Off-the-Shelf Based System for Urban Environment Video Analytics

Henry O. Velesaca[1], Steven Araujo[1], Patricia L. Suárez[1], Ángel Sánchez[2] and Angel D. Sappa[1,3]
[1]Escuela Superior Politécnica del Litoral, ESPOL, CIDIS, Guayaquil, Ecuador
[2]E.T.S. Ingeniería Informática, Universidad Rey Juan Carlos, 28933 Móstoles, Madrid, Spain
[3]Computer Vision Center, Edici O, Campus UAB, 08193 - Bellaterra, Barcelona, Spain
{*hvelesac, saraujo, plsuarez,*}*@espol.edu.ec* *angel.sanchez@urjc.es* *sappa@ieee.org*

***Abstract*—This paper presents the design and implementation details of a system build-up by using off-the-shelf algorithms for urban video analytics. The system allows the connection to public video surveillance camera networks to obtain the necessary information to generate statistics from urban scenarios (e.g., amount of vehicles, type of cars, direction, numbers of persons, etc.). The obtained information could be used not only for traffic management but also to estimate the carbon footprint of urban scenarios. As a case study, a university campus is selected to evaluate the performance of the proposed system. The system is implemented in a modular way so that it is being used as a testbed to evaluate different algorithms. Implementation results are provided showing the validity and utility of the proposed approach.**

***Keywords*—greenhouse gases, carbon footprint, object detection, object tracking, website framework, off-the-shelf video analytics.**

I. Introduction

A *smart city* is a place where traditional services become more flexible, efficient, and sustainable with the use of information and communication technologies for the benefit of its inhabitants ([1], [2]). Currently, there are several initiatives for the development of technologies in the context of smart cities; among the most challenging approaches are those focused to reduce the impact of pollution due to the emission of *greenhouse gases*. For example, in Europe, the set plan supports cities to take measures to arrive in the next years to reduce 40 percent of greenhouse gas emissions through sustainable use and energy production [3]. Urban carbon emissions can be generated due to several factors, from moving vehicles till outdoor lighting systems—e.g., street lights, lights for building facade valorization, etc. [4]. In short, there is a massive effort on measuring and controlling carbon emissions within the context of smart cities, as mentioned above, some times referred to in the literature as carbon footprint.

In order to reduce emissions, the sources need to be identified and the amount of environmental pollution needs to be measured. As mentioned above one of the sources of carbon emissions are vehicles. In recent years some effort has been put on counting the number of vehicles in urban areas and classifying them according to their size. Studies on specific places on urban scenarios were performed and statistics information was collected. Such a kind of procedure consists of installing physical devices (e.g., piezoelectric loops) to collect information for a couple of days.

978-1-7281-7539-3/20/$31.00 ©2020 IEEE

Nowadays, most of the cities have hundreds of thousands of video cameras, which are mainly used for urban monitoring and video surveillance. Generally, these cameras are being supervised by a person, which is susceptible to errors due to the large amount of information that the person has to handle. In this way, it is very important to invest efforts in computer vision systems to assist the people who are in charge of taking control and thus reducing the margin of error to the minimum [5]. Such video systems can be used not only for video surveillance applications but also for generating statistics about different urban indicators, such as the number of vehicles through an avenue, number of people in a certain place among others as mentioned in [6]. This information is useful for governments that are interested in the reduction of greenhouse gases. The current work aims to provide a technological solution to measure the impact by generating statistics related to the vehicle's usage in urban environments. The system is based on the usage of a deep learning approach that allows nowadays to obtain a reliable solution to most of the computer vision-based problems.

Deep Learning has taken importance in video analytics and allows us to solve most of the tasks that a human operator would execute, in addition of doing it in a more efficient way. For example, Fernández et al. [7] indicates that at present the principal interest is on applications such as fight detection, identification of vandalism, theft detection, among others. The authors also mention that the increase in video surveillance cameras and mass production of videos all around the world has increased in recent years, so it is necessary to use models based on Deep Learning for the tasks of classifying each one of the activities to be identified, in the same way, that no person can be done.

Another important concept is the *off-the-shelf* technologies, which are solutions made, tested and well maintained by third parties that are available either free or paid [8]. In the current work, the usage of a large amount of open source available will be exploited. In particular, code for detecting and tracking objects in a given scene, to develop and implement a system that integrates these two elements and allows to authorities of the public sector to be able to generate different type

of statistics from urban scenarios (e.g., amount of vehicles, numbers of persons, etc.). The obtained information could be used for traffic management, security of people in public places, estimation of carbon footprint in urban scenarios, among other applications.

The manuscript is organized as follows. Section II presents works related to both the computer vision approaches needed for the video processing and understanding and the software development frameworks. The solution proposed in the current work is detailed in section III. Implementation results are depicted in section IV to illustrate a case study. Finally, the conclusions are given in section V.

II. State of the art

This section reviews recent approaches related to the proposed system. It covers topics from the pattern recognition and tracking till software development frameworks and databases engines.

A. Object detection

During the last five years, a large amount of convolutional neural network-based approaches have been proposed for object detection. This section just reviews some of them, related to the system implemented in the current work.

1) SSD: One of the existing methods for detecting objects in images using a single deep neural network is presented in [9]; it is named as Single Shot multi-box Detector (SSD), which discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location [9]. This network, at prediction time, estimates scores for the presence of each object category in each default box and generates adjustments to each box to improve the matching of the object shape. The network also combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes [9].

2) Faster R-CNN: Another object detection method based on region proposal to determine object locations is introduced in [10]; in this work, the authors propose an algorithm that performs a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network. This network uses as input an image of any size and returns a prediction of object bounds and scores at each position. The RPN network is initialized with ImageNet and fine-tuned for region proposal. The obtained regions are used to train Fast R-CNN; it shares convolutional layers weights, forming a unified network based on image regions to reduce the time of execution of object detection [10].

3) YOLO: The last network architecture reviewed in this section is YOLO (You Only Look Once) [11]. This architecture is intended for object detection tasks, specialized to determine the location on the image where target objects are present, as well as to predict the type of object. YOLO tackles object detection as a single regression problem, using a convolutional neural network. It receives as input an image and returns a vector of bounding boxes coordinates together with their corresponding class probabilities. YOLO also exploits multi-scale training which increases the robustness of the solution. The last version of YOLO, named YOLOv3, uses multi-label classification to calculate the likeliness of the input belonging to a specific label, also uses binary cross-entropy loss for each label, reducing the computation complexity. This type of algorithm is commonly used for real-time one-shot object detection [11].

B. Object tracking

Once an object is detected in a given frame, it needs to be tracked through the whole video sequence in order to get the same label and counted just one time through the video sequence. Hence, this section reviews state-of-the-art techniques for object tracking.

1) Template matching OpenCV function: Template matching is a technique that applies an exhaustive search for coincidences between regions of different images. It works by comparing a region from the template image with regions in the source image, the image in which it is expected to match the template. It is recommended that the template image be smaller than the source image. In OpenCV there are six different methods to compute the matching cost; although all of them have a similar performance, CCOEFF NORMED works better when there are differences between illuminations of both images [12].

2) Simple online and realtime tracking: Simple Online and Realtime Tracking (SORT) is a software tool that allows tracking objects in real-time. It uses a Kalman filter and the Hungarian algorithm to predict the track of previously identified objects and matches them with new detections [13]. It is a simple and effective algorithm.

3) Extension to simple online and realtime tracking: In [14] an extension of the SORT algorithm, named Deep SORT, is proposed. It improves the performance of detection using appearance information. This method makes possible to track objects through longer periods of occlusions, effectively reducing the number of identity switches. The computational complexity is placed into an offline pre-training stage using a deep association metric on a largescale person re-identification dataset. With the deep network, a vector that can describe all the features of a given image is obtained, improving the uncertainties coming from the Kalman filter related to the location of objects over time. During the online application, the method establishes measurement-to-track associations using nearest neighbor queries in visual appearance space, achieving competitive performance at high frame rates.

C. Website development frameworks

Currently, there is a large number of website development frameworks, and depending on the context where it can be used will have advantages over another.

1) Java server faces: The Java Server Faces (JSF) is a web development framework oriented to the user interface for web applications based on the JAVA programming language; it uses Java Server Page (JSP) as a technology that allows the deployment of pages, but also uses other technologies such

as AJAX, XML, JavaScript, CSS, HTML, RichFaces, among others [15].

2) Symfony: Another very popular framework for web development is Symfony; its popularity is because it is one of the frameworks for the PHP programming language with better performance, one of its characteristics is that it presents an internal architecture based on modules, which allows replacing or eliminating components that are not needed within a project, also has a community that supports new features and existing ones [16].

3) Django: Before presenting the following framework we will talk about the language in which it is based; Python is a programming language interpreter that allows a fast performance for different types of programs. It is an object-oriented programming language of general-purpose in general popularity in the area of deep learning, machine learning, and other areas. Django is a web framework based in python that allows easy development, clean and pragmatic design [17]. Django uses the Model Template View (MTV) paradigm, which is essentially the Model View Controller (MVC) paradigm but with different names on its components. Table I shows the equivalence between MVC and MTV architecture [17].

TABLE I: Equivalence MVC to MTV

Typical MVC	***Django MTV (file)***
Model	Model (models.py)
View	Template (template.html)
Controller	View (views.py)

D. Databases engines

After reviewing the different web development frameworks we will now review two database engines based on two different types of paradigms.

1) MySQL: MySQL is an RDBMS (Relational DataBase Management System) that allows running databases on different operating systems, stable, good performance, has community support maintenance, and also a large amount of documentation is available. It is an engine that allows managing both desktop and web applications [18].

2) MongoDB: MongoDB is a document-oriented database that is based in collections instead of tables as in RDBMS. Documents are a structure similar to rows as in RDBMS and are organized in collections, that not have any restrictions by creation. The content of documents can be any elemental data type, such as string, date, number, or other document [19].

III. PROPOSED SOLUTION

This section presents details of the implemented system for urban video analytics.

A. Software architecture

The proposed solution has been implemented following a web-oriented approach, which allows scalability, robustness, and continuous growth. Django has been chosen as a development framework due to the support, a large developer community, in addition to the fact of using Python as a programming language, which is currently widely used by the scientific community. On the other hand, the database has been implemented in MongoDB, which is a document-oriented database, Not Only SQL "NoSQL", that allows multiple structural information that permits flexibility and scalability feature and also massive real-time data flow, this database engine is efficient in-memory processing and complex data type feature as indicated in [19]. Figure 1 shows a general scheme of the software architecture used by the proposed solution.

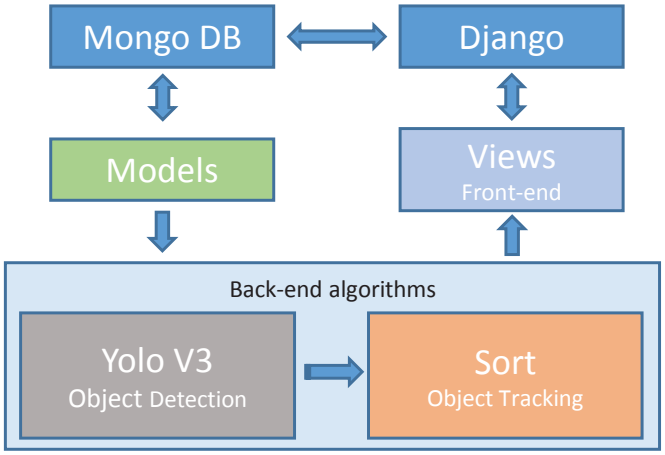

Fig. 1: General scheme of software architecture.

B. Front-end design

The front-end has been designed offering to the users all the possible system configurations. This section details all the components included in the front-end.

1) Software configuration: The first module of the web application allows the construction of the functional architecture dynamically; that is, it allows the user to create and configure the functional parameters such as options menu, profiles, controls, actions to be performed by users, projects, and users. To implement this application design, one of the most powerful components of the Django web framework has been used, which has an interface that, by default, allows managing previously created and registered models, but also leaves the possibility of being able to perform new customization in the future. Figure 2 shows a view of the models that are part of the functional architecture of the system.

2) Security administration and profiles: The next module to analyze is the default security mechanisms offered by Django; although it is a robust and safe scheme for our proposed solution, modifications were made by including additional functionalities such as displaying the options menu based on the profile assigned to a given user, visualization of actions within the views shown to the user and projects permissions assigned to a given user. Figure 3 depicts a snapshot of a part of the administrator profile view.

3) Parameters setting of back-end algorithms: Another characteristic of the proposed solution is the possibility to set all the parameters needed by the detection and tracking algorithms

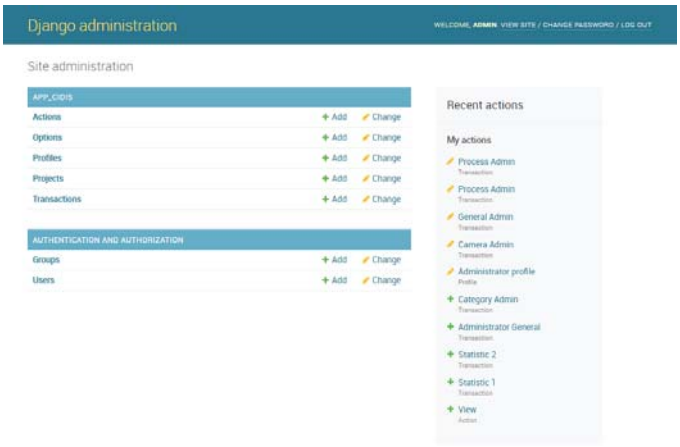

Fig. 2: View of models that allows parameterization of the system.

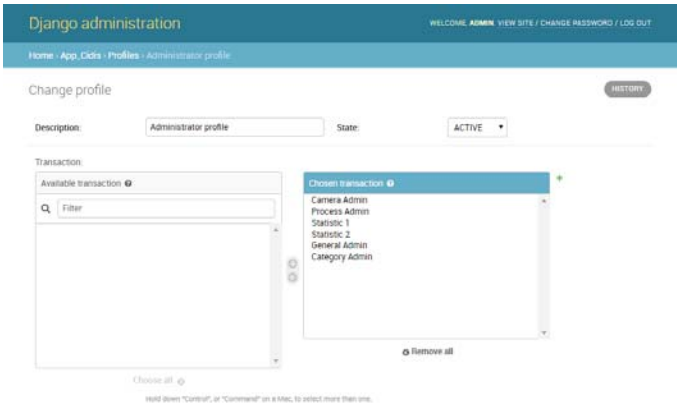

Fig. 3: View of an administrator profile.

(implemented in the back-end); it avoids having to change this configuration through the code. For example, it is possible to choose the types of objects to be detected and also the corresponding detection threshold for each object.

4) Cameras configuration: In order to give the required portability and flexibility, the system offers the possibility to connect video cameras for processing. The configuration allows defining cameras, set the camera's values, and select the one/s that will be used by the detection and tracking algorithms. The interface allows marking in a map the camera position for a further reference.

5) Execution of detection and tracking processes: After performing the configuration of the IP cameras, it is possible to execute the detection and tracking algorithms. This option allows you to select the previously configured cameras and first execute the object detection algorithm in background mode and then execute the algorithm that tracks the object of interest. The information obtained from these processes is stored in the MongoDB database so that users can analyze the information and obtain the corresponding statistics based on the information processed and stored in the system.

6) Statistics: Finally, the module of statistics obtained from the data generated by the detection and tracking algorithms is described. This module allows to generate video analytics reports according to the user requirements (e.g., day, time, type of vehicle, frequency, etc.), this is done using drill-down reports that allow users to navigate among different layers of data granularity by navigating and clicking a specific data element on the report.

C. Backend algorithms

This section presents the backend algorithms that were selected for the proposed solution.

1) Object detection: After reviewing the state of the art approaches related to the object detection problem in section II, the most promising approaches (i.e., Fast R-CNN, YOLOv3, and SSD) have been evaluated. Table II shows, mAP and FPS values for each of the evaluated approaches, when the MS COCO dataset [20] is considered. These values are indicators of the accuracy and execution times of the studied models. Despite that YOLOv3 has a mAP value lower than Fast R-CNN (about 5% smaller mAP), it is considerably faster than Fast R-CNN (about 5 times faster). Furthermore, YOLOv3 is better than SSD in both metrics. The values presented in Table II were obtained from the original publications (i.e., [10], [21] and [9]) were each network has been presented, in all the cases evaluated with the same MS COCO dataset [20].

As a conclusion, the YOLOv3 network has been selected since it is ideal for real-time application. This aspect is very important in the proposed solution because the images will be obtained using IP cameras and the processing and visualization tasks should run in real-time in a web-based platform.

TABLE II: Comparison of state-of-the-art object detection approaches, evaluated on MS COCO dataset.

Network	mAP	FPS
Faster R-CNN	**34,9**	17
SSD	26.8	59
YOLOv3	33.0	**91**

2) Object tracking: A set of experiments have been carried out for selecting the best tracking method. In the experiments, the three methods explained in the state of the art section have been analyzed. As a result, the Simple Online and Realtime Tracking (SORT) has been selected since it obtains better performance compared to the other options. It was concluded that the match template method has delays in runtime because bit-level comparisons are made with each of the recognized objects (bounding boxes), being the worst case when the execution time is $O(n \times m)$, where n and m are the numbers of objects detected in two different frames; concerning the extension of the SORT (DEEP SORT), it presents a delay since it has an added value with the appearance vector, presenting the same problem of the previous method with only the characteristic vector that has an execution time of $O(n \times m)$, thus decreasing the performance compared to the original SORT method. Table III shows metrics from execution times for the three evaluated algorithms.

TABLE III: Metrics from execution times of tracking algorithms

Elapsed time (seconds)			
-	**SORT**	**Deep SORT**	**OpenCV**
avg	0,15	2,99	3,01
max	1,53	8,26	9,04
min	0,12	2,72	2,97
std	0,13	0,44	0,52

IV. Implementation results

This section presents snapshots of the most representative views of the platform implemented for the urban video analytic application. The illustrations presented in this section correspond to results obtained from cameras of a university campus connected to the system; a server with a Titan X GPU was used to process and visualize the results in real-time.

A. Camera configuration view

The camera configuration interface presented above was used to set the IP camera addresses and other parameters. In Fig. 4 a snapshot of the camera configuration interface is shown; the camera parameters, as well as the camera position (i.e., latitude and longitude), are defined. In the bottom part, a live view of the camera is shown to check the added information is correct.

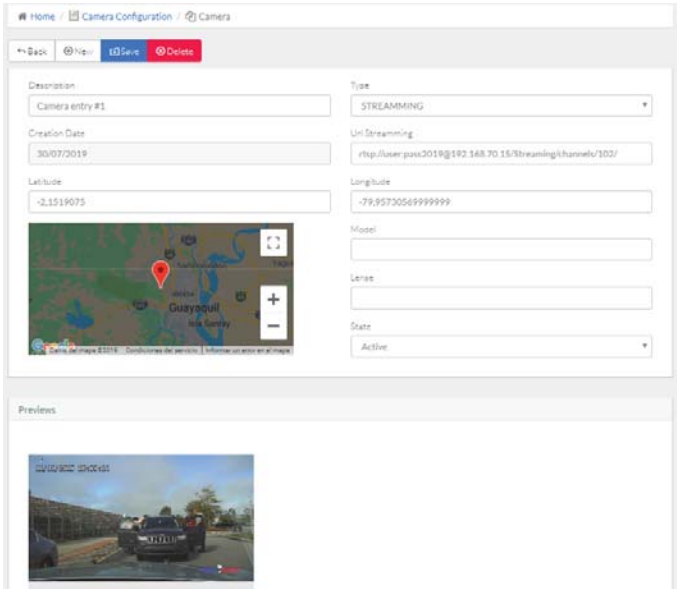

Fig. 4: View of a camera configuration.

B. Execution of detection and tracking process view

Once all cameras have been configured in the system, the videos acquired by them are processed according to the user's specifications. This section shows some results of the detection and tracking algorithms, which are subsequently stored in the database for further processing and analysis. As a case study, one of the IP cameras is used to show the system performance. The selected chamber is located geographically at the entrance of the university campus. It should be noted that the algorithms run in the background, which allows a user to use all the functionalities of the system simultaneously. Figure 5 shows the results of the detection and monitoring process.

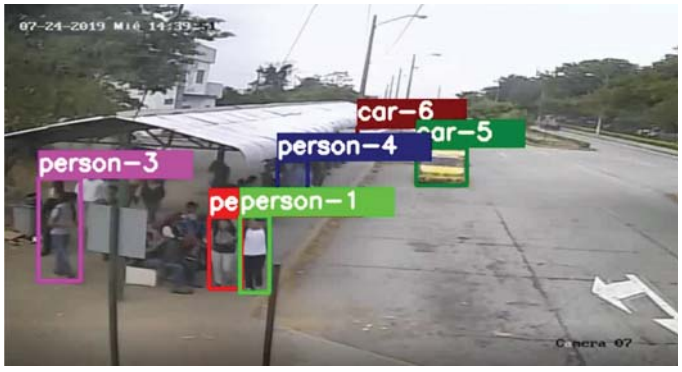

Fig. 5: View of an execution of detection and tracking processes.

C. Statistic view

The system allows obtaining statistics related to previously processed and stored video sequences. For example, the user can obtain information regarding: the number of vehicles and/or pedestrians detected by a particular camera in a certain period of time; the class of vehicles detected (for example, buses, trucks, cars, motorcycles); day and period of time with the highest frequency of vehicles and/or pedestrians; frequency of buses detected from a certain camera (in the case of study this information corresponding to the buses entering the university campus); among other possible queries that the user can make. Figures 6 and 7 show examples of the reports generated by the system; This interactive interface allows the user to visualize in greater detail the information of interest to analyze by selecting one of the existing categories in the system.

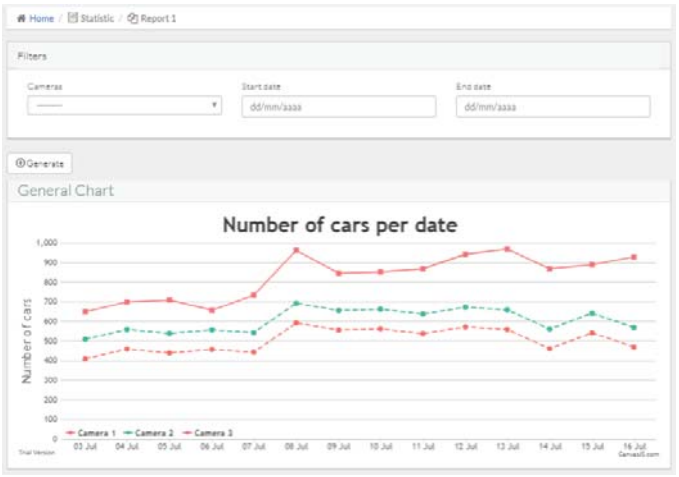

Fig. 6: View of a generated report.

V. Conclusions

The manuscript presents a computer vision system based on off-the-shelf algorithms for urban video analytics. The

main objective is to develop a system able to be used in a smart city context to process information from already existing video surveillance networks. The proposed approach has been implemented using available open-source and evaluated in a real scenario showing its validity. As a future work other deep learning-based algorithms will be evaluated in the proposed framework, trying to reduce processing time as well as to improve accuracy on results.

ACKNOWLEDGMENT

This work has been partially supported by: the ES-POL project "Aplicaciones TICs para Ciudades Inteligentes" (REF: FIEC-16-2018); the Spanish Government under Project TIN2017-89723-P; the Spanish MICINN RTI Project RTI2018-098019-B-I00; the "CERCA Programme / Generalitat de Catalunya". The authors gratefully acknowledge the support of the CYTED Network: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559) and the NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] S. P. Mohanty, U. Choppali, and E. Kougianos, "Everything you wanted to know about smart cities: The internet of things is the backbone," *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 60–70, 2016.

[2] R. R. Harmon, E. G. Castro-Leon, and S. Bhide, "Smart cities and the internet of things," in *Portland International Conference on Management of Engineering and Technology (PICMET)*. IEEE, 2015, pp. 485–494.

[3] A. Kylili and P. A. Fokaides, "European smart cities: The role of zero energy buildings," *Sustainable cities and society*, vol. 15, pp. 86–95, 2015.

[4] F. Rossi, E. Bonamente, A. Nicolini, E. Anderini, and F. Cotana, "A carbon footprint and energy consumption assessment methodology for uhi-affected lighting systems in built areas," *Energy and Buildings*, vol. 114, pp. 96–103, 2016.

[5] F. J. López Rubio *et al.*, "Detección de objetos en entornos dinámicos para videovigilancia," 2016.

[6] H. Zhang, V. Sindagi, and V. M. Patel, "Joint transmission map estimation and dehazing using deep networks," *arXiv preprint arXiv:1708.00581*, 2017.

[7] L. C. Fernández Martínez, "Identificación automática de acciones humanas en secuencias de video para soporte de videovigilancia," 2018.

[8] E. Ventocilla and M. Riveiro, "Visual analytics solutions as 'off-the-shelf'libraries," in *21st International Conference Information Visualisation (IV)*. IEEE, 2017, pp. 281–287.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

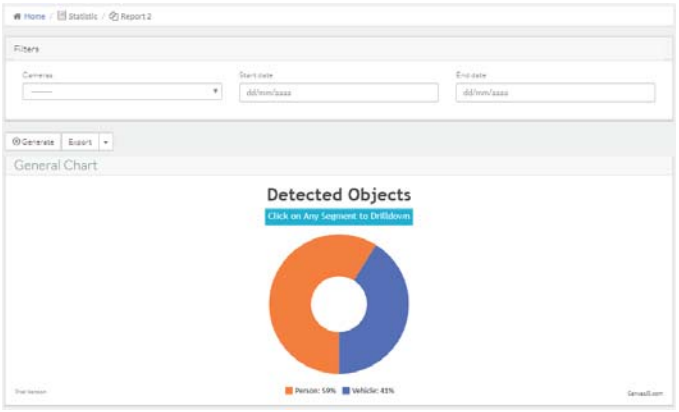

Fig. 7: View of a generated report.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] M. Marengoni and D. Stringhini, "High level computer vision using opencv," in *24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*. IEEE, 2011, pp. 11–24.

[13] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[14] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE International Conference on Image Processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[15] D. Palacios, J. Guamán, and S. Contento, "Análisis del rendimiento de librerías de componentes java server faces en el desarrollo de aplicaciones web," *NOVASINERGIA*, vol. 1, no. 2, pp. 54–59, 2018.

[16] M. C. Valle Dávila, "Estudio del framework symfony 2 para el desarrollo de aplicaciones empresariales," B.S. thesis, 2017.

[17] J. Vainikka, "Full-stack web development using django rest framework and react," 2018.

[18] A. MySQL, "Mysql," 2001.

[19] N. Q. Mehmood, R. Culmone, and L. Mostarda, "Modeling temporal aspects of sensor data for mongodb nosql database," *Journal of Big Data*, vol. 4, no. 1, p. 8, 2017.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.