouple the monitoring capabilities in collecting video streams by distributed sources from the analytics phases deployed in virtualized environments.

3. Proposed 5G-CAGE deployment

This section outlines the 5G-CAGE deployment in which the CODet VNF is embedded, by detailing the implementation and its main operational workflow, as well as the 5GINFIRE platform used for exercising the 5G-CAGE proposal in a virtualized environment.

[1]The acronym stands for *5G-enabled **C**ontext and situational **A**wareness detection with machine learnin**G** techniques of city objects in **E**xperimental vertical instances*.

3.1. 5GINFIRE platform

In the context of upcoming 5G advanced communication networks, the existence of a complex ecosystem made of multiple physically interconnected devices implies that they will have to address issues such as flexibility, scalability, and extensibility. 5GINFIRE is an open and extensible 5G NFV-based ecosystem that lays down the foundations for instantiation of fully softwarized architectures and deployments of vertical industries and experimenting with them [14]. It therefore provides an experimental playground where new components, architecture designs, vertical deployments and APIs may be tried and proposed before they are ported to more mainstream 5G networks.

Following experimental principles, for the 5G-CAGE deployment the CODet city safety solution is instantiated as a new NFV Network Service into 5GINFIRE Experimental Vertical Instances (EVI), which are simply integrated applications designed for a specific purpose. As shown in Figure 1, the 5GINFIRE platform provides multiple video streams produced by heterogeneous sources, namely moving cameras of citizens (e.g. smartphones), connected car cameras, traffic light cameras and, in general, any available city video surveillance and visual sensing source. Our new CODet VNF is devoted to localize sought vehicle license plates in this context as an initial proof-of-concept implementation, and is integrated with other existing video related 5GINFIRE VNFs such as the ones providing video cache, storage, and transcoding functionalities.

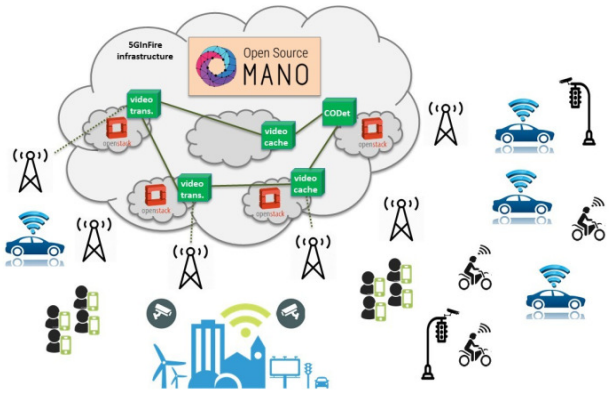

Figure 1. Integration in the 5GINFIRE infrastructure.

Of course, computing resources are required for the operation of the proposed CODet automated detection and recognition services. And, obviously, the more advanced the techniques, the more computing resources are needed. Instances of the designed CODet VNF are virtualized as replicable images, and subsequently executed using the standard *OpenStack* VIM platform[2]. On top of it, the NFV orchestration is carried out by the *Open Source MANO*[3] (management and orchestration module for the automation of the deployment, configuration, coordination, and lifecycle management of the whole process), which is included in the 5GINFIRE platform. This module is provided by the global 5G Telefonica Open Innovation Laboratory (5TONIC)[4].

Once the CODet VNF instances are deployed in the 5GINFIRE experiment virtualized infrastructure, they can be dynamically configured to access specific video streams available from video transcoders and/or video caches, thus enabling the automated image/video analysis techniques to detect specific sought objects.

3.2. CODet typical lifecycle

The lifecycle of a CODet VNF deployment and subsequent exploitation is made up of the following steps:

1. The CODet VNF is deployed in the 5GINFIRE infrastructure, which allows connecting/accessing to other VNFs of the platform.

2. Video streams from selected moving sources (e.g. Smart City bikers, connected cars, or surveillance cameras) are made available in video caches either directly or after video transcoding.

3. An external end-user (for example, a policeman) accesses the CODet VNF to configure the detection of a specific object (in our case, a given vehicle license plate number) in live video streams.

4. The CODet VNF executes the needed computer vision algorithms over the available video streams to detect the given required city object (a specific plate in our initial proof-of-concept).

5. If detected, the CODet VNF generates an alert to the end-user consisting of an image of the found object, as well as some additional information such as the corresponding timestamp and location of the detection, capturing device details, and so on, depending on the availability of such data from the corresponding video source.

Within an actual 5G network, the ideal case would be to deploy the CODet VNF (together with caches and transcoders) at edge locations, which means to use NFV Points-of-Presence (PoP) as close as possible to the video sources. This would allow reducing latency in the video streams transfer, while optimizing the performances of the early detection system itself.

[2] https://www.openstack.org

[3] https://osm.etsi.org

[4] https://www.5tonic.org

3.3. CODet VNF implementation

The implementation was wrapped in a Linux-based Virtual Machine (VM) image for being instantiated in OpenStack, incorporating the computer vision algorithms and libraries needed for the desired object recognition tasks. The operational workflow implemented in the 5G-CAGE proposal is depicted in Figure 2.

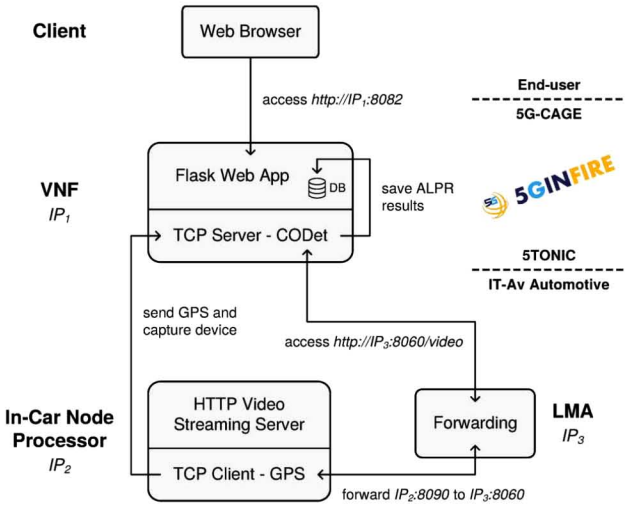

Figure 2. Operational workflow of the 5G-CAGE proposal.

The *In-Car Node Processor* runs a *TCP Client* and an *HTTP Video Streaming Server*, while the VNF runs a *TCP Server* and a *Flask Web App*. Both communications are oriented to feed the CODet VNF with video streams captured by the In-Car Node Processor, in addition to geolocation data synchronized with those video streams. The In-Car Node Processor gets the GPS coordinates and sends them to the VNF, along with the capture device URL so the VNF can access it. The In-Car Node Processor starts the HTTP Video Streaming Server, which will be forwarded through the *Local Mobility Anchor* (LMA) gateway. This way, the VNF can get all information (live video streams and GPS coordinates) from the In-Car Node Processor along the vehicle's path of motion.

The implementation supports multiple vehicles per CODet VNF, each running its own video streaming server and sending its GPS location. The CODet VNF, which is multithreaded, then keeps accessing the respective video devices of each connected TCP Client, processing the captured frames of each one with the specific implemented computer vision procedure. Every time a sought object (a given plate number in our case) is detected, the CODet VNF saves the event in a local database (DB). A Flask-based web application, which is also started in the VNF, manages this DB and provides the end-user (e.g. policy staff) a way to interact with the 5G-CAGE framework through a simple *Web Browser*. The database and the web application are unique, so the data collected from all vehicles can be available in a single place to the end-user.

In our current implementation, the main packages making up the CODet VNF are OpenCV[5] for general image processing and computer vision algorithms; the OpenALPR[6] for the specific task of Automatic License Plate Recognition (ALPR); the Python Flask library[7] to enable a web application that external entities (e.g. police staff) can use to request the detection of particular vehicle license plate numbers in which they have an interest; and GeoPy[8] to access geocoding services and be able to determine the specific location where each detection has taken place. Python has been the main language used in the implementation.

As aforementioned, at the core of the specific automatic license plate recognition task that we have used to test our proposal is the OpenALPR library. This library, developed in C++ and made available to CODet through a Python binding, operates as a pipeline with the following stages:

1. **Detection**: A classical binary classification technique based on multiscale Local Binary Pattern (LBP) features is first executed over the whole input image to find possible license plate regions. For training this detector, a previous offline training phase consuming 40-60 hours of computing was needed, using different subsets of subimages labelled as positive (real plates) and negative examples from an input dataset composed of 3000 raw input images. One such detector can support many different plate styles, as long as they have a similar aspect, though different detectors must be trained individually to find the location of plates belonging to different countries.

2. **Binarization**: The set of candidate regions obtained in the previous phase are then binarized using some dynamic thresholding technique (e.g. the Wolf-Jolien or Sauvola methods), in order to isolate individual characters in the plate.

3. **Character analysis**: Every connected component on each binarized candidate region is isolated, and a global consistency test is executed on this set of components to check if their overall sizes and locations are consistent with the typical layout of a license plate.

4. **Alignment**: If a candidate region passed the former test, the exact limits of the plate are located using an edge finding procedure along with the approximate character size and alignment information obtained by the previous phase.

[5] https://opencv.org
[6] https://www.openalpr.com
[7] http://flask.pocoo.org
[8] https://pypi.org/project/geopy

5. **Rectification**: The rectangle obtained in the previous phase is used to perform an homographic perspective rectification to get a fully frontal image of the plate.

6. **Character segmentation**: In this stage, a vertical integral projection is used to find spaces between characters, in order to isolate and prepare them to be processed individually. Small regions corresponding to slight imperfections are also eliminated in this segmentation stage.

7. **OCR**: The well-known Tesseract OCR library[9] is finally used to perform an optical character recognition on each individual letter/digit. In its latest version, this library uses an LSTM recurrent network to also perform a pattern matching algorithm to test the consistency of the sequence of letters and digits, with an input regular expression catching the country licensed plate syntax. This stage produces the final plate number, along with a probabilistic estimation of the confidence level on the obtained output.

3.4. Deployment and operation

To perform a preliminary test of 5G-CAGE in a realistic environment, we had access to the vehicular network of the automotive EVI environment available in the Technological Institute of Aveiro (*IT-Av Automotive*, Portugal)[10], which is ascribed to the 5GINFIRE project. It consists of *On-Board Units* (OBU) mounted in vehicles connected to *RoadSide Units* (RSU), which are in turn connected to the Internet through an Ethernet interface, as shown in Figure 3.

The OBU connectivity capabilities are mainly *4G*, *GPS*, *Wi-Fi* and *802.11p*, though of course, in the future the target OBU would be a 5G connected one. The GPS provides location data, while the Wi-Fi link is used for in-vehicle connectivity, specifically with the In-Car Node Processor in charge of the camera. The 11p/4G communications enable offloading the computation and allow the OBU to communicate with the virtualized infrastructure through the RSU or the *Cloud Radio Access Networks* (C-RAN). The *Unified Gateway* (GW) is used to reach the IT-Av Cloud through the 4G link. The OBU selects the most convenient interface (11p or 4G), and seamless handovers are performed. Specific *Mobility Servers* could also be deployed where needed in order to provide better routing performance. Finally, and in the upper level of the framework deployment, the Open Source MANO is the component in charge of creating and managing the virtualized CODet VNF instances deployed at the IT-Av Automotive Cloud.

Considering all the aforementioned components, a QEMU VM Ubuntu 18.04 image incorporating all the needed software presented in Section 3.3 was created. Once available, a number of corresponding NFV descriptors were created and uploaded to the 5GINFIRE portal[11]. As specified above, of the four main services implemented by the 5G-CAGE proposal as shown in Figure 2, the first two (the CODet software itself and the Flask Web App) are deployed in the VNF in the virtual image, while the GPS sender and the HTTP Video Streaming Server should be deployed as close as possible to the moving cameras (in our testbed case, in the In-Car Node Processor itself). Parameters such as the port number where the CODet will be listening connection requests from new In-Car Node Processors, the elapsed time between successive frames to be processed from the same source, as well as all sort of streaming options –including spatial and temporal resolution– are of course fully configurable.

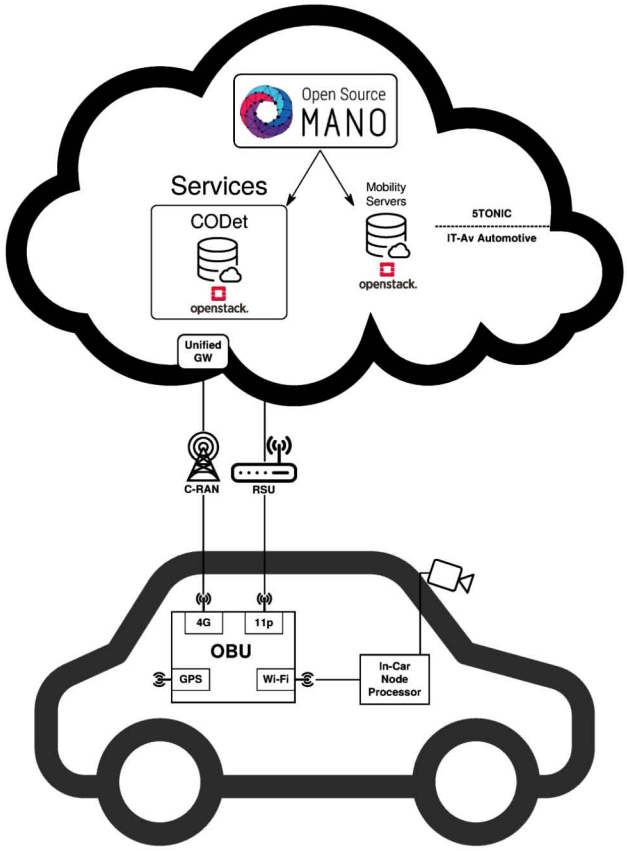

Figure 3. Deployment of the 5G-CAGE framework.

Once all components are up and running, an end-user (e.g. a policeman) could finally start searching for a license plate and consult the results from any client device (i.e. smartphone or computer) through a simple web interface. In order to achieve this, the end-user types the license plate to search for in a text input field, and if the plate were

[9] https://github.com/tesseract-ocr/tesseract
[10] https://5ginfire.eu/it-av-automotive-testbed
[11] https://portal.5ginfire.eu

successfully detected, all the relevant information would be displayed by the web application as shown in Figure 4: original text entered as license plate; GPS coordinates of the physical location where the detection took place; embedded web frame showing this location on the map; detection date and time; confidence of the result; and the image of the license plate, but cropped from the original complete frame. The image is cropped to just the interest region around the plate in order to preserve privacy issues, since its utility is just to allow the end-user to check visually if the result really corresponds to the license plate sought.

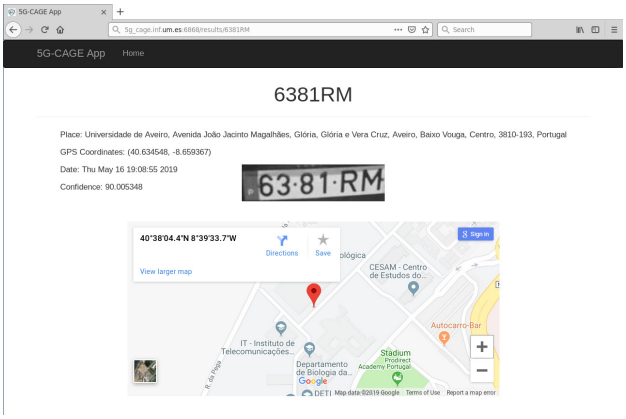

Figure 4. License plate results in detection.

Finally, all this information is also saved in a central database which is later accessible through a REST API in JSON format. Observe as well that the system also keeps continually providing some relevant performance statistics, such as the global processing time per input image or the time taken to process each individual plate, among others.

4. Experimental results

This section outlines the experimental results obtained when executing the CODet VNF, both in local tests for initial testing and in the 5GINFIRE platform to get more real results in the IT-Av Automotive Environment testbed. Both types of tests are detailed in the following two subsections, where the experimental results depicted in Figure 5 will be presented.

4.1. Results in local laboratory tests

We first conducted a local test by running the OpenALPR based CODet implementation in a real (i.e. not virtualized) machine whose main processor was an Intel 3.7 GHz CPU with 6 cores and hyperthreading (for a total of up to 12 real threads running in parallel). The associated performance results are shown in Figure 5(a). In it, we show the processing time stacked for each execution in which at least one license plate has been found in the image. Each individual bar in the graph illustrates the processing time for each detected plate, as well as an extra time (in green) that, as can be clearly seen, mostly corresponds to the detection phase. It is worth noting, therefore, that the computational load for input images which do not contain any readable plate tends to be a bit lower, but not that much, because the detection stage (and not the posterior OCR interpretation of the detected plates, shown in red in the figure) is the one that takes more computational load. The yellow processing times shown for some input frames stand for (sporadic) false plate positives that can appear in some cases due to adverse environmental conditions.

The average total processing time is 141ms, which may seem a bit high (as it allows only for $\sim$7 processed frames per second on average). The reason is that the input image in this test had a (relatively high) resolution of 1920x1080 pixels, a fact that logically makes the detection stage more involved (127ms on average). In comparison, the average processing time for interpreting each detected plate is just 13ms, clearly much lower. In certain cases, the detection time can even reach slightly higher values, of up to 140$\sim$160ms. These peaks tend to be caused by input frames eventually containing regions that, though lately correctly discarded, were initially considered as license plates, making the detection phase longer than usual.

It is finally worth mentioning that the image resolution, which has a very important impact on the obtained performance, imposes a logical trade-off between the time required by the ALPR process and its accuracy: a plate image needs to have a certain minimum of pixels to be detected and correctly interpreted by the APLR process, whereas increasing the pixels of the image will make the detection phase slower and could cause a variable quantity of input frames to be lost.

As of the average confidence obtained by the overall plate recognition system is 87.78% in our tests, which however can be considered as acceptable. A result can be considered good in case the license plate has been detected with an OCR confidence over $\sim$80%. This however not necessarily means that all detections are correct, as some of the high confidence values sometimes correspond to wrong results. However, it is obviously better to consider these images rather than discarding them directly, configuring the CODet VNF to allow for some uncertainty in the detected plates so as to allow the end-user to check these potential matches visually in the corresponding cropped images shown in Figure 4. This makes more sense than simply discarding the plate, which could eventually lead to lose an important (i.e. searched for) one. Of course, it is also possible for a given plate to be repeated in the detection output, which can also be used to reinforce the confidence on the obtained result.

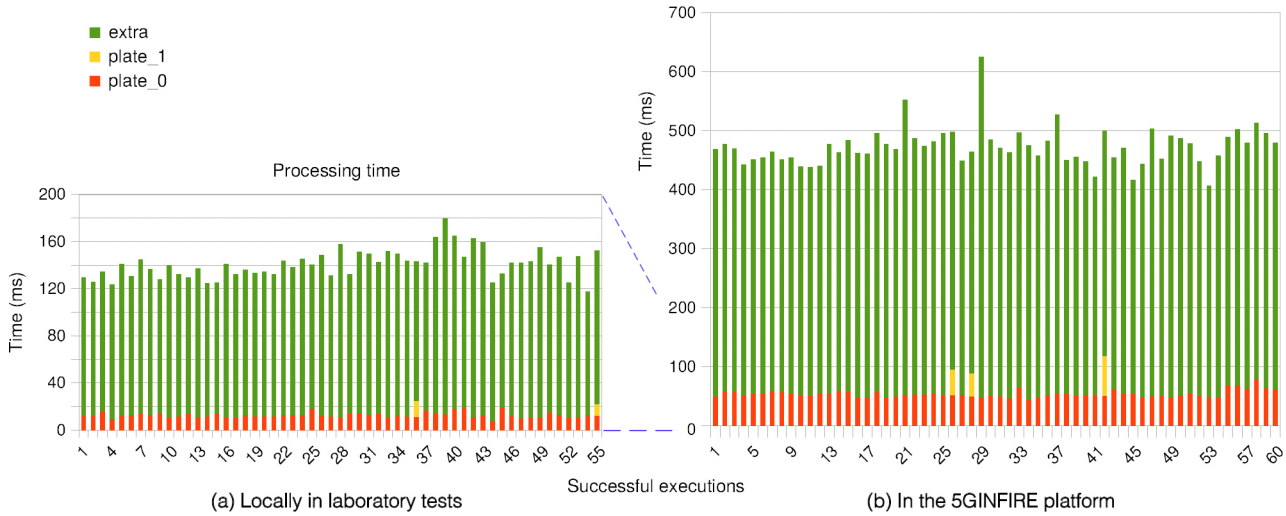

Figure 5. Performance results in processing time when executing the CODet implementation (a) in local laboratory tests and (b) the 5GINFIRE virtualized platform.

4.2. Results in the 5GINFIRE platform

A vehicle with an OBU, a Raspberry Pi attached to the In-Car Node Processor and its connected camera (see Figure 3) was taken out at the IT-Av Automotive Environment testbed in order to execute this second set of tests. The vehicle went around a circuit with 6 known a priori Portuguese license plates. The camera resolution was 1280x720 to reduce the amount of network traffic, which obviously conditioned the distance at which license plates were analyzed correctly.

Worse results than the ones obtained in the local lab tests of the previous section were obtained, which was expected for several reasons. The first one is that the VNFs deployed in the 5GINFIRE platform runs on much less powerful virtual CPUs running at just 2 GHz, having only 1 core with no hyperthreading. A second handicap comes from network issues (low bandwidth and high volume of traffic) due to the current network infrastructure used by the IT-Av Automotive Environment testbed, which, as specified above, conditioned the image resolution and compression level of input streams. Besides, the relatively lower performance of the virtualized machine made the OpenALPR system discard several real time frames while processing each one, a fact that, together with the diminished resolution, reduced significatively the number of processed frames where the license plates were seen closely enough.

Still, platform results in terms of performance are still acceptable, as can be seen in Figure 5(b). These are qualitatively similar to the results obtained from tests conducted in our local lab, formerly shown in Figure 5(a), though with processing times of all stages approximately multiplied by 3 (for a total of ∼473ms on average). The detection phase is still the most time-consuming phase and maintains its time spikes depending on the complexity of the input frame.

Regarding accuracy, the total number of 6 different license plates on the controlled experiment was known in advance, so that would be the final number of different results expected in case everything worked properly. But the input video stream performed a total of 60 detections, since in the route the same plates appeared in different frames. The confidence was of 84.11% on average, a bit lower than in the local results, but still consistent with the experiment conditions.

Finally, Figure 6 shows the amount of network traffic required by the CODet VNF to gather video streams from the In-Car Node Processor of the vehicle used in the experiment. The plot displays only the transmission rate of the data plane network interface, by using the Speedometer tool which measures and displays the rate of data across a network connection.

We observe in Figure 6 a sustained data rate of around 121 KB/s on average for the particular conditions of our experiment. But, of course, this will be very dependent on aspects (available bandwidth, image resolution, compression codecs, overall traffic, etc.) that will clearly vary a lot in the upcoming large scale 5G scenarios. In any case, it is clear that the volume of traffic will be an important issue to consider in real automotive scenarios with a greater number of vehicles in motion. However, we believe that this potential traffic congestion could be mitigated by decoupling some of the CODet functions and transferring them, when possible, to each vehicle's OBU, trying to reduce the amount of traffic in video stream delivery.

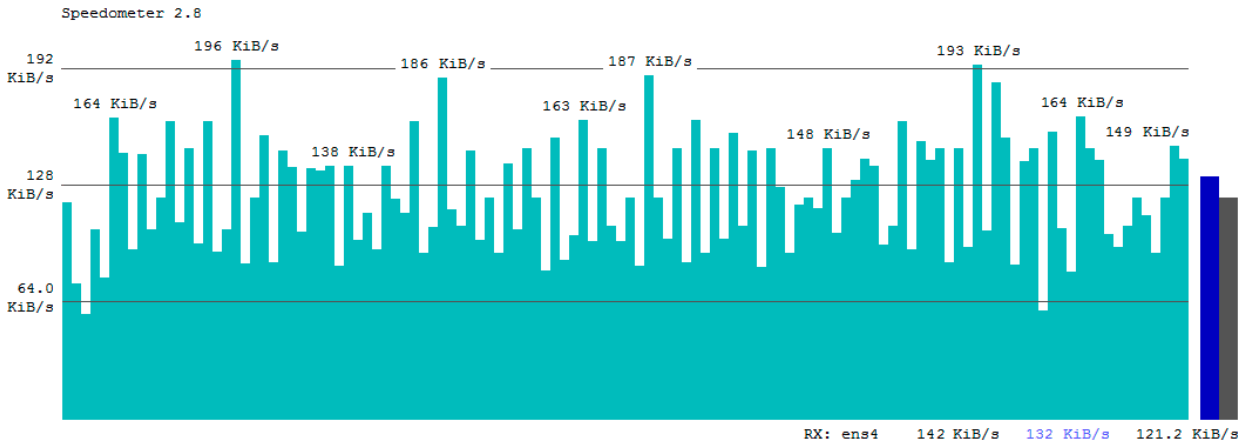

Figure 6. Network traffic performance (in KB/s) displayed by the Speedometer tool.

5. Conclusion and future works

In this work we have described 5G-CAGE, a project aimed to implement a proof-of-concept distributed vehicle license plate recognition system using a heterogeneity of input video stream sources as the underlying monitoring infrastructure, taking advantage of the upcoming 5G networks in a Smart City ecosystem. Our approach leverages on the NFV paradigm to scale adequately using the available virtualized computing infrastructure located on both the edge and/or the network core, converging with other vertical services such as video caching or transcoding already present in those infrastructures. In this context, our specifically developed CODet VNF has been deployed as a new asset in the existing 5GINFIRE platform, which ensures that it will be easy to reuse and adopt in future experiments and platform enhancements.

Our experimental results obtained in a real vehicular environment confirm the viability of the proposal, encouraging us to explore a number of lines of future work. First, it will be a priority to test the developed virtualized environment in a more realistic 5G scenario, without the current limitations imposed by the current platform hardware. The influence on both the computational performance and the accuracy of key aspects such as the specific placement of the moving cameras in realistic environments (from cars to static webcams or surveillance cameras) and the quality heterogeneity of the images obtained from the different sources should also be further studied. Finally, adding wider detection functions that might be linked to PSN applications, such as vehicle brands or human faces recognition, can be also an attractive scenario in the near future.

References

[1] 5GINFIRE H2020 Project. Evolving FIRE into a 5G-Oriented Experimental Playground for Vertical Industries, 2017. https://5ginfire.eu. 2

[2] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali. Smart cities of the future. *The European Physical Journal Special Topics*, 214(1):481–518, 2012. 2

[3] C. Dai, X. Liu, J. Lai, P. Li, and H.-C. Chao. Human behavior deep recognition architecture for smart city applications in the 5G environment. *IEEE Network*, pages 1–6, 2019. 2

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

[5] Y. Gao and H. J. Lee. Local tiled deep networks for recognition of vehicle make and model. *Sensors*, 16(2), 2016. 2

[6] H. Habibzadeh, T. Soyata, B. Kantarci, A. Boukerche, and C. Kaptan. Sensing, communication and security planes: A new challenge for a smart city system design. *Computer Networks*, 144:163–200, 2018. 1

[7] N. Laopracha and K. Sunat. Comparative study of computational time that HOG-based features used for vehicle detection. In *Recent Advances in Information and Communication Technology 2017*, pages 275–284, 2017. 2

[8] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Gonçalves, W. R. Schwartz, and D. Menotti. A robust real-time automatic license plate recognition based on the YOLO detector. In *2018 International Joint Conference on Neural Networks*, pages 1–10, 2018. 2

[9] B. Li, Y. Dong, Z. Wen, M. Liu, L. Yang, and M. Song. A machine learning-based framework for analyzing car brand styling. *Advances in Mechanical Engineering*, 10(7):1–17, 2018. 2

[10] M. Magrini, D. Moroni, G. Palazzese, G. Pieri, G. Leone, and O. Salvetti. Computer vision on embedded sensors for traffic flow monitoring. In *18th International Conference on Intelligent Transportation Systems*, pages 161–166, 2015. 2

[11] N. Mohammad, S. Muhammad, A. Bashar, and M. A. Khan. Formal analysis of human-assisted smart city emergency services. *IEEE Access*, 7:60376–60388, 2019. 2

[12] U. Raza, M. Usman, M. R. Asghar, I. S. Ansari, and F. Granelli. Integrating public safety networks to 5G: Appli-

cations and standards. *Enabling 5G Communication Systems to Support Vertical Industries*, pages 233–251, 2019. 1

[13] P. Shivakumara, D. Tang, M. Asadzadehkaljahi, T. Lu, U. Pal, and M. H. Anisi. CNN-RNN based method for license plate recognition. *CAAI Transactions on Intelligence Technology*, 3(3):169–175, 2018. 2

[14] A. P. Silva, C. Tranoris, S. Denazis, S. Sargento, J. Pereira, M. Lus, R. Moreira, F. Silva, I. Vidal, B. Nogales, R. Nejabati, and D. Simeonidou. 5GinFIRE: An end-to-end open5G vertical network function ecosystem. *Ad Hoc Networks*, 93:101895, 2019. 2, 3

[15] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Communications Magazine*, 55(4):54–61, 2017. 2

[16] J. Wang, B. He, J. Wang, and T. Li. Intelligent VNFs selection based on traffic identification in vehicular cloud networks. *IEEE Transactions on Vehicular Technology*, 68(5):4140–4147, 2019. 1

[17] Y. Zhou, H. Nejati, T. Do, N. Cheung, and L. Cheah. Image-based vehicle analysis using deep neural network: A systematic study. In *2016 IEEE International Conference on Digital Signal Processing*, pages 276–280, 2016. 2