The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-4/W3, 2017
2nd International Conference on Smart Data and Smart Cities, 4–6 October 2017, Puebla, Mexico

A CLOUD-BASED ARCHITECTURE FOR SMART VIDEO SURVEILLANCE

Luis Valentín*, Sergio A. Serrano, Reinier Oves García, Anibal Andrade, Miguel A. Palacios-Alonso, L. Enrique Sucar

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)
Sta. María Tonantzintla, CP 72840, Puebla, Mexico.
{luismvc, sserrano, ovesreinier, anibalandrade, mpalacio, esucar}@inaoep.mx

KEY WORDS: Smart City, Smart Sensing, Smart Security, FIWARE

ABSTRACT:

Turning a city into a smart city has attracted considerable attention. A smart city can be seen as a city that uses digital technology not only to improve the quality of people's life, but also, to have a positive impact in the environment and, at the same time, offer efficient and easy-to-use services. A fundamental aspect to be considered in a smart city is people's safety and welfare, therefore, having a good security system becomes a necessity, because it allows us to detect and identify potential risk situations, and then take appropriate decisions to help people or even prevent criminal acts. In this paper we present an architecture for automated video surveillance based on the cloud computing schema capable of acquiring a video stream from a set of cameras connected to the network, process that information, detect, label and highlight security-relevant events automatically, store the information and provide situational awareness in order to minimize response time to take the appropriate action.

1. INTRODUCTION

Ensuring the safety of people should be a priority for every city. In order to address this issue some approaches have been proposed. Monitoring systems are the simplest solutions, while architectures capable of analyzing human behavior and determining whether there exists any possible dangerous scenario, such as fighting, theft, etc, are the most complex. Even though the development of complex surveillance schemes constitutes a great challenge, the importance along with the necessity of preserving the safeness of society have played a decisive role as one of the main incentives for researchers and developers to work on the integration of some technologies, such as data management and computer vision, to produce systems that are reliable and effective enough to serve as solution for tasks like cities surveillance, video analytics and efficient video management in order to support city officials and/or security employees in their duty.

Nowadays, video surveillance systems only act as large-scale video recorders, storing images from a large number of cameras onto mass storage devices. From these schemes users have access to information that must be analyzed by themselves to detect and react to potential threats. These systems are also used to record evidence for investigative purposes. However, people are prone to be affected by mental fatigue as a consequence of performing the same task for a long period of time, resulting in a substantial increase of reaction time, misses and false alarms (Boksem et al., 2005). This fact has been one of the main reasons for the development of smart video surveillance systems.

In order to solve the problem of people's lack of concentration over long periods of time, one feasible solution might be the integration of automatic video analysis techniques. These techniques are based on computer vision algorithms that are capable of perform tasks as simple as detecting movement in a scene, to more complex ones such as classifying and tracking objects. The more advanced the algorithms are, the more sophisticated the system will be, and so will increase its capability to aid the human operator on real-time threat detection.

In this paper we present an architecture for automated video surveillance based on the cloud computing schema. Besides of acquiring video stream from a set of cameras, the approach we are proposing is also capable of extracting information related to certain objects within the scene. The extracted data is interpreted by the system as context information, from which we are able to detect security-relevant events automatically. For testing purposes, we have implemented a prototype of our proposed architecture.

The rest of the paper is organized as follows. Section 2 summarizes the main advances in smart surveillance systems. In Section 3 we describe the FIWARE platform that we are using to deploy our architecture. Section 4 describes the proposed architecture of the smart video surveillance system based on cloud computing. In Section 5 we describe the set of computer vision filters that we have implemented. Section 6 explains how the information flows into the proposed architecture. In Section 7 we describe our implemented system prototype. Finally in Section 8 the Conclusions and Future work are presented.

2. RELATED WORK

Video surveillance and video analysis constitute active areas of research. In general, a video surveillance system includes the following stages: modeling of the environments, detection of motion, classification of moving targets, tracking, behavior understanding and description and fusion of information from multiple cameras (Brémond et al., 2006, Ko, 2008, Hu et al., 2004, Wang et al., 2003).

*Corresponding author

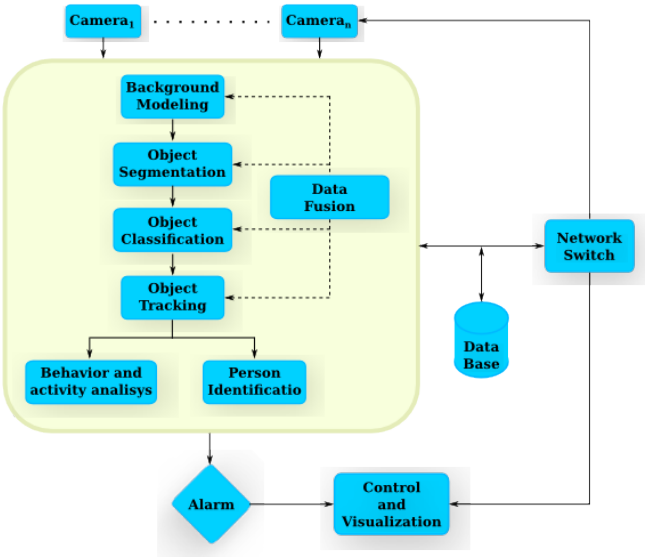


Figure 1. General video surveillance schema (Ko, 2008).

Figure 1 shows the way that all the different stages described above are connected. This general representation can be seen as an active video surveillance system. Currently, there is a wide range of video surveillance systems that have been implemented to address problems such as intrusion detection or traffic surveillance. In works like (Mukherjee and Das, 2013) and (Connell et al., 2004), for example, the authors propose systems which perform human detection and tracking. Also, systems like the one proposed by (Calavia et al., 2012) are capable to detect and identify abnormal situations based on an analysis performed on object movement, then, they use semantic reasoning and ontologies to fire alarms.

On the other hand, there is an emerging research topic related to the integration of cloud-based services to video surveillance systems. In the work presented in (Xiong et al., 2014), for instance, the authors have proposed a general approach for implementing cloud video surveillance systems. Another example of the integration of cloud-based services is presented in (Rodríguez-Silva et al., 2012), in where the authors optimize video streaming transmission based on the network requirements, process and store videos based on cloud computing.

However, most of the work developed so far is focused on solving specific tasks in the context of smart security, either integrate cloud-based services or develop computer vision algorithms, moreover, very few of them propose a model for a complete surveillance system that takes care of both aspects.

For this reason, due to the incapability classic surveillance systems present on monitoring and processing data generated by large scale video surveillance applications, in this paper we propose a general architecture for a smart video surveillance system which integrates cloud-based services and image processing algorithms.

3. FIWARE PLATFORM

FIWARE[1] or FI-WARE is a middleware platform, driven by the European Union. FIWARE was created with the idea of facilitating the development and global deployment of applications for Future Internet. According to its website, FIWARE provides a set of APIs that facilitate the design and implementation of smart applications at several levels of complexity. The API specification of FIWARE is open and royalty-free, where the involvement of users and developers is critical for this platform to become a standard and reusable solution.

FIWARE offers a catalogue that contains a rich library of components known as Generic Enablers (GEs), along with a set of reference implementations that allow developers to instantiate some functionalities such as the connection to the Internet of Things or Big Data analysis.

FIWARE is supported by the Future Internet Public-Private Partnership (FI-PPP) project of the European Union.

4. SYSTEM ARCHITECTURE

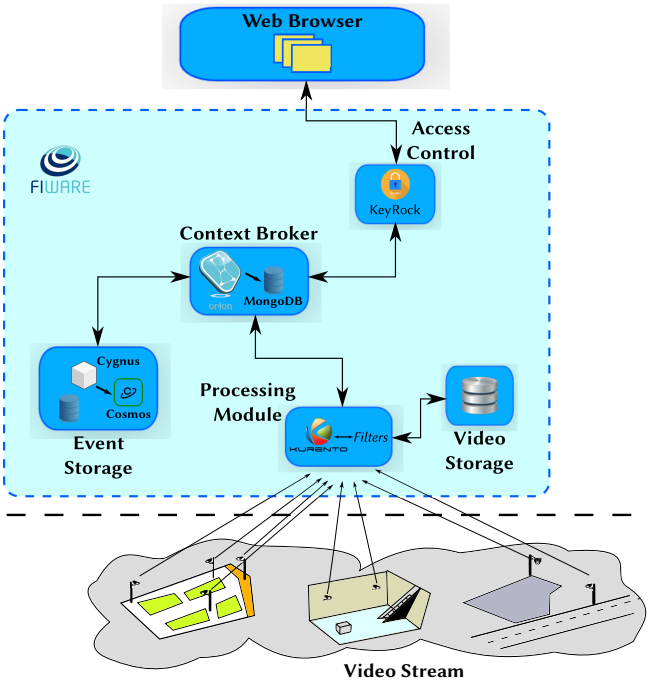


Figure 2. System Architecture.

In this work a system architecture for smart video surveillance based on the idea of cloud computing is proposed. The architecture is composed by five main functional blocks: Access control, Context Broker, Event Storage, Video Storage and Processing module, in Figure 2 the overall architecture of the smart video surveillance system is presented, each block has a unique role in the process of synthesizing data from real-time video stream into a human understandable format.

Access control: The role of this module is to establish a secure connection between the user and the system, while it prevents strangers of gaining access. In other words, here

[1] https://www.fiware.org

is where the system grants predefined specific permissions to each user according to its role. The implementation was made using the KeyRock[2] GE, which is an identity manager, developed by FIWARE, that takes care of a variety of tasks related to cyber security, such as users' access to the network and services, private authentication from users to devices, user profile management, etc.

Context Broker: To implement this module we use the Orion Context Broker[3] (OCB) from FIWARE. The OCB component is a context information manager, thus, it enables the creation, update and deletion of entities, it is also possible to register subscriptions in order for other applications (context consumers) to retrieve the latest version of all the variables that constitute an entity when some event occurs. This component can be seen as the moderator that carries out the communication process between the other modules, so that, once a module has defined the entities it will send and receive, the OCB takes care of the rest.

Event Storage: This module persists the data related to the context information, this information might be an alarm from the system or a simple notification. By saving this information, we can retrieve it for later analysis. In order to implement this block we have used two GE, Cygnus[4] and Cosmos[5]. The first one is in charge of the data persistence, it handles the transfer of information from a given source to a third-party storage, serving as a connector, which is a great feature that increases the flexibility of the system and its scalability if required. While the second one provides a means for BigData analysis, so their users avoid the deployment of any kind of infrastructure.

Video Storage: This block is employed to storage raw video data, so that users have access to the video related to an event detected/stored according to the processing module.

Processing module: This block is conformed by two submodules: Kurento and Computer Vision Filters. The Kurento sub-module provides video streaming from IP cameras through the Kurento Media Server (KMS)[6]. The KMS is based on Media Elements (ME) and Media Pipelines. Media Elements are the modules that perform a specific action on a media stream by sending or receiving media from other elements, while a Media Pipeline is an arrangement of connected ME's, that can either be a linear structure (the output of every ME is connected to a single ME) or a non-linear one (the output of a ME might be connected to several ME's). The ME's used in the implemented processing module were four: WebRtcEndpoint, PlayerEndpoint, Vision Filters and RecorderEndpoint. Figure 3 shows the Media Pipeline we implemented for our architecture prototype, *i.e.*, the logic arrangement in which we connected the four ME. In this pipeline, we get the video stream with the PlayerEndpoint through a rtsp url, after that, the output goes to to the computer vision filters, then, the output of this ME is sent to the WebRTCEndpoint, after that, the processed video is ready to be visualized. Additionally, by using the RecorderEndpoint we are able to store video from the PlayerEndpoint and thus, we give the user the capability to play stored videos any time in the future.

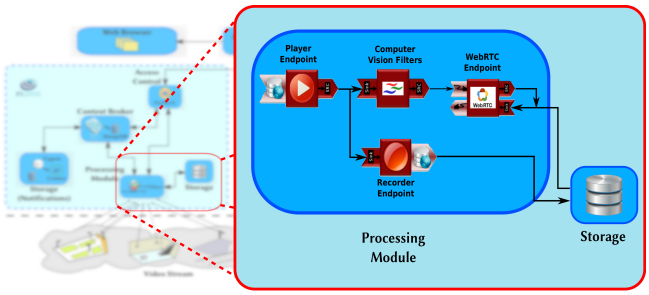


Figure 3. Kurento's Media Pipeline implemented.

Among all of our system's modules, the processing module could be considered as the most relevant one, since it is here where we develop the set of *filters* required to detect all sort of events. In the following section we specify the set of events our system is capable of detecting, by describing the computer vision filters we have implemented so far.

5. COMPUTER VISION FILTERS

In order to extract relevant information about monitored scenes from the incoming video streams, Kurento provides a set of tools that enable the integration of computer vision algorithms to a Media Pipeline. For the implementation of any computer vision procedure, Kurento uses OpenCV (Bradski and Kaehler, 2008) libraries.

In our video surveillance application, we have implemented a set of submodules that are capable of detecting people and vehicles. To do so, three filters were designed, a background subtraction, classification and tracking filters, which are described below.

(i) Background subtraction: Background subtraction is a major preprocessing step in many vision based applications. However, detecting motion based background subtraction is not as trivial nor easy as it may appear at a first glance. In order to cope with such challenge, we integrated to our system SuBSENSE (St-Charles et al., 2015), one of the state of the art background subtraction algorithms. SuBSENSE can be considered as a foreground/background segmentation algorithm, based on the adaption and integration of features known as *local binary similarity patterns* in a background model that, as time goes by, is adjusted using a feedback loop at a pixel level. In other words, what this filter really does is that it takes the incoming video stream from a camera and yields another video stream of binary images, where the white pixels are part of a foreground object (blob) and the pixels colored in black are considered to be part of the background scene.

(ii) Object classification: In the context of computer vision, a classification algorithm enables a system to take actions that require discrete information about a real world scene, such as the identity or category of every entity inside it. In our case, to be able to detect vehicles and people, we have implemented a variation of K-nearest neighbor

[2]https://catalogue.fiware.org/enablers/identity-management-keyrock
[3]https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker
[4]http://fiware-iot-stack.readthedocs.io/en/latest/cygnus/
[5]https://fiware-cosmos.readthedocs.io/en/latest/index.html
[6]http://www.kurento.org/

(Peterson, 2009) algorithm. Once the foreground objects (blobs) have been segmented from the scene by the background subtraction filter, this filter starts by extracting a set of shape features (constituted by the area, aspect ratio and compactness) from every blob in the scene. After this, the features of every object are stacked to form vectors. These vectors are passed to the classification algorithm (KNN) which queries a database in order to assign a class label. Contrary to the classical KNN which assigns an object to the class of the best match, our variation of KNN has an extra criteria that must be satisfied. That is, the similarity value between an input vector and its best match must surpass a threshold value, otherwise, it will be classified as an unknown object. This condition was integrated to reduce the amount of false positive detections by our system. One of the main reasons we integrated KNN for the classification task is that if we want to add new classes of objects we only need to modify the database file, thus, no retraining is required. Moreover, this classification algorithm has turned to be efficient enough for our real-time processing requirements.

(iii) Object tracking: In order for a system to be able to monitor what is going on among a set of entities in a scene within a period of time, this system must have the capability to gather information and persist it so as to relate it to data of following iterations, and so, create context information. In addition to the information extracted on each instant of time, with context information, it is possible to query the system about actions, temporal events and other high level concepts. For this task, we implemented a multi target tracking algorithm, which has no limit on the amount of objects it can track simultaneously and takes as input the binary image provided by the background subtraction sub-module. The algorithm extracts a set of features from each blob (a different set from the one used in the classification filter), and establishes a relation between the blobs in the current frame and those in the previous one. The most important feature this filter adds to our surveillance system is the ability to generate information about objects along time, this could be seen as a description of behavior.

Once we have designed our set of filters, we have to connect them to build logic arrangements so that a specific processing task is performed over the video stream. In Figure 4 we show these logic arrangements.

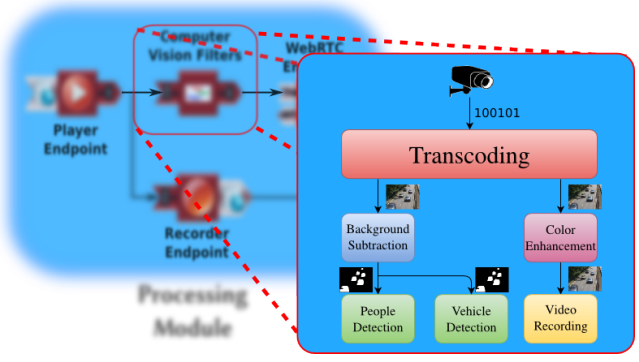


Figure 4. Example of filter logic arrangements based on Kurento's pipeline..

Any filter in the processing module is able to communicate with the OCB. The communication between Kurento's filters and the OCB goes in both directions. The filters send information about objects and events they detect and also perform queries onto variables controlled by other modules or even by the user. Rebroadcasting data from one to many, is one of the context broker's greatest features.

6. SYSTEM WORKFLOW

So far we have described each of the elements that constitute our architecture and what their role is within the system. However, part of the definition of an architecture, in addition to its components, is the description of the relation between each element and the environment. In terms of our model, an internal interaction is when two components of the system are involved, and there is the external one, where an element within the system interacts with an entity that is outside the system, which in our case is the user. In this way, we can fully define the architecture.

As shown in Figure 2, the processing module interacts with other three elements: storage, context broker and cameras. Although the context broker has an important role by managing the data coming in and out the system, due to the context of video surveillance, the processing module could be considered the core module of the architecture.

Every time an entity in the OCB, which is how objects are called, is updated or modified, the context broker sends a notification to every client that has previously subscribed to this entity. In this way, every module keeps track on the information it needs. In Figure 5 we show an example of modules communication, in this case the processing module (through the GE Kurento) sends the event detection to the OCB, while the OCB sends commands to processing modules, such as start recording, add labels to video, etc., this communication is done through *JSON* messages.

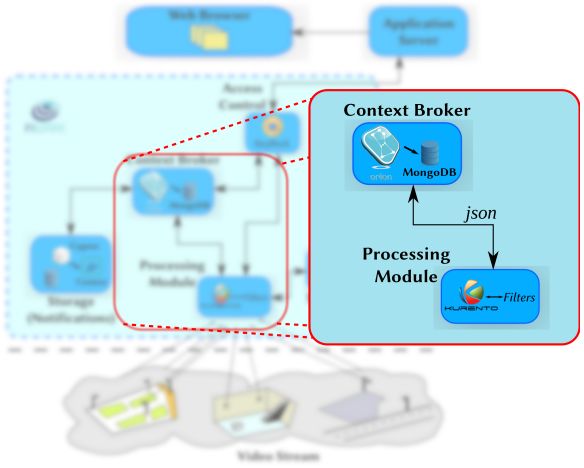


Figure 5. Processing module and OCB communication through JSON messages.

In addition to being in charge of the management of real-time data, the OCB persists information by sending it to the notifications storage module (Cygnus). The data persistence is a necessary step in order for the system to provide a means for the user to query data with respect events that occurred at a given time. In this way, it turns a very

easy task to find in a video a specific event within a defined span of time.

As a final step, our system sends all the detected events as well as the video stream of each camera to the user. In other words, by using this approach we have a video surveillance system as a service.

7. PROTOTYPE

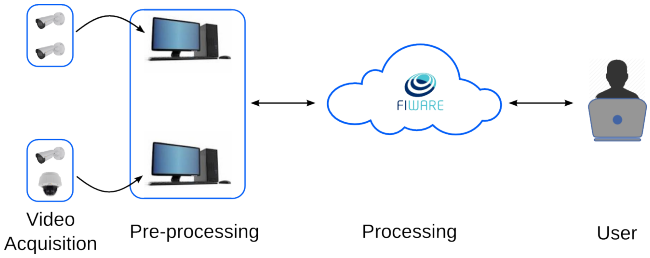
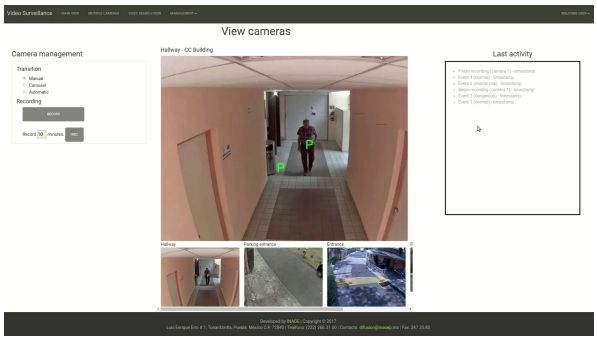


Figure 6. The implemented prototype consists of a set of cameras connected to the network, a couple of desktop computers for pre-processing and a processing system implemented on FIWARE.

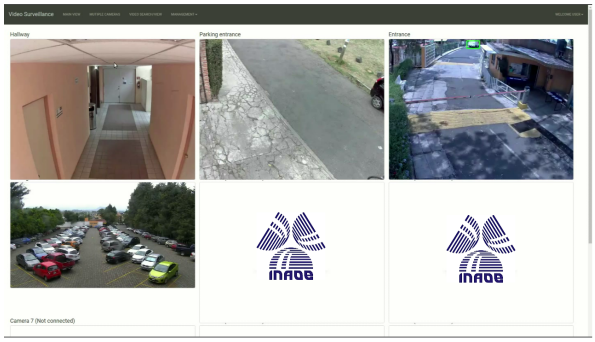
We have designed and implemented a prototype to test our system. For this prototype we are using four ip cameras and two desktop computers for video pre-processing tasks. Functionalities of the prototype are complemented on the cloud through FIWARE GEs.

The proposed prototype can be separated in three stages. The first stage includes the sensors, which in this case are cameras, however, we can include other kind of input devices such as fire and movement detectors, for instance. In this stage, we get the video stream, which is the information we send to the next phase. In the second stage, we proceed to compute the background subtraction, motion detection and object tracking, from this process we get information as position, speed, number of objects, etc. In the last stage, we perform the analysis of the video, which from all the video processing tasks, it is the one that requires the greatest processing capacity. At the end of this process, we get an interpretation of what is happening in the scene; furthermore, in this stage we also store both the video stream and the relevant information for posterior analysis if necessary.

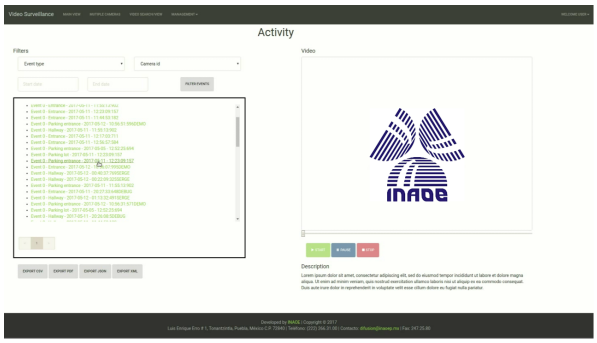
We have also implemented a graphical user interface (see Figure7). The GUI was implemented in Web2Py. Four different tabs are at the user's disposal in order for him to interact with the set of functionalities our system offers. The main tab is used for visualizing a single camera and a historical view of events detected so far. In the multiple tab, the user may visualize all of the cameras integrated in the surveillance system. Within the video search tab, the system allows the user to query videos previously stored by defining a search criteria based on different attributes such as date, type of event, camera id, etc. The management tab, displays the options available for customizing the way detected events are highlighted. It also enables the user to register new cameras.



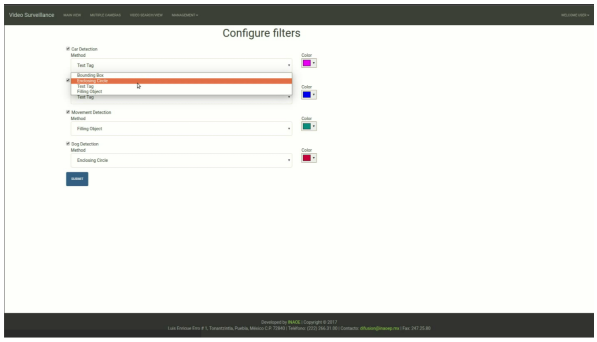
(a) Main tab.



(b) Multiple tab.



(c) Search tab



(d) Management tab

Figure 7. GUI for the video surveillance system prototype.

8. CONCLUSIONS AND FUTURE WORK

To achieve more intelligent video surveillance systems, large amounts of data need to be collected at each instant, and then analyzed to extract useful information to make decisions and create an intelligent response. In this work we have proposed a video surveillance architecture based on

the idea of cloud computing. By using this approach it is possible to provide the video surveillance as a service, this gives us the possibility of having a portable and scalable system that can be used in different scenarios. Furthermore, as a result of the video analysis it is possible to obtain a description of what is happening in a monitored area, and then take an appropriate action based on that interpretation. In addition, with this approach we are also able to add different kind of sensors besides cameras, this gives us the possibility to manage digital devices as in a IoT framework. Our system is based on the middleware FIWARE and has been implemented in a real scenario.

As a future work we want to implement more filters to incorporate them to the processing module. We also want to incorporate another type of sensors such as motion detection sensor, temperature sensor, etc., which we believe would improve the understanding the system has of a given situation. From scalability point of view, as the number of cameras and algorithms increase more computing resources are required. For this, we are also exploring a distributed approach.

ACKNOWLEDGEMENTS

This work was supported in part by FONCICYT (CONACYT and European Union) Project SmartSDK - No. 272727.

REFERENCES

Boksem, M. A., Meijman, T. F. and Lorist, M. M., 2005. Effects of mental fatigue on attention: an erp study. *Cognitive brain research* 25(1), pp. 107–116.

Bradski, G. and Kaehler, A., 2008. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.".

Brémond, F., Thonnat, M. and Zúniga, M., 2006. Video-understanding framework for automatic behavior recognition. *Behavior Research Methods* 38(3), pp. 416–426.

Calavia, L., Baladrón, C., Aguiar, J. M., Carro, B. and Sánchez-Esguevillas, A., 2012. A semantic autonomous video surveillance system for dense camera networks in smart cities. *Sensors* 12(8), pp. 10407–10429.

Connell, J., Senior, A. W., Hampapur, A., Tian, Y.-L., Brown, L. and Pankanti, S., 2004. Detection and tracking in the ibm peoplevision system. In: *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, Vol. 2, IEEE, pp. 1403–1406.

Hu, W., Tan, T., Wang, L. and Maybank, S., 2004. A survey on visual surveillance of object motion and behaviors. *Trans. Sys. Man Cyber Part C* 34(3), pp. 334–352.

Ko, T., 2008. A survey on behavior analysis in video surveillance for homeland security applications. In: *AIPR*, IEEE Computer Society, pp. 1–8.

Mukherjee, S. and Das, K., 2013. Omega model for human detection and counting for application in smart surveillance system. *arXiv preprint arXiv:1303.0633*.

Peterson, L. E., 2009. K-nearest neighbor. *Scholarpedia* 4(2), pp. 1883.

Rodríguez-Silva, D. A., Adkinson-Orellana, L., Gonz'lez-Castano, F., Armino-Franco, I. and Gonz'lez-Martinez, D., 2012. Video surveillance based on cloud storage. In: *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, IEEE, pp. 991–992.

St-Charles, P.-L., Bilodeau, G.-A. and Bergevin, R., 2015. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing* 24(1), pp. 359–373.

Wang, L., Hu, W. and Tan, T., 2003. Recent developments in human motion analysis. *Pattern Recognition* 36(3), pp. 585 – 601.

Xiong, Y.-H., Wan, S.-Y., He, Y. and Su, D., 2014. Design and implementation of a prototype cloud video surveillance system. *Journal of Advanced Computational Intelligence and Intelligent Informatics* pp. 40–47.