A SCALABLE AND VERSATILE FRAMEWORK FOR SMART VIDEO SURVEILLANCE

ANTONIO CARLOS DE NAZARÉ JÚNIOR

A SCALABLE AND VERSATILE FRAMEWORK FOR SMART VIDEO SURVEILLANCE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: WILLIAM ROBSOM SCHWARTZ
COORIENTADOR: RENATO ANTONIO CELSO FERREIRA

Belo Horizonte
Setembro de 2014

ANTONIO CARLOS DE NAZARÉ JÚNIOR

A SCALABLE AND VERSATILE FRAMEWORK FOR SMART VIDEO SURVEILLANCE

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: WILLIAM ROBSOM SCHWARTZ
CO-ADVISOR: RENATO ANTONIO CELSO FERREIRA

Belo Horizonte
September 2014

Ficha catalografica elaborada pela Biblioteca do ICEx – UFMG

Nazaré Júnior, Antonio Carlos de

N335s A Scalable and Versatile Framework for Smart Video Surveillance / Antonio Carlos de Nazaré Júnior. — Belo Horizonte, 2014

xxvi, 61 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais - Departamento de Ciência da Computação

Orientador: William Robsom Schwartz

Coorientador: Renato Antonio Celso Ferreira

1. Computação – Teses. 2. Visão por computador – Teses. 3. Gravações de video - Sistemas de segurança – Teses. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*82.10(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A scalable and versatile framework for smart video surveillance

ANTONIO CARLOS DE NAZARE JUNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. WILLIAM ROBSON SCHWARTZ - Orientador
Departamento de Ciência da Computação - UFMG

PROF. RENATO ANTÔNIO CELSO FERREIRA - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. ARNALDO DE ALBUQUERQUE ARAÚJO
Departamento de Ciência da Computação - UFMG

PROF. ERICKSON RANGEL DO NASCIMENTO
Departamento de Ciência da Computação - UFMG

PROF. RICARDO DA SILVA TORRES
Instituto de Computação - UNICAMP

Belo Horizonte, 01 de setembro de 2014.

To Ronaldo Ferreira da Cunha, for believing in me from the beginning.

Acknowledgments

I am extremely grateful to my family, specially my parents Antonio Carlos de Nazaré and Inês Conceição Reis de Nazaré for their continuous and unconditional love, for always believing in me, and for their support in my decisions and I am very grateful to Flávia Alvarenga, for her patience, companionship and love throughout my course.

I also would like to express my sincere gratitude to my advisor Prof. William Robson Schwartz. His entrepreneurship, experience, sincerity, critical view, and focus on results helped shape this thesis, and contributed considerably to my path towards an academic and research career. I am also grateful to Prof. Renato Ferreira, my co-advisor, for the contributions to the development of this Master's Thesis.

I thank my labmates at the SSIG, in particularly Victor Hugo Cunha de Melo, Cássio Elias Jr. and Marco Túlio Alves, for the stimulating discussions and for the sleepless nights we were working before deadlines. Also, I thank my colleagues in Federal University of Minas Gerais: Suellen Almeida, Itamar Hata, Roberto Oliveira, Angelo Assis, Thales Filizola, Alex de Sá, Carlos Caetano, Alessandro Sena, Renato Miranda, Heitor Motta, Phillipe Samer, Rosklin Juliano, Samuel Evangelista, Sabir Ribas and Bruno Coutinho.

A special thanks to David Menotti, for the sincere friendship.

Lastly, I thank the partial support given by the Brazilian government, particularly the CAPES and CNPQ for the financial support.

"Divide each difficulty into as many parts as is feasible and necessary to resolve it."
(René Descartes)

Abstract

The availability of surveillance cameras placed in public locations has increased vastly in the last years, providing a safe environment for people at the cost of huge amount of visual data collected. Such data are mostly processed manually, a task which is labor intensive and prone to errors. Therefore, automatic approaches must be employed to enable the processing of the data, so that human operators only need to reason about selected portions.

Focused on solving problems in the domain of visual surveillance, computer vision problems applied to this domain have been developed for several years aiming at finding accurate and efficient solutions, required to allow the execution of surveillance systems in real environments. The main goal of such systems is to analyze the scene focusing on the detection and recognition of suspicious activities performed by humans in the scene, so that the security staff can pay closer attention to these preselected activities. However these systems are rarely tackled in a scalable manner.

Before developing a full surveillance system, several problems have to be solved first, for instance: background subtraction, person detection, tracking and re-identification, face recognition, and action recognition. Even though each of these problems have been researched in the past decades, they are hardly considered in a sequence. Each one is usually solved individually. However, in a real surveillance scenario, the aforementioned problems have to be solved in sequence considering only videos as the input.

Aiming at the direction of evaluating approaches in more realistic scenarios, this work proposes a framework called Smart Surveillance Framework (SSF), to allow researchers to implement their solutions to the above problems as a sequence of processing modules that communicates through a shared memory.

The SSF is a C++ library built to provide important features for a surveillance system, such as a automatic scene understanding, scalability, real-time operation, multi-sensor environment, usage of low cost standard components, runtime

re-configuration, and communication control.

Resumo

A disponibilidade de câmeras de vigilância dispostas em locais públicos tem aumentado significativamente nos últimos anos, provendo um ambiente seguro paras as pessoas ao custo de uma enorme quantidade de dados visuais coletada. Estes dados são, em sua maioria, processados manualmente, uma tarefa que é trabalhosa e propensa a erros. Entretanto, é desejável que abordagens automáticas possam ser utilizadas no processamento dos dados, de modo que os operadores humanos necessitem tomar decisões apenas em determinados momentos.

Focados em solucionar problemas no domínio de vigilância visual, técnicas de visão computacional aplicadas a este domínio têm sido desenvolvidas durante vários anos com o objetivo de encontrar soluções precisas e eficientes, necessárias para permitir a execução de sistemas de vigilância em ambientes reais. O principal objetivos destes sistemas é a análise de cenas focando na detecção e reconhecimento de atividades suspeitas efetuadas por humanos, para que a equipe de segurança possa focar sua atenção nestas atividades pré-selecionadas. Entretanto estes sistemas são raramente escaláveis.

Antes de desenvolver um sistema de vigilância completo, é necessário resolver vários problemas, por exemplo: remoção de fundo, detecção de pessoas, rastreamento e re-identificação, reconhecimento de faces e reconhecimento de ações. Mesmo que cada um destes problemas tenham sido estudado nas últimas décadas, eles são dificilmente considerados como uma sequência. Cada um é geralmente solucionado de forma individual. No entanto, em um ambiente real de vigilância, os problemas citados precisam ser solucionados em ordem, considerando apenas o vídeo como a entrada.

Com o objetivo de avaliar abordagens em um cenário mais realista, este trabalho propõe um *framework* chamado *Smart Surveillance Framework* (*SSF*), que permite os pesquisadores a implementar suas soluções para os problemas acima citados como uma sequência de módulos de processamento que se comunicam por meio de uma memória compartilhada.

O *SSF* é uma biblioteca *C++* desenvolvida para prover características importantes a um sistema de vigilância como: uma interpretação automática das cenas, escalabilidade, operações em tempo real, ambientes multi-sensores, utilização de componentes padrões de baixo custo, reconfiguração em tempo de execução e controle da comunicação.

List of Figures

List of Tables

List of Acronyms

CCTV Closed-Circuit Television

CQS Complex Query Server

FES Feature Extraction Server

FPS Frames per Second

GLCM Gray-Level Co-occurrence Matrix

GPGPU General Purpose Graphics Processing Unit

GUI Graphical User Interface

HOG Histogram of Oriented Gradients

LAP Looking at People

OpenCV Open Source Computer Vision Library

PTZ Pan–tilt–zoom

S3 IBM Smart Surveillance System

SIFT Scale Invariant Feature Transformation

SISS Software Infrastructure for Smart Space

SQL Structured Query Language

SSD solid-state drive

SSF Smart Surveillance Framework

STL C++ Standard Template Library

VSAM Video Surveillance and Monitoring

Contents

Chapter 1

Introduction

Due to the reduction in prices of cameras and the increase 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, which complicates the storage of all data and prevents their manual processing, posing a problem for monitoring such areas [Porikli et al., 2013].

While the ubiquity of video surveillance is advantageous for protection since it provides safer environments, the monitoring of such large amount of visual data is a challenging task when performed manually by a human operator. In addition, most of the visual data do not present interesting events from the surveillance standpoint, turning it into a repetitive and monotonous task for humans [Hampapur, 2008; Davies and Velastin, 2007]. Hence, automatic understanding and interpretation of activities performed by humans in videos present great interest because such information can assist the decision making process of security agents [Hampapur, 2008]. For instance, instead of a security agent monitoring continually about 50 screens with live security video feed (tasks which humans do not present high performance due to the lack of important events during most of the time), an automated system might perform a filtering in the videos and indicate only those video segments that are 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 actions and interactions of the observed agents based on the visual information 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. Nevertheless, there is still a lack of contributions from the field of system engineering to the research [Valera and Velastin, 2005].

Humans are the main focus in the surveillance since they are the agents that perform actions that change the state of the scene. For instance, a person may interact with objects in the scene to execute a task, such as the removal of an object from a vehicle, or interact with other people to accomplish a goal, which may characterize a suspicious activity. Therefore, the design of processing methods focusing on humans is extremely important to being able to determine what is the role of each person in the scene so that responsibilities can be attributed, for example, to determine which subjects have been involved in a specific activity.

A sequence of problems have to be solved before one is able to analyze activities being performed in a video. Among them are the background subtraction [Piccardi, 2004], pedestrian detection [Dollár et al., 2012], face recognition [Zhang and Gao, 2009], tracking and re-identification [Bedagkar-Gala and Shah, 2014], and action recognition [Poppe, 2010]. All these problems present several solutions in the literature, however, they are usually treated individually, which is not suitable for applying in to real surveillance systems where the only inputs are video feeds without annotations such as in current available datasets, *i.e.*, the evaluation of face recognition methods is performed using already detected faces, which is not the case in surveillance scenarios.

To allow researchers to evaluate their methods in more realistic scenarios, this work proposes a framework called SSF[1]. This framework is composed of a shared memory structure and a set of independent processing modules that communicates through data written and read from the shared memory. One module is fed with data provided by another module in both synchronous or asynchronous way, allowing the establishment of a sequence of execution. Therefore, one can use already implemented modules to solve some of the problems and implement his/her own module to solve a specific problem. An advantage is that modules can be implemented individually without knowledge regarding the implementation details, such as internal structures or input/output interfaces of other modules. In other words, the only external information that a module needs to report to the framework is data types it will consume and produce. Thus, the inputs and outputs of other modules are irrelevant to it. Figure 1.1 shows how the modules are indepen-

[1]The SSF is available for download at http://www.ssig.dcc.ufmg.br/ssf/

dent of each other. The details of framework will be discussed in the next chapters.

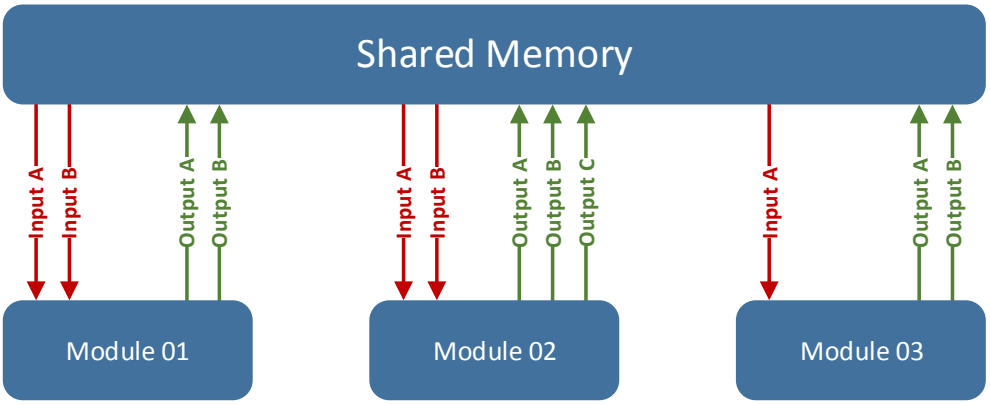


Figure 1.1: The modules must know only their inputs and outputs since there is no direct communication between modules, but only through the shared memory, which makes the framework more flexible.

1.1 Motivation

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, Huang [2014] searched the keywords *video* and *surveillance* in IEEE Xplore Digital Library[2] and the IEEE Computer Society Digital Library[3]. Figure 1.2 shows a histogram of theses publications as a function of the year. The large number of publications in the past ten years indicates that research on surveillance video is very active.

Although visual surveillance has been subject to a huge growth, most frameworks for developing methods and applications for this research area do not address some problems in a comprehensive manner. Problems such as scalability and flexibility, described in more details in the following chapters, were the motivation for developing this Master's Thesis.

The small number of frameworks that are open and focus on visual surveillance usually require a steep learning curve. In addition, with the contemporary advances in video sensors and increasing availability of network cameras allowing the deployment of large-scale surveillance systems, distributed in a wide coverage area, the design of smart and scalable surveillance system remains a research prob-

[2] http://www.ieeexplore.ieee.org/
[3] http://www.computer.org/csdl/

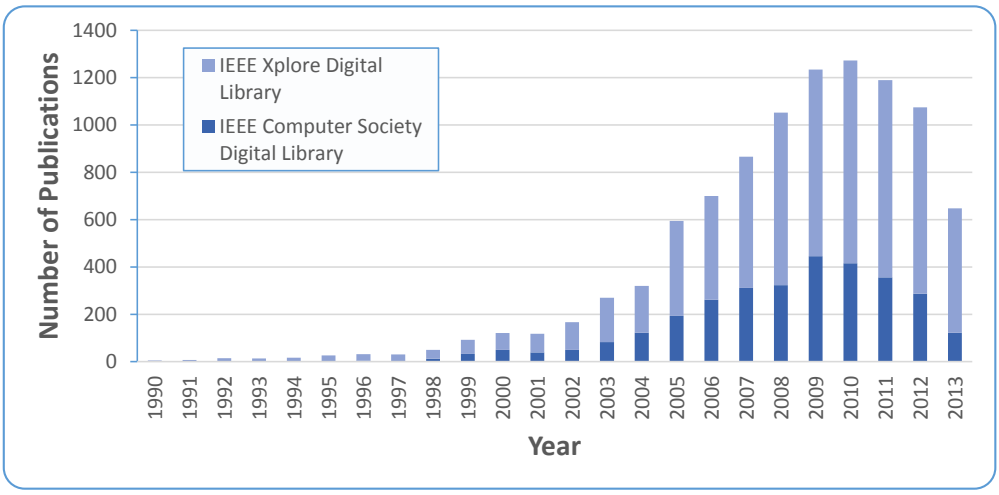


Figure 1.2: Histogram of publications in IEEE Computer Society Library and IEEE Xplore Digital Library whose metadata contains the keywords *video* and *surveillance* (Adapted from [Huang, 2014]).

lem: how to design scalable video surveillance systems considering aspects related to processing power, memory consumption and network bandwidth?

In general, when the researchers work with high-level problems, they have to deal with a sequence of problems. These researchers present several solutions in the literature, but in general, they treat the problems individually. Hence, to find the best composition of such problems, they need to spend time working with these applications first, instead of working directly with their application of interest. For instance, before approaching the individual action recognition problem, the researcher usually have to perform pedestrian detection to locate each pedestrian in the image and, only after that, the approach for action recognition may be employed. Therefore, dealing with the problems individually does not allow to find out what are the effects of the processing on the following steps.

It is desirable to employ an automatic mechanism to test various system components and to enable the comparison with other methods already developed. Such a mechanism is very important to the research community since it will facilitate comparison and validation of algorithms usually employed in visual surveillance applications.

Considering the aforementioned aspects, this Master's Thesis was developed with the objectives listed in the next section.

1.2 Dissertation's Goal

This work proposes a framework for a scalable video analysis able to readily integrate different computer vision algorithms into a functional surveillance system of third generation (the third generation of surveillance systems is presented in Section 2.2.1).

The Smart Surveillance Framework (SSF) aims to bring several improvements providing scalability and flexibility, allowing the users to focus only on their application by treating the sequence of problems as a module set which communicates through a shared memory.

The framework will also be an important tool for the research community, since it makes easier to compare and evaluate algorithms used in visual surveillance applications.

1.3 Contributions

The main contributions provided by the development of the SSF are the following:

- A novel framework to allow the processing of large amounts of data provided by multiple surveillance network cameras;
- A platform to compare and exchange research results in which researchers can contribute with modules to solve specific problems;
- A framework to allow fast development of new video analysis techniques since one can focus only on his/her specific task;
- Creation of a high-level semantic representation of the scene using data extracted by low-level modules to allow the execution of activity recognition;
- 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;
- A scheme to allow scalable feature extraction that uses the full power of multicore architectures;

Another important contribution is 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.

Finally, the SSF may also contribute to improve teaching and learning activities related to computer vision and image processing, for instance in introductory courses, because the modularization of problems enables the identification and characterization of the steps involved in diverse application domains, which help instructors and students in keeping their focus on specific subjects.

During the development of this work, we have produced two technical papers which have been submitted for publication. The following list provides references to these documents.

- **Published:** Nazare, A. C., Santos, C. E., Ferreira, R., and Schwartz, W. (2014). *Smart surveillance framework: A versatile tool for video analysis*. In IEEE Winter Conference on Applications of Computer Vision (WACV 2014).
- **Accepted:** Nazare, A. C., Ferreira, R., and Schwartz, W. (2014). *Scalable Feature Extraction for Visual Surveillance*. In Iberoamerican Congress on Pattern Recognition (CIARP 2014).

In addition to these publications, a tutorial on the SSF will be presented during the Conference on Graphics, Pattern and Images (SIBGRAPI 2014). The acceptance of this tutorial also resulted in an invitation for publication of a survey on Revista de Informática Teórica e Aplicada (RITA).

1.4 Dissertation Organization

This dissertation is organized into the following chapters. Chapter 2 reviews the published papers in the past years about the issues and challenges on visual surveillance systems. Chapter 3 describes the proposed Smart Surveillance Framework (SSF). Chapter 4 presents our experimental evaluation. Finally, Chapter 5 points our final remarks.

Chapter 2

Related Works

Several works related to video surveillance have been proposed in the past years. In this chapter, we review mainly works that focus on developing of visual surveillance applications. First, Section 2.1 presents the most common problems tackled in visual surveillance, as well as the relationship among these problems. In Section 2.1, is also focus to feature extraction problem, which is approached by one of the tools provided by the proposed framework, the Feature Extraction Server (FES), discussed in Section 3.3. Then, Section 2.2 presents a review of published papers in recent years that discuss the issues and challenges involved in the deployment of modern visual surveillance systems and discusses works similar to the proposed framework.

2.1 Visual Surveillance

Since interactions among humans provide relevant information for activity understanding, the analysis of images and videos involving humans (application domain known as Looking at People (LAP) [Gavrila, 1999]) presents large interest of the research community, being widely employed to applications such as visual surveillance, biometrics and forensics. In this scope, solving computer vision problems such as feature extraction [Li and Allinson, 2008], background subtraction [Piccardi, 2004], pedestrian detection [Dollár et al., 2012], face recognition [Zhang and Gao, 2009], person tracking [Yilmaz et al., 2006], person re-identification [Bedagkar-Gala and Shah, 2014], gesture recognition [Mitra and Acharya, 2007], pose estimation [Poppe, 2007], action recognition [Poppe, 2010], and activity recognition [Aggarwal and Ryoo, 2011] is fundamental to model interactions among agents to understand high-level activities performed in a scene under surveillance.

According to the taxonomy described in Nazare et al. [2014], the problems above might be divided into four groups: visual information representation, regions of interest location, tracking and identification, and knowledge extraction, summarized in Table 2.1. Figure 2.1 shows these groups and the relationship among the problems. 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.

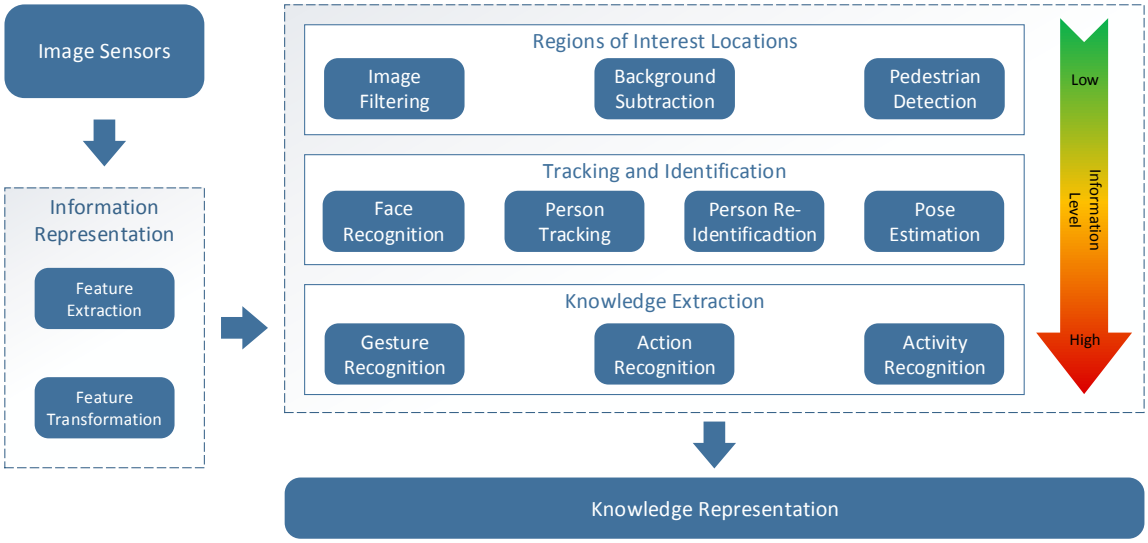


Figure 2.1: Diagram illustrating the main problems considered in visual surveillance applications, and their dependencies. Visual information is captured by the feature extraction which feeds several modules. The results obtained by each module are employed to perform scene analysis and understanding. Adapted from [Nazare et al., 2014].

The arrow in the right-hand side of Figure 2.1 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 so that one will 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).

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 main tasks related to this category are feature extraction and feature space transformation. Even though it is not shown in the diagram of Figure 2.1, to maintain the readability, the majority of the tasks depends on the feature extraction.

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. A motivation for locating regions of interest is to reduce the computational cost and therefore to focus the processing power on the higher level processing tasks. Among the tasks in this category are image filtering (salience detection), background subtraction and pedestrian detection.

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. Such information will be necessary later for recognizing which actions an agent has performed over the time, for instance.

The last category, referred to as *Knowledge Extraction*, deals with problems responsible for extracting high level knowledge from the scene. Therefore, once the objects and agents have been located, identified and their trajectories have been estimated, their actions will be recognized so that collaborations among agents characterizing suspicious activities can be recognized.

Besides the aforementioned categories, the *Knowledge Representation* is an important component in a surveillance system. It is responsible for building a scene representation based on the results of each problem so that one can use such information to make inferences and perform scene analysis.

The final stages in a surveillance system are storage and retrieval. In the past years, many research has been done in how to store and retrieve all the obtained surveillance information in an efficient manner, especially when it is possible to have different data formats and types of information to retrieve [Valera and Velastin, 2005]. Among them, we can cite the works published in [Hampapur et al., 2007; Choe et al., 2013].

The framework developed in this work has been designed to allow researchers to tackle with the problems shown in Figure 2.1 in such a way that the results achieved by solving these problems feed an inference system and the knowledge can be used to understand the scene and the activities performed by the agents (persons).

Visual Information Representation		
Overview	Since videos and images provide only pixels, this first category is responsible for converting this representation to a feature space by employing a data transformation referred to as feature extraction. The resulting feature space is usually transformed due to its high dimension or need for a more flexible representation.	
Problems	- Local Feature Descriptors	- Feature Space Transformation
Regions of Interest Location		
Overview	Problems on this category are responsible for locating the objects or regions of interest aiming at reducing the search space for the problems on the higher level categories. The main goal of the methods to solve these problems is to perform as fast and as accurately as possible so that the algorithms in the next categories can focus only on the relevant parts of the scene.	
Problems	- Filtering Regions of Interest - Background Subtraction	- Pedestrian Detection
Tracking and Identification		
Overview	Once the agents and relevant objects have been located, the algorithms on this class are responsible for providing their identification and trajectories in the scene based on information provided by multiple cameras that are capturing the scene.	
Problems	- Person Tracking - Person Re-identification	- Face Recognition - Pose Estimation
Knowledge Extraction		
Overview	This category comprises problems aiming at obtaining relevant information of the scene that will allow the security personnel to receive high level information regarding events such as suspicions activities and agent's intentions, which will aid in the decision making process.	
Problems	- Gesture Recognition - Action Recognition	- Activity Recognition

Table 2.1: Overview of computer vision problems applied to visual surveillance.

2.1.1 Local Feature Descriptors

The visual information contained in an image (or video) can only be accessed through its pixels, but the direct use of pixels presents undesired effects such as being affected by noise and illumination changes. Therefore, many general classes of low-level descriptors have been proposed [de Siqueira et al., 2013; Nascimento et al., 2012; Randen and Husoy, 1999; Li and Allinson, 2008; van de Sande et al.,

2010; Mikolajczyk and Schmid, 2005; Zhang et al., 2007; Gauglitz et al., 2011] focusing on different image characteristics, such as color, shape, and texture.

Local feature descriptors are used to describe local regions in the images. Two main approaches are employed to sample these regions. The first is based on the detection of interest points (discriminative points located usually in corners of objects detected by feature detectors [Mikolajczyk and Schmid, 2005; Li and Allinson, 2008]), and the sampling of regions around them. The second approach simply samples local regions from the image in a uniform manner. Even though the latter approach generates more data, it tends to miss information from regions that cannot be captured by the feature detector. At the end, each local regions will be described by a feature vector according to the extraction method being employed.

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. Researchers have also devoted their studies to optimize the feature extraction methods. One of the early works was proposed by Viola and Jones [2001], the integral image, an intermediate representation that allows faster computation of rectangle features. Dollar et al. [2009] proposed linear and non-linear transformations to compute multiple registered image channels, called Integral Channel Feature. Authors employed these descriptors into their CHNFTRS detector achieving state-of-the-art results in pedestrian detection. Based on their previous work on Integral Channel Feature, Dollar et al. [2010] proposed a feature extraction method that exploits the interpolation of features in different image scales, significantly reducing the cost and producing faster detectors when coupled with cascade classifiers. Recently, Marin et al. [2013] proposed the use of Random Forests to combine multiple local experts. To reduce computational cost, the multiple local experts share the extracted features. Another approach is the use of parallel architectures, as multi-core processors and General Purpose Graphics Processing Unit (GPGPU), for feature extraction. For instance, Prisacariu and Reid [2009] showed in their work efficient ways to extract Histogram of Oriented Gradients (HOG) descriptors using GPGPU, achieved speedups of over 67$\times$ from the standard sequential code.

Among the several known feature descriptors, we can mention few relevant methods. *a*) Scale Invariant Feature Transformation (SIFT) [Lowe, 2004] - a local image region is divided into a grid (*e.g.*; 4×4 pixels and a gradient orientation histogram is computed for each cell of the grid; *b*) Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] - a histogram of location and orientation of image gradients is constructed and used as feature vector (see details on Figure 2.2);

c) Gray-Level Co-occurrence Matrix (GLCM) [Haralick et al., 1973] - the occurrence of pairs of pixel intensities is tabulated in a matrix, from which statistical measures are computed and used as feature descriptors. The last two feature descriptors will be considered in our experiments to evaluate the proposed framework.

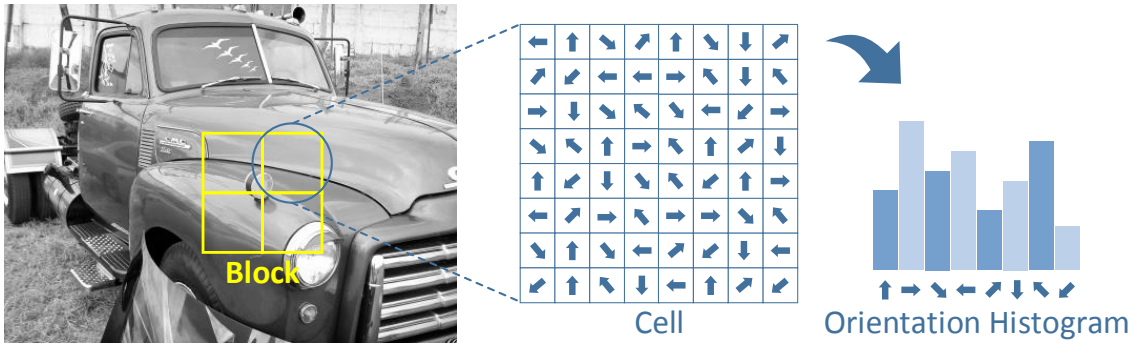


Figure 2.2: Illustration of HOG computation.

To address the feature extraction problem, the SSF provides a powerful tool: the Feature Extraction Server (FES), which 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) and also allows researchers to use feature descriptors implemented by third parties. The Feature Extraction Server (FES) is detailed in Section 3.3.

2.2 Surveillance Systems

Nowadays, there is an increasing interest in surveillance applications because of the availability of low-cost sensors and processors. There is also an emerging need from the public for improving safety and security in urban environments and the significant utilization of resources in public infrastructure. These two factors associated with the growing maturity of algorithms and techniques, enable the application of technology in public, military and commercial sectors [Regazzoni et al., 2001].

Smart visual surveillance systems deal with the real-time monitoring of objects within an environment. The main goal of these systems is to provide an automatic interpretation of scenes and to understand and predict the actions and interactions of the observed objects based on the information acquired by video cameras.

Current research in automated visual surveillance systems tends to combine multiple disciplines such as those mentioned earlier with signal processing, telecommunications, management and socio-ethical studies. Nevertheless there is

be lack of contribution from the field of system engineering to the research [Valera and Velastin, 2005].

The next sections will overview the state of art in Smart Visual Surveillance Systems, introducing the evolution of these systems, as well as their applications, requirements and challenges.

2.2.1 Evolution of Surveillance Systems

Security surveillance systems are becoming crucial in situations in which personal safety could be compromised resulting from criminal activity. For this, video cameras are constantly being installed for security reasons in prisons, parks, banks, automatic teller machines, gas stations, and elevators, which are the most susceptible for criminal activities [Räty, 2010]. For instance, Figure 2.3a shows a set of cameras placed at Tom Lee Park, Memphis, Tennessee, USA.

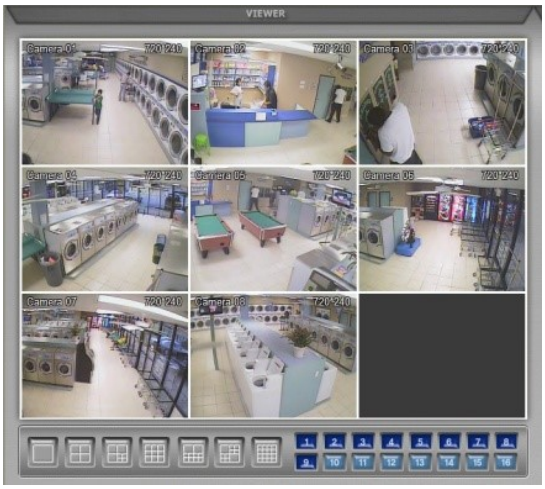
In general, images provide by a set of cameras may be monitored in real time at the command center (Figure 2.3b), where exists many display screens from which security personnel constantly monitors suspicious activities (Figure 2.3c). Images can also be archived for investigative purposes. However, the entire burden of watching video, detecting threats, and locating suspects are assigned to the human operator. This process of manually watching video is known to be tedious, ineffective, and expensive [Hampapur, 2008], because the attention span of human observers is inevitably limited [Davies and Velastin, 2007]. Therefore, the addition of computational intelligence to alert the observers to the infrequent image feed which contained events of possible importance was thus a natural development as computing resources became both cheaper and more powerful.

According to Valera and Velastin [2005] and Räty [2010], the technological evolution of surveillance systems can be divided into three generations, which are summarized in Table 2.2.

The *first generation of surveillance systems* started with analogue CCTV (Closed-Circuit Television). These systems consist of a number of cameras placed in multiple locations and connected to a set of monitors, usually placed in a single control room, via switches (a video matrix). The main disadvantages of these systems concern the reasonably small attention span of operators that may result in a significant miss rate of the events of interest. The advantage is that they provide good performance in some cases and the technology is mature. To perform computational processing on this type of system conversion from analog to digital video is required which may cause quality degradation.

(a) Surveillance cameras.

(b) Example of a command center.



(c) Surveillance system screen.

Figure 2.3: Elements of a surveillance environment: (a) Cameras at Tom Lee Park, Memphis, Tennessee, USA (Extracted from: http://goo.gl/XsvBpb); (b) Integrated command and control center of Minas Gerais, Brazil (Extracted from: http://goo.gl/B7hmgp); (c) Example of a surveillance system screen which shows camera images (Extracted from: http://goo.gl/nvYVhG).

The advent of digital CCTV and high performance computers have led to the development of semi-automatic systems, known as *second generation of surveillance systems*. This generation benefited from the early progress in digital video communications, e.g., digital compression, robust transmission and bandwidth reduction. The advances of the second generation are that the surveillance efficiency of CCTV is enhanced. The difficulties lie within the robust detection and tracking algorithms needed for behavioral analysis. Most of the research in this category is based on the creation of computer vision algorithms aiming at improving results for identification, tracking of multiple objects in complex scenes, human behavior com-

First Generation	
Techniques	- Analogue Closed-Circuit Television (CCTV) systems
Coverage	- Small/Medium areas (*i.e.* shop, banks, schools)
Smart	- No
Data processing	- None
Advantages	- Good performance in some situations and mature technology
Problems	- Use analogue techniques
Research	- Digital versus analogue - CCTV video compression
Second Generation	
Techniques	- Automated by combining computer vision with CCTV systems
Coverage	- Small/Medium areas (*i.e.* shop, banks, schools)
Smart	- Yes
Data processing	- Low
Advantages	- Increase the surveillance efficiency of CCTV systems
Problems	- Robust algorithms required for behavioral analysis
Research	- Automatic learning of scene variability and patterns of behaviors
Third Generation	
Techniques	- Automated wide-area surveillance system
Coverage	- Large areas (*i.e.* cities, highways)
Smart	- Yes
Data processing	- High
Advantages	- More accurate information and distribution of different sensors type
Problems	- Distribution of information (integration and communication) - Moving platforms, multi-sensor platforms
Research	- Distributed versus centralized intelligence - Data fusion and multi-camera surveillance techniques

Table 2.2: Summary of technical evolution of intelligent surveillance systems. (Adapted from [Valera and Velastin, 2005]).

prehension, and multi-sensor data fusion. The second generation also improved intelligent human-machine interfaces, performance evaluation of video processing algorithms, signal processing for video compression and multimedia transmission for video-based surveillance systems [Räty, 2010].

In the *third generation*, the technology revolves around wide-area surveillance systems, dealing with a large number of cameras, geographically distributed resources and several monitoring points. Such factors allowed the acquisition of more accurate information by combining different types of sensors and the distribution of the information. The difficulties are in achieving efficient information integration and communication, the establishment of design methodologies, and the task of designing and deploying multi-sensor platforms. The current research concentrates on distributed and centralized intelligence, data fusion, probabilistic reasoning frameworks, and multi-camera surveillance techniques [Valera and Velastin, 2005]. According to Räty [2010] the main objective of the fully third generation system is to ease efficient data communication, management, and extraction of events in real-time video from a large collection of sensors. To achieve this goal, improvements in automatic recognition functionalities and digital multiuser communications are required.

2.2.2 General Surveillance Systems

Several surveillance systems of the third generation have been designed and developed both in the industry and in the academia. These systems can be classified into two groups: general purpose and specialized in a certain function. Most works in the literature describe systems in the latter group ()discussed in Section 2.2.3).

Different from the specialized systems, the SSF can be classified as general purpose because the user (researcher) has the freedom to develop his/her modules (as described in Section 3.6) and use them for any purpose involving surveillance. The following paragraphs present examples of known general-purpose systems and their similarities and differences with the framework proposed in this work.

Several technologies for video-based surveillance have been developed under a United States government funded program called Video Surveillance and Monitoring (VSAM) [Collins et al., 2000]. This program, which can be considered one of the pioneers among the third-generation systems, looked at several fundamental issues in detection, tracking, auto-calibration, and multi-camera systems. The goal of VSAM was to develop efficient wide-area video surveillance systems using a distributed network of cameras. The system provided the capability to detect, track, lo-

calize and visualize objects within the known environment. Similar to other newer systems, the SSF incorporates several concepts based on VSAM, such as scalability, modularization and code reuse.

Knight [Shah et al., 2007] is a fully automated system with multiple surveillance cameras that detects, categorizes and tracks moving objects in the scene using computer vision techniques. Although it can be used in various types of surveillance environments, the Knight is a closed framework that does not allow the implementation of new methods to replace or extend to the existing ones. In addition, it is a commercial system, hindering its use in academia.

Another system is the IBM Smart Surveillance System (S3) [Tian et al., 2008], which is among the most advanced surveillance systems nowadays. It provides the following capabilities: automatic monitoring of a scene, management of surveillance data, perform event based retrieval and receive real-time event alerts. In S3, computer vision routines are not implemented directly into the system, but as plugins. One of its disadvantages it that it requires the use of technologies from *IBM*, such as *IBM DB2* and *IBM WebSphere*, which reduces its applicability for research purposes.

San Miguel et al. [2008] and Suvonvorn [2008] proposed two general-purpose frameworks for processing and analyzing surveillance videos. Similarly to the SSF, they enable the development of modules for processing images and videos. However, they have adopted a different approach for data communication between the modules. In [San Miguel et al., 2008], the communication between modules is mapped through a database system, while in [Suvonvorn, 2008], the modules communicate directly, where a buffer is used as an exchange zone. In contrast, modules in the SSF do not communicate directly, but through a shared memory, which allows modules to be launched in an asynchronous way and the dependency among them can be defined as parameters, making the SSF versatile and flexible.

Xie et al. [2002] proposed a Software Infrastructure for Smart Space (SISS), called Smart Platform. A smart space is a typical multi-modal system which typically involved dozens of distributed computation and perception modules that are usually not developed for running together, such as speech recognition, person-tracking and gesture recognition. The Smart Platform is a flexible and extensible cross-platform system that allows modules to be restarted or moved to different hosts and system reconfigurations in execution time. It was designed for pervasive computing, so it does not meet some requirements of video surveillance analysis, such as lack of mechanisms to facilitate the representation of object tracking, actions and activities.

The work proposed by Afrah et al. [2009] addresses two aspects in the development of vision-based systems that are not fully exploited in many current frameworks: abstraction above low-level details and high-level module reusability. They proposed a systematic classification of subtasks in vision-based system development. However, this framework is inflexible in according to the exchange of modules, preventing researchers from comparing results obtained by different methods, which would be an important feature for the academic community.

With a proposal similar to the SSF, the work proposed by Wang et al. [2012] presents a vision system architecture that can readily integrate computer vision processing and make application modules share services and exchange messages transparently. The model of computation assumed by the authors is the same used in the SSF. In this model, modules communicate with each other through a shared memory and are executed independently and in parallel.

Despite their similarities, there are some key difference between the two approaches: *a*) Wang et al. [2012] system, the processing is centralized for some tasks, such as capturing sensor data, encoding and decoding video streams, and transforming different types of data, but on the SSF all processing is performed in parallel on modules, which allows a better use of the processing power; *b*) the shared memory on the SSF stores the scene information in a hierarchy based on the necessary structures for surveillance environment to avoid data redundancy, allowing low memory consumption (for more details, see Section 3.2); *c*) the SSF allows one to perform complex queries on data in shared memory through the Complex Query Server (CQS) (Section 3.4).

Another aspect that differentiates SSF from other systems is that SSF implements the Feature Extraction Server (FES), described in Section 3.3, which allows the feature extraction to be performed using the entire computational power available in the system with the objective of maximizing the performance (one can use all available CPU cores). In the other systems mentioned earlier, the feature extraction process receives no special treatment, being under the user's responsibility.

2.2.3 Applications

To design efficient systems, it is necessary that researchers understand the nature of the environments in which the systems will be used. Another issue is to be able to interpret the requirements of the end user. Regazzoni et al. [2001]; Valera and Velastin [2005]; Sedky et al. [2005]; Hampapur et al. [2003] classified real-world applications into the following monitoring categories:

Public Area

Detect anomalous behavior from a person or a group of people in subways, parking lots, stadiums, large facilities and other public areas;

Interior and Exterior of Buildings

Improve safety in buildings, such as banks, shopping malls and houses. Access control, intrusion detection, object removal/abandoned alert and people counting are common surveillance tasks in this category.

Transport

Monitoring for railway stations, airports and maritime environments, traffic measure, accident detection and autonomous navigations.

Military

Surveillance of strategic infrastructure, enemies movements in the battlefields and air monitoring.

Entertainment

Interactive games interface, sport analysis, broadcast of abstract and sports events.

Efficiency Improvement

Long routine tasks, personalized training, coordination in workplace, compiling consumer demographics and monitor.

There are several published papers on surveillance applications. Among them we can mention, the work of Xia et al. [2013] that focuses on wide-area traffic monitoring for highway roads. Odobez et al. [2012], in turn, designed a metro station monitoring system that aims at automatically detecting dangerous situations which may lead to accidents or violence. The system proposed by Thornton et al. [2011] allows an operator to search through large volumes of airport surveillance video data to find persons that match a particular attribute profile. Siebel and Maybank [2004] especially deal with the problem of multi-camera tracking and person handover, on metro stations, within the *ADVISOR* surveillance system. A framework for people searching, where the user can specify personal attributes through queries such as "*Show me the bald people who entered a given building last Saturday wearing a red shirt*", was proposed by Vaquero et al. [2009]. It is important to notice that, many surveillance applications are of commercial license, and thus, there are none scientific sources that describe them.

Being a general tool, the SSF enables the development of many types of applications since coding specific modules to address many visual surveillance problems allows the user to develop various applications types. In addition, the exchange/-combination of these modules can generate new applications. Appendix A illustrates an example of real application developed using the SSF.

2.2.4 Challenges

As mentioned earlier, surveillance systems of the third generation contribute significantly to the design of various types of secure environments. Meanwhile, along with improvements, several challenges have emerged, causing many researchers devote their studies to do so. The work published by Liu et al. [2009] discusses some challenging issues faced by researchers. Other papers addressing the challenges of smart surveillance systems have been published recently, such as Räty [2010]; Haering et al. [2008]; Hampapur et al. [2003]; Regazzoni et al. [2001]. The next paragraphs present an overview on these challenges.

Quality and Consistency of Data Image

Images are not always perfect in such systems. For instance, objects of interest can be partially occluded, camera lenses maybe covered or damaged, the person being identified may have covered himself/herself by purpose. Even when these problems do not exist, there are other aspects causing decreasing the image quality, such as, poor illumination, sensor noise, particularly in poor lighting conditions and low resolution of the cameras.

The detection of events related to certain individuals comes from different cameras when the individuals are moving, for instance in an airport. Therefore, events detected from multiple cameras/sensors relating to the same object (person-/people) must be combined to reduce uncertainty and inconsistency. A typical scenario is that from a camera with poor visibility a male is detected while from the audio recording it strongly indicates a female. So adequate methods must be applied to resolve this inconsistency.

This type of challenge comprises several sub-challenges in computer vision. Since they are out of the focus of this work, these problems will not be detailed. Please, see [Liu et al., 2013; Valera and Velastin, 2005; Räty, 2010] for further reading.

Flexibility and Scalability

A large-scale video surveillance system comprises many video sources distributed over a large area, transmitting live video streams to a central location for monitoring and processing. Contemporary advances in video sensors and the increasing availability of networked digital cameras have allowed the deployment of large-scale surveillance systems over existing network infrastructure. However, designing a smart and scalable surveillance system remains a research problem: how to design scalable video surveillance systems according to aspects related to processing power, memory consumption and network bandwidth?

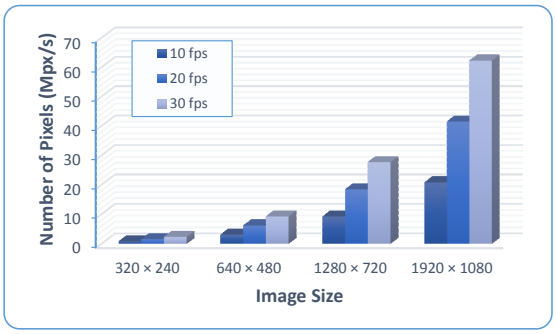
Besides the wide availability of cameras, the emergence of high-resolution image sensors at higher frame rates (Frames per Second (FPS)) contribute to the increase of the amount of generated data. From the charts in Figure 2.4, it can be concluded that the quality of images generated by a camera is directly proportional to the computational power needed to process them. This is a problem for surveillance systems: being able to process data in real time. Thus, novel solutions are needed to handle restrictions of video surveillance systems, both in terms of communication bandwidth and computing power. A solution to decrease the necessary bandwidth is to allocate machines, responsible for processing, close to the sensors.

The framework proposed in this work deals with the scalability problem through the implementation of modules (see Section 3.6) which are executed in parallel. Thus, the researcher can partition his/her problem into smaller problems and execute them as a pipeline. Another feature that contributes to the performance is the FES, detailed in Section 3.3.

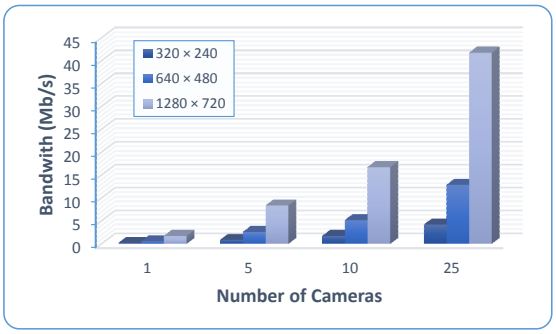
Privacy

According to Fleck and Strasser [2010], the privacy is a fundamental and very personal property to be respected so that each individual can maintain control of the flow of information about himself/herself. According to Gilbert [2007], privacy comprises confidentiality, anonymity, self-determination, freedom of expression, and control of personal data.

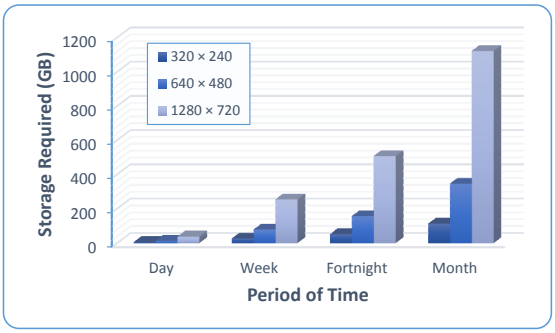
In the surveillance environment, it is important to guarantee privacy, as persons within a perimeter covered by cameras have very little choice of being filmed or not, whereas e.g., in the case of cell phone tracking the user still has the choice to turn his phone off. Additionally, it is not always apparent where cameras are located. Another problem is that operators are not always well-intentioned, such as recently happened on Araraquara (São Paulo, Brazil) in which the operators were



(a) Pixels Acquired per Camera



(b) Bandwith Required



(c) Video Storage per Camera

Figure 2.4: Illustration of the impact caused by the large amount of data generated: (a) Number of pixels acquired by a single camera in terms of resolution and frame rate; (b) Bandwidth required by a variable number of video cameras; (c) Space necessary for video data storage. The charts were generated using information from the tool available in http://goo.gl/PzLF0c.

using the surveillance cameras to inappropriately look at women[1]. Therefore, an automated and privacy-respecting surveillance system is a desirable goal.

According to Fleck and Strasser [2008], the latest video analysis systems emerging are based on centralized approaches that impose strict limitations to privacy.

An example of functionality able to maintain the privacy of individuals is mask out some portions of the image. The whole moving object, or just the face of the person can easily be masked out or pixellated, as illustrated in Figure 2.5. Other functionalities, such as abstraction, multiple privacy levels and encryption, are described in Winkler and Rinner [2010].

Even though the face information should be maintained during the visualization, it for instance, need to be available in the system so that one can make inferences. Thus, only when necessary (for instance, during a crime investigation), the

[1]Story available at: http://folha.com/no1384502 (in Portuguese)

faces may be viewed and only by authorized persons.

Figure 2.5: Masking out the face of a personal to address privacy concerns. (Extract from: http://goo.gl/maps/pdioc).

Besides of the papers which discuss the privacy of technical manner, there are several others that deal with the subject of a sociological point of view, as the work published in Posner [2008], where is discussed how surveillance systems must address some aspects of privacy, which are guaranteed by the law of the United States.

System Evaluation

Second Haering et al. [2008], one of the major challenges of developing a smart surveillance system is that it has to operate robustly during the entire time in a in wide range of scenarios. The only way to ensure robust and reliable performance is to perform extensive testing.

The following questions are relevant for system evaluation. Is it possible to establish a repository containing some common surveillance scenarios? Who are the people providing these scenarios, and what are the evaluations criteria? To answer these questions, Venetianer and Deng [2010] discuss some of the major challenges involved and provides a case study for addressing the evaluation problem.

For algorithms in other areas, such as machine learning, there are standard data sets to validate, evaluate and compare the algorithms. However, for visual surveillance systems, each security concern is different, the objects being recognized and events being detected are more specific according to the application. Therefore, it is a very difficult task to evaluate a complete surveillance system from a case awareness viewpoint [Liu et al., 2009].

The performance evaluation of video analysis systems requires significant amount of annotated data. Typically, annotation is a very expensive and tedious

process. Additionally, there can be significant errors in annotations and part of the evaluation of the surveillance systems depends on what the system operator considers as relevant action since they are not objective. All of these issues make performance evaluation a significant challenge [Hampapur et al., 2003].

For the aforementioned reasons, it is desirable an automatic mechanism that allows to test various system components and that facilitates comparison with existing methods. With the proposed framework is possible to solve the comparison of methods problem since it is very easy to change only the modules that perform a particular function without having to recode the entire rest of the process. Thus, the results generated by these modules may be fairly compared. The automated test for the entire surveillance system with a given purpose can also be done in the SSF just by writing specialized modules for this task.

Chapter 3

Smart Surveillance Framework

The SSF is a C/C++ library built using the Open Source Computer Vision Library (OpenCV) and the C++ Standard Template Library (STL) to provide a set of functionalities to aid researchers not only on the development of surveillance systems but also on the creation of novel solutions for problems related to video surveillance, as those described in Section 2.1.

One of its main goals is to provide a set of data structures to describe the scene allowing researches to focus only on their problems of interest and to use this information without creating such infrastructure for every problem that will be tackled, as it is done in the majority of cases nowadays. For instance, if a researcher is working on individual action recognition, he/she would need firstly to capture data, detect and track people, and only then perform action recognition. By using the SSF, one just needs to launch the detection and tracking modules (that might have been implemented by somebody else), to provide the people's location. In this case, one may concentrate only on the problem at hand, action recognition without concerning with the design of data representation, storage and communication.

The framework was designed to provide features for a third generation surveillance system [Räty, 2010; Valera and Velastin, 2005], such as tools to perform scene understanding, scalability, real-time operation, multi-sensor environment, usage of low-cost standard components, runtime re-configuration, and communication control. The next sections describe the design choices of the SSF to provide such desirable features.

3.1 Architecture

Figure 3.1 presents the architecture of the proposed framework containing its main components. Such components can be divided into two main parts: SSF kernel and user modules. The first part is composed of the SSF core that can be configured and its components which allow the researcher (user) to develop his/her applications and surveillance-related methods focusing only on the computer vision algorithms without concerning with data communication, storage, search and module synchronization. The second part are the modules, (described in Section 3.6), which are components written by the user using an interface to communicate with the shared memory using specific data types (*SMData*). Such components are independent and do not communicate directly, only through the shared memory. Such design 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.

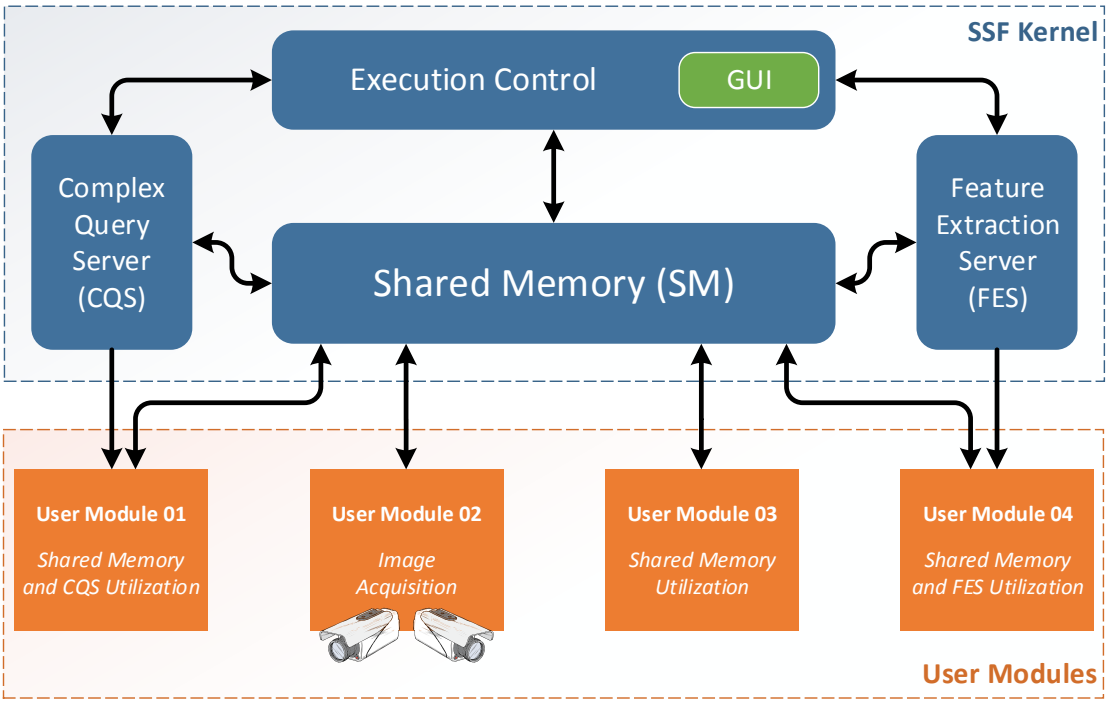


Figure 3.1: Architecture of the Smart Surveillance Framework (SSF).

The SSF kernel is composed of the following components: *a*) *Shared Memory*: the backbone of the SSF, it allows the communication among all other components of the framework once they do not communicate directly to each other; *b*) *Feature Extraction Server (FES)*: allows the user to implement and develop feature extraction methods that will be executed in an asynchronous manner aiming at maximiz-

ing the usage of the computational resources available in the system; *c*) *Complex Query Server (CQS)*: this component allows modules to search for specific data in the shared memory by taking advantage of Prolog, queries in SQL databases, among others; *d*) *Execution Control*: this component controls the execution of the modules, internal components of the SSF and is responsible for the SSF initialization. In addition, this component has a graphical interface to aid the user to configure the runtime environment.

3.2 Shared Memory

To allow modules to be designed and implemented independently from each other, it is necessary preventing direct data transmission among them, otherwise, one modules would need to be aware of other module interface, which would reduce the flexibility when integrating a set of modules to solve a given task. To address this constraint, the SSF provides a resource to store and control of the data communication between the user modules. This feature, referred to as shared memory, defines an interface to modules write and read data items.

The shared memory was designed to enable the development of many types of applications, including applications that are not in the visual surveillance scope. For this proposed, it was composed of three components, as illustrated in Figure 3.2, described as follows.

The first component, called *Memory Manager*, is responsible for the storage and management of the handled data. In the SSF, the data items are created by user modules and their references are passed on to the shared memory and the Memory Manager becomes responsible for the management of these references.

Since surveillance systems must handle large volumes of data (see Section 2.2.4), the memory on the SSF host machine can be easily filled. Thus, to deal this problem, the *Memory Manager* has a mechanism to detect when the primary memory is almost full and store the oldest entries on a secondary storage device (*i.e.* a hard disk or a solid-state drive (SSD)), thus increasing the memory limit that can be used by the SSF. In this way, when a data is required, the memory manager first checks whether it is in primary memory, otherwise it is retrieved from the secondary memory.

The second component is the *Basic Shared Memory*, responsible for the functions to access the data. This component does not depend on the context of the application, that is, their interface functions are general (*i.e.*, functions to write and

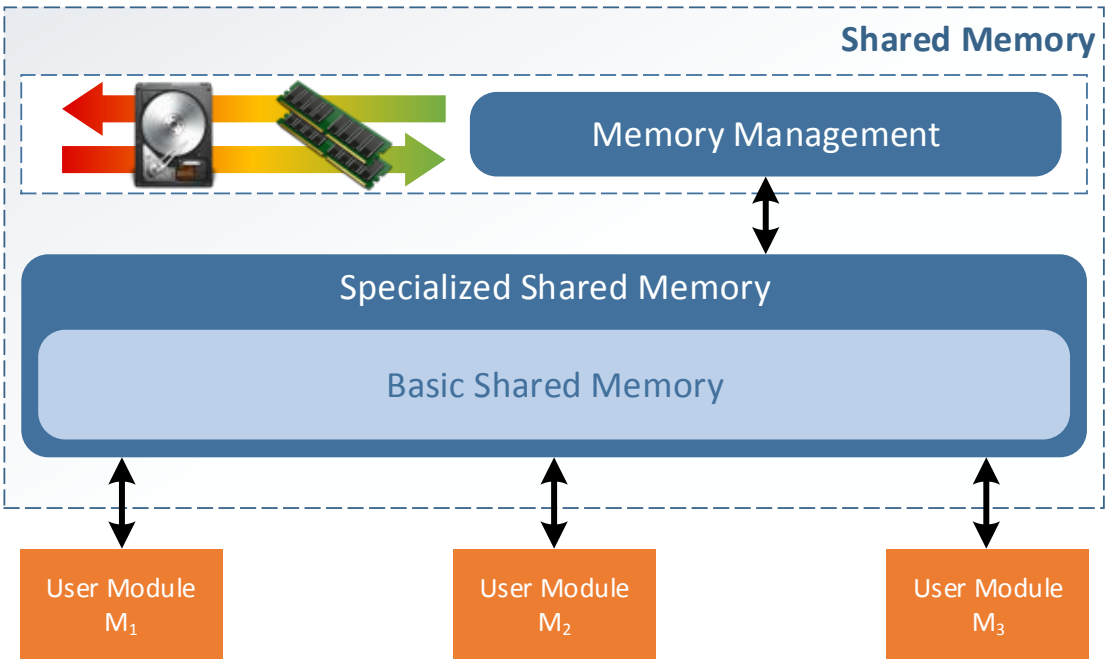


Figure 3.2: Components of the shared memory.

read data items) and have no knowledge of the data type being manipulated.

The third component, the *Specialized Shared Memory*, is a specialization of the shared memory, with surveillance purposes. This component provides methods and specific data types for the surveillance domain and is available when the user is developing user modules.

Focusing on surveillance, the shared memory stores the scene information in a hierarchy to avoid data redundancy, as showed in Figure 3.3. All data structures are stored in lists and only their unique identifiers on the lists are stored in the elements of the hierarchy, which not only avoids the need for updating the information every time the data structures are changed, but also reduces the data redundancy.

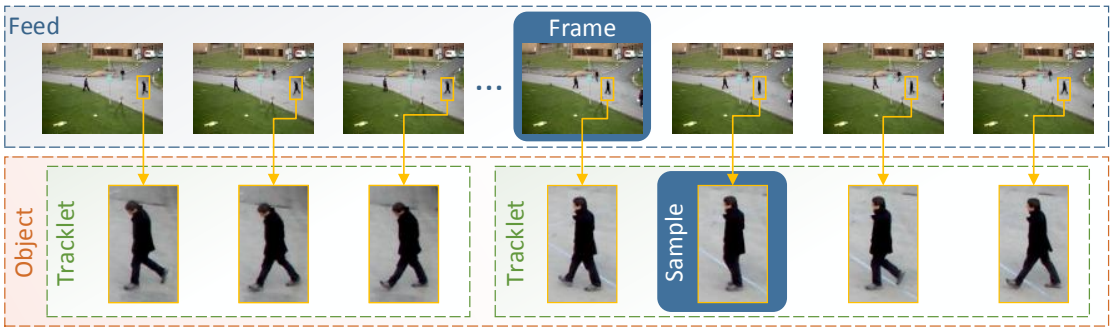


Figure 3.3: Hierarchical structure in the shared memory to store information regarding the scene under surveillance.

The following data structures and attributes are used in the shared memory to represent the scene under surveillance:

Feed is a sequence of frames that may have been obtained from a video file, a set of image files or frames obtained from a surveillance camera.

Frame contains an image of the feed and the attributes associated to it. Its attributes contain feature descriptors extracted from the frame, masks provided by the background subtraction and filtering methods and samples with possible object locations in the frame.

Sample represents the region of a frame containing an object. Its attributes contain feature descriptors, reference to the frame, sample location and possibly the gesture and pose when the sample belongs to a person.

Tracklet contains a set of samples from consecutive frames belonging to a single object. Its attributes contain feature descriptors extracted from the tracklet (usually temporal features) and the actions performed by the person during the tracklet duration.

Object is defined as being a set of tracklets belonging to a single individual associated with an identifier (for instance, the person's name).

Besides the standard structures in the SSF, it is also possible to create new data structures by heritage of a prototype data, referred as *user data*. The user data allows specific data definition such as sensors output (audio, temperature, multi-spectral images) or exchange of specific data types between modules, such as classification models.

Even though the hierarchical design chosen for the shared memory results in low memory consumption because there are no data duplication, the amount of data generated during processing can still be very large (for instance, a video feed being recorded for hours). To handle that, the SSF has a management mechanism that detects when the amount of memory allocated is close to the maximum available (or a maximum set by the user) and transfer to the secondary memory (hard disk) the least-requested data items. If any data stored on disk is requested again, it is transferred back to the main memory. This mechanism assures memory availability for processing, thereby contributing to the scalability of the system and allowing the use of low-cost computers with limited memory.

Another feature of the shared memory is that it is incremental in the sense that when a new data item is stored, it receives a new and unique identifier together

with a creation time stamp. With such information, one can trace back the entire execution of the system. For instance, one could verify when tracklets were merged and when new objects were created, which might be useful in the development of novel object tracking and recognition approaches.

As mentioned earlier, the shared memory also allows the communication between modules in an indirect manner - a module M_1 writes a data item to the shared memory, then other modules, say M_2 and M_3 can request this data item by setting the data type and the module that has generated it, as illustrated in Figure 3.2. The producer (module M_1) writes the data item that can be read by any other consumer modules (modules M_2 and M_3 in the example), which makes the framework more flexible in the sense that only the consumer modules have to indicate from which modules the will receive a given data type, the producer only writes its outputs to the SM.

3.3 Feature Extraction Server

As pointed out earlier, feature extraction is required to solve several problems in surveillance. Due to the large amount of data, this step must be efficient. However, even though local feature extraction methods have been proposed [Dollar et al., 2009; Viola and Jones, 2001], the feature extraction is still a time-consuming task. To reduce the computation cost, we developed the Feature Extraction Server (FES), a runtime framework which allows leveraging of modern parallel architectures aiming at increasing the performance of such methods.

The FES relies on an asynchronous approach to receive requests, process them and return feature vectors to modules with the objective of maximizing the occupancy of the processing units available. Once a request is 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. For instance, the module might be processing the feature vectors already extracted while others are being extracted. Therefore, all features vectors do not need to be stored in memory before processing, preventing from high memory consumption. In fact, the maximum amount of allocated memory can be set to avoid the process from using the virtual memory.

Figure 3.4 illustrates the main components of the feature server: *request control*, *extraction method* and *feature extraction memory*. Using the FES, a feature extraction request is performed as follows. First, a module sends extraction requests by passing image regions from which the features will be extracted by a given method. Such re-

quests are sent to a queue in the *request control*, which allows the module to make all requests for an image and continue its processing while the features are extracted. Then, the request control selects the extraction method chosen by the module and forward the requests to the *extraction method*, which process them using N instances. First, it checks the memory availability in the *feature extraction memory*, if there is not memory available, the extraction waits until some memory has been released. Finally, once the feature extraction is completed, the feature vector is pushed to the output queue and it is ready to be retrieved by the requesting module.

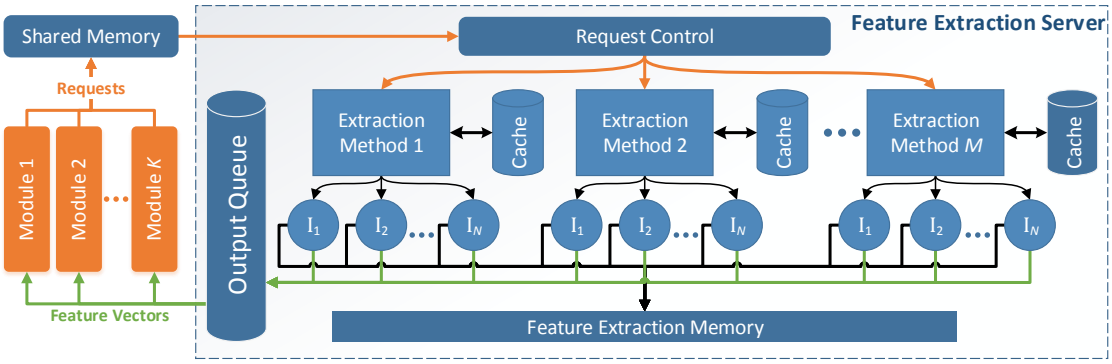


Figure 3.4: Feature Extraction Server (FES) and its interface with a module.

The *request control* is responsible for screening the requests made by the modules. It is composed of an input queue a data structure for storing information regarding the feature extraction methods available. Once a request enters the queue, the request control forwards it to the correct feature extraction method. The request control is useful in the sense that the feature extraction becomes centralized, such that two modules requiring the same feature extraction method will use the same instance of the extraction method, which will allow the usage of cached features if two modules request feature extraction for the same image region.

The *extraction method* manages the feature extraction for a specific feature descriptor, such as HOG, GLCM and others [Li and Allinson, 2008]. When the extraction method receives a request, it first verifies in the cache if the same request had been made before and the feature descriptors are already available, if so, return them, otherwise it checks in the *feature extraction memory* whether there is memory available in the feature extraction memory (Experiments show that the usage of cache reduces greatly the computational cost for feature extraction, see Section 4.3.2.).

The feature extraction memory allows the FES to set a limit of memory that can be used for the feature extraction process, otherwise the entire memory available in

the machine could be consumed quickly compromising the execution. If there is no memory available, the extraction method is blocked until some memory is released (some module retrieves an extracted feature vector from the output queue, process it, and sets as released), otherwise, it sends the request to one of its instances to perform the actual feature extraction for an image region. Also, it avoids memory reallocation, since the probability of feature vectors being the same size is great.

The advantages provided by the FES include the following. Besides of using methods already implemented, the user can implement his/her own feature extraction methods which will have their processing distributed according to the computational power at hand or according to the parameter setting chosen by the user. In addition, it allows users to develop novel feature descriptors and evaluate them easily on problems such as detection and recognition. Finally, this centralization approach based on a server to extract features allows the caching of features vectors so that several modules might share the same vectors for different purposes.

3.4 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 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. The following paragraphs describe how these methods are used in the SSF.

The *initialization* method requires that the user informs which fields for instance, time-stamp of image, location of sample and time interval of a tracklet, will

be stored in the CQS for future search. This information is given at the definition of each data type and allows the framework to grow in a scalable way, *i.e.*, without modifying CQS structure when new data types are incorporated to the framework.

At execution time, the CQS initializes by iterating over each data type and registering the searchable fields. This initialization is required in some solutions to create underlying structures such as tables in SQL Databases. Then, the CQS retrieves data items from the shared memory and passes them to *storing* methods so, they can be registered in the underlying structure (a row in SQL Database or a fact in *Prolog*).

For the *querying* methods, a user module retrieves a copy of a CQS instance with access only to query methods. Query methods are subdivided into *filter* and *retrieve* methods: *filter* methods are simple operations (“equal to", “less than", “or", among others) that receive field and data types and change the internal state of the CQS instance by building a partial filter of the field and integrating it with the previously state; *retrieve* methods return data to the user considering the filtered state of the CQS instance.

As an example, suppose that one is interested in recognizing a fighting activity between two subjects by analyzing the output of an identity recognition module and an action recognition module. The fighting activity is characterized by two subjects, close together, facing each other, and at least one of them is performing punching actions. Examples of queries to identify this fighting activity are given in *Prolog*, SQL and in CQS query format in Figure 3.5. In this example, tracklets are represented by horizontal lines and the action being performed is shown inside a rectangle. A fighting activity may be characterized by any two subjects that are close together facing each other and at least one subject is performing punching actions. The query result R is the reference to the video segment containing the action.

3.5 Execution Control

The SSF components may be set 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.

The SSF first initializes the internal components (i.e., FES and CQS), by assigning values to its parameters. Then, the instantiation and configuration of the

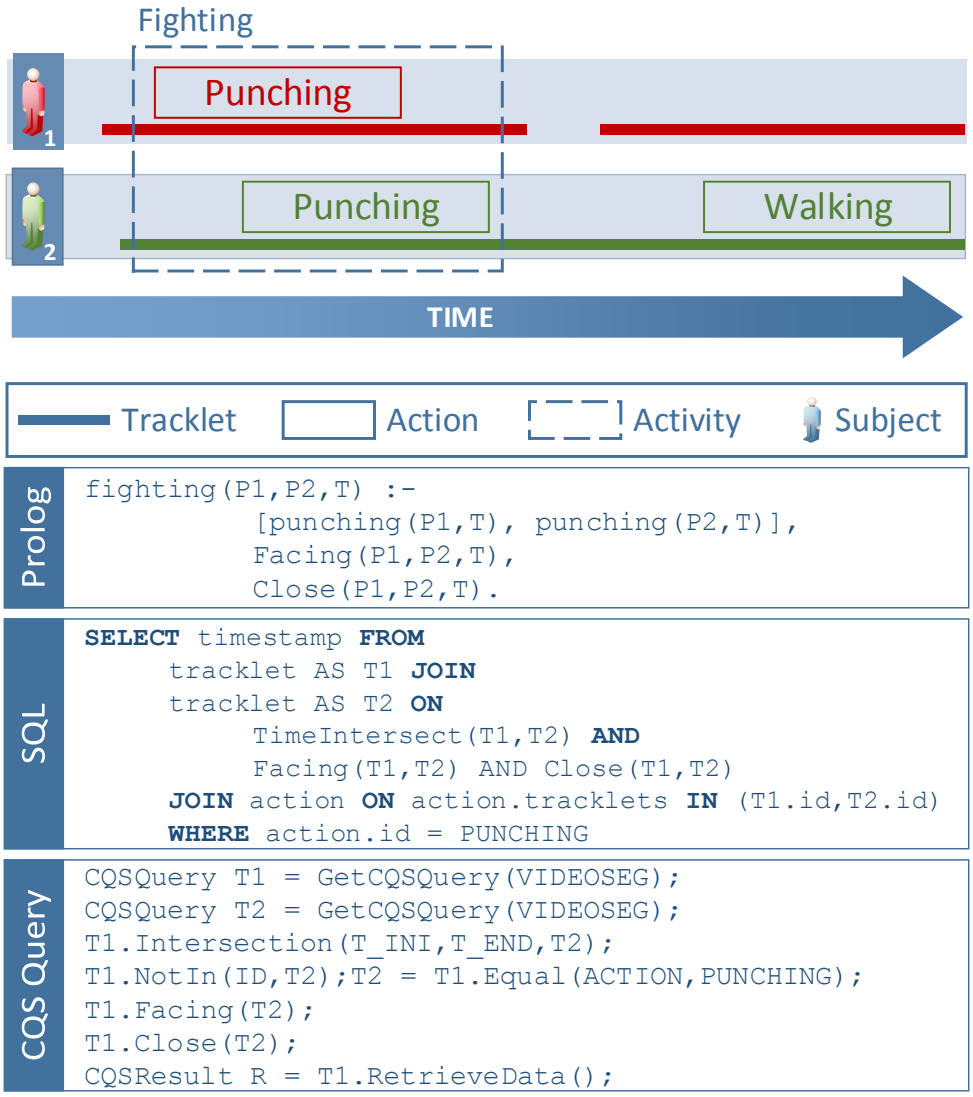


Figure 3.5: Examples of queries in *Prolog*, SQL and CQS to identify the fighting activity by analyzing the output of a module that recognizes tracklets of subjects and another module that performs action recognition.

parameters of the user modules is performed. It is worth noting that only the modules that are listed on the configuration are initialized. The execution control also defines data flows referred to as data streams, between modules and shared memory. These streams are declared in the configuration file (or in the GUI), in which the user defines how the modules will communicate with shared memory, stating which types of data will be transmitted, according to those that are implemented in the user module.

Due to the large number of parameters, the configuration file becomes complex and difficult to maintain. Thus, to deal with this problem, the SSF provides

a Graphical User Interface (GUI) component, as shows the Figure 3.6. Its goal is help the user to configure the runtime environment for the SSF. Through it, we can perform the following tasks: *a*) configure modules defining the parameters values; *b*) create and setup pipelines; *c*) define the data flow between the modules and/or pipelines; *d*) configure the SSF internal components, such as the shared memory and CQS.

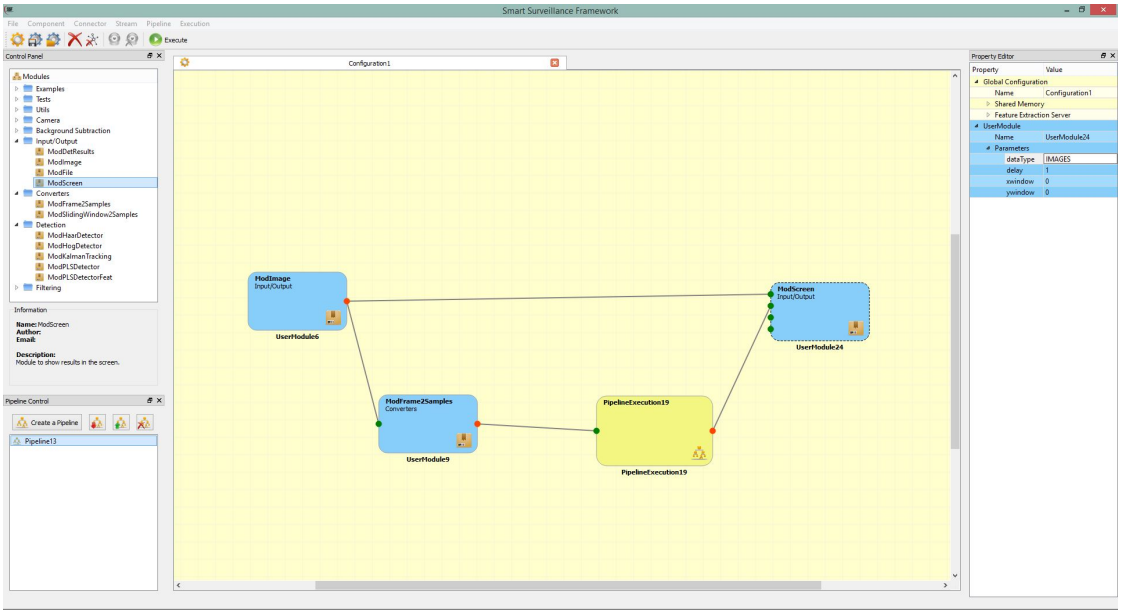


Figure 3.6: Screenshot of the Parameter Setup Interface. Red and green dots indicate inputs and outputs of a module, respectively. Blue box are User Modules, while light green box represents pipelines.

3.6 User Modules

The user modules are designed to allow the development 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. This routine reads from and writes to the shared memory using a standard interface provided by the framework.

As mentioned in Section 3.2, the modules work independently, in other words, a running module is not aware of the existence of others. This allows the exchange of modules of the same type without affecting the operation of the system. Moreover, modules are executed in different threads, which increases the performance of the system, enabling real-time processing. For each module is specified what types of data it will provide and also, if necessary, what types of data it will consume. This way, the module creates information but is not aware of which modules will use it.

Another important feature 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. For instance, Figure 3.7 illustrates a face recognition pipeline which consists of the following modules: *a*) background subtraction; *b*) face detection; and *c*) face recognition. After 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.

Figure 3.7: Illustration of an execution pipeline.

Since the modules are executed asynchronously, it is the responsibility of shared memory to perform data synchronization because a module might consume information faster than another module can provide. Figure 3.8 illustrates an example of synchronization between two modules (M_1 and M_2) and the shared memory (SM). In the first instant of time (t_1), M_1 writes a new data item to the SM while M_2 reads and processes the current data item in the memory. At time t_2, the module M_1 is processing, while M_2 reads and processes the only available data item. Since there is no more data to read in t_3, M_2 is locked until a new information is made available, which occurs in t_4. Finally, at t_5, M_2 is unlocked and performs a new reading. Therefore, by using locking mechanisms in the data reading, the SM is able to synchronize dependencies among modules without compromising the performance of independent modules.

In real scenarios, it might be relevant that the surveillance system is able to process different types of sources besides images, such as audio and proximity sensors. To achieve that, users can implement specific data definitions for sensor readings and process them inside the modules. Another advantage of using user data and

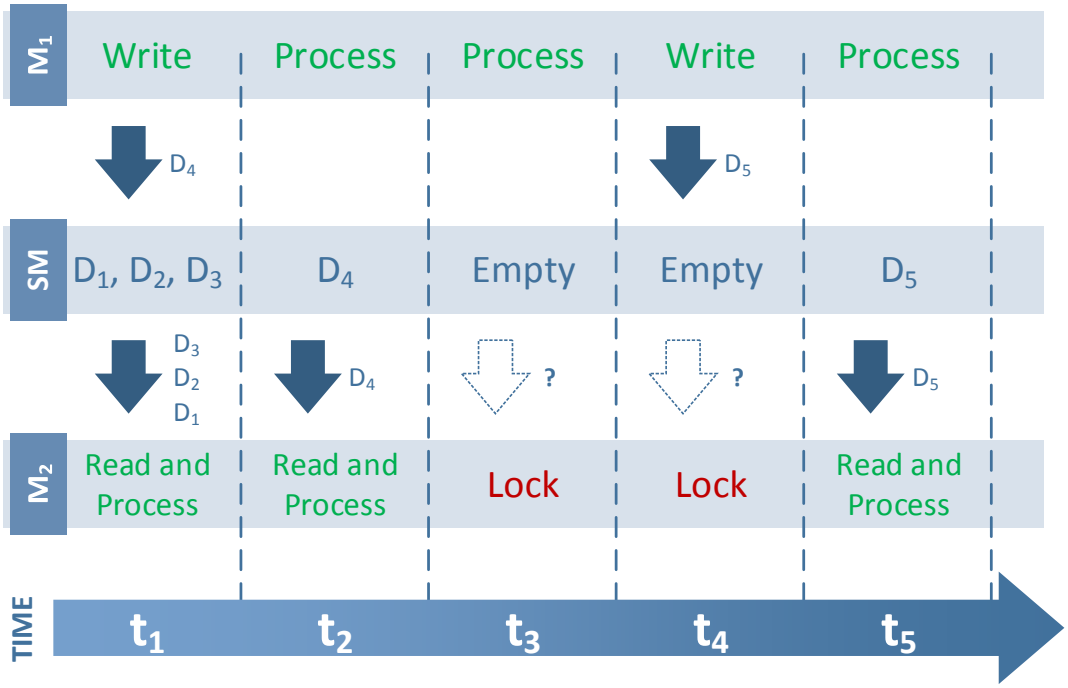


Figure 3.8: Example of the SSF module synchronization approach.

user modules for data acquisition is that one can take advantage of specific sensor hardware, such as camera built-in filters, Pan–tilt–zoom (PTZ) control and sensor alarms. Indeed, the SSF implements a PTZ camera control module that allows other modules to send command to IP cameras, allowing operations such as zooming in to the face region for proper face recognition, or moving a camera around to track a suspicious person.

Chapter 4

Experimental Results

This chapter evaluates important aspects of the framework proposed in this work. Section 4.1 explores the framework scalability, Section 4.2 describes experiments to evaluate the communication latencies caused by the architecture of Shared Memory and Section 4.3 discusses the performance of the Feature Extraction Server (FES). Finally, Section 4.4 discusses the results obtained.

All experiments were conducted using computer with two Intel® *Xeon*™ 2.40GHz processor with 6 physical cores each, 32GB of main memory and running Windows™ operating system.

4.1 Framework Scalability

This section presents two experiments to demonstrate the scalability of the framework. The SSF allows the user to perform parallelization of methods by decomposing the problem into subproblem. The following will describe briefly the types of problem decomposition which are dealt in these experiments.

The goal of a decomposition is to divide the problem into independent subproblems. They can mostly be written independently. Two general methodologies are commonly used. The first, termed *data decomposition*, assumes that the overall problem consists in executing computational operations or transformations 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. In a sense, the SSF supports both task decomposition (by the decomposition of a large problem into smaller parts, which in turn, are implemented through the modules) and data decomposition (by the instantiation of

several modules, with the same purpose, where, each module will handle a portion of the dataset) [Kumar, 2002].

Figure 4.1a shows a problem being treated in the traditional way, *i.e.*, sequentially. In this approach, each input data is processed sequentially and individually. In the SSF, this approach is equivalent to solve the entire problem inside a single user module containing all problem tasks, for instance, a standalone user module performing the detection and recognition tasks.

Figure 4.1b shows a task decomposition of the problem shown in Figure 4.1a. In this case, each sub-problem is addressed in parallel and various subsets of the data are processed simultaneously in a pipeline composed of specific modules for each task. The implementation of specific user modules for each task enables the implementation of this decomposition type on the proposed framework. From the above example, the detection and recognition would be implemented in separate modules, which would allow them to run in parallel.

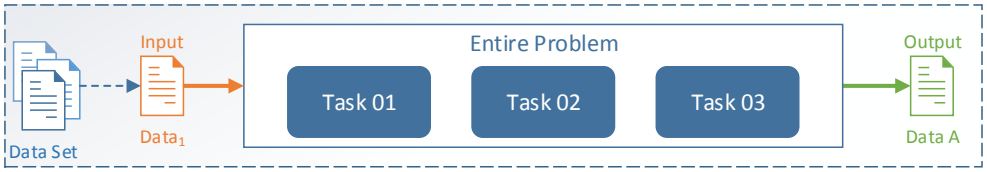
Figure 4.1c illustrates a data decomposition approach for the same problem. Here, the entire problem is replicated, the dataset is partitioned and each replica is responsible for a subset of the data. The SSF enables the decomposition of data through the pipeline replication.

The aforementioned decomposition techniques are not exclusive and can often be combined, as illustrated in Figure 4.1d, a decomposition commonly implemented in the SSF. It allows not only data decomposition but also task decomposition at the same time, taking fully advantage of the processing power of multi-core processors.

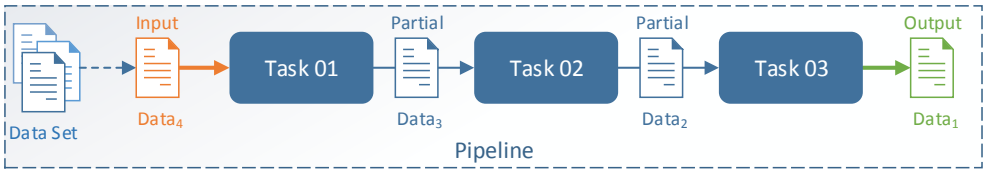
The following experiments demonstrate the application of parallelism on the SSF through the approaches of problem decomposition presented above. As a starting problem (sequential method), twelve series of similar image processing operations were implemented. This number of sequence operations was chosen because of the number of available cores on the test machine. Therefore, the sequential method of example has as input an image I and a series of operations p_i, where the computational cost C of all operations is similar, in other words, $C(p_i) = C(p_j)$, for all values of $1 \leq i, j \leq 12$. A dataset containing 100 images of 640×480 pixels was used.

4.1.1 Data Decomposition Evaluation

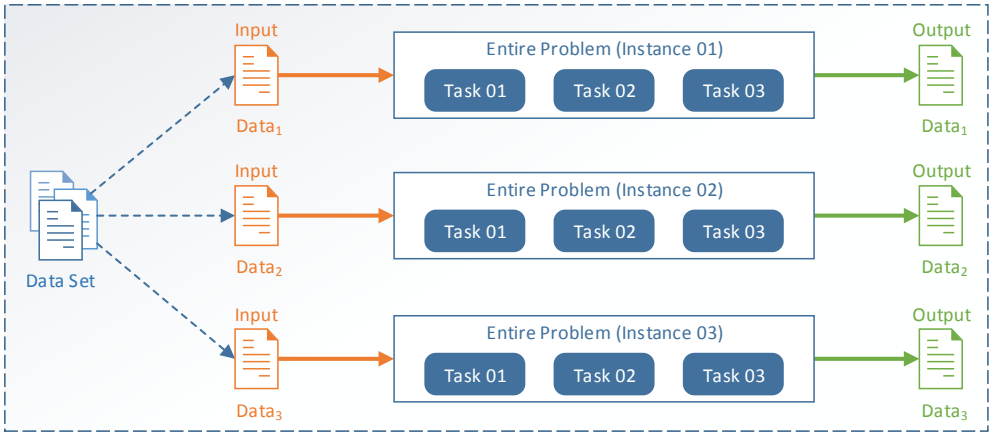
In this experiment, the sequential method has been implemented as a single SSF module and replicated n times, as illustrated in Figure 4.1c, in which each instance of the method is responsible for processing $100/n$ images from the dataset. The



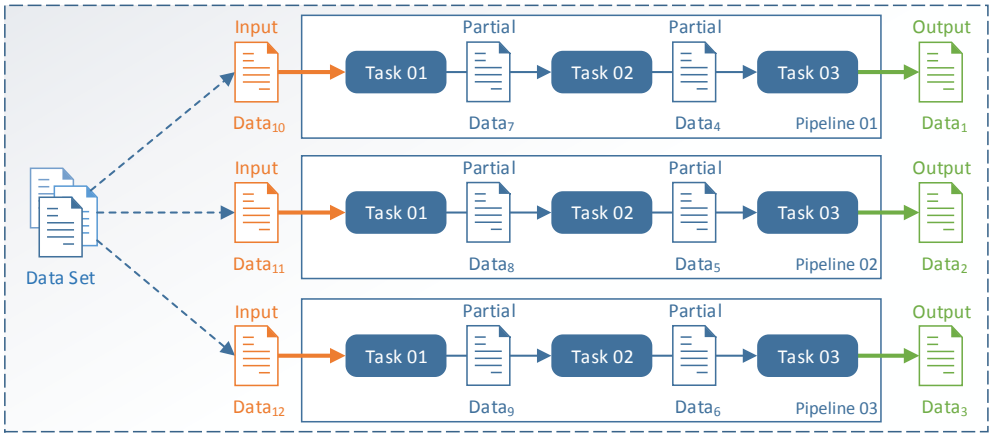
(a) Sequential approach to a problem. In the SSF, the entire problem is solved within a single user module.



(b) Task decomposition. In the SSF, each tasks runs in an independent module in parallel.



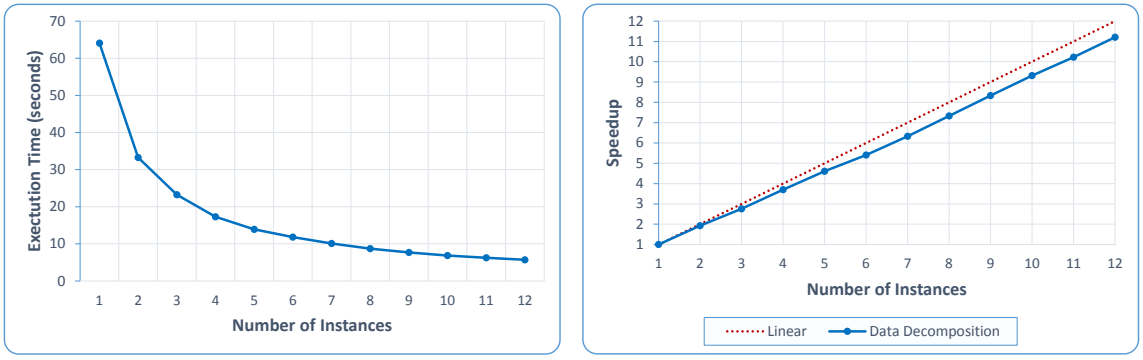
(c) Data decomposition. In the SSF, the data is split into subsets and each subset is presented to a sequence of independent modules.



(d) Hybrid decomposition. Combination of task and data decomposition.

Figure 4.1: Examples of problem decomposition

value of n was varied from 1 to 12 and each experiment was executed ten times. Figures 4.2a and 4.2b report the average execution time and speedup achieved by the data decomposition approach, respectively.



(a) Execution time. (b) Speedup.

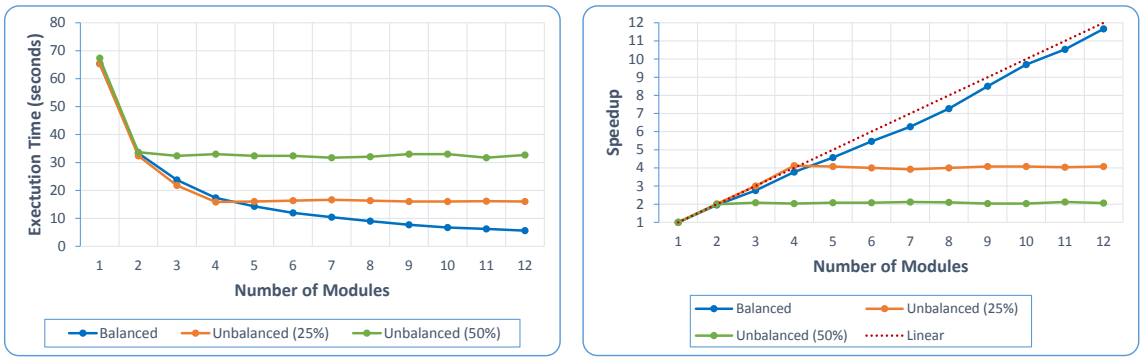
Figure 4.2: Results of the experiments considering the data decomposition approach.

As shown in Figures 4.2a and 4.2b, it is advantageous to use of the framework to parallelize the processing of a considerable number of images. The speedup obtained by the data decomposition approach is very close to linear. This fact demonstrates that the communication overhead caused by the SSF is minimal. Section 4.2 presents a detailed evaluation of the communication overhead caused by the SSF.

4.1.2 Task Decomposition Evaluation

This experiment demonstrates whether the use of task decomposition approach is advantageous in the framework. The sequential method is divided into n sub-problems, each of which implemented as a SSF module. Then, these modules are interconnected forming a pipeline, similar to Figure 4.1b. The value of n was varied from 1 to 12, *i.e.*, the sequential method was divided into up to 12 sub-problems.

The division of the sequential method in sub-problems was conducted in three distinct ways. In the first, the n modules had the same computational complexity, *i.e.*, the problem was equally divided and the computational load of modules were balanced. In the second way, at least one of the n modules had at least 25% of the operations of sequential method while the other 75% were equally distributed among the $n - 1$ remaining modules. The third is similar to the second. In his case, however, only 50% of the operations were reserved for only one module. The results are presented in Figures 4.3a and 4.3b, according to the execution time and speedup, respectively.



(a) Execution time.

(b) Speedup.

Figure 4.3: Results of the experiments considering task decomposition.

Analyzing the results, one can observe that the task decomposition is successful when there is a balance between the parts. Figure 4.3b presents an almost linear speedup for the case where the operations of the sequential method were divided equally among the modules. However, the two other examples, in which the division into sub-tasks was unbalanced, did not show any cost reduction starting from a certain number of modules. For the example where 25% of the operations are under the responsibility of only one module, the time improvement was observed only for up to 4 modules. From then on, there was no performance gain because the time spent by the module that was overloaded will always be greater than or equal to 25% of the total time. To the overhead of 50%, the improvement was observed only for up to 2 modules due to the same reasons previously discussed. This experiment demonstrates that the task decomposition in SSF can be scalable if conducted in a balanced way.

4.2 Communication Latency

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 between 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. Figure 4.4 illustrates this experiment.

This experiment considered pipelines with sizes of 1, 3, 5, 7, 10 and 15 mod-

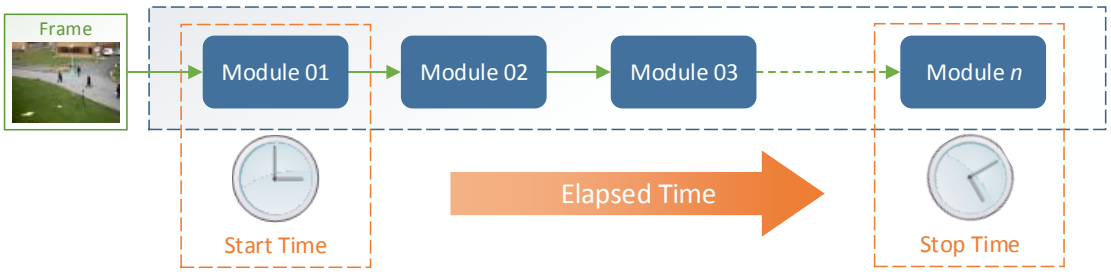
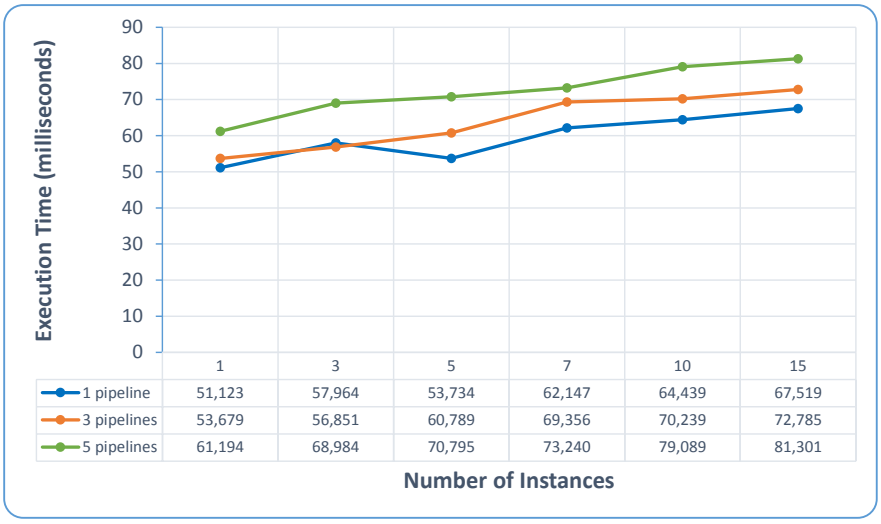


Figure 4.4: Setup of the experiment performed to compute the data latency in the SSF.

ules. In addition, for each pipeline size, executions with 1, 3 and 5 simultaneous pipelines were tested. To perform this experiment, a total of 100 different images were transmitted and the average time spent for each move across the pipeline was calculated. The results are showed in Figure 4.5.



	1	3	5	7	10	15
1 pipeline	51,123	57,964	53,734	62,147	64,439	67,519
3 pipelines	53,679	56,851	60,789	69,356	70,239	72,785
5 pipelines	61,194	68,984	70,795	73,240	79,089	81,301

Figure 4.5: Results of the experiments regarding the data latency for the framework.

Results shown in Figure 4.5 demonstrate that the overhead caused by the increased number of modules simultaneously connected to the shared memory is low. Although this overhead exists, it is negligible when compared with the processing time of the data, usually orders of magnitude higher.

4.3 Feature Extraction Server (FES) Evaluation

This section describes the experiments conducted to evaluate the performance of the Feature Extraction Server (FES). The evaluation was conducted using three

traditional methods of features extraction: *Pixel Intensity*, Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005] and Gray-Level Co-occurrence Matrix (GLCM) [Haralick et al., 1973]. Even though there are many other feature extraction methods, we have chosen these three methods because they present different computational cost and memory consumption, allowing us to evaluate different aspects of the Feature Extraction Server (FES).

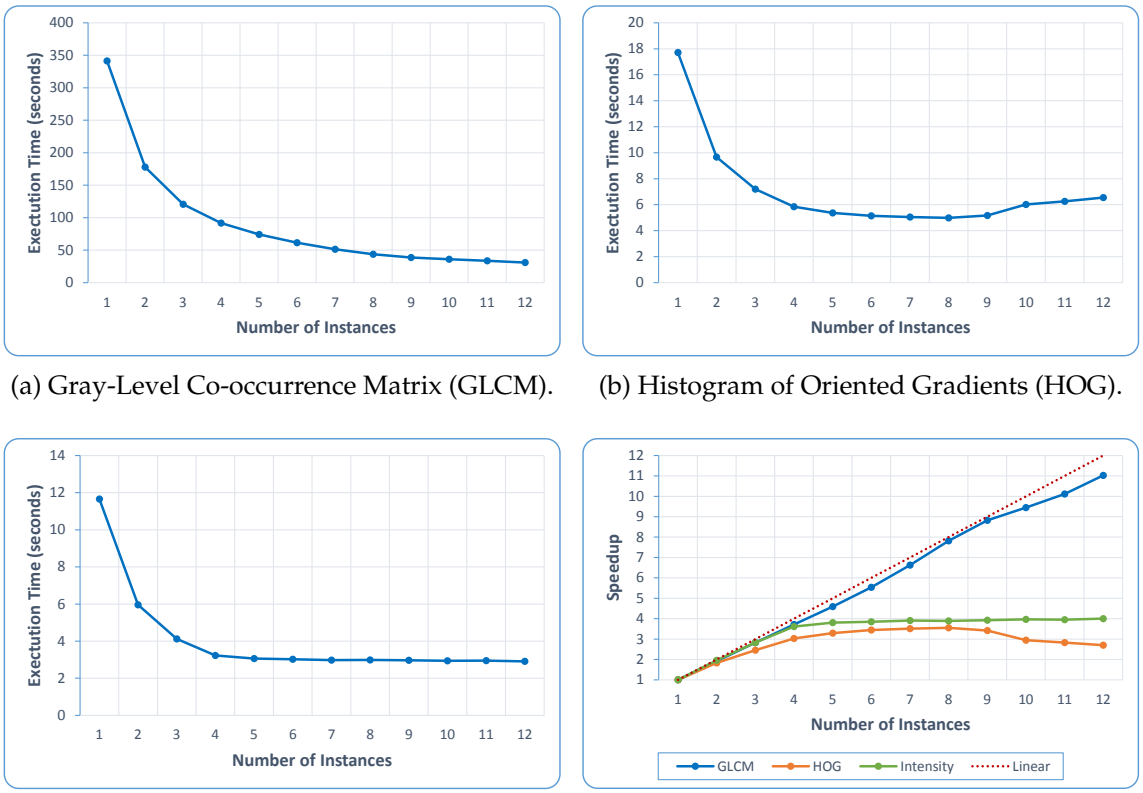
The experiments consist in extracting feature descriptors of an image with a resolution of 640×480 pixels, using the aforementioned methods. To represent a realistic scenario, we employ the sliding window algorithm [Forsyth and Ponce, 2011], widely used in object detection, to sample the image regions from which the feature descriptors are extracted. This algorithm works by exhaustively scanning an input image to generate a set of coordinates of several detection windows in multiple scales. For this work, we follow the block setup used in Dalal and Triggs [2005], in which each detection window is split into 105 blocks and we set the stride and scales parameters to generate a total of $48,495$ detection windows per image. We evaluate the FES regarding two aspects: the performance of parallel feature extraction by increasing the number of extraction instances and improvements obtained by the cache memory in the feature extraction.

4.3.1 Number of Instances

To demonstrate the performance of parallelism provided by FES, we conducted experiments based on the number of instances used in the extraction. Each experiment consisted in the execution of a method repeated ten times and varying the number of instances from 1 to 12.

As shown in Figure 4.6, one can 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 GLCM (Figure 4.6a) method showed a proportional reduction in run time on all experiments, while for the other two methods, HOG and intensity, Figures 4.6b and 4.6c respectively, it was only observed up to six instances. In the HOG case, there is a slightly increase in the run time starting from nine instances. This is because the computational complexity of HOG and intensity is smaller when compared to the GLCM, hence there is an overhead caused by the FES, starting at nine instances. This behavior can be explained by the *Amdahl's Law*[1]. This law states that a fraction

[1] *Amdahl's law*, also known as *Amdahl's argument*, is named after computer architect Gene Amdahl, and is used to find the maximum expected improvement to an overall system when only part of



(a) Gray-Level Co-occurrence Matrix (GLCM).

(b) Histogram of Oriented Gradients (HOG).

(c) Pixel Intensity.

(d) Speedup of the methods.

Figure 4.6: Computation time obtained for the feature extraction as a function of the number of extraction instances.

of sequential operations, even in small numbers, can significantly limit the speedup achieved by a multi-core computer.

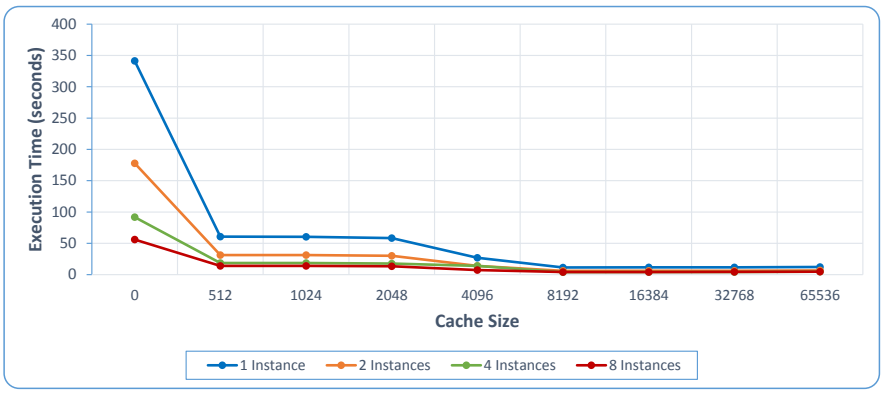
Figure 4.6d shows the speedup obtained for each feature extraction method. The speedup achieved with the GLCM method presents a linear growth, demonstrating the scalability of the FES for computationally expensive methods. For the HOG and intensity methods, the speedup presented a linear growth up to only five instances due to the overhead present in the FES which is more evident when the method is not very computational expensive.

4.3.2 Cache Size

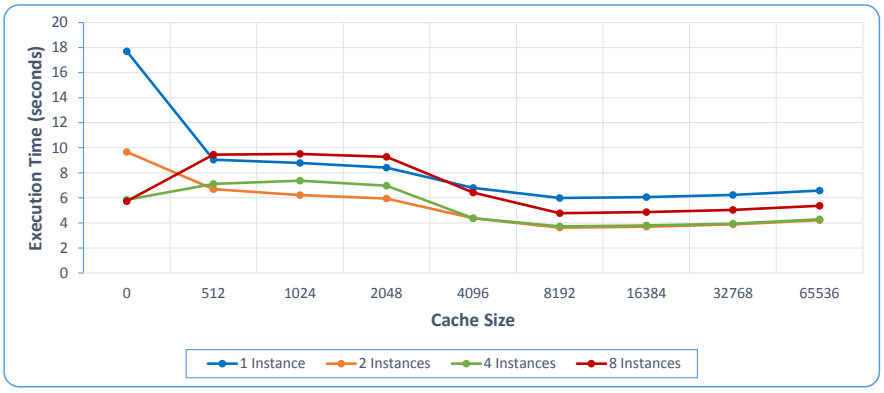
This set of experiments aims at showing the performance gain obtained when the cache memory is used for the feature extraction method and when its size is increased. We performed experiments where each extraction method is individually

the system is improved. It is often used in parallel computing to predict the theoretical maximum speedup using multiple processors Amdahl [1967].

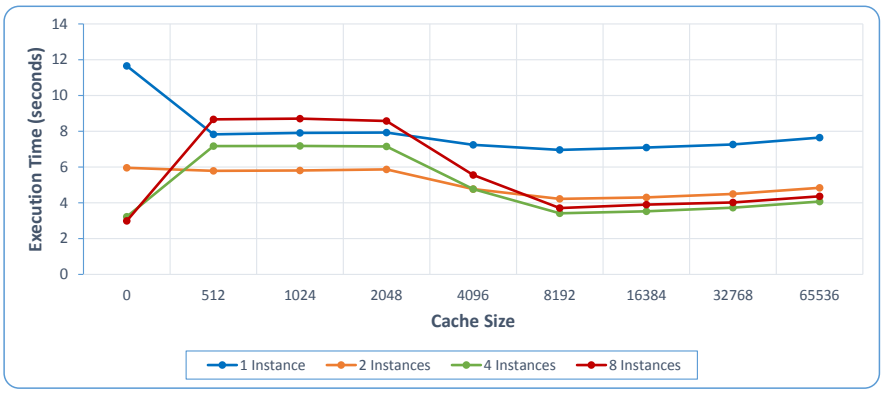
executed for a different cache with at most C entries, where $C \in \{0, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536\}$ by varying the number of instances in 1, 2, 4, and 8. Each experiment was executed ten times. The average computational time is reported in Figure 4.7.



(a) Gray-Level Co-occurrence Matrix (GLCM)



(b) Histogram of Oriented Gradients (HOG)



(c) Pixel Intensity

Figure 4.7: Computation time with the addition of cache memory with multiple sizes (maximum number of entries).

Figure 4.7a shows a significant reduction in time for the GLCM without using cache (near 80% for a cache of size 512), for every number of instances. The improvement is also observed for 1024, 2048, 4096, and 8192 cache size. However, starting from 16834 entries, the runtime does not decrease. This is because the number of extracted features is not enough to fill the entire cache.

The cache utilization also significantly contributed to the performance of HOG and intensity. However, this contribution is only observed when the HOG is performed in one or two instances and when the pixel intensity is executed for a single instance, although two instances result in a slightly reduction in the run time.

Unlike the previous results, experiments with 4 and 8 instances for HOG and intensity increased the run time due to the overhead caused by competition for access to the cache, since the low computational cost of the methods yields the instances to quickly compute the features and consequently making them wait to have access to write to the cache memory. One may also notice a small increase in run time for cache values above 8192, which we believe is also caused by poor spatial locality of the memory.

4.4 Discussion and Remarks

This chapter has presented the experimental evaluation of our proposed framework. In this section, we presented a discussion and remarks of the achieved results.

The experiment regarding the scalability of the framework showed good results with nearly linear speedup in most cases. The data decomposition can be easily implemented in SSF and shows promising results in cases where the dataset can be partitioned equally. However, partitioning a dataset into several sub-sets is not always possible, as there may be dependence among the data. The speedup presented by task decomposition only has a linear behavior when the modules have a similar computational cost, which leads us to conclude that the manner of task decomposition is performed interferes on the framework scalability.

According to the experiments, the latency of communication between the modules and the shared memory was low, when compared with the time required to perform processing pertaining to video surveillance operations.

Results regarding the Feature Extraction Server (FES) demonstrated that we are able to achieve almost linear speedup, provided that the method is computation intensive, and also demonstrated that enabling cache decreases by 80% the runtime.

Chapter 5

Conclusions

This work proposed a novel framework to allow further development on computer vision methods and surveillance applications. The architecture of the Smart Surveillance Framework (SSF) allows the 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 framework also provides two important components, the feature extraction server and the complex query server, these components maximize the computational resource usage and facilitate the scene understanding, respectively.

The proposed framework will be made publicly available and besides of making surveillance research using real data and in real-time processing easier, it will also allow researchers to provide their methods (implemented as modules) to be used by other researchers to compare how results. Nowadays, it is difficult to compare results to previously published works since the code is not always available or it is necessary to adapt the code to work on new data sets. By using the SSF, one can provide the source-code (or just its compiled version) of the module to solve a computer vision problem and when another researcher proposes a novel solution, he/she can use that module to compare the results different data sets or to compare the computational cost in the same machine. Therefore, the SSF might also contribute to a more accurate validation of computer vision algorithms, mainly those related to surveillance.

5.1 Future Works

As future works, we propose: *a*) extensions of the SSF to provide new features including the distribution of the processing and the data to multiple computers to

make it even more scalable; *b*) improvement of the Graphical User Interface (GUI) to allows data processing visualization, which currently is not focus of many research but is very useful to provide a better understanding of the behavior of computer vision algorithms, mainly when large amounts of data are used and the processing is done in parallel; *c*) improvement of persistence mechanism in secondary memory using the information of which modules are running and which data requests are being performed; and, finally, *d*) the incorporation of security and privacy to the framework by adding data encryption and user permission levels to preserve person's identities, which will allow the SSF to be employed in real surveillance applications.

Bibliography

Afrah, A., Miller, G., and Fels, S. (2009). Vision system development through separation of management and processing. In *Proceedings of IEEE International Symposium on Multimedia (IISM 2009)*, pages 612--617.

Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Computing Surveys*, 43(3):1--43. ISSN 0360-0300.

Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of Spring Joint Computer Conference (AFIPS 1967)*, pages 483--485.

Bedagkar-Gala, A. and Shah, S. K. (2014). A survey of approaches and trends in person re-identification. *Image and Vision Computing*, 32(4):270--286. ISSN 02628856.

Choe, T. E., Deng, H., Guo, F., Lee, M. W., and Haering, N. (2013). Semantic Video-to-Video Search Using Sub-graph Grouping and Matching. In *Proceedings on IEEE International Conference on Computer Vision Workshops (ICCVW 2013)*, pages 787--794.

Collins, R., Lipton, A., Kanade, T., and Fujiyoshi, H. (2000). A system for video surveillance and monitoring. Technical report, Carnegie Mellon University.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, volume 1, pages 886--893. ISSN 1063-6919.

Davies, A. C. and Velastin, S. A. (2007). A progress review of intelligent cctv surveillance systems. In *Proceedings of IEEE Intelligent Data Acquisition and Advanced Computing Systems (IDAACS 2007)*, pages 417--423.

de Melo, V. H. C., Leão, S., Menotti, D., and Schwartz, W. R. (2014). An optimized sliding window approach to pedestrian detection. In *Proceedings on International Conference on Pattern Recognition (ICPR 2014)*, pages 1 -- 8.

de Siqueira, F. R., Schwartz, W. R., and Pedrini, H. (2013). Multi-scale gray level co-occurrence matrices for texture description. *Neurocomputing*, 120:336 -- 345. ISSN 0925-2312.

Dollar, P., Belongie, S., and Perona, P. (2010). The fastest pedestrian detector in the west. In *Proceedings of the British Machine Vision Conference (BMVC 2010)*, pages 68.1--68.11.

Dollar, P., Tu, Z., Perona, P., and Belongie, S. (2009). Integral channel features. In *Proceedings of the British Machine Vision Conference (BMVC 2009)*, pages 91.1--91.11.

Dollár, P., Wojek, C., Schiele, B., and Perona, P. (2012). Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743--61. ISSN 1939-3539.

Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627--1645. ISSN 0162-8828.

Fleck, S. and Strasser, W. (2008). Smart camera based monitoring system and its application to assisted living. *Proceedings of the IEEE*, 96(10):1698--1714. ISSN 0018-9219.

Fleck, S. and Strasser, W. (2010). Towards secure and privacy sensitive surveillance. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2010)*, pages 126--132.

Forsyth, D. A. and Ponce, J. (2011). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference. ISBN 0130851981.

Gauglitz, S., Höllerer, T., and Turk, M. (2011). Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, 94(3):335--360. ISSN 0920-5691.

Gavrila, D. (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82--98. ISSN 1077-3142.

Gilbert, N. (2007). Dilemmas of privacy and surveillance: Chaljenges of technological change. Technical report, University of Surrey.

Haering, N., Venetianer, P. L., and Lipton, A. (2008). The evolution of video surveillance: An overview. *Machine Vision and Applications*, 19(5-6):279--290. ISSN 0932-8092.

Hampapur, A. (2008). Smart video surveillance for proactive security. *IEEE Signal Processing Magazine*, 25(4):136--134. ISSN 1053-5888.

Hampapur, A., Brown, L., Connell, J., Pankanti, S., and Senior, A. (2003). Smart surveillance: applications, technologies and implications. In *Proceedings of International Conference on Information, Communications and Signal Processing (ICICS 2003)*, pages 1133--1138.

Hampapur, A., Brown, L., Feris, R., Senior, A., Shu, C. F., Tian, Y., Zhai, Y., and Lu, M. (2007). Searching surveillance video. In *Proceedings on IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS 2007)*, pages 75--80.

Haralick, R. M., Shanmugam, K., and Dinstein, I. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610--621. ISSN 0018-9472.

Huang, T. (2014). Surveillance video: The biggest big data. *Computing Now*, 7(2). ISSN 0823-6437.

Kumar, V. (2002). *Introduction to Parallel Computing*. Addison-Wesley Longman Publishing Co., Inc., 2nd edition. ISBN 0201648652.

Li, J. and Allinson, N. M. (2008). A comprehensive review of current local features for computer vision. *Neurocomputing*, 71(10-12):1771--1787. ISSN 0925-2312.

Liu, H., Chen, S., and Kubota, N. (2013). Intelligent video systems and analytics: A survey. *IEEE Transactions on Industrial Informatics*, 9(3):1222--1233. ISSN 1551-3203.

Liu, W., Miller, P., Ma, J., and Yan, W. (2009). Challenges of distributed intelligent surveillance system with heterogenous information. *Proceedings of QRASA*, pages 69--74.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91--110. ISSN 0920-5691.

Marin, J., Vazquez, D., Amores, J., Lopez, A., and Leibe, B. (2013). Random forests of local experts for pedestrian detection. In *Proceedings on IEEE International Conference on Computer Vision (ICCV 2013)*, pages 2592--2599. ISSN 1550-5499.

Mikolajczyk, K. and Schmid, C. (2005). Performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615--30. ISSN 0162-8828.

Mitra, S. and Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 37(3):311--324. ISSN 1094-6977.

Nascimento, E. R., Schwartz, W. R., and Campos, M. F. M. (2012). Edvd - enhanced descriptor for visual and depth data. In *Proceedings on International Conference on Pattern Recognition (ICPR 2012)*, pages 2776 -- 2779. ISSN 1051-4651.

Nazare, A. C., Santos, C. E., Ferreira, R., and Schwartz, W. R. (2014). Smart surveillance framework: A versatile tool for video analysis. In *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV 2014)*, pages 753--760.

Odobez, J.-M., Carincotte, C., Emonet, R., Jouneau, E., Zaidenberg, S., Ravera, B., Bremond, F., and Grifoni, A. (2012). Unsupervised activity analysis and monitoring algorithms for effective surveillance systems. In *Proceedings of International Conference on Computer Vision (ICCV 2012)*, pages 675--678.

Piccardi, M. (2004). Background subtraction techniques: A review. In *Proceedings of International Conference on Systems, Man and Cybernetics (ISMC 2004)*, volume 4, pages 3099--3104. ISSN 1062-922X.

Poppe, R. (2007). Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1-2):4--18. ISSN 1077-3142.

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976--990. ISSN 0262-8856.

Porikli, F., Bremond, F., Dockstader, S., Ferryman, J., Hoogs, A., Lovell, B., Pankanti, S., Rinner, B., Tu, P., and Venetianer, P. (2013). Video surveillance: Past, present, and now the future. *IEEE Signal Processing Magazine*, 30(3):190--198. ISSN 1053-5888.

Posner, R. A. (2008). Privacy, surveillance, and law. *The University of Chicago Law Review*, 75(1):245--260. ISSN 0041-9494.

Prisacariu, V. and Reid, I. (2009). fasthog - a real-time gpu implementation of hog. Technical report 2310/09, Department of Engineering Science, Oxford University.

Randen, T. and Husoy, J. (1999). Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291--310. ISSN 0162-8828.

Räty, T. D. (2010). Survey on contemporary remote surveillance systems for public safety. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 40(5):493--515. ISSN 1094-6977.

Regazzoni, C. S., Ramesh, V., and Foresti, G. L. (2001). Special issue on video communications, processing, and understanding for third generation surveillance systems. *Proceedings of the IEEE*, 89(10):1355--1539. ISSN 0018-9219.

San Miguel, J. C., Bescós, J., Martínez, J. M., and García, A. (2008). Diva: A distributed video analysis framework applied to video-surveillance systems. In *Proceedings of International Workshop on Image Analysis for Multimedia Interactive Services (IWIAMIS 2008)*, pages 207--210.

Sedky, M. H., Moniri, M., and Chibelushi, C. C. (2005). Classification of smart video surveillance systems for commercial applications. In *Proceeding of IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS 2005)*, pages 638--643.

Shah, M., Javed, O., and Shafique, K. (2007). Automated visual surveillance in realistic scenarios. *IEEE Multimedia*, 14(1):30--39. ISSN 1070-986X.

Siebel, N. T. and Maybank, S. J. (2004). The advisor visual surveillance system. In *Procedings of ECCV Workshop Applications of Computer Vision (ECCV-ACV 2004)*, pages 1--9.

Souza, J., Ferreira, C. A. M., Júnior, C. E. S., de Melo, V. H., and Schwartz, W. (2014). Self-organizing traffic lights: A pedestrian oriented approach. In *Proceedings of SIBGRAPI Workshop of Undergraduate Works (WUW-SIBGRAPI 2014)*, pages 1--6.

Suvonvorn, N. (2008). A video analysis framework for surveillance system. In *Proceedings of IEEE Workshop on Multimedia Signal Processing (MMSP 2008)*, pages 867--871.

Thornton, J., Baran-Gale, J., Butler, D., Chan, M., and Zwahlen, H. (2011). Person attribute search for large-area video surveillance. In *Proceedings of IEEE International Conference on Technologies for Homeland Security (HST 2011)*, pages 55–61.

Tian, Y. L., Brown, L., Hampapur, A., Lu, M., Senior, A., and Shu, C. F. (2008). Ibm smart surveillance system (s3): Event based video surveillance system with an open and extensible framework. *Machine Vision and Applications*, 19(5-6):315--327. ISSN 0932-8092.

Valera, M. and Velastin, S. (2005). Intelligent distributed surveillance systems: A review. *IEE Proceedings - Vision, Image, and Signal Processing*, 152(2):192. ISSN 1350-245X.

van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582--96. ISSN 1939-3539.

Vaquero, D. A., Feris, R. S., Tran, D., Brown, L., Hampapur, A., and Turk, M. (2009). Attribute-based people search in surveillance environments. In *Proceedings of Workshop on Applications of Computer Vision (WACV 2009)*, pages 1--8.

Venetianer, P. L. and Deng, H. (2010). Performance evaluation of an intelligent video surveillance system – a case study. *Computer Vision and Image Understanding*, 114(11):1292--1302. ISSN 1077-3142.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume 1, pages I–511--I–518. ISSN 1063-6919.

Wang, G., Tao, L., Di, H., Ye, X., and Shi, Y. (2012). A Scalable Distributed Architecture for Intelligent Vision System. *IEEE Transactions on Industrial Informatics*, 8(1):91--99. ISSN 1551–3203.

Winkler, T. and Rinner, B. (2010). A systematic approach towards user-centric privacy and security for smart camera networks. In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2010)*, pages 133–141.

Xia, J., Rao, W., Huang, W., and Lu, Z. (2013). Automatic multi-vehicle tracking using video cameras: An improved camshift approach. *KSCE Journal of Civil Engineering*, 17(6):1462--1470. ISSN 1226-7988.

Xie, W., Shi, Y., Xu, G., and Mao, Y. (2002). Smart platform - a software infrastructure for smart space (siss). In *Proceedings of IEEE International Conference on Multimodal Interfaces (ICMI 2002)*, pages 429--434.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45. ISSN 0360-0300.

Zhang, J., Marszałek, M., Lazebnik, S., and Schmid, C. (2007). Local features and kernels for classification of texture and object categories: A comprehensive study. *International Journal of Computer Vision*, 73(2):213--238. ISSN 0920-5691.

Zhang, X. and Gao, Y. (2009). Face recognition across pose: A review. *Pattern Recognition*, 42(11):2876--2896. ISSN 0031-3203.

Appendix A

Application Example: Self-Organizing Traffic Lights

This appendix on based in Souza et al. [2014] and it described how an application can be developed using the SSF.

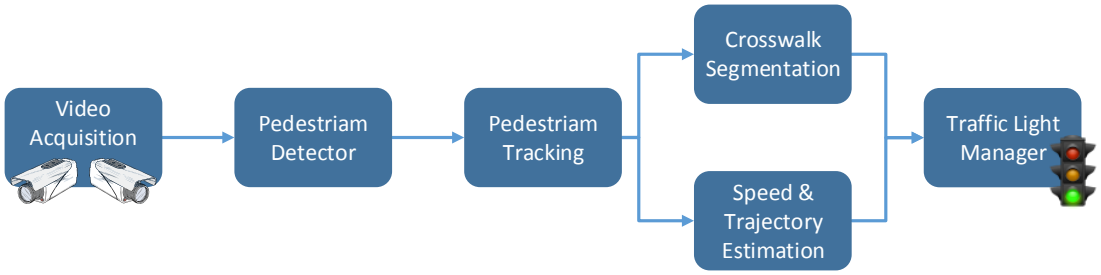
The traffic light is a valuable device to control the vehicular and pedestrian traffic. One of its main issues is that several traffic lights might be improperly calibrated once they do not consider the differences in pedestrian mobility from region to region. As each region presents different pedestrians with different characteristics, there is a need for automatic approaches.

For such purpose, two challenging cases in transport engineering literature must be handled. The first case happens when pedestrians with reduced speed cannot cross the street within the available time. The second case happens when the traffic light for pedestrians remains open for a long time even when there are no pedestrians waiting to cross. Such problem may be tackled by the same approaches adopted by traditional visual surveillance methods.

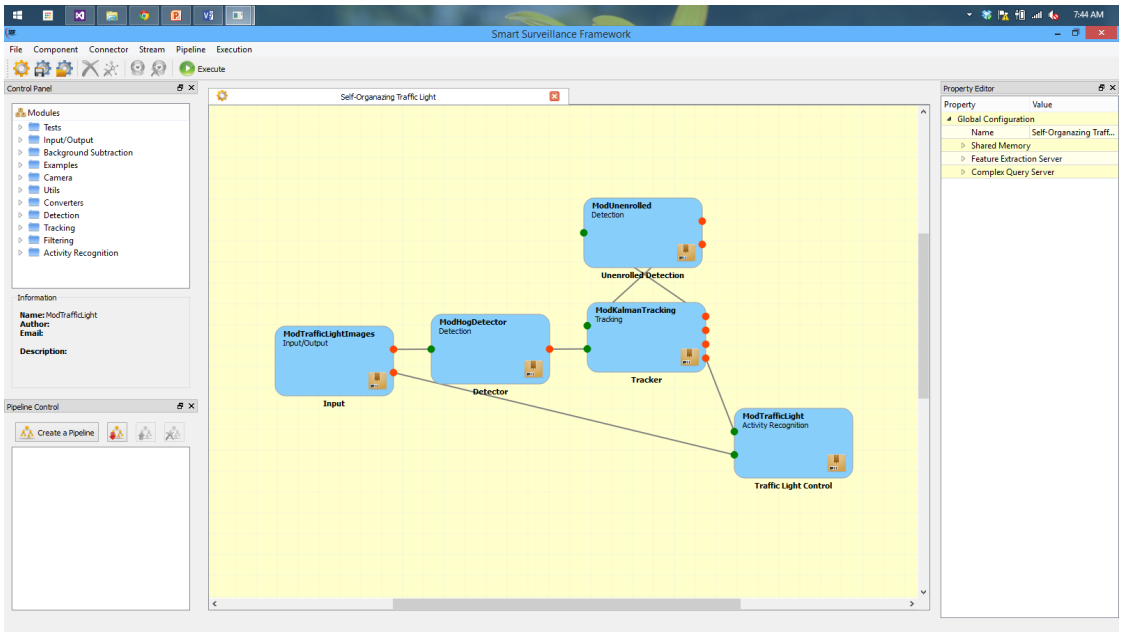
The problem can be solved by the following approach, divided in the steps depicted in Figure A.1a and Figure A.1b presents the application configuration on the SSF Graphical User Interface (GUI).

The steps of application are described as follows:

1. First, the pedestrians must be detected using a pedestrian detection method, such as the HOG Detector [Dalal and Triggs, 2005], the *LatSVM* Detector Felzenszwalb et al. [2010] or [de Melo et al., 2014] (Pedestrian Detection Module).



(a) Workflow of the application example. The pedestrian position in each frame is determined in the pedestrian detector and pedestrian tracking modules. Then, the pedestrian position is used to estimate a distribution map where pedestrians walk when crossing the street and to estimate the speed and trajectory of these pedestrians to set the red flashing time in the pedestrian traffic light. (Adapted from Souza et al. [2014].)



(b) Configuration of the Traffic Light application.

Figure A.1: Self-Organizing Traffic Lights application example.

2. Based on the detected pedestrians, one must follow the pedestrians based on a tracking approach to estimate where they are heading to and their velocity (Tracking Module).

3. The next step is splitted into two parts. The first part handles the crosswalk segmentation based on the pedestrian's velocity (Crosswalk Segmentation Module). In this scenario, pedestrians that are moving belongs to the crosswalk, while non-moving pedestrians are more likely to be at the crosswalk's border. The second part estimates the speed and trajectory of each pedestrian (Speed Module).

4. The information collected by both parts are employed by the traffic light manager to determine the amount of time of red and flashing red, which will allow the pedestrians to finish their cross or to lock the traffic light if there are no pedestrians waiting to cross (Traffic Light Manager Module).

Such solution to the problem may be easily implemented using the SSF since it allows to abstract several layers of the problem. The user may split each of these steps into its equivalent modules. Most modules that compose the traffic light pipeline do not need to be implemented. Hence, the major concern of the user is to handle the data through the modules and implementing the traffic light manager.

A of the SSF main advantages is its flexibility to test several parameters and methods for each module, which allows the user to select the best ones for the given application. For instance, the user can easily try different pedestrian detectors to find which is the more suitable for his/her application.

Finally, the implemented solution to this problem can be easily shared and assessed by peers, once the user makes the module publicly available.