A Scalable and Versatile Framework for Smart Video Surveillance

Antonio C. Nazare Jr.
Renato Antonio Celso Ferreira (*Co-advisor*), William Robson Schwartz (*Advisor*)
Department of Computer Science – Universidade Federal de Minas Gerais, Brazil
Email: {antonio.nazare, renato, william}@dcc.ufmg.br

***Abstract*—The large amount of visual data generated by surveillance cameras is usually analyzed manually, a challenging task which is labor intensive and prone to errors. Therefore, automatic approaches must be employed to enable the proper processing of the visual data. The main goal of automated surveillance systems is to analyze the scene focusing on the detection and recognition of suspicious activities. However, these systems are rarely tackled in a scalable manner. With that in mind, this Master's thesis[1] proposed a framework for scalable video analysis called Smart Surveillance Framework (SSF) to allow researchers to implement their solutions to the surveillance problems as a sequence of processing modules that communicate through a shared memory. The framework provides useful features to the researchers, such as memory management to allow handling large amounts of data, communication control among execution modules, predefined data structures specifically designed for the surveillance environment and management of multiple data input. Our experimental results evaluate important aspects of the Smart Surveillance Framework (SSF) and demonstrate the scalability of the framework, the lower overhead caused by the communication between the modules and the shared memory and the high performance of our feature extraction mechanism.**

***Keywords*-Smart Surveillance Framework, Surveillance Systems, Computer Vision Video Analysis, Video Surveillance**

I. Introduction and Motivation

Due to the reduction in prices of cameras and the improvement in network connectivity, the number of surveillance cameras placed in several locations increased significantly in the past few years. If on one hand, a distributed camera network provides visual information in real time covering large areas, on the other hand, the number of images acquired in a single day can be easily in the order of billions, preventing their manual processing and posing an intricate problem for monitoring such areas [1].

While the ubiquity of video surveillance provides safer environments, the monitoring of large amount of visual data is a challenging task when performed manually by human operators since most of the visual data do not present interesting events from the surveillance standpoint, turning it into a repetitive and monotonous task for humans [2], [3]. Hence, automatic understanding and interpretation of activities performed by humans in videos are of great interest since such information can assist the decision making process of security agents [2].

[1]This work summarizes the M.Sc. thesis of the first author.

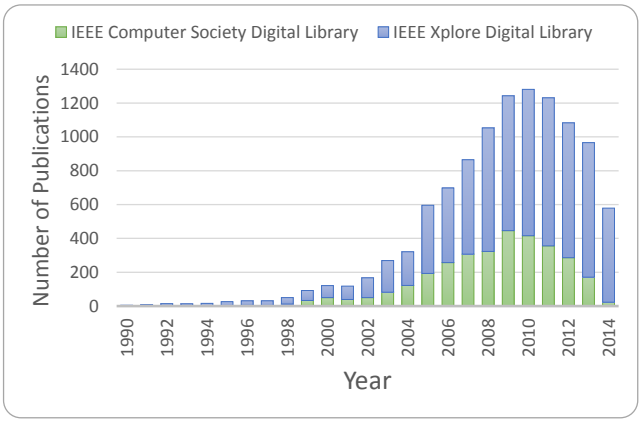


Figure 2. Histogram of publications in IEEE Computer Society Library and IEEE Xplore Digital Library whose metadata contains the keywords *video* and *surveillance* (Adapted from [4] and updated on December 17, 2014).

The addition of automatic understanding and interpretation to surveillance systems does not entail the replacement of human operators as foreseen on several Sci-Fi movies, on the contrary, it aims at supplying additional information to the operator. For instance, instead of a security agent monitoring continually up to 50 screens with live security video feed (task which humans do not present high performance due to the lack of important events during most of the time [5]), an automated system might perform a filtering in the videos and indicate only those video segments more likely to contain interesting activities, such as suspicious activities that might lead to a crime.

Smart visual surveillance systems deal with the real-time monitoring of objects within an environment. The main goal of these systems is to provide automatic interpretation of scenes and understand activities and interactions of the observed agents based on the visual information being acquired. Current research regarding these automated visual surveillance systems tend to combine multiple disciplines, such as computer vision, signal processing, telecommunications, management and socio-ethical studies.

In the last two decades, professionals of industry and researchers have dedicated their studies to improve surveillance systems. To understand the increase of works related to video surveillance and inspired by Huang [4] study, we

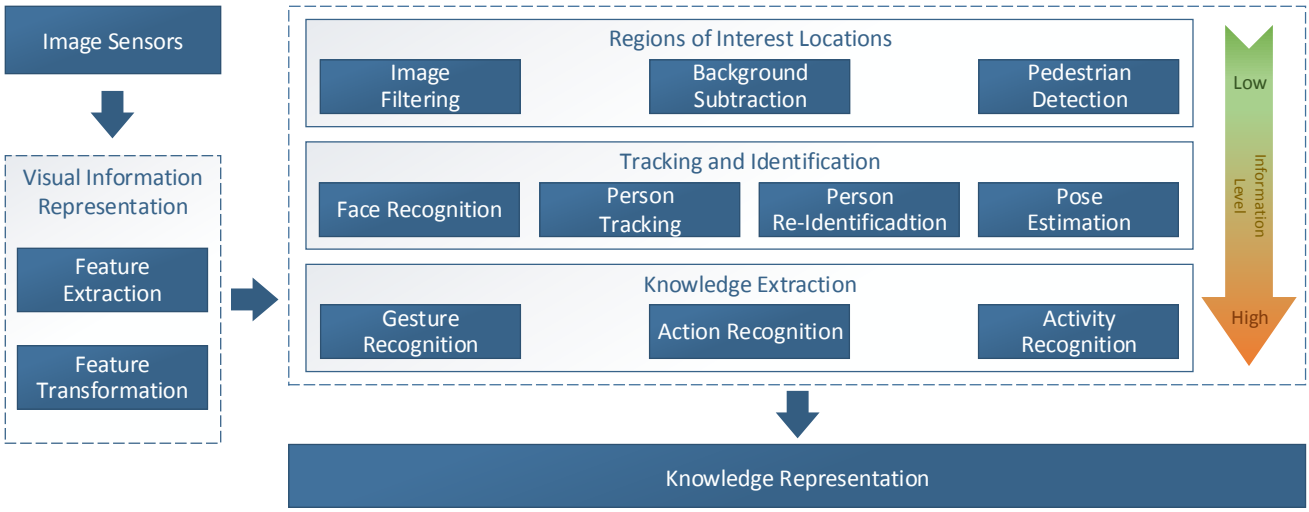


Figure 1. Diagram illustrating the main problems considered in visual surveillance applications, and their dependencies. Visual information is captured by the feature extraction which feeds several modules. The results obtained by solving each problem are employed to perform scene analysis and understanding.

searched the keywords *video* and *surveillance* in IEEE Xplore Digital Library (within metadata only) and the IEEE Computer Society Digital Library (by exact phrase). The findings are shown in Figure 2 as a function of the publication year. The large number of publications in the past ten years indicates that research on surveillance video has been very active.

One of the great challenges of automatic surveillance systems is to interpret what is happening in the scene, a sequence of problems need to be solved, which is highly prone to noise generated during the process. Among the problems are the background subtraction [6], pedestrian detection [7], face recognition [8], pose estimation [9], person tracking [10] and re-identification [11], action recognition [12] and activity recognition [13].

The problems considered in the surveillance domain might be divided into four groups: visual information representation, regions of interest location, tracking and identification, and knowledge extraction. Figure 1 shows these groups and the relationship among the problems within each. While modules located at the top of the diagram define low-level problems, in the sense that they present low dependency to solutions obtained by other problems, *e.g.*, background subtraction and pedestrian detection, modules at the bottom comprise high level problems since they depend on the results of other problems, *e.g.*, action and activity recognition.

The arrow in the right-hand side of Figure 1 represents the dependencies among the problems. For example, to solve the action recognition, one first needs to correctly detect and track the person who is executing an action. Tasks composing this process might be affected by errors propagated along the task chain (e.g., detection errors will affect the tracking of a person, which will prevent the recognition of the action executed by this person). Therefore, it is necessary to solve the tasks in an accurate manner be able to solve problems presenting several dependencies, such as the activity recognition, responsible for making inferences regarding the activities being executed in a scene (e.g., loitering, identification of suspicious collaborations or carjacking).

In the diagram shown in Figure 1, *Visual Information Representation* comprehends tasks aiming at representing the information contained in the visual data, e.g., converting pixel information to a feature space which is more robust to noise and transformations taking place in the video. The goal of the *Regions of Interest Location* is to narrow down efficiently the locations of the scene where information regarding activities taking place can be extracted. Then, once the tasks in the previous category have located the relevant regions in the scene for each frame, the problems in the *Tracking and Identification* category will estimate their trajectories and identify the agents based on information including their appearance or their faces. Finally, after the objects and agents have been located, identified and their trajectories have been estimated, their actions and activities can be recognized, these problems refer to the *Knowledge Extraction* category. All the information collected by executing the tasks will be used to generate a knowledge representation regarding the scene, so that one can use such information to make inferences and perform scene analysis.

Even though each one of these problems present a vast literature, they are usually considered independently such as in currently available evaluation data sets, e.g., the evaluation of face recognition methods is performed using already detected, cropped and aligned faces [14], which cannot be accomplished in real surveillance scenarios where the only inputs are video feeds without annotations. Therefore, dealing with the problems individually does not allow one to identify what are the effects of the results obtained by solving one problem on the following steps in the processing sequence.

Although visual surveillance has been subject to a huge growth, most development frameworks do not couple with the advances in the domain. According to Valera [15], the technological evolution of surveillance systems can be divided

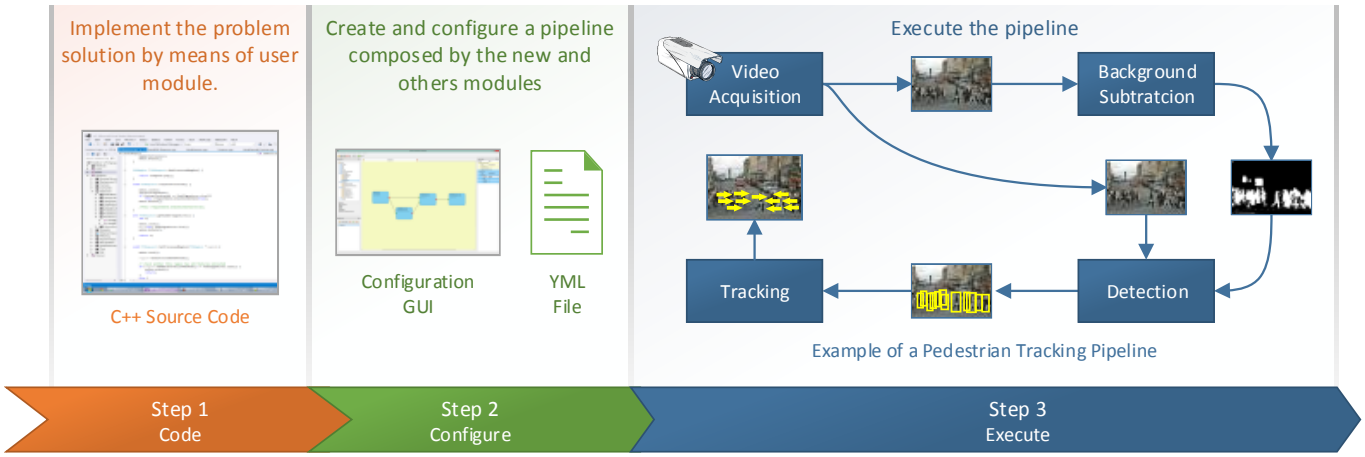


Figure 3. Overview of the Smart Surveillance Framework, an development environment that allows the researcher to implement his/her surveillance algorithms in an integrated manner by setting up a sequence of modules that will be executed in a pipeline. The researcher can take advantage of the transparent communication control provided by a shared memory using surveillance-focused data structures (Section II-A), a feature extraction server (Section II-B) to reduce the computational cost, and a high level reasoning might be performed using information stored in the Complex Query Server (Section II-C).

into three generations, The first generation consists of analogue Closed-Circuit Television (CCTV) which are composed of cameras distributed in the scene and connected to monitors by switches. The second generation is characterized by high performance computers and digital cameras, allowing the development of computer vision algorithms to assist humans in many surveillance tasks. The third generation deals with wide-area scenes and large amount of data acquired by several different types of sensors, leading to more challenging scenarios which require improved solutions. Several surveillance systems of the third generation have been designed and developed both in the industry and in the academia. Among the systems are Knight [16], IBM Smart Surveillance System (S3) [17], Smart Platform [18] and the work proposed by Wang *et al.* [19].

Motivated by the presented issues, this Master's thesis proposed a framework for a scalable video analysis able to readily integrate different computer vision algorithms into a functional surveillance system. This framework, called *Smart Surveillance Framework* (SSF), aims at bringing several improvements providing scalability and flexibility, allowing the users (researchers) to focus only on their application by treating the sequence of problems as a set of processing modules that communicates through data streams, stored in a shared memory.

More specifically, the Smart Surveillance Framework is a development environment in which the researcher can implement and evaluate his/her algorithms related to surveillance in an integrated manner, as illustrated in Figure 3. It is based on execution modules that communicate to each other using data streams controlled by a shared memory. The framework provides the following features to aid the researcher: memory management to allow handling large amounts of data in regular computers; communication control among execution modules; predefined data structures specifically designed for surveillance environment; management of multiple data input, such as cameras or stored videos; feature extraction server to maximize the usage of the processing power available to compute local descriptors; query server to allow high level reasoning and scene understanding; and a configuration interface to help setting up sequences of execution. These features and the framework architecture will be discussed in details in Section II.

The main contributions provided by the development of this Master's thesis are the following:

i) A novel framework to allow the processing of large amounts of data provided by multiple surveillance network cameras;

ii) A platform to compare and exchange research results in which researchers can contribute with modules to solve specific problems;

iii) A framework to allow fast development of new video analysis techniques since one can focus only on their application, by treating the sequence of problems as a whole and consequently providing the lower and higher level application modules.

iv) Creation of a high-level semantic representation of the scene using data extracted by low-level modules to allow the execution of video event analysis based on individual or group activities;

v) A testbed to allow further development on activity understanding since one can focus directly on using real data, instead of annotated data that may prevent the method from working on real environments;

vi) A platform to allow scalable feature extraction that uses the full power of multi-core architectures;

vii) A review of published papers in recent years that discuss the issues and challenges involved in the deployment of modern visual surveillance systems, as well the discussion of similar works to the proposed framework.

This document summarizes the work carried out during the Masters and is organized as follows. Section II describes

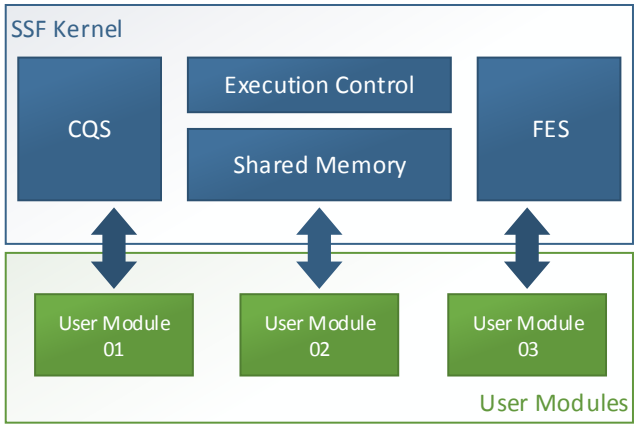


Figure 4. Architecture of the Smart Surveillance Framework.

the proposed Smart Surveillance Framework (SSF). Then, Section III presents our experimental evaluation, and finally Section IV points our final remarks.

II. Smart Surveillance Framework

Written in C/C++, the Smart Surveillance Framework (SSF) is a tool built to provide a set of functionalities to aid researchers not only on the development of surveillance systems, but also on the creation of novel algorithms for problems related to video surveillance.

The SSF has been designed to allow the development of modern surveillance systems, providing features as tools to perform scene understanding, scalability, real-time operation, distributed multi-sensor environment and communication control. The architecture of framework can be divided into two main parts: *user modules* and *SSF kernel*, as illustrated in Figure 4. While the former is where the user implements his/her surveillance and computer vision algorithms, the latter, responsible for controlling data communication, parallelism and data structures, lies outside of the user domain, being accessible only through configuration parameters.

The SSF kernel is composed of the following components. (i) *shared memory*: the backbone of the SSF, it allows the communication among all other components and stores the data generated by user; (ii) *Feature Extraction Server (FES)*: it processes feature extraction requests and return feature vectors to user modules to maximize the occupancy of the processing units available; (iii) *Complex Query Search (CQS)*: this component allows user modules to search for specific data in the shared memory by using Prolog or queries in SQL databases; (iv) *Execution Control*: this component controls the execution of the other SSF components and is responsible for the SSF initialization. In addition, this component has a graphical interface to aid the user to configure the run-time environment.

The user modules are components written by the researchers to solve surveillance and computer vision problems (in fact, any algorithm can be implemented in the user modules). These modules use a well-defined interface to communicate with the the kernel components and the concept of *data stream* to communicate to other modules through specific data types).

The following sections describe the SSF components.

A. Shared Memory

To achieve a flexible and modular software architecture, it is necessary that the modules be designed independently without knowing each other specific interfaces, which would reduce the flexibility when integrating a set of modules to solve a given task. Therefore, to address this constraint, the SSF provides a resource to store data and control the data communication between user modules. Such feature is referred to as *shared memory* and is responsible for the data communication control. This way, the modules only need to know the interfaces provided by the shared memory and not each other specific interfaces.

B. Feature Extraction Server

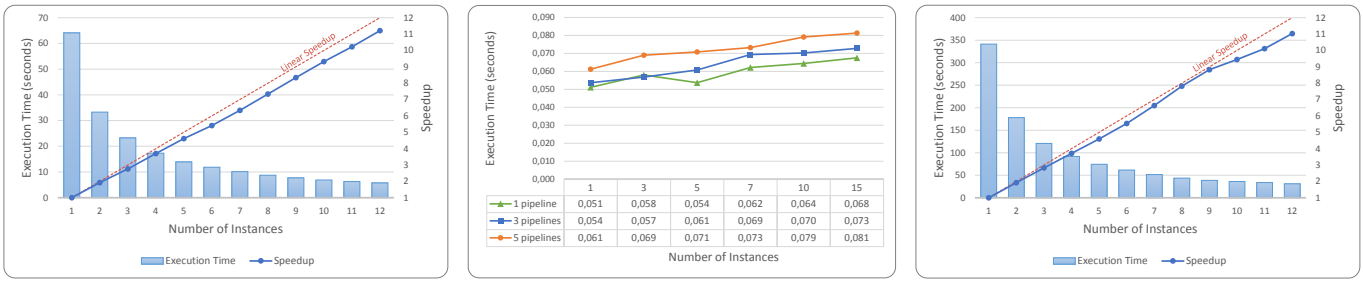
Feature extraction is critical for surveillance systems since several algorithms require feature descriptors as input. However, most feature extraction algorithms are highly time consuming and not suitable for real time applications. To address the feature extraction problem, the SSF provides a powerful tool: the Feature Extraction Server (FES). It allows the feature extraction to be performed using the entire computational power available in the system to maximize the performance (one can use all available CPU cores). More specifically, researchers implement their feature extraction methods based on a template class and the feature extraction server will be responsible for splitting the task among the available processing units.

The feature extraction server relies on an asynchronous approach to receive requests, process them and return feature vectors to the user modules with the objective of maximizing the occupancy of the processing units available. Once a request has been sent to the FES, it does not block the processing being executed in the module, which can continue working while the request is been processed by the FES.

C. Complex Query Server

To search for specific data, such as actions being performed in a given time interval or tracklets intersection of two given subjects, one may retrieve data from the shared memory by implementing the query in a module. However, such approach may be inefficient since the architecture of the shared memory is optimized for simple write and read requests. To allow user modules to search efficiently for specific data in the shared memory, the SSF provides the Complex Query Server (CQS).

CQS is independent of the underlying query/inference solution, for instance Relational or Big Data Databases and logic programming such as Prolog. Therefore, the user modules are not required to know how to write a query in a specific solution. To achieve this independence, the CQS defines a common interface with modules so that each complex query solution underlying must implement this CQS common interface which either may be simplified to allow easily integration with as



(a) Framework Scalability. (b) Communication latency. (c) FES Performance

Figure 5. Examples of computational experiments performed on Smart Surveillance Framework (SSF)

many underlying solutions as possible or may also be complete enough to easily allow complex queries. Any implementation of an underlying query/inference solution in the CQS common interface can be performed by implementing *initialization*, *storing* and *querying* methods.

D. Execution Control

The SSF components that will be used for an execution are chosen by parameter settings, which increases the customization of the framework. The parameters might be supplied via a configuration file or assigned through the Graphical User Interface (GUI). Once the configuration file is provided, the Execution Control is responsible for initializing the remaining components and for assigning values to the parameters.

E. User Modules

The user modules are the framework mechanism where the researcher implements his/her algorithms of typical routines of a surveillance system, such as person detection, background subtraction, face recognition, person tracking and re-identification, and action and activity recognition.

Every module follows the same standard interface, in which the user (researcher) defines its input and output data types and its parameters without specifying which module will provide or receive them. This is done later, in execution time by reading the dependencies from a parameter file (or the GUI), which makes the framework highly flexible and versatile. Once the module is launched, an execution routine (where the user implement his/her method), is called and executed.

Another important feature related to modules, is the creation of execution pipelines – collections of user modules behaving as a single module. A pipeline allows one to group several modules of individual methods in a sequence. Once defined, multiple instances of the pipeline can be launched just by changing their inputs. For instance, one pipeline can be launched to process data from each surveillance camera attached to the system. Such a feature also makes the framework more scalable.

To the user's perspective, the communication between modules does not exist directly because when implementing his/her module, the user requests data types as input without specifying which module will provide it, and provides data types as output also without specifying target modules. The actual communication, controlled by the shared memory, is only set in the execution time when the user specifies the input and output modules. This communication scheme, known as publish-subscribe messaging pattern, allows the reuse of modules as components of applications with different goals and increases the flexibility of the framework once the modules with the same purpose are interchangeable.

III. Experimental Results

A functional version of framework is available to download[2]. The currently version has about $60,000$ lines of code and 90 user modules developed by other researchers, including algorithms for person detection, person tracking, camera control, classifiers and feature extraction approaches. Currently, we are working on a new and improved version that will be available as open source licensed software, along with its documentation and user module examples.

We evaluate three important aspects of the framework proposed in this work: *scalability*, *latency* and *FES performance*[3].

Figure 5a presents an experiment to demonstrate the scalability of the framework. The SSF allows the user to perform parallelization of methods by decomposing a problem into independent sub-problems. Two general methodologies are commonly used. The first, termed *data decomposition*, assumes that the overall problem consists in executing computational operations to one or more data structures and, further, that these data structures may be divided and operated upon. The second, called *task decomposition*, divides the work based on different operations or functions. Here, we present the evaluation of data decomposition on SSF. In this experiment, the sequential method has been implemented as a single SSF module and replicated n times, where n represents the number of cores used. Each instance of the method is responsible for processing $100/n$ images from the data set, where n was varied from 1 to 12 and each experiment was executed ten times. Figure 5a reports the average execution time and speedup achieved by the data decomposition approach. According to Figure 5a, it is advantageous to use of the framework to parallelize the processing of a considerable number of images.

[2]The SSF can be requested at http://www.ssig.dcc.ufmg.br/ssf/

[3]Due to the lack of space, we presented only a subset of the results. The complete experiment set is available in the M.Sc. thesis (https://www.dcc.ufmg.br/pos/cursos/defesas/1729M.PDF)

The speedup obtained by the data decomposition approach is very close to linear, demonstrating that the communication overhead caused by the SSF is minimal.

To evaluate the overhead caused by the communication between the modules and the shared memory, we conducted an experiment in which an image (SSF frame data type) was transmitted through a certain number of modules. For that, a pipeline with n modules was created and each module just forwards the image frame (without performing any processing) to the next module. The time elapsed between the instant at which the first and the last module of the pipeline (Modules 01 and n, respectively) performed the reading of the image was computed to estimate the data latency. According to results shown in Figure 5b, the overhead caused by the increased number of modules connected to the shared memory at the same time is low, even when multiple pipelines are instantiated simultaneously. Although this overhead exists, it is negligible when compared with the processing time of the data, usually orders of magnitude higher.

The Figure 5c presents a experiment conducted to evaluate the performance of the Feature Extraction Server (FES). This evaluation was conducted using a widely employed features extraction methods: Gray-Level Co-occurrence Matrix (GLCM) [20]. According to Figure 5c, it is possible to observe an improvement in the computational performance as a function of the number of instances used in the FES, which demonstrates the advantage of its usage in multi-core environments. The speedup achieved with the GLCM method presents a linear growth, demonstrating the scalability of the FES for computationally expensive methods.

IV. Conclusion

This Master's thesis proposed a novel framework to allow further development of Computer Vision techniques and surveillance applications. The architecture of the Smart Surveillance Framework (SSF) allows simultaneous execution of multiple user modules that can be developed independently since they have communication and synchronization through a shared memory, which contributes to the scalability and flexibility. The proposed framework will also allow researchers to provide their methods (implemented as modules) to be used by other researchers to compare their results.

Scientific Publications

As a result of this Master's thesis, we published two papers on international conferences and submitted another to a scientific journal (currently in under review after a major revision). The following list provides references to these documents.

1) A. C. Nazare, C. E. Santos, R. Ferreira, and W. R. Schwartz, "*Smart surveillance framework: A versatile tool for video analysis*," in Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV), 2014, pp. 753–760. (Qualis B1)
2) A. C. Nazare, R. Ferreira, and W. R. Schwartz, "*Scalable feature extraction for visual surveillance*,"" in Proceedings of Iberoamerican Congress on Pattern Recognition (CIARP), 2014, pp. 375–382. (Qualis B2)
3) A. C. Nazare and W. R. Schwartz, "*A scalable and flexible framework for smart video surveillance*," Journal of Computer Vision and Image Understanding, 2015. (*Under Review*). (Qualis A1)

Acknowledgments

This work was supported by the CNPq and FAPEMIG, Brazilian funding agencies.

References

[1] F. Porikli, F. Bremond, S. Dockstader, J. Ferryman, A. Hoogs, B. Lovell, S. Pankanti, B. Rinner, P. Tu, and P. Venetianer, "Video surveillance: Past, present, and now the future," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 190–198, 2013.

[2] A. Hampapur, "Smart video surveillance for proactive security," *IEEE Signal Processing Magazine*, vol. 25, no. 4, pp. 136–134, 2008.

[3] A. C. Davies and S. A. Velastin, "A progress review of intelligent CCTV surveillance systems," in *Proceedings of IEEE Intelligent Data Acquisition and Advanced Computing Systems (IDAACS 2007)*, 2007, pp. 417–423.

[4] T. Huang, "Surveillance video: The biggest big data," *Computing Now*, vol. 7, no. 2, 2014.

[5] G. J. Smith, "Behind the screens: Examining constructions of deviance and informal practices among CCTV control room operators in the UK," *Surveillance & Society*, 2004.

[6] M. Piccardi, "Background subtraction techniques: A review," in *Proceedings of International Conference on Systems, Man and Cybernetics (ISMC 2004)*, vol. 4, 2004, pp. 3099–3104.

[7] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–61, 2012.

[8] X. Zhang and Y. Gao, "Face recognition across pose: A review," *Pattern Recognition*, vol. 42, no. 11, pp. 2876–2896, 2009.

[9] R. Poppe, "Vision-based human motion analysis: An overview," *Computer Vision and Image Understanding*, vol. 108, no. 1-2, pp. 4–18, 2007.

[10] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[11] A. Bedagkar-Gala and S. K. Shah, "A survey of approaches and trends in person re-identification," *Image and Vision Computing*, vol. 32, no. 4, pp. 270–286, 2014.

[12] R. Poppe, "A survey on vision-based human action recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010.

[13] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys*, vol. 43, no. 3, pp. 1–43, 2011.

[14] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, October 2007.

[15] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: A review," *IEE Proceedings - Vision, Image, and Signal Processing*, vol. 152, no. 2, p. 192, 2005.

[16] M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," *IEEE Multimedia*, vol. 14, no. 1, pp. 30–39, 2007.

[17] Y. L. Tian, L. Brown, A. Hampapur, M. Lu, A. Senior, and C. F. Shu, "IBM smart surveillance system (S3): Event based video surveillance system with an open and extensible framework," *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 315–327, 2008.

[18] W. Xie, Y. Shi, G. Xu, and Y. Mao, "Smart platform - a software infrastructure for smart space (SISS)," in *Proceedings of IEEE International Conference on Multimodal Interfaces (ICMI 2002)*, 2002, pp. 429–434.

[19] G. Wang, L. Tao, H. Di, X. Ye, and Y. Shi, "A Scalable Distributed Architecture for Intelligent Vision System," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 91–99, 2012.

[20] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 6, pp. 610–621, 1973.