DEGREE OF DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING AND SCIENCE

DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXII Cycle

UNIVERSITY OF MODENA AND REGGIO EMILIA

INFORMATION ENGINEERING DEPARTMENT

Ph.D. DISSERTATION

Video Streaming and Video Surveillance in Mobile Scenarios

Candidate:
Giovanni Gualdi
Advisor:
Prof. Rita Cucchiara
Co-Advisor:
Prof. Andrea Prati
The Director of the School:
Prof. Sonia Bergamaschi

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXII Ciclo

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Video Streaming and Video Surveillance in Mobile Scenarios

Tesi di:
Giovanni Gualdi
Relatore:
Prof. Rita Cucchiara
Correlatore:
Prof. Andrea Prati
Il Direttore:
Prof. Sonia Bergamaschi

To Him,

who is so immense
to hold the universe within the span of His hand,

who is so wise
to create in a blink all the things that we are striving to understand
and those that we will never discover,

who came to look for me
well before I could even think of finding Him,

who gave to my heart the first biological beat
and to my life the first eternal breath.

Contents

List of Figures

List of Tables

Acknowledgments

After the exciting academic and professional experiences, that started in 2002 before my graduation as a computer engineer and took place one after another in the following years, I eventually ended up in 2006 to realize the need and (above all) the desire to go back to school, for the sake of learning new things... better late than never.

This thesis is the result of more than three years spent at the ImageLab laboratory (University of Modena and Reggio Emilia, Italy), during my PhD. It was an exiting time, a tough time, a time with long working days (and sometimes nights); definitely a time for learning new things (and a lot of them!). Yes, looking back, now I feel myself definitely grown from a professional point of view; and yes, I would do it all over again.

First of all I want to acknowledge my advisor Prof. Rita Cucchiara and my co-advisor Prof. Andrea Prati. Actually, it was Andrea that created the conditions for that crazy spark of going back to school. Without their assistance I would have never produced the pages that you are holding (or browsing) now.

I want to express all my gratitude to my fiancee Alice, to dad, mum and my brother Carlo. A few formal words in an acknowledgment section will never express the love and the thankfulness I have for you. Without you there would be no professional life neither private life for me; you are color to my life. This is just a truth that comes from my heart.

I wish to list the names of all the other people that I am thankful to, for having assisted, sustained and encouraged me during these years; but I would definitely skip some important names. A joyful thought to all my precious friends I have in Italy and in California.

Before closing, the most important acknowledgment: endless thanks to God my creator, and my wonderful Lord, Jesus Christ: the dedication of the thesis is to Him.

Giovanni Gualdi,
Modena, January 2010

Chapter 1

Introduction

The last two decades have witnessed a surge of interest in automated video surveillance from business and research. The first prototypes and commercial products were typically embodied by monolithic systems, where video sensors, processing units and output devices were tightly coupled; thanks to the significant advances of the last ten years in algorithms, hardware platforms and networks, the paradigm of video surveillance is increasingly drifted toward the mobility and distribution of its modules (i.e. source, processing, monitoring, storage, etc.).

This mobile and distributed approach is what typically goes under the name of *mobile video surveillance*, that is still at the present time a quite general and blurry term, since it can assume different flavors depending on the scientific perspective from which video surveillance is considered. Indeed, it is not a matter of debate the fact that video surveillance touches a very broad spectrum of different scientific topics: a non exhaustive list would comprehend disciplines from computer and electrical engineering and computer science: computer vision, image processing, machine learning, pattern recognition, graph theory, signal theory, networks design and infrastructures, multimedia streaming, hardware design, power consumption, parallel and distributed computation, etc.

Nevertheless, reviewing both scientific literature and business strategies, from the broad multi-disciplinary nature of video surveillance, it is possible to identify a set of topics that have been involved in a deeper and more direct manner than others into this move toward mobility and distribution, namely computer vision algorithms, data compression and communication, systems and applications for embedded hardware.

Remaining focused on these aspects, the thesis will tackle some key issues that are strategic at the present time for an organic and consistent development of *mobile video surveillance*: at first we deal with video compression and streaming, that can be considered as a preliminary step; then we focus on computer vision techniques, as the real core for automatic surveillance; in both cases we design and evaluate processes and algorithms specifically aimed to succeed in mobile scenarios.

The thesis is organized as follows.

In the introductory *Chapter 2* we analyze the meaning that mobile video surveillance takes according to the different observation perspectives; then we propose a unified *mobile video surveillance* definition that is intended to be modern and comprehensive of all the other scattered terminologies. This proposal is founded on the understanding of what really makes “mobile” a video surveillance system. We will see that there is not a defined and sharp border, rather it is a property that increases together with some features of the modules that constitute the system. We will also propose an overview of the most remarkable works that directly marked the development of mobile video surveillance or that opened its way with preparatory findings.

The thesis is then divided in three parts, part I, part II, part III. Each part is made of a typical introductory chapter that contains a description of the topic, the statement of a problem that is not solved yet, a review of related works; eventually we propose innovative solutions or methodologies to improve the state of the art solutions. After a detailed discussion of these points, the thesis part closes with experimental results and evaluations. There is also a concluding part (part IV) that contains some appendices as well.

Part I is focused on video streaming. In *Chapter 3* we analyze at first what the requirements and the conditions for a successful video streaming are, remaining within the context of mobile video surveillance; then we discuss about the main shortcomings of general purpose video streaming architectures and systems (typically aimed for entertainment) when applied to surveillance context.

In *Chapter 4* we propose an architecture called Moses (MObile Streaming for vidEo Surveillance) that is designed to work with low-bandwidth and ubiquitous wireless networks and to overcome the main limitations of the other streaming systems, namely latency, image quality, video fluidity and frame losses. Moses is made of two open-source applications: an encoder (called Movie, Mobile Video Encoder) and a decoder (Movide, Mobile Video Decoder); the latter is devised to work on PCs and PDAs.

Evaluating the performance of a live video streaming system is not an easy task, since it is mostly based on the perceived quality of live (not stored) videos by means of the human receiver. In *Chapter 5* we propose an innovative methodology to extract quantitative measures on the four aforementioned aspects on any streaming system, even on those proprietary and commercial systems that are provided with binaries executables only (no source code) and therefore cannot be modified for performance assessment, or those systems where it is not possible to access, modify or monitor the network traffic.

Part II is focused on object tracking. This work can take very different paths depending on the architectural set up of the mobile surveillance systems; we focus on two extreme cases: as a first case, we propose a scenario with cameras that are installed on movable platforms but operate being fixed; the processing is supposed to be remote and the challenging part is to provide video compression and streaming that do not hinder or compromise remote video processing aimed to object tracking. This scenario is analyzed in *Chapter 7*.

As a second case, in *Chapter 8* we propose a scenario where the challenges of video tracking are shifted from video compression and streaming to computer vision: we tackle the problem of object tracking with a camera that is subject to unconstrained and unpredictable motion and changing focal length (i.e. zooming in and out). We will define a framework that translates the object tracking problem to a special case of clique extraction, that is solvable with an euristic coming from graph theory. Experimental results, provided in *Chapter 9* are collected exploiting publicly annotated datasets.

Part III deals with the case of video surveillance in mobile context where it is not feasible to rely on object tracking due to the presence of mobility and extreme visual clutter of real-world outdoor scenarios (e.g. night vision, construction working sites, dense vehicular traffic scenes, etc.). As a matter of facts, the approaches described in Part II would likely fail in these contexts and our proposal is to perform object detection through appearance-based methods: this cue is indeed more robust to visual scene clutter than motion segmentation. We exploit a state of the art classifier technique based on covariance descriptors and apply it to pedestrian detection; nevertheless the results in terms of accuracy and efficiency that we have collected testing this method in the aforementioned challenging scenarios are rather poor. Therefore we propose a twofold set of improvements: the first on the classifier side (Chapter 11), the second on the detector side (Chapters 12 and 13).

In *Chapter 11* we show how the use of the relevance feedback technique along with the training phase of a boosting classifier can improve its accuracy, without adding significant computational burden to that procedure that is by itself very CPU-consuming. The relevance feedback can be split in two independent parts: one is implicit (i.e. totally automatic, since it exploits data obtained through background modeling), the other explicit (i.e. takes advantage of human assessment).

In the same chapter we also propose a modification of the classifier in order to deal with objects characterized by circular shapes, since the original method performed quite weakly on this kind of features. In the specific, we show how the use of polar transformations before the computation of the covariance descriptors can significantly improve classification accuracy. We also demonstrate that the use of multi-spectral (color) image derivatives benefits the covariance descriptor for classification purposes. Both modifications can be successfully applied for detecting the accurate position of the head of pedestrians. Furthermore, we extend the evaluation of the polar transformation proposal to polymer classification in photomicrograph contexts.

The typical technique used for detecting objects in an image through a (binary) classifier is the sliding window paradigm, that is a brute force search of the learned pattern (e.g. pedestrian) over the whole space of possible window states. As soon as the dimension of the state space grows (x,y position is the least parameter set, but others can be added, like scale, aspect ratio, rotation, etc.), this approach becomes intractable from a computational point of view. *Chapter 12* proposes to exploit a set of visual cues, orthogonal to appearance, in order to make the sliding window approach faster; the approach has the additional advantage to benefit detection accuracy as well. In the specific, we propose to exploit motion information to focus the attention over areas where motion is present or was present in the recent past and to use a rough estimation of the scene perspective in order to reject all the windows that are significantly out of scale.

Applying a probabilistic view to object detection, in *Chapter 13* we propose a Monte Carlo sampling approach for estimating the likelihood density function of the searched objects, using Gaussian kernels. The estimation is performed with a multi-stage strategy where the proposal distribution is progressively refined by taking measurements into account. For videos, this approach is plugged into a Bayesian-recursive framework which exploits the temporal coherency of the pedestrians. This proposal is demonstrated to yield a detection accuracy very similar to the sliding window paradigm, but requiring a much lower computational load.

After concluding remarks, *Part IV* contains a series of appendices. *Appendix A* describes the prototype devised to video secuity in construction working sites. This application is aimed to the automatic and real-time detection of workers that do not wear the protective helmet and exploits the techinques described in part III.

Appendix B describes the freeware implementation of MOSES with its two applications MOVIE and MOVIDE. Both softwares can be freely downloaded from the author's home page[1].

In *Appendix C* we provide for the reader's convenience the mathematical definitions, lemmas and a theorem for the Dominant Set framework that were not exaustively treated in Chapter 8.

1.1 Contributions of the Thesis

Several main contributions of the thesis are listed below.

- An up-to-date and unified definition of mobile video surveillance, that is comprehensive of the several scientific perspectives from which it can be observed.
- The definition of an innovative methodology for video-streaming evaluation, that provides a quantitative measure of latency, image quality, video fluidity and frame losses even on closed and proprietary streaming systems.
- The design of an open-source video compression, streaming and decoding system tailored for mobile video surveillance. The encoder software works on x86 architecture, the decoder on x86 and StrongArm architectures.
- The performance evaluation of traditional object tracking algorithms fed with videos that passed through low bandwidth networks.
- The introduction of an algorithm for structural object tracking in videos with freely moving camera, translating the problem from the tracking domain to graph theory.
- The use of relevance feedback for enriching the last stages of a rejection cascade of boosting classifiers in order to improve classification performance.
- The use of polar transformation to improve classification performance for the detection of objects with circular features.
- The demonstrated evidence that multi-spectral (color) image derivatives generate covariance descriptors that are stronger for classification purposes w.r.t. luminance derivatives only.
- The use of a pedestrian classifier for automatic inference of scene perspective.

[1]imagelab.ing.unimore.it/imagelab/~gualdi/

- The use of motion history images and scene perspective to reduce the computational burden of object detection based on sliding window approach.
- The use of a multi-stage Monte Carlo sampling with boosting cascades for pedestrian detection.

Chapter 2

What is Mobile Video Surveillance?

Thanks to the spread of both mobile devices and wireless network accessibility, Ubiquitous Multimedia Access (UMA) has become a very common topic within the multimedia community during the last few years. Research centers and telecom providers address new, smart and efficient solutions for the ubiquitous access to multimedia data and in particular videos, from everywhere with mobile devices (laptops, PDAs or last generation cellular phones). Possible applications of such technology include consumer entertainment and digital TV broadcasting, video conferencing, telemedicine and telemanipulation, military applications, and eventually *mobile video surveillance*.

The meaning of mobile video surveillance is quite hazy and might assume very different meanings, depending on the contexts; it is often referred as a broad and generic super-class of emerging real-time video surveillance applications where the computational load is not completely in charge of a fixed platform directly connected to the camera. Sometimes it is the degree of motion of the front-end to give the flavor of mobility to a video surveillance system. Conversely, in multimedia or telecommunication contexts, the term mobile is generally related to the type of data connection, making mobile video surveillance a mere extension of traditional video surveillance systems (attended by human operators) to ubiquitous wireless connectivity.

Each of these definitions is partially correct but certainly not exhaustive, since the word “mobile” could be easily exchanged with more specific terms like distributed, embedded, battery powered, moving, wirelessly interconnected, etc.

Since the present dissertation deals with the research efforts spent in studying, optimizing and advancing some founding parts of mobile video surveillance, the next sections provide a unified and thorough analysis of what mobile video surveillance really is. It is not our intention to create a new dictionary definition, but to analytically dissect the constituting parts of mobile video surveillance systems (Section 2.1), understand what their features and interrelations are (Section 2.2), and eventually provide an overview of the most important contributions in the scientific literature over these topics and how they can be contextualized into our analysis (Section 2.3).

Publications related to Chapter 2: [b,i]; see the list of author’s publications, page 147.

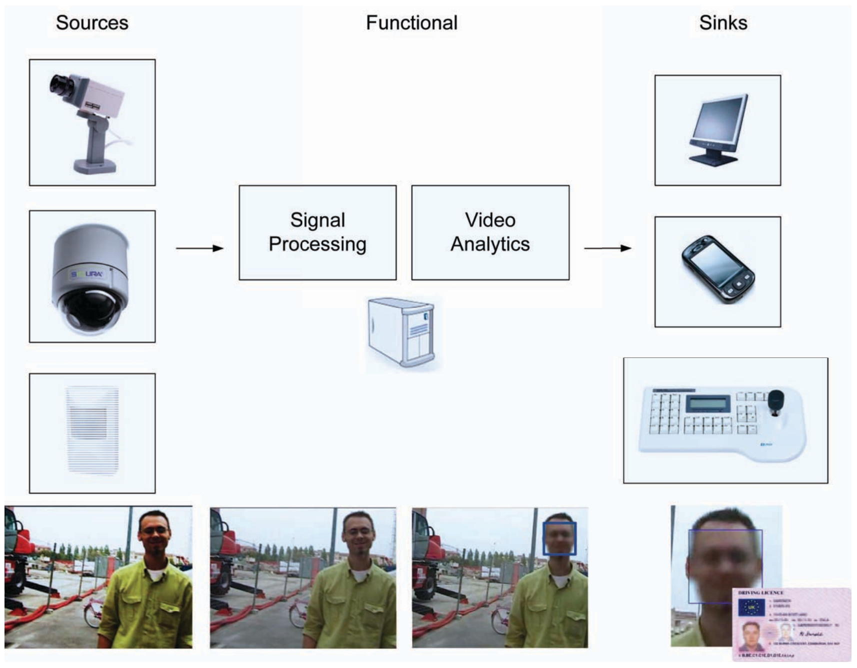


Figure 2.1: Sources, functional and sink modules in an example of video surveillance system providing face recognition.

2.1 Modules of Video Surveillance Systems

The modules (or layers) of generic video surveillance systems can be grouped in three main classes: source, functional and sink modules (see also Tab. 2.1 and Fig. 2.1):

- **Source Modules**: cameras are obviously the main sensors belonging to this group, but given the multi-sensory nature of the events to capture, video surveillance systems can be equipped with sensors belonging to domains that go beyond the vision: audio sensors, ID sensors, (e.g. RFIDs, bar codes, biometric sensors like fingerprint, etc.), or context sensors (e.g. temperature, PIR, etc.); in some cases even storage devices can provide the functionalities of source modules, especially in off-line applications (e.g. forensic analysis), but they logically belong to the sink category. The source module is analog, being analog the nature of the sensed events, and it is followed by an analog-to-digital converter, in order to perform digital signal processing. Even if the modern sensors generally integrate the analog-to-digital unit inside the device itself, any operation from the digital conversion onward is not to be accounted as part of the source module, but it is typically a part of the functional processing modules;

- **Functional Modules**: the functional modules perform a processing task on the input, providing an output on the same or on a different domain; they can be divided in two further parts:

  - *Signal Processing Modules*: they aim to modify or transform the properties and the quality of the input, without necessarily extracting knowledge from it. In this class we include analog-to-digital converters, as well as signal encoders and decoders, streamers, but also some sensor pre-processing; in case of video, pre-processing is typically made of low-level and context-blind operations, such as quality enhancement (SNR), pixel-wise operations (e.g. filtering, morphology, etc.), projective transformations (e.g. homographies), warping, distortion removal, region-of-interest operations (resizing, cropping), etc. Pre-processing typically precedes analysis of higher complexity and it can be directly connected to the sensors.

  – *Purposive Modules*: these modules are devoted to extract information and thus knowledge from video or other sensor data; the commercial softwares, often called "video analytics", are based on computer vision and pattern recognition techniques and range from the most basic motion detection up to the new generation of algorithms that will provide, in the (coming) future, event detection, behavioral analysis, situation assessment and automatic focus of attention.

- **Sink Modules**: the sink modules are qualified by the lack of outputs to other modules (e.g. storage devices, back-end monitors, etc.) or, in case there are outputs, they run into different systems: for instance, the output of an alarming module could be connected to external actuators of alarms.

We did not mention here the modules providing network functionalities: even if they play an essential role in mobile video surveillance, in this dissertation they will be treated as black boxes which simply provide IP network connectivity, offering specific performance and properties (bandwidth, protocols, etc.).

2.2 Features of Video Surveillance Systems

There are three features, namely distribution, mobility and degree of motion of the sensors, that qualify the aforementioned modules and help in defining what mobile video surveillance systems are. We provide here an brief overview of such features:

- **degree of distribution of the system**, i.e. what the physical distance among modules is. A typical distinction is:

  – *monolithic systems*: all the modules are physically tightly coupled, from the sources to the sinks, including also the functional modules. In this type of systems, the network connectivity does not play any functional role, being used at most to control the whole system. Also those multi-camera systems where the cameras, even if located at different places, are functionally connected to a monolithic system (e.g. wide-baseline stereo pairs) can be considered part of this group;

  – *distributed systems*: there is physical distance among the modules, with variable degree of distribution: from one extreme, where only one portion of the system is detached from the whole rest (e.g. the video source and grabbing of a distributed network camera system), to the other extreme, where all the parts of the system are scattered far apart from all the others (e.g. a

	Module	Input	Output
Source	- video (cameras)		analog video
	- audio (microphones)		analog audio
	- ID (biometrics, RFIDs, bar codes, etc.)		analog/digital data
	- context sensors (Temperature, PIR, etc.)		analog/digital data
Functional: Signal Processing	- analog to digital converter	analog data	digital data
	- video encoder	digital video	bit stream
	- video decoder	bit stream	digital video
	- (other sensors) encoder	digital sensor data	bit stream
	- (other sensors) decoder	bit stream	digital sensor data
	- video pre-processing	video	video
	- (other sensors) pre-processing	digital sensor data	digital sensor data
Functional: Purposive	- video analytics	video	meta data
	- (other sensors) analytics	digital sensor data	meta data
	- sensors fusion	meta data	meta data
Sink	- storage	video, meta data	
	- user front-end	video, meta data	
	- alarms, actuators handler	meta data	*to actuators*

Table 2.1: Modules of video surveillance systems.

totally distributed system). The network connectivity in these systems plays a dominant role, and the properties to be aware of or to keep monitored are bandwidth, latency, degree of ubiquitousness of accessibility (especially for radio-mobile networks) and robustness on errors. The distributed surveillance systems are also defined as *remote surveillance systems*: indeed, the survey [3] points to the remote surveillance as one of the most challenging intelligent surveillance scenario.

This distinction is not only architectural but directly affects the system functionalities: monolithic systems with respect to distributed ones are obviously less expensive, smaller, simpler and the direct connection with the source modules makes video data available at high resolutions and frame rates; on the opposite such systems have a limited capability of covering wide areas of surveillance and a poor flexibility in processing parallelism. Conversely, distributed systems are potentially more effective: sensors can be displaced over wide spaces and computing resources can be organized in a flexible manner. This added value is obtained at the cost of a higher architectural complexity. The streaming of sensor data among the distributed modules plays a key role in distributed systems; part I of this dissertation is completely devoted to the topic of video streaming in such contexts.

- **degree of mobility of the system**, i.e. how does the location of the modules change over time:

  - *fixed modules*: the location of all the modules of the system never changes. Typically, these systems are powered through external outlets and connected by wires. From an algorithmic point of view, the video analytic modules can take advantage by this condition, since parameters and models (e.g. background suppression, camera geometry, camera color calibrations, etc.) can be calculated once and then repeatedly exploited thanks to the lack of motion;
  - *mobile modules*: the modules (or a part of them) are not constrained to stay at all time in the same location, but operate being fixed. Typically, the mobile blocks are provided with wireless network connectivity, even if there are counter-examples (e.g. [4] describes a cable-based mobile sensor architecture). From an algorithmic point of view, the analytic modules will require a bigger effort to obtain useful information from the sensed environment, since parameters and models might need a re-computation jointly to every module movement;
  - *moving modules*: the physical location of the modules (or a part of them) is not only unconstrained over time, but it is also varying while the module itself is operative.

  It is important to stress here the distinction between *mobile* and *moving*: the mobile module is free to move but operates while it is static, the moving module operates in motion. An example of mobile source module might be the set of cameras that observe a road construction site which moves from time to time: the cameras are moved together with the site, but they are operated being in a fixed position. An example of moving source module is a camera installed over a vehicle for traffic patrolling and automatic license plate recognition.

  The degree of mobility of the system has a strong impact on architectural properties of the surveillance systems such as the type of electric powering, data connection and processing architecture. The higher is the degree of mobility, the more the system heads toward battery powered solutions (rather than external outlet), small, embedded or specific purpose processor architectures (rather than general purpose ones), wireless network connectivity (rather than wired). In addition, the degree of mobility strongly affects the type of algorithms. Many video processing tasks cannot be performed with moving modules for the lack of geometric references, the reduced computational power of typical mobile processors and for the intrinsic different nature of the sensed data. Parts II and III of this dissertation are devoted to computer vision algorithm that can effectively work when some modules of the surveillance systems are mobile or moving.

- **degree of movement of the optical sensor**, with respect to the body of the camera (and not necessarily with respect to the sensed context). There are basically two categories: *static optical sensors* (fixed focal length is a necessary condition to be considered static) and *movable optical sensors*, such as varifocal or PTZ (pan-tilt-zoom) cameras. A new kind of sensor, that belongs to the first group even if the name could generate misconceptions, is the so called EPTZ (Electronic PTZ, [5]), that exploit high definition sensors (several mega pixels) in order to virtually reproduce the behavior of pan, tilt and zoom even if the camera and its focal length are static. There is another common misconception that confuses the mobility of the whole camera body and the mobility of its optical sensor: a clarifying counter-example is the case of a camera attached to the body of a vehicle: because of the vehicle motion, the camera might be moving with respect to the context, but its sensor is static with respect to the camera body.

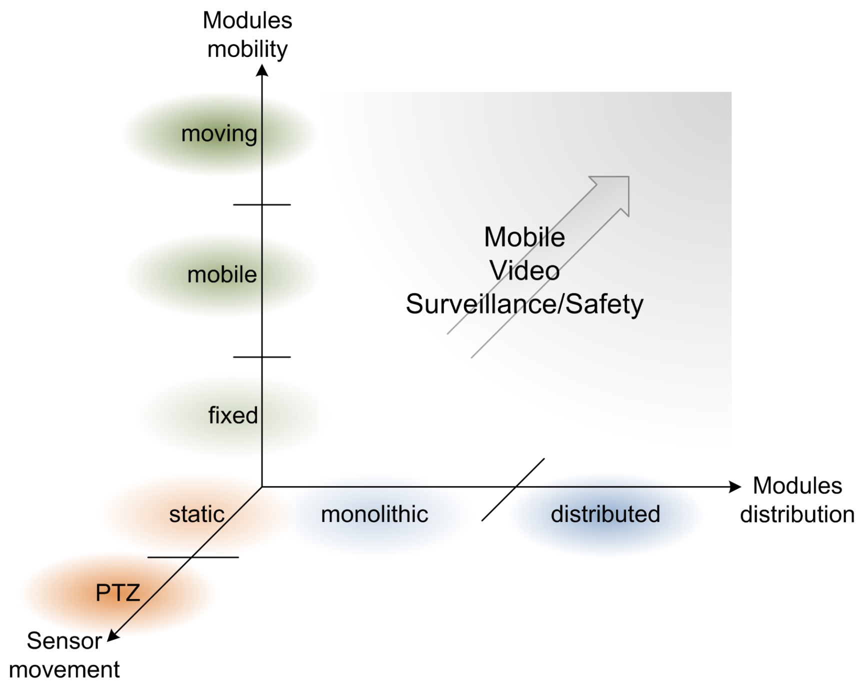
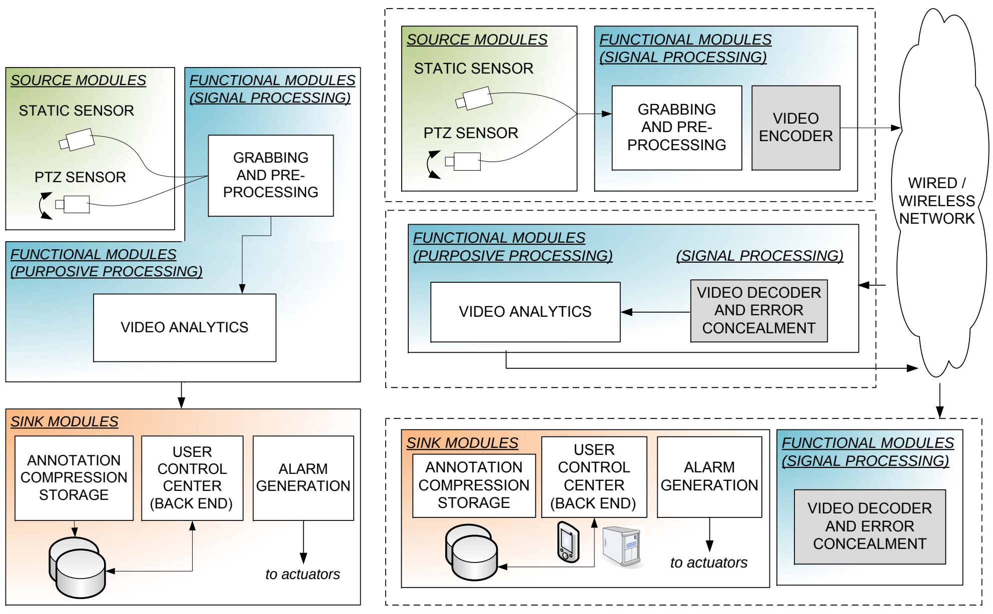


Figure 2.2: The three features of a system for video surveillance and the place taken by mobile video surveillance.

The three features proposed here are reciprocally orthogonal; provided this taxonomy, in the thesis we define as *mobile video surveillance* those systems that are characterized by a not-null degree of distribution, or mobility or both. The more distributed and the higher the degree of mobility of the modules, the more the system can be considered mobile; the third feature (degree of movement of optical sensor) does not bring any contribution. Fig. 2.2 gives a visual representation of the modules, their features and where mobile video surveillance is placed according to the provided taxonomy; typical video surveillance systems are depicted in Fig. 2.3: Fig. 2.3(a) shows a potential monolithic system, Fig. 2.3(b) a distributed one, where each dashed block can be fixed, mobile or moving.

2.3 Systems, Applications and Algorithms for Mobile Video Surveillance

This section provides an overview of the most remarkable works that directly marked the development of mobile video surveillance or that opened its way with preparatory findings.


SOURCE MODULES
STATIC SENSOR
PTZ SENSOR
FUNCTIONAL MODULES (SIGNAL PROCESSING)
GRABBING AND PRE-PROCESSING
FUNCTIONAL MODULES (PURPOSIVE PROCESSING)
VIDEO ANALYTICS
SINK MODULES
ANNOTATION COMPRESSION STORAGE
USER CONTROL CENTER (BACK END)
ALARM GENERATION
to actuators
VIDEO ENCODER
WIRED / WIRELESS NETWORK
(SIGNAL PROCESSING)
VIDEO DECODER AND ERROR CONCEALMENT

(a) (b)

Figure 2.3: Video surveillance systems: example of monolithic system (a) and of distributed system (b). Any dashed block could be fixed, mobile or moving.

Monolithic, fixed:
this is the basic architecture of the classical video surveillance systems, being compact and lacking of any degree of mobility. As depicted in Fig. 2.3(a), the source module sends data to a cascade of functional modules (from pre-processing to purposive) which exploit computer vision techniques. Detected objects and events of interest are sent to the sink modules that generate alarms, annotate the videos or store them for future tasks of posterity logging. This basic architecture is well established from several years: among the first important experimentations, we find the VSAM project [6] financed by Darpa (as we will see later in this section, the VSAM project has expanded from a fixed system to a very generic moving and distributed system). The videos, the annotations and the alarms are sent to the connected back-end. Being the cameras fixed, a reference model of the scene can be inferred or reconstructed to exploit background suppression for segmenting moving objects; the popular proposal of Stauffer and Grimson [7] uses Mixture of Gaussians for background modeling and suppression; a simplified method was proposed in W4 [8]. After these seminal works, tens of modified solutions were proposed in order to account for additional issues, like shadows [9], multi-layer motion segmentation [10] and many others. After segmentation, discriminative appearance-based tracking is often adopted [3, 11–13], where the appearance (color, texture, etc.) of segmented data is matched against the previously segmented and tracked objects. Still inside the monolithic systems, but closer to the border with distributed ones, there are video surveillance systems based on multiple cameras that are typically wire-connected to a central video grabber. In such systems the labels of the objects (i.e. the identity) are to be kept consistent not just along the time but also across the camera views. An example of approach for consistent labeling with overlapping views is [14],

where auto calibration is adopted and bayesian inference allows the discrimination over uncertainty due to overlapped views of groups of people. In traditional fixed systems, on top of static cameras, PTZ cameras are employed also. In such cases, it is possible to construct a reference mosaic in order to adopt detection methods similar to background suppression [15], or activity maps, to focus the PTZ camera on specific regions of interest out of the complete viewable field [16]. Otherwise, knowing the camera motion, the viewing direction of the PTZ camera can be synchronized and guided by the information extracted with the static cameras [17].

Distributed, fixed:
the concept of distributed surveillance system, synonym of remote surveillance system, is several years old [18]: at that time, the video analytics was performed at the camera side and only meta-data was sent over the network to the control center. A much wider distributed system was proposed under the name DIVA [19] which compressed and streamed the video for remote analysis. Another work that exploits the DIVA framework is [20], where the video streaming of some cameras flows on wireless connections. Many proposals have been defined in this area, among the others [21, 22]: the latter provides some considerations on the future of distributed (and implicitly fixed) systems.

In fixed systems, either monolithic or distributed, the most used technique for the detection of objects is undoubtedly the foreground object segmentation through background suppression.

Monolithic, mobile:
there is not much difference between these systems and the *monolithic, fixed* ones, a part for the fact that the video analytic algorithms need to be working in mobile contexts. Indeed the architecture of Fig. 2.3(a) is correct also in this scenario, with the addition that the whole block is installed on a mobile platform. The computer vision techniques that exploit prior knowledge of the context and the scene (e.g. techniques based on the scene geometry or on the background image), can be used in this mobile contexts only if they are given the ability to be properly re-initialized or updated. An example of geometry recovery used for people detection and counting that fits mobile scenarios, is the one depicted in [23], where the geometry calibration parameters of the whole system is re-computed using a fast and semi-supervised approach.

Distributed, mobile:
a distributed and mobile system often requires the introduction of wireless network communications, introducing limitations in the bandwidths: this can pose serious problems to video analysis, in case it is performed *after* the compression; [24] addresses tracking on low frame rate videos, well suited for compression that reduces temporal resolution. On the side of compression that degrades video quality, Chapter 7 discusses the performances of object tracking on highly compressed video data.

Monolithic, moving:
those stand-alone systems that offer an unconstrained mobility, like portable and hand-held systems (e.g. smart-phones, laptops, etc.) usually belong to this category; computer vision applications limited to be fully functional within the computational borders of this kind of platforms were considered unfeasible up to few years ago, but are now spreading and there is actually a strong research activity in this area. [25] tackles face

detection over PDAs, dealing with the problems of limited computational power and power consumption; [26] uses computer vision techniques over a smart phone for the visually impaired people, but it does not hinder the applicability to visual surveillance. Regarding the video analysis over the monolithic moving systems, the segmentation of moving objects is completely different from what described up to this point. Frame differencing becomes useless in most of the cases (background-foreground separation is still possible but it needs further processing: [27] is an example of foreground segmentation using KDE techniques), and detection-by-tracking instead of tracking-by-detection is often adopted [28]: after a selection of the region of interest, several probabilistic tracking methods can be used [11], such as Kalman and particle filters, mean-shift, etc. An approach based on contour tracking that fits monolithic mobile systems is described in [29]. [30] proposes a unified framework for handling tracking with occlusions, non-overlapping sensor gaps and moving sensors that switch fields of regard. All the augmented reality works based on the SLAM [31] represent the state of the art of what can be done on geometry recovery in monolithic moving systems.

Distributed, moving:
given the wide complexity of this last sub-portion of the mobile video surveillance systems, we divide it in several classes:

- *moving sources, fixed sinks*: robot-surveyor applications for assessment in disaster or extraordinary scenarios were studied even in the 1980s: for example [32] deals with remote recognition of nuclear power-plants with human-based remote optical analysis; these applications did not employ any computer vision algorithm because of the limited computational capabilities of the processors at that time. Conversely, in modern applications it is possible to depict three very different cases, depending on the position of the functional (and especially of the purposive) modules:

    - *the purposive module is on the source side*, that is moving and typically has limited computational power, physical size and electric power autonomy; on the opposite, the advantages are: process uncompressed video data, (possibly) stream over the network only meta-data, offer an architecture that is scalable under the processing point of view (the system can easily handle the increase of video sources, since each one has its own processing unit). This is the approach based on *smart cameras*: an interesting platform of smart cameras for video surveillance, equipped with grabbing, pre-processing, encoding and wireless radio mobile network card, is described in [33]; also [34] shows a similar example of smart cameras, mounted on autonomous robots. As we mentioned at the beginning of the section, VSAM [6] belongs to this category also: it streams over wireless network only "symbolic data extracted from video signals", that is, meta-data. Actually, VSAM is a very wide project and deploys source modules of the three kinds: fixed, mobile and moving (airborne).
    - *the purposive module is on the sink side*, that is fixed: the computation can typically rely on fixed and unconstrained processing platforms: systems with mobile sources that are based on very complex computer vision techniques, need to rely on this kind of set up. The tracking algorithms proposed in [35]

or in Chapter 8 are a couple of examples. The network links, used to transmit the whole video from the sources to functional modules can be a critical part: the (often limited) bandwidth must support video compression with a quality that is to be sufficient for accomplishing video processing; in some cases it is possible to adopt compression schemes to improve its efficacy (as in the case of airborne videos [36]). Anyway, the recent increase in bandwidth of wireless networks (WiMAX 802.16 is just an example) opens the way to new scenarios. Transmitting the whole video, and not just meta data, has the advantage that video itself can be used for multiple purposes; for instance, storage, human-based/human-assisted surveillance; these tasks cannot be performed when just meta data is streamed over the network. In case video applications do not need the streaming of the whole video (e.g. remote face recognition [37]), even limited bandwidths can suffice. In any case, the set-up with a central processing unit positioned at the sink side looses scalability.

  – *the purposive module is split*, partially over the source and the rest over the sinks. This can be a solution for some specific cases where it is possible to operate a clear division of functionalities inside the purposive module (e.g. low resolution processing at source side and spot-processing on high resolution image patches at sink side). [38] describes and analyzes the best techniques for optimized processing distribution and partition.

- *fixed sources, moving sinks*: the purposive module for this category is typically on the side of the source, therefore being able to benefit from the fixed site (unconstrained physical dimensions and power supply) and from the availability of uncompressed video to process; moreover the architecture remains scalable. The sinks are usually reduced to human assisted back-ends and often use web-services techniques [39,40]. Some works analyze strategies for the processing of fixed camera videos and for the tuning of video data transmission to mobile sinks in order to optimize the quality perceived by human operators [41,42].

- *sources and sinks are independently moving*: the wireless data communication will cross two radio bridges, suffering more latency and being more prone to network errors: in Chapter 5 we briefly evaluate the performance of video streaming under these conditions. This is the only scenario where it is reasonable to keep the functional module fixed (and therefore detached from both sources and sinks), in order to exploit all the architectural advantages of fixed modules. iMouse [43] depicts a generic architecture based on multi-sensory surveillance where the camera sources are mounted on a moving robot and the back-end sinks are moving smart phones; instead of moving robot, [44] depicts a mobile video server based on web-services for cellular phones; [45] describes an architecture for vehicle-to-vehicle video streaming based on WiFi, while [46] a PDA-to-PDA streaming based on GPRS.

Part I

Video Streaming Architecture for Mobile Video Surveillance

Chapter 3

Introduction

The video streaming for mobile video surveillance presents several technological challenges. On the one side, videos pose serious problems in terms of both amount of data transferred on the network and computational resources. On the other side, mobile devices and Ubiquitous Multimedia Access scenarios require accessibility through different and often limited wireless networks, either 802.11 WiFi, 3G networks such as HSPA (High Speed Packet Access) and UMTS (Universal Mobile Telecommunications Service), or even 2/2.5G networks such as GPRS (General Packet Radio Service) or EDGE-GPRS (Enhanced Data rates for GSM Evolution). These conflicting requirements - high data volumes and limited resources - emphasize the need for efficient codecs for both downloading and streaming applications. Given our main focus on video data, the goal is to allow UMA services to maintain a quality sufficient for both human and automatic surveillance.

Recalling the three-layer architecture depicted in Fig. 2.3(b), with source, functional and sink modules that are distributed and linked together via IP-based connectivity, the focus of this thesis part is kept on the video streaming process that takes place from source to the functional module (for automated video surveillance) or to the sink modules (for human-based surveillance or for data storage). As we mentioned in Section 2.3, this is not the only possible architecture for mobile video surveillance: the growth of smart cameras makes the shifting of some processing tasks on-board on the local encoder side more feasible and interesting. In general, part of the computer-based video surveillance algorithms could be implemented on the side of the source module, with the twofold advantage of reducing the transmission bandwidth and working with uncompressed images. However, our focus will be kept on a first layer that performs video encoding and streaming only, demanding any surveillance task to the further steps: this solution remains the most flexible since no specific assumptions are made on the applications implemented on the functional and sink modules. In our work the first layer is embodied by a standard PC architecture, being aware that standard video encoders can often rely on embedded hardware implementations to support real-time compression.

Despite the general architecture, we will focus on the most challenging case in which the video source is provided by a live camera and acquired on demand. There are many

Publications related to Part I: [a,c,d,f,k]; see the list of author's publications, page 147.

examples of interesting applications; in the case of mobile-to-fixed scenarios: a camera mounted on a police vehicle patrolling a city area, a robot equipped with a camera for monitoring disaster areas, a security camera used in a construction working site that moves over on daily basis; any of these might stream the live videos to a control center; vice versa the case of fixed-to-mobile could be embodied by a police officer viewing through his PDA the video collected by a camera installed inside a building (or by a camera mounted on a moving vehicle for the mobile-to-mobile case).

In Chapter 4 we propose a streaming system called Moses (MObile Streaming for vidEo Surveillance), that supports video streaming in different conditions, aiming at low-latency transmission over limited-bandwidth networks. Additionally, the video stream is provided with a sufficient quality to be correctly analyzed by both human-based or computer-based video surveillance layers. To this aim we propose a suitable optimization of the streaming process with an adaptive control of the streaming parameters. Moses is made of two applications: the first is called Movie (MObile VIdeo Encoder) devoted to live video grabbing, compression and streaming, the second called Movide (MObile VIdeo DEcoder), devoted to video down-streaming, decompression, and playback.

Moses is built upon open-source software components. The reason is twofold. First, the availability of the complete source code permits modifications and optimization at any level. Second, most of open-source software or components can be cross-compiled on different architectures and/or operating systems with just specific adjustments: cross-architecture software is required since Movide is meant to work not only on PCs but also on PDAs.

Evaluating real-time video streaming is neither simple nor clearly defined. In Chapter 5, we propose a new methodology with an effective image analysis step to provide comparative performance evaluation over four key parameters, namely latency, image quality, video fluidity and frame losses. Eventually, we compare results achieved with Moses and other streaming systems over such parameters.

The rest of the thesis part is structured as follows. In the next section we define the system requirements for effective human-based and computer-based video surveillance. Then, in Sections 3.2 and 3.3, we review some commercial and scientific approaches to video streaming. Chapter 4 presents the full details of Moses, reporting on the video encoder layer Movie (Section 4.1), on the video decoding layer Movide for PC platforms (Section 4.2) and for PDA platforms (Section 4.3). Performance evaluation (Chapter 5), contains five sections: Section 5.1 describes the proposed methodology, Section 5.2 provides a detailed description of the test bed and the operational conditions used during the tests, and Sections 5.3, 5.4 and 5.5 for experimental results.

3.1 System Requirements

Mobile video surveillance calls for live video streaming in order to dispatch an on-line view of the controlled scene. Beyond this basic feature, there are other requirements (listed in Tab. 3.1) that need to be considered in order to define a successful mobile video surveillance system.

Given the requirement #1 the video coding must be sufficiently adaptable to different wireless network supports and not only to the ones with large bandwidth (such as WiFi or UMTS/HSPA), that do not still offer ubiquitous coverage. In this work GPRS/EDGE-GPRS network has been selected as reference mobile data service, since

#	Requirement	Video Surveillance	
		Human-based	**Computer-based**
1	**Ubiquitous accessibility**	$\surd$	$\surd$
2	**Low latency**	$\surd$	$\surd$
3	**High image quality**	preferable	$\surd$
4	**Video fluidity**	preferable	-
5	**No frame skipping / loss**	-	$\surd$
6	**High frame rate**	-	preferable

Table 3.1: Summary of requirements for streaming in mobile video surveillance systems. '$\surd$' and '-' indicate 'required' and 'not required', respectively.

it provides wide coverage over the European territory. This is merely an implementation choice that does not compromise the generality of the system which can provide video streaming over any IP-based network. Given the effective bandwidth available for EDGE-GPRS and GPRS connections, Moses will be tested on 80 and 20 Kbps, respectively. Moreover, in mobile video surveillance a multiple delivery of video sources could be requested, therefore tests on 20 and 5 Kbps will be performed to simulate a four cameras video delivery. The requirement #2 of *low latency* is necessary because surveillance systems should exhibit high reactivity to changes in the scene. Moreover, mobile video surveillance needs *high image quality* (requirement #3) and *good video fluidity* (requirement #4). Both are preferable for a satisfactory human-based surveillance and the first is mandatory in computer-based surveillance to allow a correct video analysis and scene recognition. This process is very sensitive to noise and visual conditions and it is, thus, greatly influenced by the coding artifacts or the quantization introduced by the video compression: Chapter 7 is completely devoted to this topic and it will be clear at that point that #3 is a crucial parameter. The tracking algorithms are often based on object-to-track association on frame basis and they usually assume a fixed frame rate for effective status predictions. Therefore the requirement #5 (*no frame skipping*) becomes necessary. Finally, the search area for a tracked object in a new frame is generally proportional to the displacement that the object might have between two consecutive frames: thus requirement #6 (*high frame rate*) prevents an excessively-enlarged search area.

Part of the listed requirements are necessary for the majority of UMA services but the last-generation commercial streaming systems typically fulfill them just in part. Our system Moses, on the other hand, is specifically designed to fulfill them all. It is based on H.264/AVC (MPEG-4 part 10) [47–49], suitably devoted to work on low-capacity networks by means of several improvements to make the streaming adaptive. H.264/AVC guarantees a better trade-off between image quality and bandwidth occupation with respect to MPEG-2 and MPEG-4 part 2 [49].

3.2 Video Streaming Solutions and Components

Several video streaming solutions handle all the steps of the process, namely video grabbing, encoding, network streaming and video playback. Two examples of popular off-the-shelf suites are Microsoft Windows Media® [1] and Real Networks® [2], based on Helix technology [3]. The encoding layers of such systems are intended to provide streaming server capabilities, i.e. to handle intensive video broadcasts. They require solid processing platforms and/or server-oriented operating systems. For example, Windows Media Streaming Server runs on Windows Server OS only. The main goal of these proprietary solutions is to provide massive access to both live and stored video resources for entertainment purposes, rather than ubiquitous, uni-cast video streaming as required for surveillance purposes. For this reason their latency is usually rather high (as shown in Chapter 5), being in conflict with requirement #2. Moreover, since the typical users are non-technical practitioners, their settings are often pretty limited mainly in terms of video coding and network streaming. Regardless of these considerations, their performance in mobile video surveillance conditions are not negligible. We measured the overall streaming performances of both Windows Media and Real Networks, since they both provide a video player for PC and PDA. Numerical results will be discussed in Chapter 5.

Skype® [4] probably represents the most popular freeware tool for live multimedia streaming, addressing audio and video. The overall performance of this system is very interesting, though it has some limitations for mobile video surveillance. In particular, the settings flexibility is even coarser than the aforementioned tools since almost everything is automatically handled: for instance, grabbing source, frame size and rate and video bitrate are not adjustable. This approach makes the system very easy to be used for audio/video calls but also very rigid and certainly not flexible enough to be used for our goals. Moreover, according to the methodologies that will be presented in Section 5.1, analytic latency measurement becomes unfeasible since the software cannot stream video files. Skype is meant for audio streaming, that cannot be disabled in favor of video, producing a bandwidth waste that becomes a critical issue on radio mobile connections. Finally, the video player is currently implemented in the PC version only.

On the side of open-source software, VideoLan Client (VLC) [5] is probably the most renowned available tool. Differently from all previous examples, VLC is designed for research or free use and not for commercial purposes; it provides a very flexible and refined setup, that reaches the lowest level of details for video grabbing, coding and network streaming. It supports many video compression standards (including H.264) and streaming technologies. However, the system shows strong limitations that conflict with many of the requirements of our project: video latency (requirement #2) can be kept pretty low only at the cost of strong video quality degradation.; regarding bandwidth usage, the H.264 video bitrate control is neither very strict nor optimized; H.264 streaming is allowed only on MPEG TS (Transport Stream) encapsulation and this is demonstrated to be a drawback when working with low bandwidths [50], since the stack MPEG-TS/UDP/IP introduces more than double the overhead of the stack RTP/UDP/IP. Eventually, the VLC parameter control of the H.264 video coding gives access to the full palette of parameters but it is not precise in handling them correctly: artifacts and strong image quality degradation are introduced as soon as the setup

[1] URL: www.microsoft.com/windows/windowsmedia. Every URL provided in this section was last accessed on Jan 20th, 2010

[2] URL: www.realnetworks.com

[3] URL: https://helixcommunity.org

[4] URL: www.skype.com

[5] URL: www.videolan.org

deviates from the standard conditions in order to be optimized for low bandwidths.

An alternative solution is to design and develop an optimized system that specifically targets mobile video surveillance, using existing components where suitable.

The most complex blocks in the video streaming pipeline are video encoding and decoding. Given the choice for H.264 codec, we compared existing encoding engines according to their computational performances and parameters flexibility. Performance is required since the encoding must be performed in real time and conversely H.264 can be computationally very demanding if configured on high encoding profiles; flexibility is needed to tune the encoder for addressing different needs, such as high encoding speed or high image quality.

JM [6] is the H.264 reference software: it is open source, completely flexible and modifiable, but has very limited computational performance. Intel Integrated Performance Primitives (Intel IPP) [7] is a suite of libraries that offers broad digital signal processing algorithms, including also video coding and specifically H.264. Such libraries are optimized but not open source and can be executed on Intel processors only. X.264 [8] represents today one of the (open source) H.264 encoders with best performance and highest completeness over the H.264 standard and for these reasons it was chosen as the software reference for our video encoder MOVIE.

Regarding the choice for H.264 decoder, the performance issue becomes very restrictive since we want to address not only standard x86 architectures, but also CPUs with limited computational power, like StrongArm architectures of PDAs. The same considerations mentioned on the JM and Intel IPP H.264 encoders are valid for their decoder tools. From the wide variety of other decoders, many being open source, the choice is limited if we consider only those that can be compiled on both PC and PDA. Our selected decoder for MOVIDE is based on FFMPEG [9], because of performance and ability to handle most of the H.264 coding profiles; it does not provide a ready-to-use PDA version, but since it is open source it can be appropriately modified for that. This library has the additional advantage to offer a network down-streaming layer implemented for many different protocols.

3.3 Scientific Contributions to Streaming for Mobile Video Surveillance

Video streaming has reached its peak of interest in the scientific community in the last ten years. A survey paper on this topic [51] reports and discusses the relevant signal processing issues and proposes possible solutions. Regarding video streaming, researchers address the problem from several different perspectives. For example, some of them tackle the optimal allocation of resources (for instance computational resources of video servers [52], or power resources to optimize consumption and video quality [53]). Another set of papers focuses on the system architecture, in terms of both communication model (as in [54], where the analysis is based on Real Networks products, but without considering low-capacity networks) and data synchronization (as in the case of [45] where an inter-vehicle communication for live video streaming is considered,

[6]URL: iphome.hhi.de/suehring/tml
[7]software.intel.com/en-us/intel-ipp
[8]URL: www.videolan.org/developers/x264.html
[9]URL: ffmpeg.org

even though based on 802.11 WiFi networks and thus not suitable for generic mobile video surveillance). Some works propose systems for video streaming over low-capacity networks, such as GPRS. For instance, Lim *et al.*in [46] introduces a live video streaming system on GPRS network, based on MPEG-4 compression and containing several computational optimizations for working on PDAs. It can achieve a frame rate of 21 fps at the encoder side and 29 fps at the decoder side for transmitting a QCIF (176x144) video at 128 Kbps. However, their system drops to 2-3 fps when transmission is over GPRS. Moreover, no information on the latency of the system is provided. The work in [55] tackles the problem of transmitting real-time videos over GPRS by using frame skipping. Chuang *et al.*in [56] deals with adaptive playback for video streaming over simulated 3G networks: a statistical model on both departure and arrival processes is built in order to avoid buffer underflows and preserve playback smoothness. Even if the paper does not deal clearly with latency measurements and supposes an additional payload for timing data exchange, the idea of adapting video playback to optimize the buffer management will be used also in our work. Eventually, many works tackle video surveillance over wireless networks but most of them do not address low-capacity networks. [57] analyzes some key issues of mobile video surveillance, such as networking requirements and digital image transmission; moreover, similarly to what we propose in Chapter 7, the author evaluated the effect of error-prone transmission and coding artifacts on the final video surveillance applications (specifically, intruder alarming and object recognition). However, mainly due to the immaturity of the technology in 1999, no effective proposal to video streaming for mobile video surveillance over low-capacity network is really proposed in the paper. Agrafiotis *et al.* [58] described the video transmission aspects of the European project WCAM (Wireless Cameras and Audio-Visual Seamless Networking) and made an excellent work in evaluating the performance (in terms of jitter buffer sizes, packet error rates, etc.) of H.264 using TCP/IP or UDP/IP stacks. However, this work is based on 802.11b/g networks only. Cai *et al.* [59] reviewed video coding techniques for wireless networks, but also in this case the test-bed demonstration system is based on a 802.11b. One interesting example of video transmission over low-capacity networks and with the specification of low latency is reported in [60], but this work is focused on video streaming and not on successive video processing. Lam *et al.*presented a very interesting work [61] with a final objective similar to ours. However, in their case frame skipping is functionally employed to fit in the low-bandwidth requirement with an intelligent filtering of frames – bandwidths close to 5 Kbps are sustainable only through a very aggressive frame skipping. As mentioned above, this could complicate the tracking task and, moreover, requires to move part of the computational load on the local side of the camera.

Chapter 4

System Proposal: MOSES

The following Section 4.1 describes MOVIE, the video encoder layer (developed for PC-based hardware only). The decoding layer MOVIDE has been designed in two different versions, depending on the computational resources of the client, namely PC-based (Section 4.2) and PDA-based (Section 4.3).

4.1 MOVIE: the Video Encoder

The typical encoder layers of streaming systems are made of three basic blocks, namely video grabbing, encoding and network streaming, plus further controlling steps. Our encoder layer aims to provide high flexibility in the control of video source and compression and to keep the latency and the frame-loss rate at lowest levels. The following peculiar aspects of the architecture (depicted in Fig. 4.1) were specifically designed to attain such objectives:

- *multi-threaded processing and pipeline*: video grabbing, encoding and network streaming are handled by dedicated threads decoupled through circular buffers. Having asynchronous threads optimizes the processing since the execution of each one is basically independent from the others; this allows the implementation of a pipeline that reduces latency compared to plain serial processing. The original X.264 source code was modified for this purpose;

- *low buffer occupancy*: buffering is necessary to avoid video data losses due to thread decoupling; as drawback, it introduces some latency and for this reason the application is tuned to keep buffers at the lowest occupancy. The best way to achieve this is to set the grabbing frame rate equal to (or slightly lower than) the average encoding frame rate; the buffering between encoder and network streamer is not crucial since the second task, being light weighted, manages to keep the buffer always close to zero occupancy;

- *UDP streaming*: raw H.264 encoded stream is seamlessly forwarded to the network streamer that packetizes it into UDP datagrams of fixed byte size. UDP is preferable with respect to TCP due to its prioritized dispatch over the network in case of congestion, but this comes at the cost of possible datagram loss. Nevertheless, thanks also to the Automatic Repeat-reQuest (ARQ) mechanism implemented on

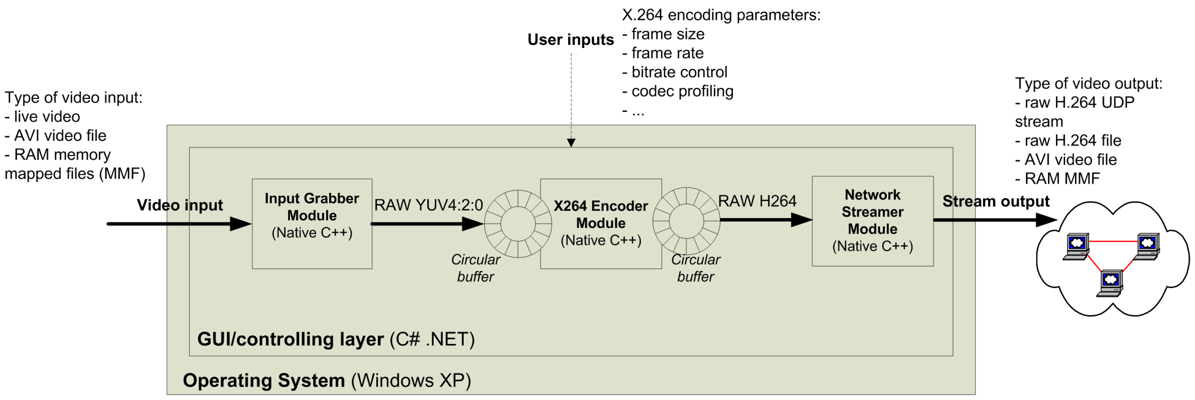


Figure 4.1: Functional scheme of MOVIE. For memory mapped files see Appendix B.

the Radio Link Control (RLC) [62], our experiments show that the transmission over EDGE-GPRS or even GPRS is very robust since the rate of lost datagrams is extremely low and none is received out of order. For this reason, and for the fact that MOSES aims to deliver only video streams with no additional audio or text to be synchronized with, we decided to use raw UDP instead of RTP/UDP.

As shown in Fig. 4.1, the development was partly based on C#/.NET Framework and partly on native C/C++ modules. In particular, the use of the .NET Framework is devoted only to non-intensive tasks (GUI and controlling layer).

4.2 MOVIDE: the Video Decoder for PC platforms

In case of PC-based client, the decoder layer has the functional scheme reported in Fig. 4.2. The computational demand of H.264 for decoding is definitely lower than encoding, therefore the most critical issue for this PC-based layer is not really the optimization of computation rather the efficient network buffering and the data flow management; in fact, even if video grabbing frame rate (encoder side) and playback frame rate (decoder side) are set to the same values, the datagram generation rate (encoder side) and the datagram extraction rate (decoder side) might differ from time to time for several reasons, such as wireless network instability, varying CPU load (either on encoder or decoder side) due to operating system tasks, video coding (changing video scene complexity), and so on. Specifically the following procedures were adopted to minimize latency and obtain the best of the video quality from the network stream:

- *dynamic buffer sizing*: if the datagram generation rate remains higher than the datagram extraction rate for a sufficiently long time, the buffer might fill up and every datagram received afterward would be lost (*buffer overflow*). We propose a simple algorithm that dynamically adapts the buffer size: it either doubles the buffer size when this gets filled up beyond a value $\chi\%$, or halves it when its level decreases below $(100-\chi)\,\%$. χ is computed empirically and depends on the network conditions, the buffer initial size and the video bitrate;

- *adaptive playback frame rate*: even if the latency time is, for obvious reasons, directly related to the occupancy of the network buffer, tuning the system to keep

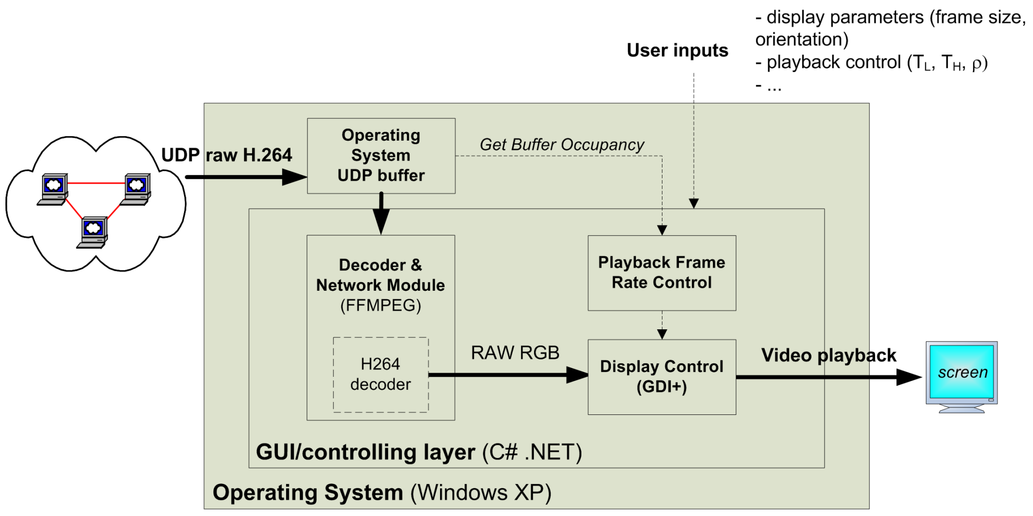

User inputs
- display parameters (frame size, orientation)
- playback control (T_L, T_H, ρ)
- ...
UDP raw H.264
Operating System UDP buffer
Get Buffer Occupancy
Decoder & Network Module (FFMPEG)
H264 decoder
RAW RGB
Playback Frame Rate Control
Display Control (GDI+)
Video playback
screen
GUI/controlling layer (C# .NET)
Operating System (Windows XP)

Figure 4.2: Functional scheme of MOVIDE in the PC version.

it as empty as possible would result in an uneven trend of the playback frame rate, due to *buffer underflow*. Therefore, to achieve the best trade-off between low latency and good fluidity, the block for playback frame rate control (see Fig. 4.2) implements a set of rules to keep the buffer occupancy between two values T_L and T_H (typical values are $T_L = 5\%$ and $T_H = 15\%$ of the buffer size). In general, the playback frame rate $FR^t_{playback}$ at time t is function of two values: the buffer occupancy B^t_{occ} (that needs to be kept between T_L and T_H) and the discrete derivative of the buffer occupancy ΔB^t_{occ}. The adaptive control can be summarized as follows:

$$FR^t_{playback} = \begin{cases} FR^{t-1}_{playback} \cdot (1+\rho) & \text{if } \Delta B^t_{occ} > W \\ FR^{t-1}_{playback} \cdot (1-\rho) & \text{if } \Delta B^t_{occ} < -W \\ FR^{t-1}_{playback} & \text{if } \left(B^t_{occ}, \Delta B^t_{occ}\right) \text{ optimal} \\ FR^{t-1}_{playback} \cdot (1+\frac{\rho}{W}\Delta B^t_{occ}) & \text{otherwise} \end{cases} \tag{4.1}$$

where W defines a window of ΔB^t_{occ} (typical value is a few thousandths), which represents the limits for a sustainable variation of the buffer occupancy; if the $\left|\Delta B^t_{occ}\right|$ consistently exceeds W, the system will end up in buffer overflow or underflow in a short time. ρ is the reactivity of the adaptive control ($0 < \rho < 1$ but typical value is approximately a few hundredths). The closer ρ gets to 0, the weaker the adaptive control is; eventually the control would be disabled with $\rho = 0$. The reactivity increases together with ρ, eventually ending up in an unstable system. The pair of values $\left(B^t_{occ}, \Delta B^t_{occ}\right)$ is considered optimal in the following cases:

$$\left(B^t_{occ}, \Delta B^t_{occ}\right) \text{ optimal if:} \begin{cases} T_L < B^t_{occ} < T_H & \wedge \quad \Delta B^t_{occ} \approx 0 \\ B^t_{occ} > T_H & \wedge \quad -W \leq \Delta B^t_{occ} \leq 0 \\ B^t_{occ} < T_L & \wedge \quad 0 \leq \Delta B^t_{occ} \leq W \end{cases} \tag{4.2}$$

In other words, the control reacts increasing (decreasing) the playback frame rate when the buffer occupancy increases (decreases) too rapidly ($\left|\Delta B^t_{occ}\right| > W$).

When the conditions are optimal the playback frame rate is kept constant. In all the other cases, the frame rate is slightly adjusted with a factor proportional to the slope of variation of the buffer occupancy;

- *decoder-display coupling*: in the absence of synchronization between the decoder and display threads, it may happen that a frame is correctly decompressed but not displayed, therefore lost, because it is overtaken by a decoded frame which follows (*frame overwriting* effect). Buffering the frames flow would solve the problem but it would also introduce some latency. A different approach, that completely avoids buffering, is to introduce a simple synchronization decoder-display that, just before a frame gets overwritten, delays the decoder until the frame is effectively displayed. Maximum delay is set to:

$$\frac{1}{\left(2 \cdot FR^{t}_{playback}\right)}$$

  As shown in Chapter 5, this solution massively reduces frame losses and, even better, has a positive effect on the latency.

4.3 MOVIDE: the Video Decoder for PDA platforms

Fig. 4.3 shows the scheme of the PDA decoder. Differently from the PC version, the successful implementation of the PDA-based decoder requires peculiar optimizations, given the limited computational power of these devices. Specifically, the most critical issues are, together with video decoding and network buffering, video data exchange between processes and video display. The most suitable operating system to rely on in such tight conditions would be Linux for embedded systems, given the important advantage of being open source, thus enabling low-level control on memory, network and management of processes and services. Unfortunately, the limited support for most of the devices and peripherals (such as GSM/GPRS modems) prevented us from adopting this solution, opening the way to Windows CE. Each module and the most important functionalities designed for a successful PDA-based decoder follow:

- *optimized display control*: in the case of PDA-based solution, writing video data directly on the graphic card memory is computationally very convenient rather than relying on the standard functions of the operating system. For this reason the display control is based on GAPI (Game API)[1] and Direct Draw[2], instead of GDI+[3] functions. For a further speed up of this control, we made use of pre-calculated look-up tables for image rescaling and 90-degrees flipping, used to fit the desired playback frame size and orientation;

- *inter-process communication*: the decoding and the GUI modules are kept completely detached in their scheduling (i.e. they run not simply on two threads, but on two separated processes), so that each processing flow will not interfere with the other (for example, when the GUI module calls the garbage collector). Since

[1] URL: msdn.microsoft.com/en-us/library/ms879884.aspx
[2] URL: msdn.microsoft.com/en-us/library/ms879875.aspx
[3] URL: msdn.microsoft.com/en-us/library/ms533798(VS.85).aspx

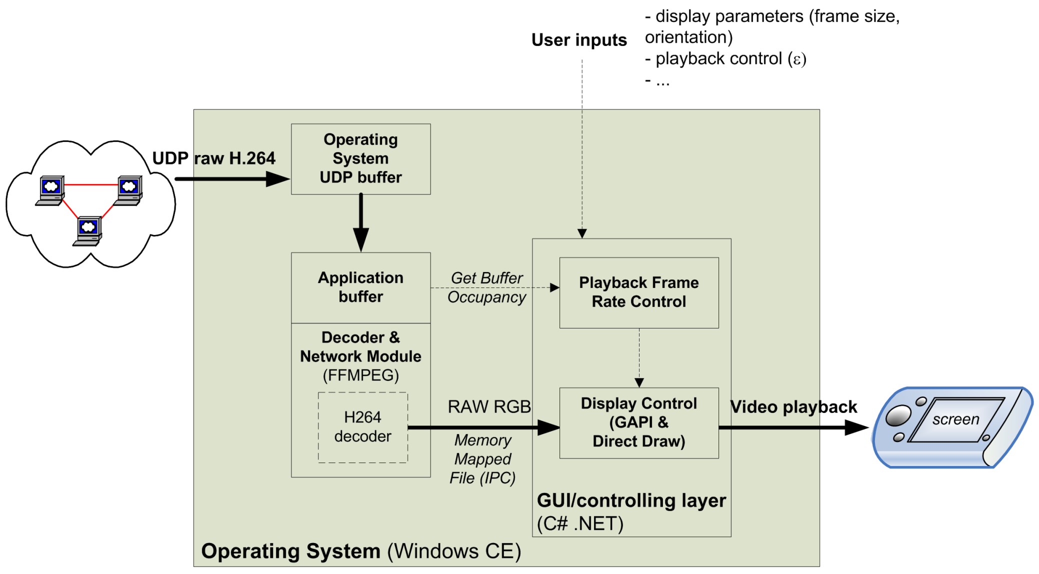

User inputs
- display parameters (frame size, orientation)
- playback control (ε)
- ...
UDP raw H.264
Operating System UDP buffer
Application buffer
Get Buffer Occupancy
Playback Frame Rate Control
Decoder & Network Module (FFMPEG)
H264 decoder
RAW RGB
Memory Mapped File (IPC)
Display Control (GAPI & Direct Draw)
Video playback
screen
GUI/controlling layer (C# .NET)
Operating System (Windows CE)

Figure 4.3: Functional scheme of MOVIDE in the PDA version.

a modest QQVGA (160x120) video, 24 bits RGB colors at 10 fps, would generate a 4.6 Mbps bandwidth communication, it is obvious that the Inter-Process Communication (IPC) must be extremely efficient to avoid frame skipping and preserve video fluidity. Data exchange through either local file system or local UDP loop-back is not feasible since both these methods cannot sustain such a high data transfer rate without compromising the overall performance of the system. Moreover, since file systems are actually based on flash memories which have a finite number of erase-write cycles, IPC based on file system would deteriorate the support in a short time. The most efficient IPC method to be used is then memory-mapped files (MMFs), i.e. a virtual file on RAM memory. For details see Appendix B. This approach allows the achievement of performance comparable to shared memory between threads;

- *adaptive playback frame rate*: the implementation of the adaptive control described in the PC-based decoder needs to be revisited to be successful on a PDA. Since Windows CE does not allow querying of the occupancy of the UDP buffer, an application buffer on top of the UDP operating system buffer must be added (see Fig. 4.3). In addition, the algorithm presented in Section 4.2 through Eq. 4.1 and 4.2, cannot be successfully implemented on a PDA, because the computational resources do not sustain the required frame rate in case it needs to be firmly increased (usually when $\Delta B^t_{occ} > W$). For this reason the PDA decoder layer employs a light-weighted control based on the key task of keeping the UDP buffer occupancy as low as possible (but greater than zero) in order to minimize the playback interruptions due to buffer underflows. Given a value ϵ that is a few thousandths above 1, the control acts as follows:

$$FR^t_{playback} = \begin{cases} FR^{t-1}_{playback}/\epsilon & \text{if } \Delta B^t_{occ} = 0 \\ FR^{t-1}_{playback} \cdot \epsilon^2 & \text{if } \Delta B^t_{occ} > 0 \end{cases} \quad (4.3)$$

Being ϵ slightly above one, the reaction in the case of buffer occupancy greater than 0 is stronger than in the other case. Analytical results will be given in Section 5.4, where $\epsilon = 1.002$ was used. This control does not claim to be an optimal algorithm for playback frame rate adaptation (for a thorough analysis of adaptive playback, refer to [56]) but our goal is to verify that even on reduced-power processors a simple adaptive control can significantly increase the fluidity of the video playback without the need of further buffering.

Regarding the *dynamic buffer sizing* and *decoder-display coupling*, the PDA decoder implements the same procedures described for the PC decoder.

Chapter 5

Performance Evaluation

5.1 A Methodology for Video Streaming Assessment

Evaluating the performance of a live video streaming system is not an easy task, since it is mostly based on the perceived quality of live (not stored) videos by means of the human receiver. In accordance with our requirements, we propose a methodology to extract quantitative measures on *image quality*, *video latency*, *frame loss* and *video fluidity*. Specifically:

1) *Image quality*: it can be easily measured with PSNR (Peak Signal-to-Noise Ratio) that gives an idea of the distortion introduced by the coding and transmission process. Even though this is not completely corresponding to the way our human visual system evaluates the quality, it is an easy and well-known method to measure image quality [63].

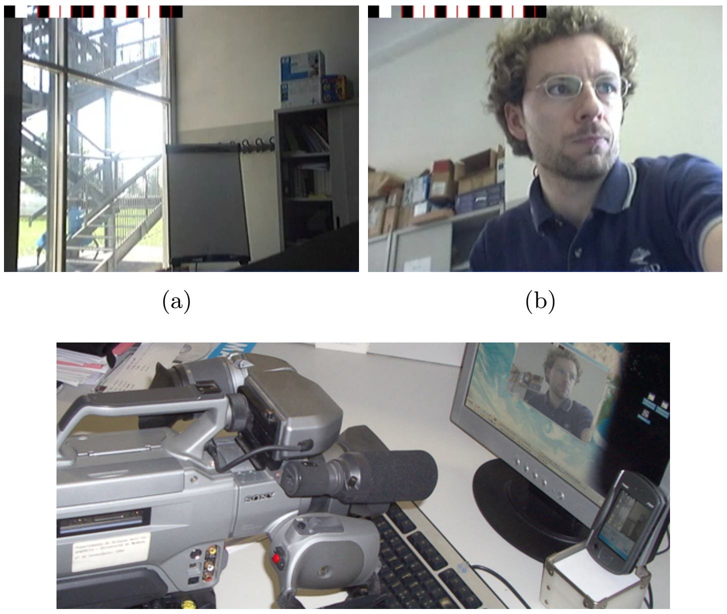
2) *Video latency*: there are not commonly accepted methods for measuring the *latency* in an analytical way. An approximate measurement can be obtained by synchronizing the encoding and the decoding unit on the same time server, and modifying the encoder so that it dispatches, together with the encoded frames, also the time stamp of their grabbing. Then, after having retrieved the time stamp of display of a decompressed frame, the decoding unit deducts the latency by time differencing [64]. This procedure has several drawbacks: on one side the synchronization with the time server should be frequent, in order to have precise time gap measurements and this would be a waste of network bandwidth; moreover embedding a time stamp for each frame would result in further bandwidth waste. In addition this kind of measurement requires modifications to core functions of the video grabbing, networking and displaying, therefore it is only feasible on open source code and cannot be employed on closed systems such as Windows Media and Real Networks, that we want to compare with.

Consequently, an alternative way to measure the latency must be used. Schmidt in [65] presents an interesting approach to measure the synchronization of synthetic multimedia data (video, audio and text) through external observation of media players using several sensors: we modified and extended the approach for real video data. The frame number is superimposed on each frame of a recorded video. We then play the video both on the encoding unit, and, after having passed it through compression and streaming, also on the decoding unit.

Let us call $\bar{t}$ a given time instant, $FN_{enc}\left(\bar{t}\right)$ and $FN_{dec}\left(\bar{t}\right)$ the frame number shown on the encoder and decoder respectively. Let us define $\bar{t}^*$ as the time such that $FN_{enc}\left(\bar{t}^*\right) = FN_{dec}\left(\bar{t}\right)$, i.e. the time such that the same frame number is visible on both sides. It holds that $\bar{t}^* < \bar{t}$. Let us then call Δt_{enc} the time gap between two grabbed frames on the encoder, which is constant since the grabbing frame rate is constant; in such conditions a generic time t can be approximated with $t = \Delta t_{enc} \cdot FN_{enc}\left(t\right)$ and exploiting the definition of latency as $L\left(\bar{t}\right) \equiv \bar{t} - \bar{t}^*$, we can write it as follows:

$$\begin{aligned} L\left(\bar{t}\right) \equiv \bar{t} - \bar{t}^* &= \Delta t_{enc} \cdot \left(FN_{enc}\left(\bar{t}\right) - FN_{enc}\left(\bar{t}^*\right)\right) \\ &= \Delta t_{enc} \cdot \left(FN_{enc}\left(\bar{t}\right) - FN_{dec}\left(\bar{t}\right)\right) \end{aligned} \tag{5.1}$$

This procedure needs to embed frame numbers directly on video frames and it could be made automatic with a tool for recognizing the numbers on the images: using plain numbers can be problematic due to the distortion introduced by strong image compression, which could make OCR task unreliable. For this reason, we prefer to adopt a code-based number representation. The binary coded frame number is superimposed on a small portion of each image. More specifically, blocks of white and black color were used to code 1s and 0s, respectively. Fig. 5.1 shows some snapshots of the methodology. The leftmost-upper two blocks are static and are used for calibration purposes. A video streaming session gets started and the video is played on the screens of both sides (encoder and decoder), which are physically placed one close by the other. At the same time, an external high-frame-rate camera acquires and stores a video of the evolution


(a) (b)

(c)

Figure 5.1: Examples of the experimental methodology used to measure latency. (a) and (b) are two frames of the same video, respectively frame number 1702 and 1718. During a streaming session (c) it happens that, due to latency, the encoder side is showing frame (b) while the decoder the frame (a).

of the streaming process on both screen (Fig. 5.1(c)). The resulting video is processed with simple image processing algorithms to automatically compute FN_{enc} and FN_{dec} by recognizing black and white blocks and deduct latency using Eq. 5.1. This tool is very flexible since it can measure the latency also on non open-source systems such as Windows Media and Real Networks.

3) *Frame loss*: the following equation quantifies the *lost frames* $LF\left(\bar{t}\right)$ exploiting the same frame number coding methodology: given

$$\Delta FN_{dec}(j) = FN_{dec}(j) - FN_{dec}(j - \Delta t_{acq}) \tag{5.2}$$

where Δt_{acq} is the discrete sampling period of the external camera and j is a generic frame of its recorded video, then

$$LF\left(\bar{t}\right) = \sum_{j=0}^{\bar{t}} \varphi(j) \tag{5.3}$$

where

$$\varphi(j) = \begin{cases} 0 & \text{if } \Delta FN_{dec}(j) \leq 1 \\ \Delta FN_{dec}(j) - 1 & \text{otherwise} \end{cases} \tag{5.4}$$

In other words, given that the encoding frame rate is constant and that the frame rate of the external camera is much higher than the encoding and decoding frame rates, if two successive frames grabbed with the external camera show the same frame number or two successive frame numbers on the decoder screen, it means that no frames were lost. Frame losses can be due to network datagram losses (this event would most likely produce a loss of several consecutive frames, since a datagram usually contains several frames when compression rates are high and frame size are limited), compression skipping or decoder frame overwriting.

4) *Video fluidity*: it can be measured as the trend of the Δt_{dec}, the time gap between two frames played by the decoder. This information can be computed again exploiting the frame number coding. In the best case, Δt_{dec} is constant and equal to Δt_{enc}; conversely the more the Δt_{dec} is scattered, the more the video playback looses in fluidity.

5.2 Test Bed and Operational Conditions

The tests for video streaming assessment are designed to verify the degree of fulfillment for the requirements listed in Tab. 3.1. The results are discussed on the basis of two different platforms: the former is PC-to-PC, the latter is PC-to-PDA.

For the PC-to-PC platform, we prepared a mobile-to-fixed system: the mobile site was mounted on a car (camera-car setup) and equipped with a standard x86 laptop (Intel Pentium Centrino 1.7Ghz), connected to either a USB Camera (Logitech Quickcam Pro 4000) or an IP camera (Axis 2420 plus infra-red illuminator for night vision), and an EDGE-GPRS modem used for uplink video transmission.

For the PC-to-PDA platform, we tested a fixed-to-mobile scenario; some tests have been performed also in a mobile-to-mobile scenario (from our camera-car setup to the

PDA). Two different PDAs have been used for the tests (i-Mate JasJar, WM 5.0, CPU: Intel Bulverde, 520Mhz; i-Mate PDA2k, WM 2003, CPU: Intel PXA263, 400Mhz); the results did not show relevant differences by using different PDAs. In this case the radio mobile connectivity was always GPRS-based.

In both hardware configurations, the following operational conditions were used:

- *video encoding*: the H.264 codec has been tested in two different profiles: a *baseline* (to achieve low latency at the cost of low quality) and a *high* profile (for best video quality at the cost of higher latency). The high profile contains several enhancements including the use of CABAC [47] encoding, wider reference window for B frames, deeper analysis for macro-block motion estimation, finer sub-pixel motion estimation and better rate distortion algorithms;

- *video sources*: the design and development of the system was always tailored for live video sources, but the performance measurements were gathered using two stored videos (VLAB, VCAR) in order to replicate the experiments on the same data. Tab. 5.1 shows the main properties of the test videos. Specifically, the video VLAB contains scenes with three different types of motion: reduced (moving people but static camera), medium (freely moving camera) and extreme (shaking camera); the video VCAR is taken from the camera-car setup while driving in a busy urban area.

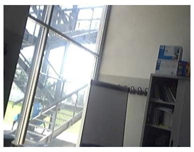
	VLAB	**VCAR**
		
Scenario	Indoor (webcam inside laboratory)	Outdoor (camera-car in urban traffic)
Frame Rate	10 fps	10 fps
Resolution	QVGA (PC-to-PC) QQVGA (PC-to-PDA)	QVGA (PC-to-PC) QQVGA (PC-to-PDA)
Length	≈ 120 s (≈ 1200 frames)	≈ 600 s (≈ 6000 frames)

Table 5.1: Recorded videos used for video streaming assessment.

- *network and bitrates*: when relying on EDGE-GPRS, video bitrate was set to 80 and 20 Kbps. A few tests were made saturating the effective EDGE-GPRS bandwidth, at 120 Kbps. When using GPRS, video bitrate was set to 20 and 5 Kbps, and a few tests were made with 10 Kbps. As mentioned in the introduction, 20 Kbps on EDGE-GPRS and 5 Kbps on GPRS are meant to generate four simultaneous and independent (not spatially multiplexed) video streams. We have experimented the wireless network transmission in several conditions: half tests were performed with the encoder as mobile site, the other half being the decoder side. Half the cases inside a building and in the other half on our camera-car setup (we drove for more than 80 km, at urban and freeway speeds, between 50 km/h and 110 km/h). We measured the network transmission on 20000 UDP datagrams, for more than 140 minutes of video streaming over GPRS at 5 and 20 Kbps and over EDGE-GPRS at 80 Kbps. In these conditions, thanks to the ARQ implemented in the RLC, a very reduced percentage of datagrams was lost (0.54%) and none was received out of order. Given this high degree of reliability of GPRS and EDGE-GPRS networks, in the next sections we will give evidence of frame losses due to compression skipping and to decoder frame overwriting only, which are definitely predominant on the effects due to network failures.

5.3 Experimental Results in PC-to-PC Scenario

	Tool	**Profile**	**Bit-rate (Kbps)**	**Dec Dsp Cpl**	**Playback Frame Rate**	**Avg Latency (sec)**	**Std Dev (sec)**
1	MOSES	baseline	80	no	fastest	1.26	0.28
2	MOSES	baseline	80	yes	fastest	1.21	0.28
3	MOSES	baseline	80	yes	adaptive	1.55	0.27
4	MOSES	high prof	80	yes	fastest	1.65	0.38
5	MOSES	baseline	20	yes	fastest	1.41	0.33
6	Windows Media		80			4.76	0.36
7	Real Media		80			3.15	0.07
8	VLC	baseline	80			2.21	0.27

Table 5.2: Latency of Windows Media, Real Media, VLC and MOSES in PC-to-PC scenario over VLAB at QVGA resolution. DecDspCpl stands for Decoder-Display Coupling.

Latency measures in PC-to-PC scenario are summarized in Tab. 5.2. MOSES is configured in five different ways and compared against Windows Media, Real Media and VLC. The latency introduced by MOSES is the lowest, whatever configuration is used. With respect to the base configuration (row #1 of Tab. 5.2), the introduction of the decoder-display coupling, for frame-overwriting reduction, decreases the average latency from 1.26 s to 1.21 s (row #2). Instead the use of the adaptive frame rate control with $T_L = 5\%$, $T_H = 15\%$, W=0.005 and $\rho = 0.04$ (row #3) has the cost of a slight increase in latency (1.55 s) but greatly improves the video fluidity, as described further in this section. The introduction of a high complexity encoding profile (row #4) adds about 0.45 s of latency, that is tolerable considering the gain obtained in image quality. Finally, the reduction of the bitrate (from 80 to 20 Kbps) increases the latency (row #5) because the time to fill the UDP datagrams (whose size is kept unchanged for the sake of the test) is longer.

In order to produce a fair comparison, the other three systems have been configured to minimize latency – smallest buffer size and fastest video codec and profiling. In Windows Media, the streaming server (a middle layer) was removed and the video was streamed directly from the encoder to the player. VLC H.264 was configured with the same baseline profile used in MOSES; since raw UDP streaming is not supported, MPEG-TS/UDP/IP (the only one available on UDP) was used: under these conditions the latency is fairly low (row #8), but this is obtained at the cost of a fully compromised video quality, since about 48% of the video frames were corrupted or lost.

A possible side effect of reducing the latency is to lose video fluidity, having an irregular trend of Δt_{dec}. The graphs in Fig. 5.2(a) and 5.2(b) show the buffer occupancy and the Δt_{dec} respectively, with MOSES configured as in row #2 of Tab. 5.2: the buffer occupancy is always close to zero, resulting in an almost-ideal latency. However, this set-up generates a frequent increase of the frame decoding time, from the expected 100 ms (due to a $FR_{playback}$ of 10 fps) up to 1.4 s. These continuous changes in the Δt_{dec} bring to poor video fluidity, affecting both the overall user satisfaction and the understanding of the scene for the human-based video surveillance.

Fig. 5.3(a) and 5.3(b) show the improvements achieved by enabling MOSES' adaptive control in the condition of row #3 of Tab. 5.2. The trend of the Δt_{dec} demonstrates that the playback is made fluid. As described in Section 4.2, when the buffer occupancy is lower than T_L (5% in this experiment), the control starts decreasing the decoding frame rate until the buffer occupancy is stable between T_L and T_H. Conversely, when the buffer occupancy is higher than T_H, the control increases the playback frame rate to empty the buffer.

Fig. 5.4 shows image quality and frame losses measurements over the VLAB at 80 Kbps. Image quality is evaluated in terms of PSNR and the lost frames are represented with superimposed symbols at the bottom of the same graph. MOSES was configured with high encoding profile (row #4 of Tab. 5.2). VLC could not work properly with the same high profile due to a massive video frame corruption, therefore it was configured with a slightly lighter profile. Windows Media and Real Media were configured with WMV9 and RMV10 codecs respectively. Tab. 5.3 summarizes the achieved results: the statistics over the PSNR are computed on the correctly received frames only. It is clear that MOSES outperforms the others, showing also very few (and fairly distributed) lost frames.

	PSNR in dB avg. (std. dev.)	**% of lost frames**
Windows Media	34.79 (2.61)	2.09%
Real Media	35.32 (2.66)	9.38%
VLC	32.76 (3.04)	73.96%
MOSES	38.23 (2.16)	4.00%

Table 5.3: PSNR and percentage of lost frames in the PC-to-PC scenario over VLAB at QVGA and 80 Kbps.

Eventually image quality on MOSES at different bitrates is measured. We used the VCAR video, encoded with baseline profile at 20 (resized at QQVGA resolution), 80 and 120 Kbps. Results are shown in Fig. 5.5. The difference in the PSNR along the time is mainly due to the different scenes in the video (moving or stationary car) which change the global image motion therefore varying the compression quality. It is interesting to notice how the image quality increases significantly (becoming higher than 35 dB even in the 20 Kbps video stream) when the camera is not moving, being when the car has stopped at the traffic light (approximately from frame 1700 to 2000).

5.4 Experimental Results in PC-to-PDA Scenario

	Tool	**Profile**	**Bit-rate (Kbps)**	**Dec Dsp Cpl**	**Playback Frame Rate**	**Avg Latency (sec)**	**Std Dev (sec)**
1	MOSES	baseline	20	no	fastest	1.25	0.28
2	MOSES	baseline	20	yes	fastest	1.16	0.30
3	MOSES	baseline	20	yes	adaptive	1.76	0.39
4	MOSES	high prof	20	yes	fastest	1.68	0.58
5	MOSES	baseline	10	yes	fastest	2.29	0.47
6	MOSES m-to-m	baseline	20	yes	fastest	2.96	0.49
7	Windows Media		20			6.42	0.14
8	Real Media		20			3.87	0.10

Table 5.4: Latency of Windows Media, Real Media and MOSES in PC-to-PDA scenario over VLAB at QQVGA resolution. M-to-m stands for mobile to mobile.

Tab. 5.4 and Fig. 5.6 show the latency measurements in the PC-to-PDA scenario on the VLAB video; VLC is excluded because its PDA player is not actually available. Fig. 5.6(a) shows the comparison between Windows Media PDA player and Real Media PDA player according to row #7/8 of Table 5.4. Both these systems show an almost-constant, rather-high latency. Moreover, Windows Media shows latency scattering from second 60 to second 90, corresponding to the part of the video sequence when the camera is shaking; also the lost frames are concentrated in this part of the video.

Fig. 5.6(b) plots the latency of MOSES, encoding video in the three conditions of row #1/2/3 of Tab. 5.4. The orange plot (Tab. 5.4, row #1) shows the latency when the decoder and display threads are running decoupled. This generates a massive presence (79%) of frame losses, due to overwriting. The plot shows sharp and regular peaks due to buffer underflows (sharp latency raises) and very fast playback (sharp latency falls). The introduction of the decoder-display coupling avoids frame overwriting and the frame losses are reduced to 2%: this is the blue plot (Tab. 5.4, row #2), that shows smaller peaks as expected; however there is a higher dependency of the latency on video sequence complexity: for example, the latency suddenly drops around second 28, when the camera starts to move. This is due to the tolerance of the bitrate control: the more complex the scene is, the higher the encoding bitrate and the lower the time to fill up and deliver a datagram. The opposite effect is visible around second 96, when the camera stops moving. As in the PC-to-PC case, also in this configuration the introduction of the decoder-display coupling yields a lower latency than with the coupling disabled. The green plot (Tab. 5.4, row #3) shows the latency when the adaptive control is turned on: since the buffer is initially empty, it reduces $FR_{playback}$ of a factor $\epsilon = 1.002$ each frame; after a few seconds (approximately 15), $FR_{playback}$ has reached a critical value and the latency starts to increase until the buffer occupancy becomes definitely greater than zero: at this point, the reaction of the adaptive control becomes effective and reduces the latency through the increase of $FR_{playback}$ of ϵ^2 each frame. The adaptive control drastically reduces the presence of peaks in the latency. Fig. 5.6(c) shows the frame number trend of the streams of Fig. 5.6(b) in a short time interval. The plot clearly shows the smoothness in the playback introduced by the adaptive control.

Eventually, Fig. 5.6(d) shows the latency measured for MOSES in the three conditions of row #2/4/5 of Tab. 5.4. As expected, the high profile increases the latency. Also the 10 Kbps stream latency is higher: as aforementioned, it is because the reduction of encoding bitrate increases the time to fill the UDP datagram.

We also measured latency in a mobile-to-mobile setup, row #6 (specifically laptop to PDA) setup: the plot of the latency is similar to the other cases, but the average and the standard deviation tend to increase. In fact this configuration introduces a further degree of instability in the network communication, due to the additional step of the video data flow on radio mobile channels.

	PSNR in dB. Avg. (std. dev.)		
	20 Kbps	**10 Kbps**	**5 Kbps**
Windows Media	28.52 (2.14)	27.02 (0.97)	27.81 (1.01)
Real Media	30.39 (2.80)	28.59 (2.24)	27.36 (1.78)
MOSES	32.80 (2.77)	28.93 (3.28)	24.47 (3.01)
MOSES, forced @3.3 fps	n/a	n/a	29.76 (3.28)

Table 5.5: PSNR in the PC-to-PDA scenario over VLAB at QQVGA resolution.

	% of lost frames		
	20 Kbps	**10 Kbps**	**5 Kbps**
Windows Media	17.27%	56.73%	96.11%
Real Media	8.93%	30.04%	60.01%
MOSES	0.32%	0.64%	15.86%
MOSES, forced @3.3 fps	n/a	n/a	67.48%

Table 5.6: Percentage of lost frames in the PC-to-PDA scenario over VLAB at QQVGA resolution.

Eventually we calculated the PSNR of the compressed frames and the frame losses of the VLAB on the three systems. MOSES was configured in high profile (row #4 in Tab. 5.4). Fig. 5.7 shows the results at 20 Kbps, 10 Kbps and 5 Kbps. The stronger the compression becomes, the higher the frame loss rate is. It is evident that MOSES outperforms, on average, the other two, especially in terms of lost frames. Tab. 5.5 reports a summary of the average PSNR and Tab. 5.6 the percentage of lost frames. MOSES could sustain 10 fps even at 5 Kbps, but, as expectable, the frame rate is maintained only at the cost of PSNR. Forcing our encoder to skip 2 frames on 3, PSNR increases significantly (Fig. 5.7): the percentage of lost frames is 67.48% (consider that 66.6% was due to the forced frame skipping at the encoder side), but the average PSNR is 29.76 dB. However, as stated in Section 3.1, this is not a suitable solution for the computer-based video surveillance due to an excessive frame skipping that prevents correct tracking.

5.5 Measuring Compression Only: Speed and Quality

We performed a further set of tests to assess the performance of the compressor Movie. In the specific we are interested in excluding the streaming and decoding parts and measuring only compression parameters, namely speed and frame-by-frame quality, assuming to use bandwidths that may be supported by UMTS or HSPA networks; to this aim we propose the following benchmarks:

- *video benchmark*: three minutes long video clip, taken from the movie "Terminator 3" from minute 3 to minute 6 in PAL resolution (720x576). The original video is compressed with MPEG2 codec (DVD compression); the proposed video clip qualifies as a challenging benchmark for video compressors since it is (obviously) taken with moving camera, contains high degree of motion and several sudden changes of visual scenarios;
- *encoder benchmark*: the video encoder is fed with the video benchmark at two different resolution (720x576, 30 fps and 360x288, 25 fps) and is requested to target the bandwidths from 100 Kbps to 1 Mbps, with 100 Kbps stride;
- *computer benchmark*: a desktop PC with Windows XP. The hardware configuration is: AMD Athlon 64 X2 Dual-Core, 2.00 Ghz, equipped with 2GB of RAM.

The tests measure: (a) the time needed to compress the whole video; (b) the PSNR computed comparing the compressed frames with their original version (that is MPEG2 compressed) and then averaging the resulting PSNRs over the whole video clip. Actually, the time measurements include also the decoding from the original MPEG2 clip and the storage of the compressed stream in the file system. It is straightforward that if the encoding time is less than three minutes, the task can be performed in real-time on the given computer benchmark.

The first test measures speed and PSNR of Movie configured with two different profiles of the H.264 encoder (quoted also in Section 5.2):

1. *baseline profile*: H.264 is configured to minimize the CPU load at the cost of low video quality compression. It uses fast motion estimation, no B frames, wide key-frames interval; CABAC and deblocking filters are disabled;
2. *high profile*: opposite of baseline, i.e. highest video quality compression at the cost of higher CPU burden. It uses exhaustive motion estimation, P and B frames, limited key-frames interval, CABAC and de-blocking filters.

. The results reported in Tab. 5.7 show that the high profile yields to slightly better PSNR at a much higher computational load; consider that the compression in this set of tests has been forced to work on one CPU core only.

The second test, reported in Tab. 5.8, compares the performance of Movie versus the commercial encoding software *MainConcept Reference*[1]: both applications have been configured to work with baseline profile and the limitation to one CPU core has be removed: indeed the compression time in Movie is approximately halved (w.r.t. the times reported in Tab. 5.7, column "Baseline Profile"), demonstrating the efficiency of the multi-threaded implementation described in Section 4.1. At full resolution

[1]URL: www.mainconcept.com

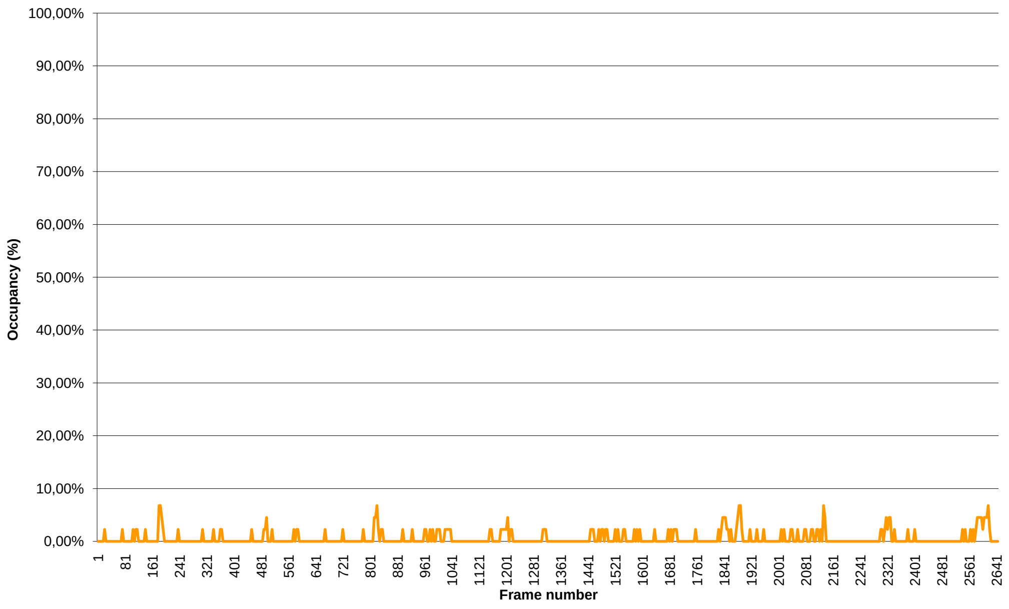
Video Clip Properties	Bit-Rate	Baseline Prof. Enc. Time	Baseline Prof. PSNR	High Prof. Enc. Time	High Prof. PSNR
720x576 30 fps	100Kbps	4.38	31.59	12:17	32.48
	200Kbps	4.52	34.11	14:58	35.10
	300Kbps	4.59	35.40	17:02	36.35
	400Kbps	5.10	36.14	18:40	36.62
	500Kbps	5.18	36.67	19:57	37.54
	600Kbps	5.26	37.07	21:03	37.88
	700Kbps	5.38	37.39	22:04	38.14
	800Kbps	5.51	37.62	23:02	38.35
	900Kbps	6.01	37.83	23:56	38.52
	1Mbps	6.12	38.01	24:38	38.67
360x288 25 fps	100Kbps	2.18	34.01	5:33	35.00
	200Kbps	2.21	35.65	6:35	36.55
	300Kbps	2.26	36.47	7:26	37.24
	400Kbps	2.30	36.97	8:05	37.67
	500Kbps	2.33	37.33	8:45	37.97
	600Kbps	2.37	37.60	9:19	38.18
	700Kbps	2.43	37.82	9:53	38.35
	800Kbps	2.49	38.00	10:22	38.48
	900Kbps	2.54	38.13	10:51	38.58
	1Mbps	2.58	38.25	11:25	38.67

Table 5.7: Encoding performance of benchmark video with Movie. The compression is forced to compute on one CPU core only.

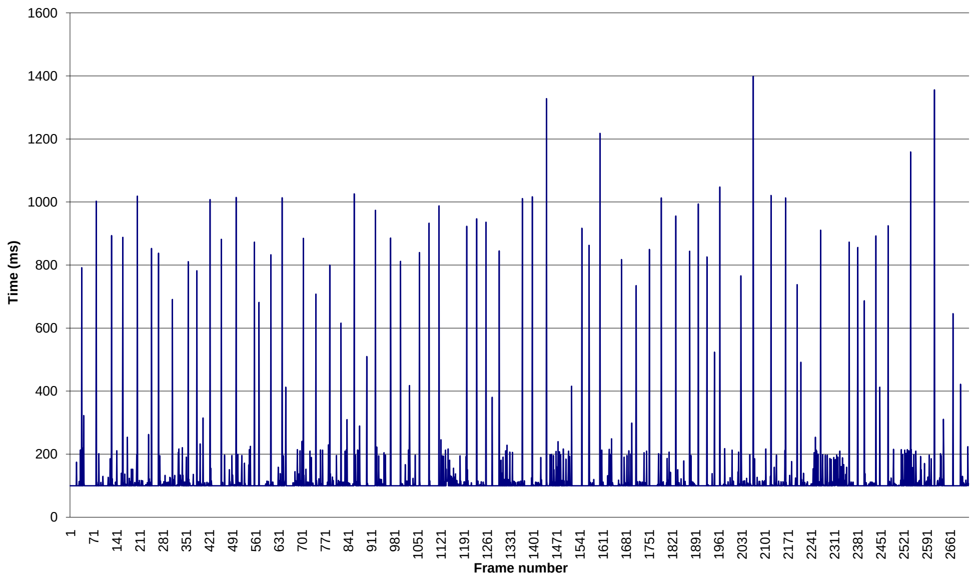
MainConcept yields slightly better results both in speed and video quality, but at half resolution the performance are reversed, showing a better video quality and a significant lower time in favor of Movie. Regardless of these minor differences, these tests demonstrate that even at higher bitrates and frame sizes w.r.t. the tests in the former sections, Movie provides state of the art performances, comparable to commercial softwares.

Video Clip Properties	**Bit-Rate**	MOVIE		**MainConcept**	
		Enc. Time	**PSNR**	**Enc. Time**	**PSNR**
720x576 30 fps	100Kbps	2:46	31.59	2:41	31.64
	200Kbps	2:47	34.11	2:42	34.80
	300Kbps	2:53	35.40	2:46	36.23
	400Kbps	3:04	36.14	2:50	37.10
	500Kbps	3:09	36.67	2:55	37.72
	600Kbps	3:18	37.07	3:00	38.16
	700Kbps	3:21	37.39	3:04	38.53
	800Kbps	3:23	37.62	3:09	38.83
	900Kbps	3:30	37.83	3:14	39.08
	1Mbps	3:33	38.01	3:17	39.29
360x288 25 fps	100Kbps	1:26	34.01	1:45	32.96
	200Kbps	1:26	35.65	1:49	34.41
	300Kbps	1:27	36.47	1:53	35.00
	400Kbps	1:27	36.97	1:56	35.34
	500Kbps	1:28	37.33	1:58	35.56
	600Kbps	1:29	37.60	2:00	35.72
	700Kbps	1:29	37.82	2:01	35.84
	800Kbps	1:31	38.00	2:02	35.93
	900Kbps	1:32	38.13	2:03	36.00
	1Mbps	1:32	38.25	2:07	36.06

Table 5.8: Encoding performance of benchmark video of MOVIE and MainConcept, configured with baseline profile. In this test the multi-threaded optimizations are enabled.

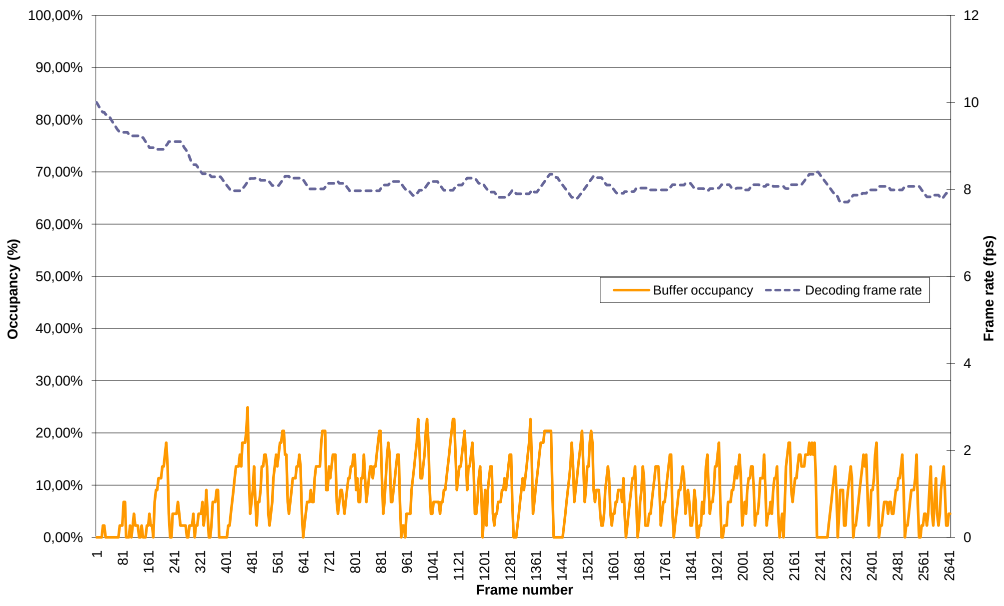


(a) Buffer occupancy (%)

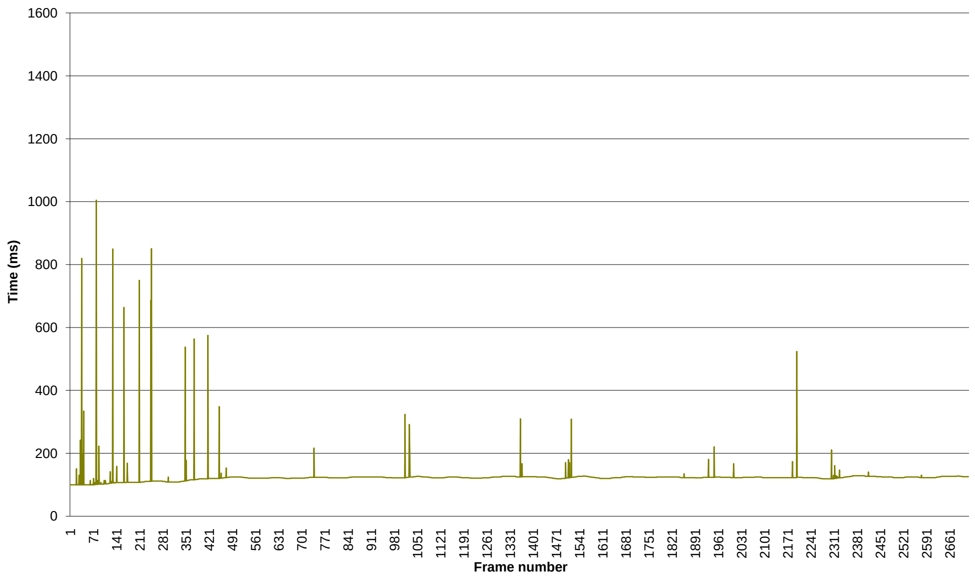


(b) Δt_{dec}

Figure 5.2: Buffer occupancy and Δt_{dec}, with MOSES adaptive control disabled.



(a) Buffer occupancy (%) and playback frame rate



(b) Δt_{dec}

Figure 5.3: Buffer occupancy and Δt_{dec}, with MOSES adaptive control enabled (configured with $T_L = 5\%$, $T_H = 15\%$, W=0.005 and $\rho = 0.04$).

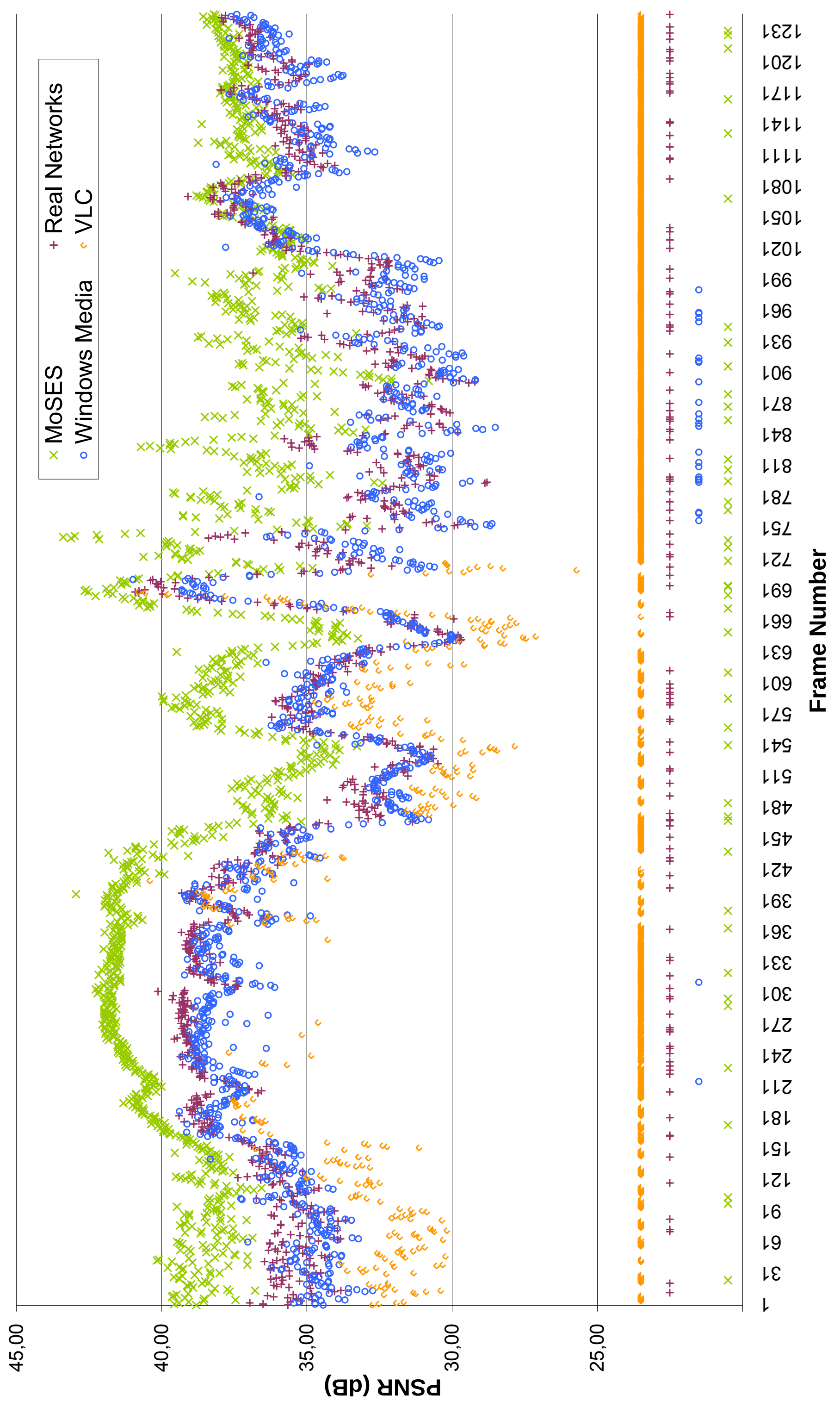


Figure 5.4: Comparison of image quality (PSNR) measured on the VLAB video at QVGA. The video contains scene with static camera (approx. frames 210-450 and 1030-end), moving camera (approx. frames 450-640), shaking camera (the rest). Lost frames are represented with the symbols in the bottom part of the graph.

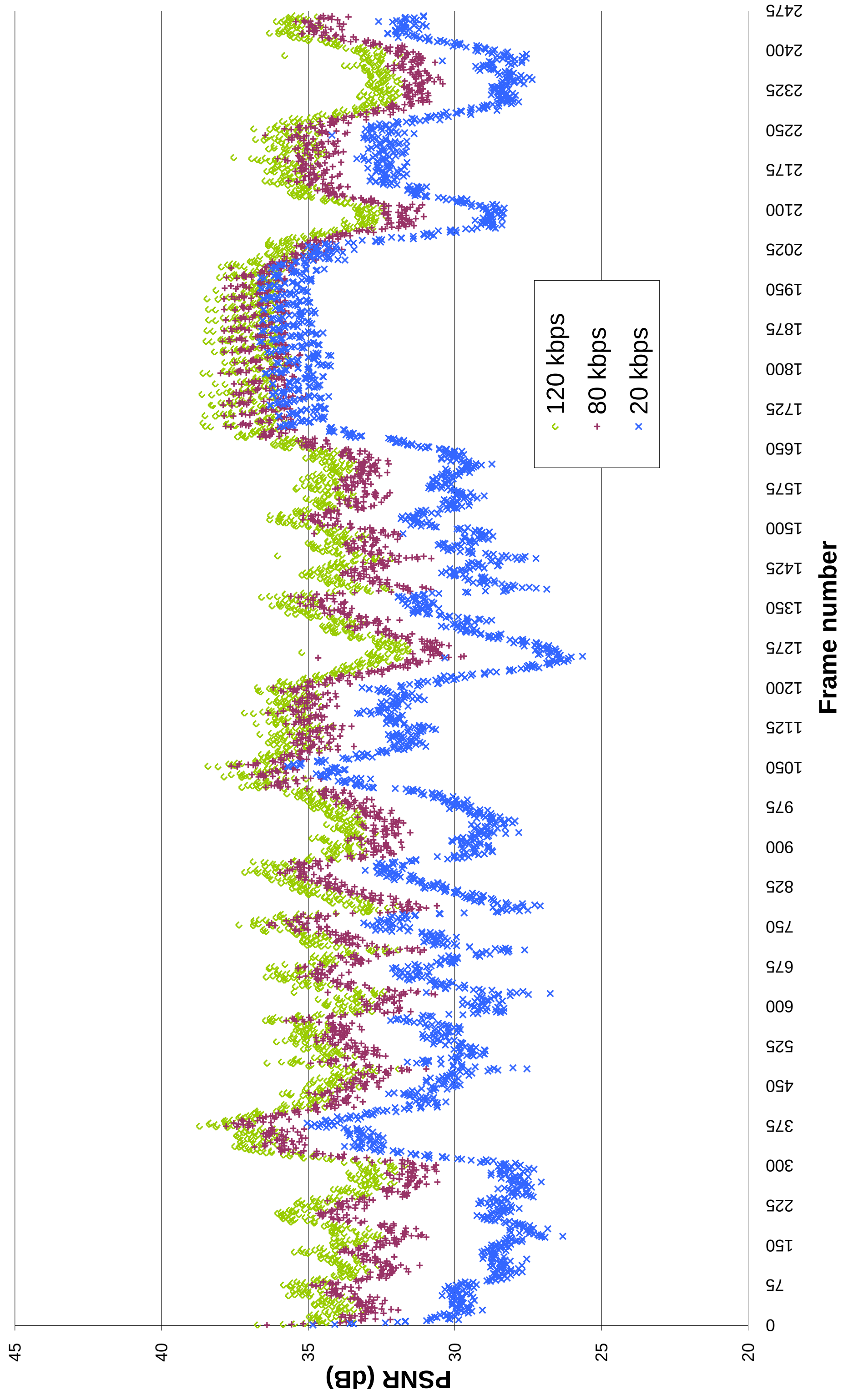
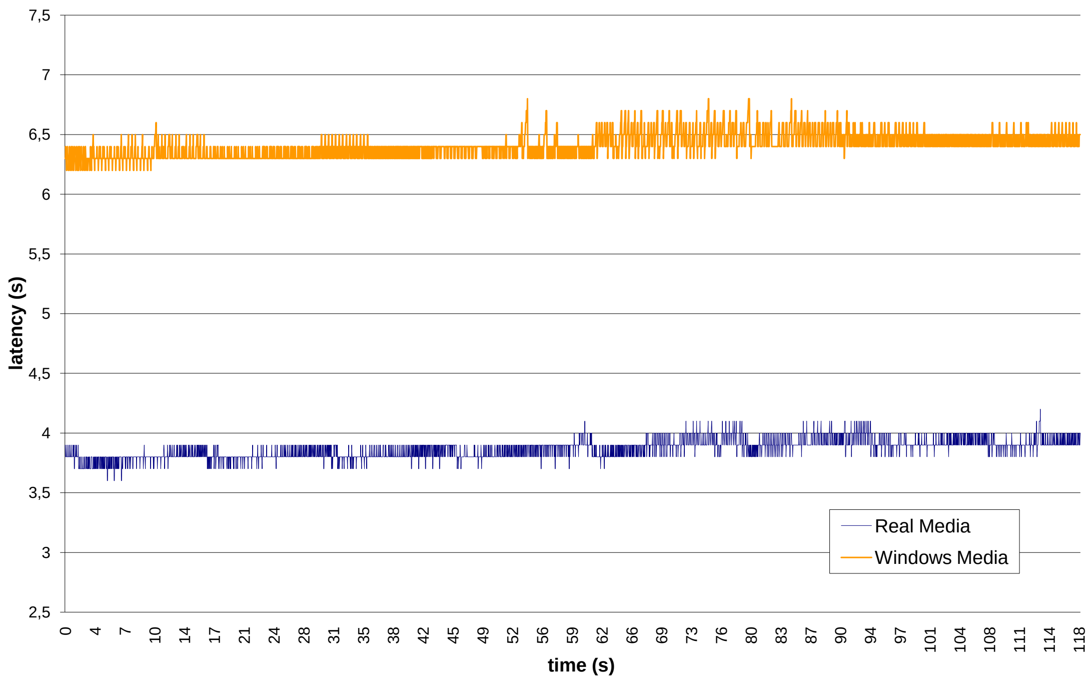
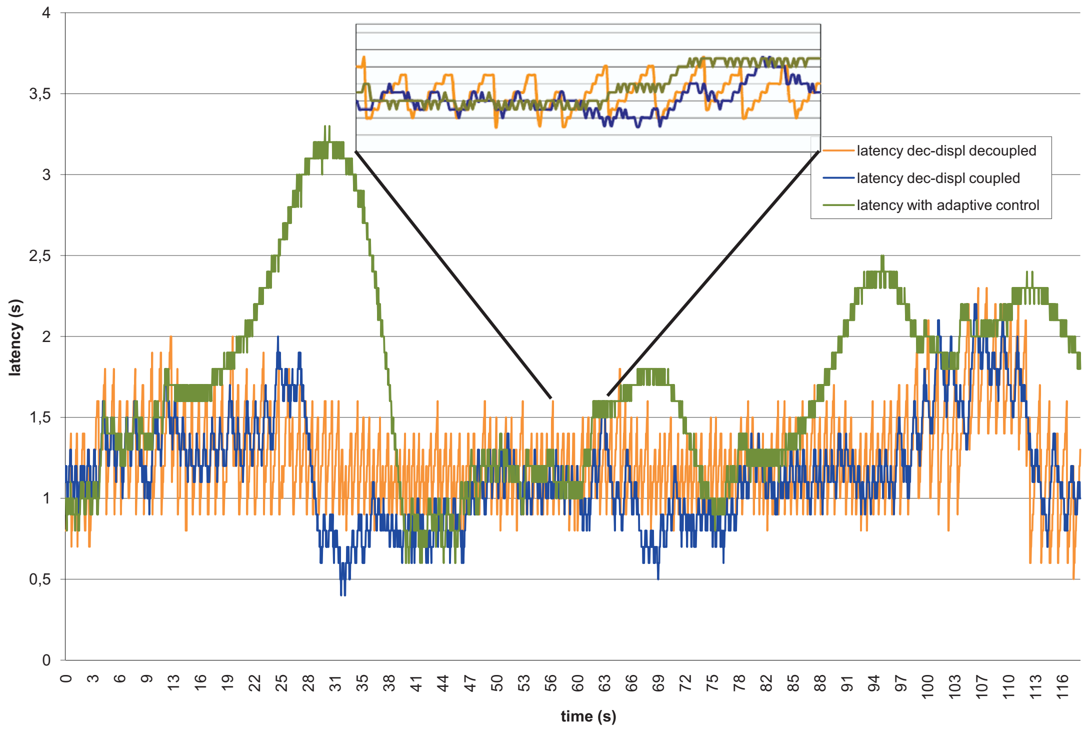


Figure 5.5: Image quality (PSNR) measured on the VCAR video using MOSES at 20 Kbps (QQVGA), 80 Kbps (QVGA) and 120 Kbps (QVGA).

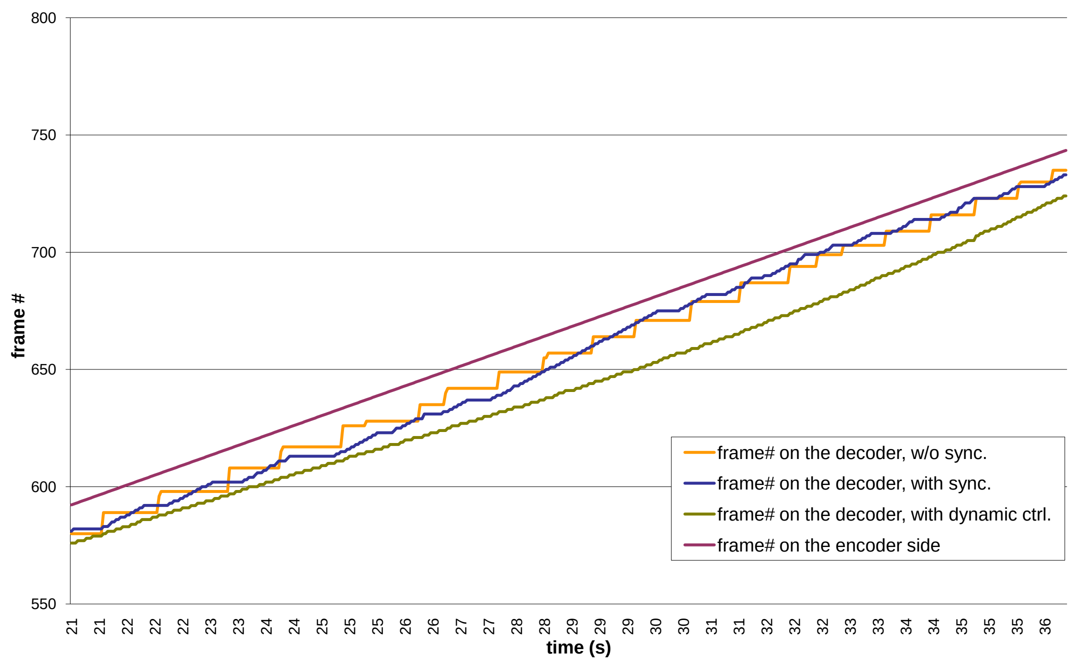


(a) Windows Media and Real Media

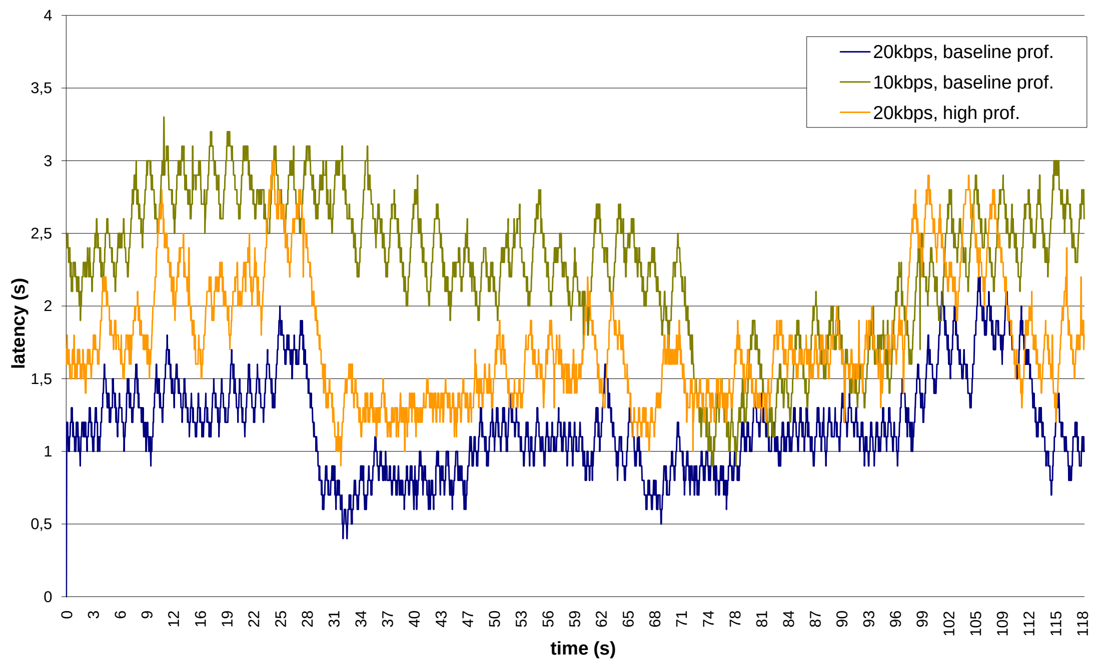


(b) MOSES, @20 Kbps, baseline profile

Figure 5.6: Comparison in terms of latency over VLAB at QQVGA resolution. The scale of the latency is different on graphs (a) and (b,d).

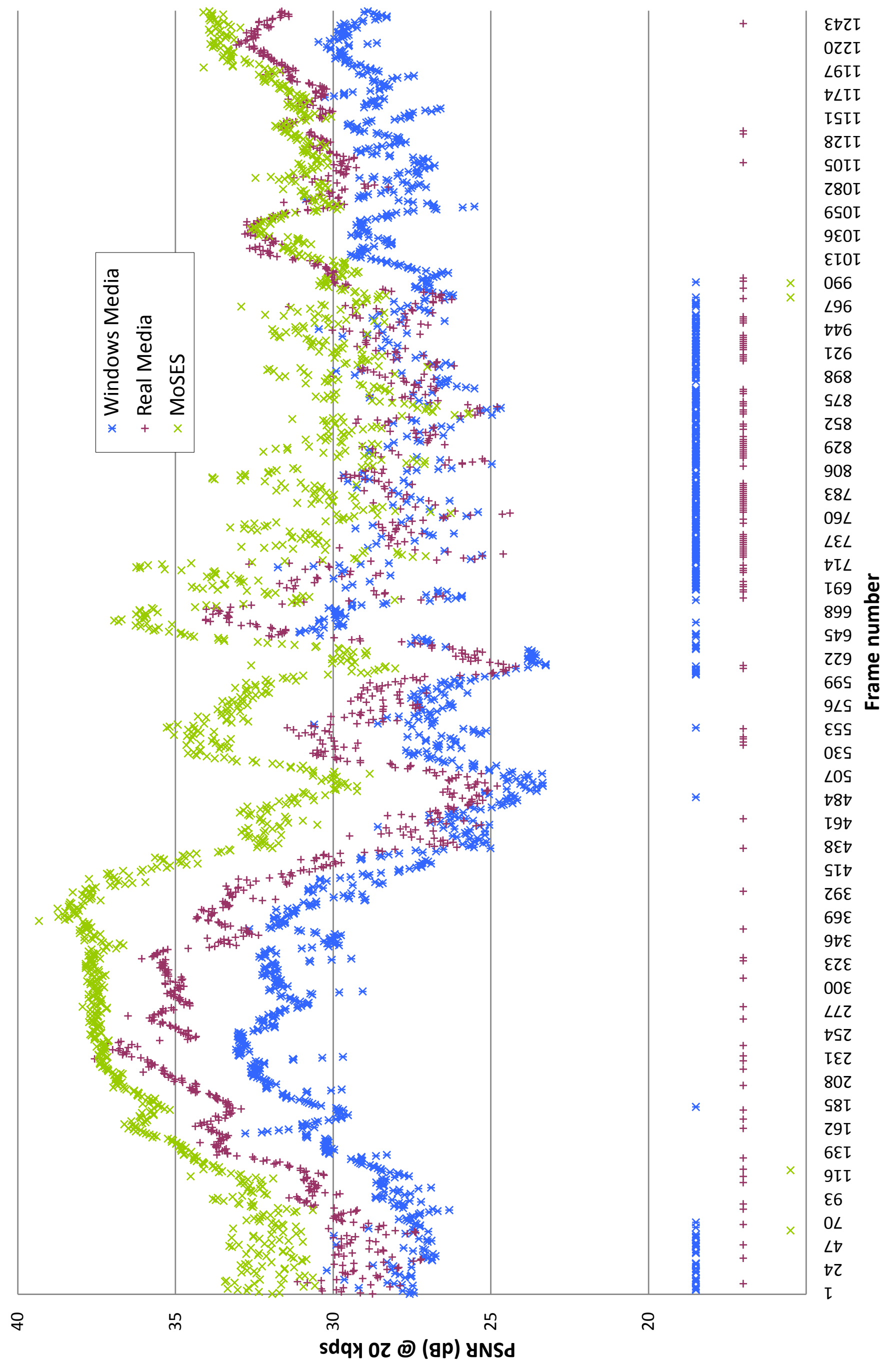


(c) Frame number trends in Moses



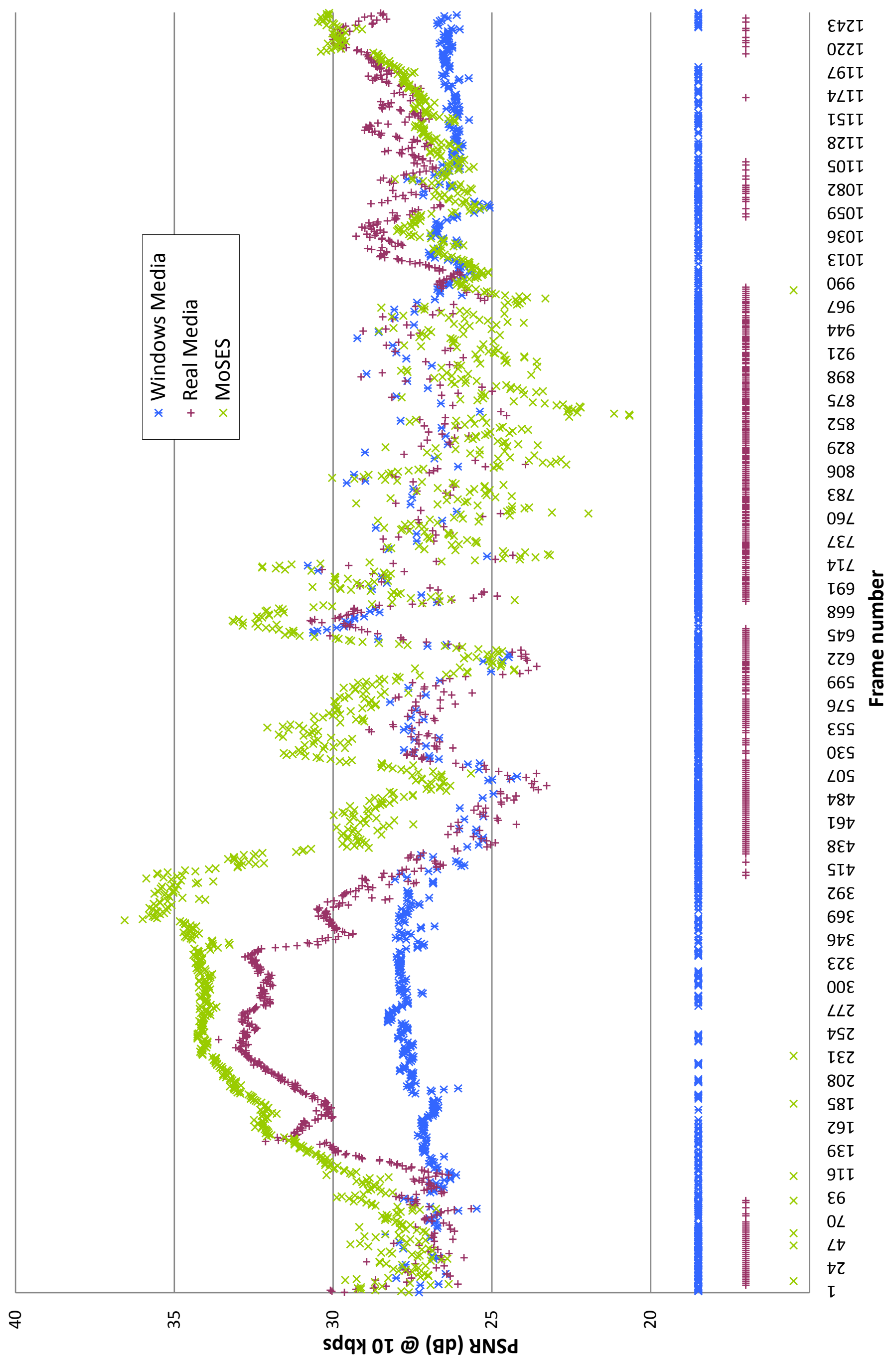
(d) Moses, with decoder-display coupling

Figure 5.6: Comparison in terms of latency over VLAB at QQVGA resolution. The scale of the latency is different on graphs (a) and (b,d).



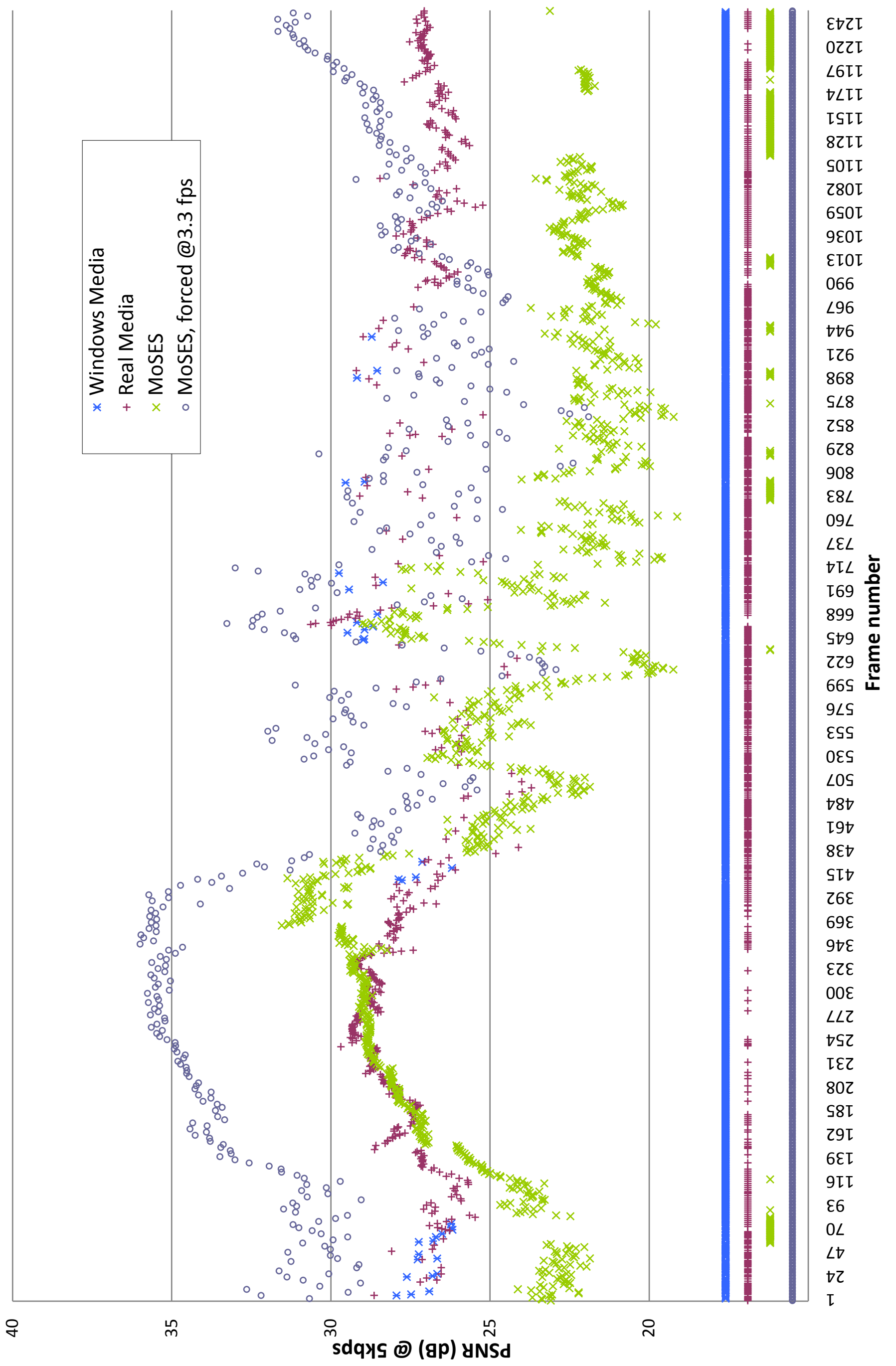
(a) 20 Kbps

Figure 5.7: Comparison of video quality (PSNR and frame losses) over VLAB at QQVGA and 20 Kbps. The symbols in the lower part of each graph indicate frame losses.



(b) 10 Kbps

Figure 5.7: Comparison of video quality (PSNR and frame losses) over VLAB at QQVGA and 10 Kbps. The symbols in the lower part of each graph indicate frame losses.



(c) 5 Kbps

Figure 5.7: Comparison of video quality (PSNR and frame losses) over VLAB at QQVGA and 5 Kbps. The symbols in the lower part of each graph indicate frame losses.

Part II

Object Tracking for Mobile Video Surveillance

Chapter 6

Introduction

In recent years, object detection and tracking have been recognized by the scientific community as fundamental tasks since they are founding blocks for most of the advanced video analysis steps. To be more specific, detection and tracking are complementary to each other, and depending on the properties of the video to process, one of the two is typically preliminary to the other: in other words, tracking by detection or detection by tracking [28].

In this thesis part we deal with the issues that arise when object tracking and detection are performed within the context of mobile video surveillance. As we mentioned in Section 2.2, there are two orthogonal contributions that must be considered in this case, namely mobility and distribution of the system modules.

Considering the several modules of surveillance systems and the large number of possible combinations mixing those two features, a very wide range of different tracking scenarios can be generated: for instance, tracking performed on embedded systems or on smart cameras, tracking on compressed videos, consistent labeling (multi-camera tracking) with synchronized or non-synchronized video streams, tracking from non-steady cameras, and so forth. Each of these tracking scenarios raise very challenging problems to be solved. We propose to tackle two specific types of tracking that lie on the extremes of the axes of Fig. 2.2, i.e. a tracking scenario with high degree of system distribution and another with high degree of system mobility.

Regarding the tracking scenario with high degree of system distribution, in Chapter 7 we envision a scenario with mobile source modules (i.e. the position of the cameras is unconstrained, but they operate remaining static), equipped with the bare minimum computational support to perform video grabbing, compression and streaming. All the steps to obtain object tracking are performed remotely. This is a typical scenario were the area to survey is subjected to frequent transformations or translations and unspecialized smart cameras are deployed. Object tracking on fixed camera videos has been thoroughly analyzed in scientific literature [11], and it could be considered a solved problem when performed in standard conditions. However, issues arise when the parameters of the system or the properties of the observed scene take extreme values. We are interested here in the evaluation of tracking performance when the video data is passed through an extreme compression and streaming process.

Publications related to Part II: [a,e,g]; see the list of author's publications, page 147.

In the case of IP-based cameras exploiting large-bandwidth networks, the problems introduced by the network are typically negligible and remote video analysis might be similar to local processing. However, this task becomes definitely more challenging when the camera is connected to the network through wireless low-capacity means, since a severe spatial and temporal compression of the video stream is unavoidable; this could compromise the automatic surveillance task that follows.

In the case of cameras that operate being fixed, the paradigm of tracking-by-detection is commonly adopted and the detection (called also segmentation) is often based on background suppression techniques which compare the actual pixel values with the corresponding values in the (statistically) learned model of the static scene. It is evident that the *frame compression* can significantly affect this step by changing pixel values and making a sophisticated statistical background model useless. For this reason, in Chapters 7 and 9 we define a system and a method to measure the performance of pixel-level moving object segmentation operated on videos that have been previously compressed and streamed over low-bandwidth networks. The tracking step is also sensitive to frame skipping and to excessively-low frame rates since it is typically based on object-to-track association (on a frame basis) and on limited search areas: thus, if an object moves too much on two successive frames, tracking is likely to lose it. For this reason, we also evaluate the ability of the system to track objects after a strong *temporal degradation* due to video transmission.

Regarding the tracking scenario with high degree of system mobility, in Chapter 8, we envision a scenario with moving source modules: more specifically, we tackle object tracking from cameras that are freely moving and changing the focal length. Differently from the previous scenario, it is not possible here to exploit statistical or geometrical models to segment the foreground objects from the background and predictive models (such as Kalman filters) are ineffective. In such cases, many scientific contribution propose the detection-by-tracking approach.

In point tracking, objects are usually represented by single or multiple points and the correspondences between two consecutive frames is established by either deterministic [66] or statistical methods [67] to provide tracking without object segmentation. An alternative is to represent the data using kernel primitives such as rectangles or ellipses. These kernel methods can be used to estimate a density-based appearance model of the object [68]. Other approaches encompass silhouette tracking, estimating the object contour evolution by means of state-space models [69] or variational methods [70].

These proposals are robust and efficient when the object can be represented by a single feature (e.g. color, texture, covariance descriptors, etc.) but in the case of complex articulated objects represented by parts which are often partially or completely overlapped they are likely to fail. To deal with such challenging scenarios structural information expressing spatial constraint among features might be used. This is the case of the pictorial structures [71] that have been proposed for object recognition and then further developed for people tracking [35]. Similarly [72, 73] are based on inference in a graphical model and can be applied again to people tracking [74]. All these approaches tend to be specifically focused on the articulated structure of the human body or human face and rely on Bayesian probabilistic frameworks; on the other hand tracking can be brought to a problem of graph matching through a graph based representation using Region Adjacency Graph (RAG), where vertices represent image regions and edges encode adjacency. This is the case of [75–78]. A notable exception is [79] where RAGs are tracked by fitting independent Kalman filters to both regions and adjacency relations. [80] uses graphs and Kalman filter for insects tracking.

Structural methods based on point features are less used than region-based ones. This is primarily due to the fact that defining relations between point features is more difficult. In [81] SIFT features are extracted from the tracked object and a nearest-neighbor graph is built on top of them. Relaxation labeling is used for matching and the object graph itself is updated by removing disappearing features and adding new ones. In [82] the tracked features are the linear borders of geometric objects and edges connect parallel or perpendicular borders.

The definition of the structural model can be inferred from the image data. This approach is very general but might suffer from the instability of the model inference, in terms of detection of both regions and features and with respect to the invariance of the relational structure to be tracked. In addition, an inferred model is inherently unable to capture detailed information about the intrinsic articulation and deformability of non-rigid objects. This lack of use of previous knowledge in tracking is indeed surprising as structural models, such as spring models [83, 84], are widely used to describe the behaviour of articulated objects.

By contrast, in Chapter 8, our proposed approach requires a prior structural model of a target object with generic shape (i.e. not necessarily bound to the human body figure or human face); this model is then enriched with features (or attributes) extracted from real data. In this way we are able to search for a match that not only maximizes the frame-by-frame feature matching, but that also accounts for the coherence of the structural relations in a way invariant to variations in scale, rotations and translations, and even blurring due to camera motions; the search for the best coherent match with the provided model is made through Dominant Set extraction [85].

Chapter 7

Tracking on Low Bandwidth Video Streaming from Steady Camera

The system used for evaluating the performance of object tracking with mobile source module and remote video analysis, is made of two main parts: MOSES, described in Chapter 4, provides video grabbing, compression, up-streaming, down-streaming and decompression; SAKBOT (Statistical And Knowledge-Based Object Tracker), described in the following section and in [9], performs moving object detection and tracking. In the specific, MOVIE grabs the video, encodes and streams it over radio-mobile networks; MOVIDE down-streams and decodes the stream and passes the video frames to SAKBOT. The inter-process communication between MOVIDE and SAKBOT is provided with Memory Mapped Files (MMF; see Appendix B for details): the two applications exchange raw uncompressed data, in order to keep the quality of the decoded video frames unchanged. Fig. 7.1 depicts the architecture used to assess the performance of the tracking performed with SAKBOT on low bandwidth video streaming w.r.t. the performance of the same system working on full quality videos; further details are provided in Chapter 9.

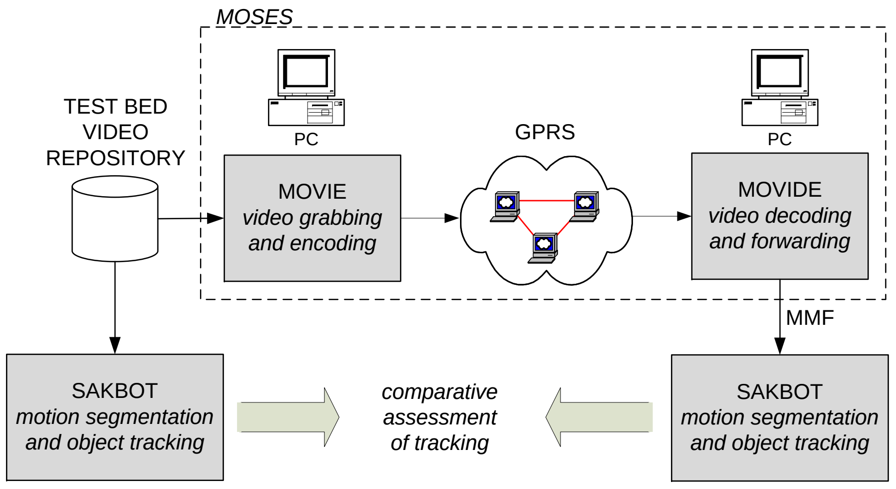

MOSES
TEST BED VIDEO REPOSITORY
PC
GPRS
PC
MOVIE *video grabbing and encoding*
MOVIDE *video decoding and forwarding*
MMF
SAKBOT *motion segmentation and object tracking*
comparative assessment of tracking
SAKBOT *motion segmentation and object tracking*

Figure 7.1: Functional scheme of the architecture used to assess tracking on low bandwidth video streaming from steady camera.

7.1 A System for Segmentation and Tracking with Fixed Background

The functional scheme of SAKBOT is depicted in Fig. 7.2: called $\mathbf{I}^t$ the current image and $\mathbf{I^t}(p)$ the value of a point p in the RGB color space, SAKBOT compares the input image with the background model $\mathbf{B^t}$, defined for each point of the image. If p is a point on the uncovered background then $\mathbf{B^t}(p)$ should correspond to its value in the current frame; however, if p is a point of a known object (i.e. that has been segmented and classified), $\mathbf{B^t}(p)$ is an estimation of the value of background covered by the object itself. Thus, if point p does not belong to any known object, the background value in p is predicted using only statistical information ($\mathbf{B_s^{t+\Delta t}}(p)$) on the set of elements $S(p) = \{\mathbf{I^t}(p), \mathbf{I^{t-\Delta t}}(p), ..., \mathbf{I^{t-n\Delta t}}(p)\} \cup w_b\{\mathbf{B^t}(p)\}$. In order to improve the stability of the model we include an adaptive factor by combining the n sampled frame values and the background past values (with an adequate weight w_b). The n frames are sub-sampled from the original sequence at a rate of one every Δt (typically one every ten). Then, the statistical background model $\mathbf{B_s}$ is computed by using the median function as follows:

$$\mathbf{B_s^{t+\Delta t}}(p) = \underset{i=1,\dots,k}{arg\,min} \sum_{j=1}^{k} Distance(\mathbf{x_i}, \mathbf{x_j}) \qquad \mathbf{x_i}, \mathbf{x_j} \in S(p) \tag{7.1}$$

where the distance is a *L-inf distance* in the RGB color space:

$$Distance(\mathbf{x_i}, \mathbf{x_j}) = max(|x_i.c - x_j.c|) \quad with \ c = R, G, B. \tag{7.2}$$

Foreground points resulting from the background suppression could be used for the selective background update also for the ghost (i.e. apparent object) suppression. When a stopped object starts to move two foreground objects, one real and one apparent (the ghost) are generated and the background must be selectively modified. Thus, foreground points are segmented into known objects whose motion and texture features are evaluated to classify them in moving visual objects (MVO), noise, shadows or ghosts. Shadows are detected as proposed in [86]. Accordingly, a knowledge-based background model $\mathbf{B_k}$ is defined as:

$$\mathbf{B_k^{t+\Delta t}}(p) = \begin{cases} \mathbf{B^t}(p) & \text{if } p \in \{O|O \text{ is MVO}\} \\ \mathbf{B_s^{t+\Delta t}}(p) & \text{if } p \in \{O|O \text{ is ghost}\} \end{cases} \tag{7.3}$$

The knowledge of the scene's components in the current frame will be used to update the background model $\mathbf{B}$:

$$\mathbf{B^{t+\Delta t}}(p) = \begin{cases} \mathbf{B_s^{t+\Delta t}}(p) & \text{if } p \notin \text{ known object} \\ \mathbf{B_k^{t+\Delta t}}(p) & \text{otherwise} \end{cases} \tag{7.4}$$

The expression in Eq. 7.4 defines a selective background update, in the sense that a different background model is selected whether the point belongs to a known object or not.

Moving objects detected by SAKBOT are then classified as person or non-person according to their geometrical shape and size (scaled according to their position on

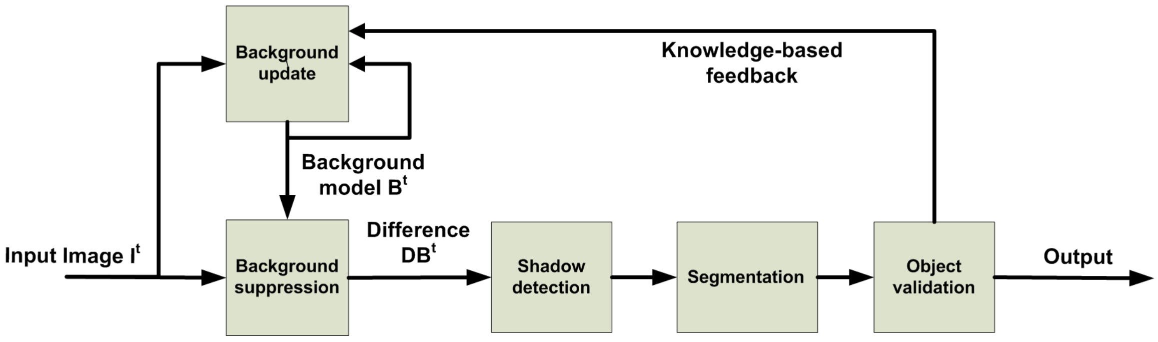


Figure 7.2: Functional scheme of SAKBOT

the ground plane, if calibration data are available). The objects validated as people are then tracked by means of an appearance-based algorithm. The algorithm uses a classical predict-update approach. It takes into account not only the status vector containing position and speed, but also the memory appearance model and the probabilistic mask of the shape [87]. The former is the adaptive update of each pixel in the color space. The latter is a mask whose values can be viewed as the probability for that pixel to belong to that object. These models are used to define a maximum a posteriori classifier that searches the most probable position of each person in the scene. The tracking algorithm includes a specific module for coping with large and long-lasting occlusions. Occlusions are classified into three categories: self-occlusions (or apparent occlusions), object occlusions, and people occlusions. Occlusion handling is very robust since it can maintain the shape of the tracked objects in a very precise manner. It has been tested in many applications and further details can be found in [87].

The approach used in SAKBOT is quite robust and has been applied in several different outdoor/indoor contexts, but requires both an almost-fixed time interval between successive frames (the above-defined Δt) and a good image quality to have correct pixel values for the background updating.

Chapter 8

Tracking with Freely Moving Camera

The approach proposed here tackles object tracking in the challenging condition of freely moving camera and varying focal length. For the sake of performance evaluation, the experimental results are generated using recorded videos, but nothing hinders to deploy the proposed algorithm on a mobile video surveillance system with moving source module. The system could be either monolithic or distributed. However, in the second case, the network bandwidth to rely on for video streaming must be definitely higher than what has been discussed in Chapter 7, where only videos with fixed background were taken in consideration. Indeed, a video footage having equal frame rate and frame size of what was proposed there but grabbed with freely moving camera, would require higher compression bit rates to maintain similar video quality. The test performed in Section 5.5 demonstrate that MOSES can provide the desired video quality even in this challenging scenario.

8.1 Overview of the Framework

Fig. 8.1 shows the conceptual scheme of our framework. An initial *Graph-Based Model Definition* provides the framework with both a model of the features to be tracked and a structural representation of their spatial arrangement. In this proposal color features are used, but different or more descriptive features could be exploited (textures, edges, covariance descriptors, etc.). Moreover, an initial frame can be taken as reference for the extraction of the feature model, the structural model or both (Fig. 8.2(a)). Each new frame I_t (Fig. 8.2(b)) is provided to the *Feature Cluster Extraction* component (Section 8.2) that applies the feature model and produces the mask of the probability of each feature class onto the current image (called *back-projection*). Each back-projection is then clustered using meanshift and, for each cluster, attributes are extracted. Most of the extracted feature clusters represent erroneous detections of the tracked object feature (see Fig. 8.2 (c-g)) and the correct candidates must be extracted using global consistency information, using the procedure that follows.

A labeling function maps each feature cluster on the originating model feature. Each pair of clusters, whose features are rigidly joined together in the structural model, are connected by edges to form the *labeled graph* G_t (Section 8.3.1). Then an edge weighted *association graph* GA_t is created between the structural model G_0 and the labeled graph G_t (Section 8.3.2) and each edge is weighted according to a global *coherence measure*

(Section 8.3.4) in such a way that each maximal edge weight clique in GA_t corresponds to a maximal coherence subgraph isomorphism of G_0 on G_t, and vice versa. Finally the Dominant Set framework [85] is used to search for the maximum coherence match (Section 8.3.3) $Match_t$ (Fig. 8.2(h)).

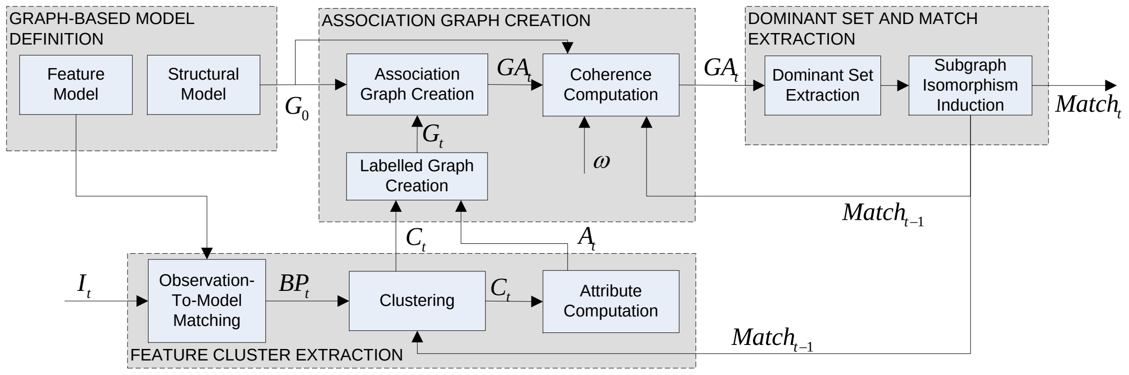


Figure 8.1: Scheme of the proposed framework.

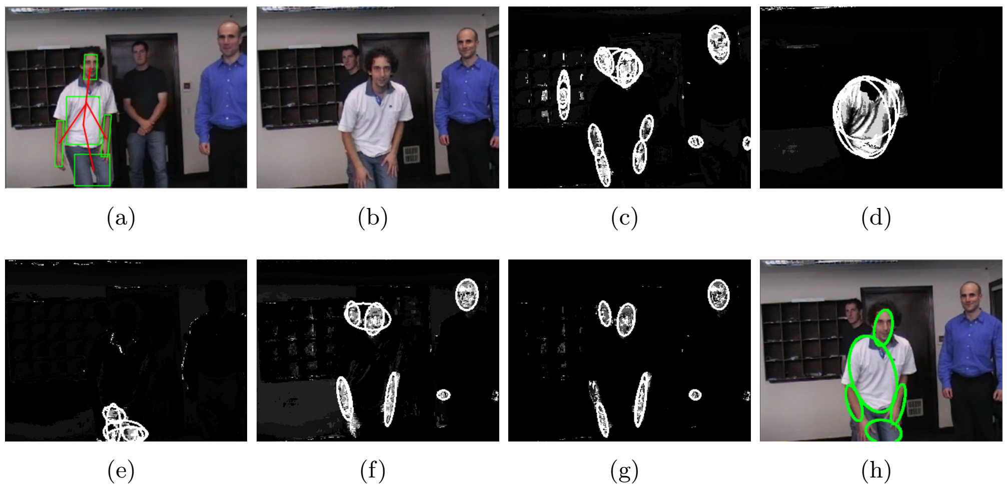


Figure 8.2: Framework steps by examples. (a) definition of structural model (red) and color-feature model (green); (b) generic input image; (c) backprojection of color-feature of the head on input image and clusters extraction. Each cluster is represented by an ellipse. (d-g) same as (c), for the following body parts: (d) for torso, (e) for legs, (f) for left arm, (g) for right arm; (h) result of tracking as a maximal subgraph isomorphism of the model (a) onto current image (b).

8.2 Extraction of Feature Clusters

For each feature of the model, the *Feature Cluster Extraction* component extracts all the possible clusters of features which might represent a part of the tracked object according to the feature model. The *Feature Cluster Extraction* operates on simple color features using a modification of the Camshift algorithm [88], but different cluster extraction algorithms could be used.

The standard Camshift tracking algorithm uses a model of the object, consisting of a color histogram, and requires a region of interest to initialize the search. For each input image a probability mask of the model is produced, evaluating each pixel according to the color histogram as if it were a pdf. The resulting value is then scaled on 256 gray levels, producing the so called back-projection. Then, iteratively alternating the meanshift gradient ascend algorithm and a size-adaptation of the region of interest, the region estimate converges to encompass the extracted features, providing the initial search location for the next frame.

For the extraction of the feature clusters, the Camshift is modified as follows. *First*, the object to be tracked is modeled with multiple histograms of colors, each corresponding to a different part (e.g. green boxes in Fig. 8.2(a)); therefore, for each input image, multiple back-projections (BP_t) are obtained (Figs. 8.2 (c-g)) and the cluster extraction proceeds independently for each BP_t. *Second*, the back-projections are obtained on the following quantization of the Hue-Saturation-Value color space:

$$(h,s,v) = \begin{cases} \left(\left\lfloor \frac{H}{16} \right\rfloor, \left\lfloor \frac{S}{16} \right\rfloor, \max_V\right) & \text{if } S > \tau_S \wedge V > \tau_V \\ \left(0, 0, \left\lfloor \frac{V}{16} \right\rfloor\right) & \text{otherwise.} \end{cases} \tag{8.1}$$

The addition of value and saturation components to the standard Camshift color space (that uses hue only) provides an enriched color description and allows to perform better with low-saturation colors (considering the V component only). *Third*, our approach scatters particles over the back-projections coming from each object part: each particle determines a different starting point for a Camshift procedure, and consequently several clusters are generated. The particles are spatially scattered over the BP_t with Gaussian or uniform distribution, depending on the object tracking status at the previous frame.

For each cluster C_t^i of the set C_t, the set of attributes:

$$A_t^i = \left[P\left(C_t^i\right), M\left(C_t^i\right), R\left(C_t^i\right), D\left(C_t^i\right)\right] \tag{8.2}$$

are computed, where $P = (x, y)$ are the coordinates of the cluster centroid, M its mass, R the area and D the density:

$$\begin{aligned} M\left(C_t^i\right) &= \sum_{p \in C_t^i} BP_t\left(p\right) \\ R\left(C_t^i\right) &= \left\|\left\{p \in C_t^i\right\}\right\| \\ D\left(C_t^i\right) &= \frac{M\left(C_t^i\right)}{R\left(C_t^i\right)} \end{aligned} \tag{8.3}$$

8.3 Tracking using relational information

Regardless of the robustness of the extraction step several factors could lead to a wrong assignment between clusters if plain part-to-part feature matching is used. Indeed, distractors, noise, pose deformations or illumination changes can easily lower the coherence between correct correspondences or make unrelated features more similar. For this reason any approach that is based only on the similarity between features is inherently sensitive to noise and clutter. To overcome this limitation we add contextual information, thus casting the feature matching into a more robust subgraph matching problem.

8.3.1 From feature clusters to labeled graphs

In order to obtain a graph from a set of feature clusters we exploit the prior knowledge about the physical structure of the object. To this end, we define a structural model where each part of the object is associated to a feature class which is known to be rigidly joined to some other parts, but can move freely from the rest. This is the case with any articulated object, while totally-rigid objects can be modeled by joining all the parts.

A *structural model* of an object is a connected graph $G_m = (P, S)$ where P is the set of distinct parts we use to represent the object, with cardinality $|P|$, and $S \subseteq P \times P$ are their structural relations, where $(p_a, p_b) \in S$ if and only if p_a and p_b are joined in the object. This model embeds our prior knowledge about the structure of the object to be tracked in terms of its parts. In Fig. 8.3 some examples of structural models are presented.

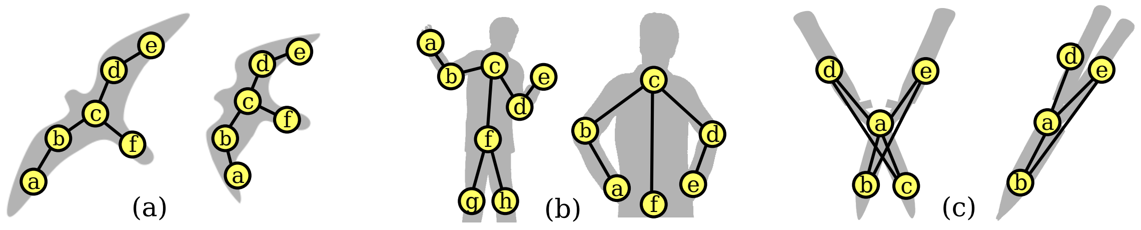


Figure 8.3: Example of objects and their structural models; the labels of the nodes are used to define the labeled graph from the structural model by means of the labeling function. For each object we depict a model transformation: (a) deformation, (b) deformation, scaling and occlusion; in (c) the model comprises two totally-rigid submodels one of which partially occludes the other.

Given

- a structural model $G_m = (P, S)$,
- a set of features clusters C,
- the set of corresponding attributes A,
- a surjective labeling function $l : C \to P$, that assign one label to each cluster,

we define the *labeled graph* as the $|P|$-partite graph $G = (C, E, A, l)$ where C is the vertex set, $E = \{(u, v) \in C \times C | \, [l(u), l(v)] \in S\}$ the edge set, A the vertex attributes and l the vertex labelling function. In this graph each edge represents a structural relation between a pair of feature clusters. The automatic extraction of feature cluster candidates from a frame T_i yields a graph with many nodes and edges. The supervised selection of the ground truth from a reference frame will result in a simpler graph with just one cluster for each part of the object to be tracked: we call this graph the *model* *graph.* Our goal is to find within each labeled graph extracted from a frame T_i the subgraph which is the most coherent with the model graph we are tracking. In other words we are looking for a maximum coherence subgraph isomorphism.

Given labeled graphs $G_1 = (C_1, E_1, A_1, l_1)$ and $G_2 = (C_2, E_2, A_2, l_2)$ a *labeled isomorphism* between them is a relation $M \subseteq C_1 \times C_2$ such that $\forall (u_1, u_2), (v_1, v_2) \in M$, with $u_1, v_1 \in C_1$ and $u_2, v_2 \in C_2$, the following properties hold:

$$l_1(u_1) = l_2(u_2) \wedge l_1(v_1) = l_2(v_2) \tag{8.4}$$

and

$$u_1 = v_1 \Leftrightarrow u_2 = v_2 \tag{8.5}$$

The first condition ensures that M does not map feature cluster of incompatible classes. The second condition forces M to be a partial injective function. It is easy to see that any labeled isomorphism is a special case of subgraph isomorphism which enforces label consistency.

We still need to define a measure of the global coherence of a labeled isomorphism M. In our context limiting the measure to a similarity between *vertex* attributes would be not enough, as in this way we would be unable to take into account structural relations among vertices. For example, considering vertices only it would be possible to measure the color similarity of a cluster (i.e. an attribute of a vertex of a labeled graph) with another cluster (from the other labeled graph); conversely it would not be possible to measure the consistency of the inter-cluster distances from one labeled graph to the other, since the concept of distance between clusters can be defined considering at least pairs of them (i.e. edges of a labeled graph) and not just singletons. Unfortunately, even measuring coherence between *edges* would not be general enough, as it would not be possible to account for invariants that depend on more than one edge, such as length ratios or angle differences: indeed, both these measurements can be obtained considering at least pairs of edges. For this reason we defined a coherence measure between *pairs of edge* matches as this allows us to deal with most of the variations in scale and articulation throughout the whole video sequence. To this end we define the set of edge matches as:

$$e(M) = \{[(u_1, v_1), (u_2, v_2)] \in E_1 \times E_2 | (u_1, u_2) \in M \wedge (v_1, v_2) \in M\} \tag{8.6}$$

and let $\omega : (E_1 \times E_2) \times (E_1 \times E_2) \to \mathbb{R}^+$ be a measure of coherence between pairs of edges matches, then the total weight of M is defined as:

$$\Omega(M) = \sum_{a \in e(M)} \sum_{b \in e(M) \setminus \{a\}} \omega(a, b)\,. \tag{8.7}$$

8.3.2 From graph matching to clique search

In order to search for a match of maximum compatibility between two labeled graphs we choose a two-step approach which first casts the matching problem into a clique search problem and then solves it using continuous optimization.

Given labeled graphs $G_1 = (C_1, E_1, A_1, l_1)$ and $G_2 = (C_2, E_2, A_2, l_2)$ and a function $\omega : (E_1 \times E_2) \times (E_1 \times E_2) \to \mathbb{R}^+$ that measures the coherence between pairs of edge associations, we define an *association graph* between them as an edge weighted graph

$Ga = (Va, Ea, \omega)$ where $Va = E_1 \times E_2$, $Ea \subset Va \times Va$ and, $\forall u_1, v_1, w_1, z_1 \in C_1$ and $\forall u_2, v_2, w_2, z_2 \in C_2$, the element:

$$\{[(u_1, v_1), (u_2, v_2)], [(w_1, z_1), (w_2, z_2)]\}$$

belongs to Ea if and only if the following three conditions hold:

$$\begin{aligned} &l_1(u_1) = l_2(u_2) \wedge l_1(v_1) = l_2(v_2) \wedge l_1(w_1) = l_2(w_2) \wedge l_1(z_1) = l_2(z_2), \\ &u_1 = w_1 \Leftrightarrow u_2 = w_2, \\ &v_1 = z_1 \Leftrightarrow v_2 = z_2 \end{aligned} \tag{8.8}$$

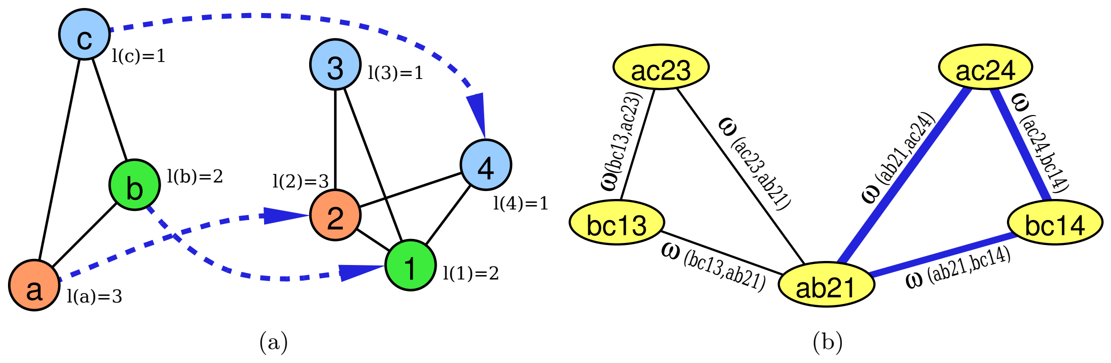


Figure 8.4: (a) two labeled graphs and a labeled isomorphism represented by the blue dotted connections; (b) association graph between the former labeled graphs; its edges are weighted by the coherence measure ω; the subgraph highlighted by blue edges is the clique associated to the labeled isomorphism shown in (a).

With this definition we are able to show some useful connections between labeled isomorphisms and complete subgraphs (cliques) in this association graph.

To this end, note that each $X \subseteq V_a$ represents a relation between edges in E_1 and E_2. In order to obtain a relation between vertices in V_1 and V_2 we define a natural map $v : \mathcal{P}(V_a) \to \mathcal{P}(V_1 \times V_2)$ as:

$$\begin{aligned} v(X) = &\{(u_1, u_2) \in V_1 \times V_2 | \\ &[(u_1, v_1), (u_2, v_2)] \in X \vee \\ &[(v_1, u_1), (v_2, u_2)] \in X\} \end{aligned} \tag{8.9}$$

That is, a match between vertices is induced by X if they are mapped by any edge match in X. It is easy to see that v is not injective, nevertheless it has a proper right partial inverse, namely the function $e(M)$ defined by (8.6).

We can now formulate the following lemmas (proofs are provided in Appendix C):

Lemma 1 *Given labeled graphs G_1, G_2 and their association graph G_a, $X \subseteq V_a$ is a clique if and only if $v(X)$ is a labeled isomorphism between G_1 and G_2.*

Lemma 2 *If $X \subseteq V_a$ is a maximal clique in G_a, then $v(X)$ is a maximal labeled isomorphism between G_1 and G_2. Conversely, if M is a maximal labeled isomorphism between G_1 and G_2 then $e(M)$ is a maximal clique in G_a.*

From the previous lemmas and the definition of the weight of the labeled isomorphism M, derives the following:

Theorem 1 *Given two feature graphs G_1 and G_2, each maximal(maximum) weight labeled isomorphism M between them induces a maximal(maximum) edge weight clique in $Ga(G_1, G_2)$ and vice versa.*

The examples of a labeled isomorphism an the correspondent clique in a labeled association graph are shown respectively in Fig. 8.4(a) and 8.4(b).

8.3.3 An effective heuristic for the weighted clique problem

Theorem 1 casts our tracking problem into a search for a maximal edge weighted clique in a novel type of association graph. In order to perform this search we use the Dominant Set framework [85]. Given an edge weighted graph $G = (V, E, \omega)$, a subset of vertices $S \subseteq V$ and two vertices $i \in S$ and $j \notin S$ the following function measures the coherence between nodes j and i, with respect to the average coherence between node i and its neighbors in S

$$\phi_S(i, j) = \omega(ij) - \frac{1}{|S|}\sum_{k \in S} \omega(ik) \tag{8.10}$$

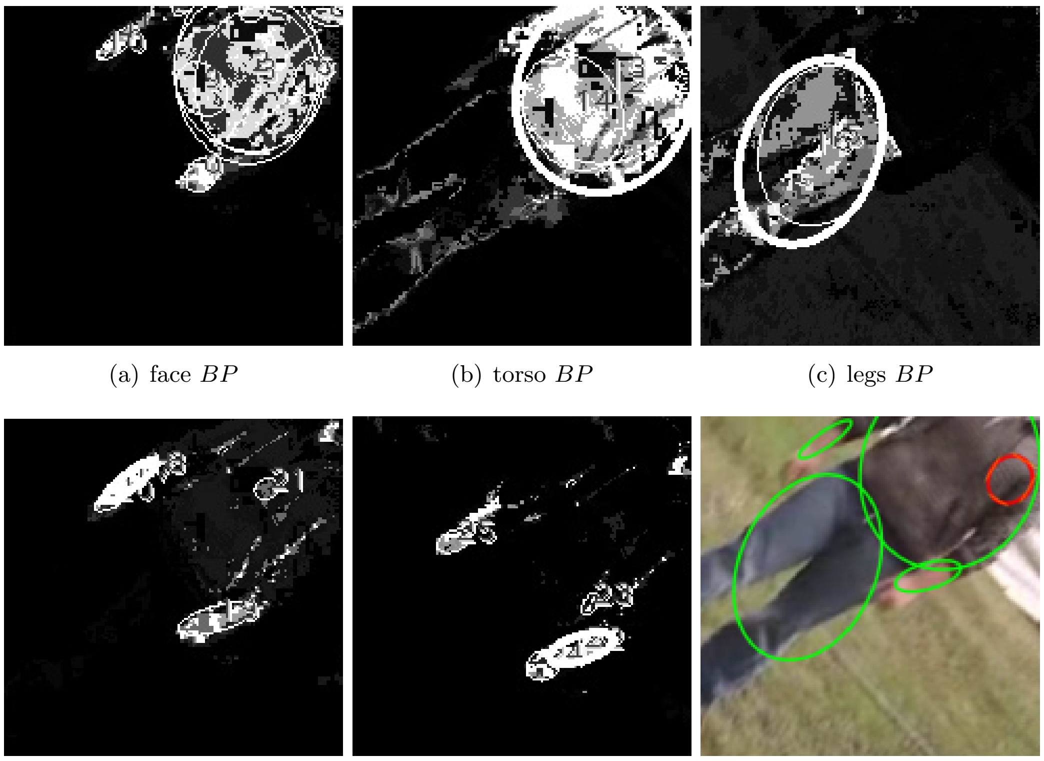
While overall weighted coherence between i and all the nodes in S is defined as:

$$w_S(i) = \begin{cases} 1 & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(i, j) w_{S \setminus \{i\}}(j) & \text{otherwise} \end{cases} \tag{8.11}$$

Intuitively, $w_S(i)$ will be high if i is highly coherent with vertices in S. Given this measure $S \subseteq V$ is said to be *dominant* if the following conditions hold:

$$\begin{aligned} &w_S(i) > 0, \forall i \in S \text{ and} \\ &w_{S \cup \{i\}}(i) < 0, \forall i \notin S \end{aligned} \tag{8.12}$$

The conditions above correspond to the two main properties of a cluster: namely internal homogeneity and external inhomogeneity. For this reason in the literature this framework has been associated to clustering. Nevertheless, its use as an heuristic for the edge weighted clique problem is justified by the fact that when applied to unweighted graphs, the notion of a dominant set is equivalent to the notion of a clique. Hence, a dominant set can be seen as a generalization of cliques to graphs with weighted edges. Moreover, there is another compelling reason to prefer dominant sets over other techniques of clique search: since we are dealing with graphs coming from real images it is quite common that a spurious node fits in the labeled isomorphism. For instance this is the case when a part of the model is occluded and distractors of that part are present. In such cases there is no correct matching node, but assigning a spurious node would nevertheless yield an isomorphism with a higher global coherency measure: therefore a greedy clique search would include it as well. On the opposite, by using the dominant set framework instead of a greedy clique search technique, the nodes with a low coherence with respect to the others are automatically discarded (see Fig. 8.5). This false positive suppression happens by exploiting the clustering property of the dominant set framework and without defining any threshold.



(a) face BP (b) torso BP (c) legs BP

(d) right arm BP (e) left arm BP (f) match

Figure 8.5: Example of the benefit of dominant set w.r.t. greedy clique search. The face is obviously not visible, but its back-projection is not null and generates cluster that are false candidates (a). The Dominant Set generates a match with highly coherent nodes only, made of the green ellipses (f); an exact graph matching algorithm instead leads to a greedy match, made of the green and the red ellipses.

In [85] it is shown that dominant sets correspond to local maximizer over the standard simplex of the quadratic function

$$f(\mathbf{x}) = \mathbf{x}^t A\mathbf{x} \tag{8.13}$$

where A is the weighted adjacency matrix of the graph (thus $A_{ij} = \omega(i, j)$). These maximizers can be found by exploiting the convergence properties of the payoff monotonic replicator dynamic

$$x_i(t + 1) = (Ax(t))_i/(x(t)^t Ax(t))$$

which is guaranteed to converge to a local maximum when the association graph is undirected and, thus, the matrix A is symmetric [89]. At convergence the value of the function f is a measure of the coherence of the extracted set; this property can be exploited in tracking when the target object disappears or is totally occluded: in such cases, any match from the labeled graph of the model to the labeled graph generated with the actual frame will yield very poor values of f and this condition can be used to automatically detect the absence of the object from the scene and therefore suspend the tracking (tests will be provided in Section 9.2). Finally, as the local maximizer found by the replicator dynamic is not guaranteed to be the global maximum, we used an enumeration strategy similar to the one presented in [90].

8.3.4 Coherence Computation

Given the association graph $Ga_{t,0}$ between G_t and G_0, our goal is to assign to each of its edges

$$\{[(u_t, v_t), (u_0, v_0)], [(w_t, z_t), (w_0, z_0)]\} \in Ea$$

a weight in the interval $[0, 1]$ which reflects the coherence between the two connected edge associations (see Fig. 8.4). This measure $\omega : Ea_{t,0} \rightarrow [0, 1]$ is the sum of several components, each referring to a specific property of the tracked object that should be consistent along the video sequence. Since different and independent properties are considered, the mis-detection of any of them (for example, due to occlusion or deformation) does not compromise the overall coherence evaluation. We define three properties that are expected to be consistent along the video sequence: *color* and *structure* with respect to the initial model, and *spatial similarity* with respect to the object tracked at the previous frame, if present.

Color-based consistency measured through cluster density and mass: "ω_d" and "ω_m". Let us define the normalized density $ND()$ and the the normalized mass $NM()$ as:

$$\begin{aligned} ND(u_t) &= \frac{D(u_t)}{\max\limits_{\forall v_t \in C_t \mid l(u_t)=l(v_t)} D(v_t)}, \\ NM(u_t) &= \frac{M(u_t)}{\max\limits_{\forall v_t \in C_t \mid l(u_t)=l(v_t)} M(v_t)} \end{aligned} \tag{8.14}$$

then we define the color-based consistency based on density ω_d and on mass ω_m, respectively as:

$$\omega_d(C_t) = \sqrt[4]{ND(u_t) \cdot ND(v_t) \cdot ND(w_t) \cdot ND(z_t)} \tag{8.15}$$

$$\omega_m(C_t) = \sqrt[4]{NM(u_t) \cdot NM(v_t) \cdot NM(w_t) \cdot NM(z_t)} \tag{8.16}$$

The clusters are defined over the back-projection that measures the color similarity of the image I_t compared to a color feature of the model: therefore the higher the density of a cluster, the higher its color similarity to the model. The densities of the four clusters are multiplied and not summed up in order to reinforce the overall $Ea_{t,0}$ color similarity. Since small clusters might show very high ω_d, the ω_m component reinforces only the $Ea_{t,0}$ that have strong masses.

Structure consistency measured through cluster sizes and inter-cluster distances: "ω_{sd}". This component reinforces the $Ea_{t,0}$ that show structural similarity with the model, i.e. cluster size variations which are supported by consistent inter-cluster distance variations. Fig. 8.6 depicts three different cases. (a) is a typical structure size reduction (for example, due to camera zoom out) that maintains consistency between area and distance variations. On the other hand, (b) and (c) depict a structure deformation that is penalized by ω_{sd}: in both cases the distance variation between top and middle clusters is not supported by a similar variation in the size of the cluster. The structure consistency measure ω_{sd} is formalized introducing the *linear area ratio*

$$lar : C_t \times C_0 \rightarrow [0, \infty), lar(u_t, u_0) = \sqrt{\frac{R(u_t)}{R(u_0)}}$$

and the *distance ratio*

$$dr : E_t \times E_0 \to [0, \infty), dr\left((u_t, v_t), (u_0, v_0)\right) = \frac{\left|\overline{P(u_t) P(v_t)}\right|}{\left|\overline{P(u_0) P(v_0)}\right|}$$

Structure consistency of $Ea_{t,0}$ is obtained when lar measures are similar to the respective dr measures, i.e. their ratio is close to 1; the consistency measure can then be obtained modelling the deviation with a Gaussian. To evenly stretch the ratio codomain from $[0, \infty)$ to $(-\infty, \infty)$, it is formally appropriate to use a logarithm. Therefore, ω_{sd} is defined as follows:

$$\begin{aligned} \omega_{sd} = & \exp \frac{-\left(Q(u) - \Delta(u,v)\right)^2}{2\sigma^2} \cdot \exp \frac{-\left(Q(v) - \Delta(u,v)\right)^2}{2\sigma^2} \cdot \\ & \exp \frac{-\left(Q(w) - \Delta(w,z)\right)^2}{2\sigma^2} \cdot \exp \frac{-\left(Q(z) - \Delta(w,z)\right)^2}{2\sigma^2} \end{aligned} \tag{8.17}$$

where

$$\begin{aligned} Q(a) &= \log\left(lar\left(a_t, a_0\right)\right), \\ \Delta(b,c) &= \log\left(dr\left((b_t, c_t), (b_0, c_0)\right)\right) \end{aligned}$$

In analogy to what is done with ω_d and ω_m, the four contributes of ω_{sd} are multiplied together and not summed up.

Structure consistency measured through cluster relative orientations: "ω_a". This component favors the maintenance of angular consistency of the $Ea_{t,0}$. Fig. 8.6 depicts two cases: regardless of the *overall* rotation of one graph compared to the other, (d) maintains the consistency of reciprocal angles of the segments, while (e) does not and is therefore penalized by ω_a. Let's introduce

$$\vartheta\left((u_t, v_t), (u_0, v_0)\right) = \arccos \frac{\overline{P(u_t) P(v_t)} \times \overline{P(u_0) P(v_0)}}{\left\|\overline{P(u_t) P(v_t)}\right\| \cdot \left\|\overline{P(u_0) P(v_0)}\right\|}$$

as the angle between the two segments connecting the centroids of the clusters, the value ω_a is defined as:

$$\omega_a = \frac{\exp\left\{m \cdot \cos\left[\vartheta\left((u_t, v_t), (u_0, v_0)\right) - \vartheta\left((w_t, z_t), (w_0, z_0)\right)\right]\right\}}{\exp\left\{m\right\}} \tag{8.18}$$

This resembles the Von Mises distribution [91], that is often used to model angular distributions.

Spatial similarity with the object at previous frame measured through overlap and rotation: "ω_o" and "ω_r". Let us consider the graph $Match_{t-1}$, which represents the tracked object at the previous frame, and the projection of its attributes over the graph G_0; the similarity components ω_o and ω_r favor the edges in $Ea_{t,0}$ that respectively maximize the area of overlap and minimize the overall graph rotations with repect to $Match_{t-1}$. In case $Match_{t-1}$ is partial/missing, these components will provide the

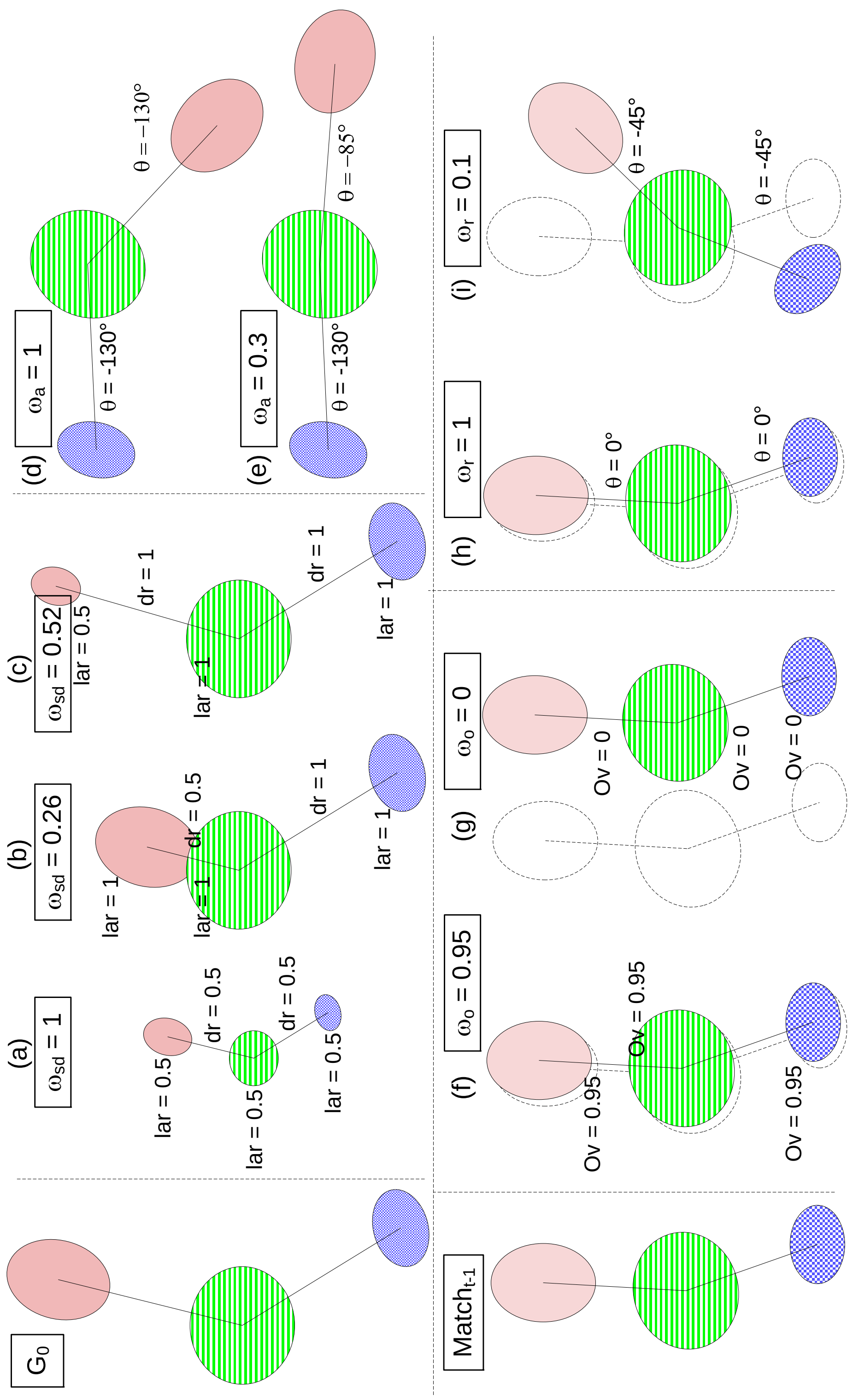


Figure 8.6: Structure consistency measure with ω_{sd} (a, b, c) and ω_a (d, e), and spatial similarity measure with ω_o (f, g) and ω_r (h, i). For the only sake of clarity and without loss of generality, the G_0 and $Match_{t-1}$ are made of only three nodes.

contribution for the detected portion of the object only. Fig. 8.6 (f,g,h,i) depicts some explanatory examples. By defining the overlap ratio Ov as:

$$Ov\left(u_t, u_{t-1}\right) = \frac{2 \cdot R\left(u_t \cap u_{t-1}\right)}{R\left(u_t\right) + R\left(u_{t-1}\right)}$$

we can calculate ω_o as:

$$\omega_o = \sqrt[4]{Ov\left(u_t, u_{t-1}\right) Ov\left(v_t, v_{t-1}\right) Ov\left(w_t, w_{t-1}\right) Ov\left(z_t, z_{t-1}\right)} \tag{8.19}$$

ω_r is defined to favor the minimization of the rotation of each single segment:

$$\omega_r = \frac{\exp\left\{m \cdot \cos\left[\vartheta\left(\left(u_t, v_t\right), \left(u_{t-1}, v_{t-1}\right)\right)\right]\right\}}{\exp\left\{m\right\}} \cdot \frac{\exp\left\{m \cdot \cos\left[\vartheta\left(\left(w_t, z_t\right), \left(w_{t-1}, z_{t-1}\right)\right)\right]\right\}}{\exp\left\{m\right\}} \tag{8.20}$$

Chapter 9

Experimental Results

9.1 Test Beds and Evaluation Methodologies

For the tests of object tracking on low-bandwidth video streaming from steady camera we propose two different surveillance scenarios: one indoor, taken at the hall of the building of our department and one outdoor, taken from a camera mounted in a public park. The first scenario is characterized by few moving people and no illumination changes, while the second is a less-controlled scenario: illumination changes and several people move in and out from the scene. The second scenario is obviously more challenging for both the video encoder and the computer-based video surveillance system. For performance analysis we recorded two videos, called VHALL and VPARK (see Tab. 9.1).

	VHALL	**VPARK**
Scenario	Indoor (static camera at building hall)	Outdoor (static camera at park)
Frame Rate	10 fps	10 fps
Resolution	QVGA	QVGA
Length	$\approx$ 420 s ($\approx$ 4200 frames)	$\approx$ 420 s ($\approx$ 4200 frames)

Table 9.1: Test bed videos for evaluation of object tracking on low-bandwidth video streaming from steady camera.

The video compression bit-rate used in Moses is set to either 5 Kpbs or 20 Kbps: in the first case, it is possible to send up to four video streams (possibly corresponding to four different cameras in a multi-camera surveillance system) on the average bandwidth of GPRS; however, this is only possible in case the video scene shows limited presence of moving objects (video motion covers less than a third of the scene) and rare illumination changes, that is the case of VHALL; if these hypotheses are not met, as in the case of VPARK, 20 Kbps video compression has been tested as well: in this condition it is possible to stream a single video over GPRS network, or four simultaneous videos over E-GPRS; the transmitted videos have resolution of QVGA - 320x240, that is large enough to make precise segmentation in automatic surveillance. The given constraints are very demanding: transmitting a video over 5 Kbps or 20 Kbps bandwidths, at QVGA resolution and with sufficient quality and frame-rate to be processed by a video surveillance system is far from being easy.

The tests measure the pixel-level segmentation and object-level tracking and are designed to highlight the performance drop due to video compression and streaming: to this aim, the ground truth is the segmentation and tracking generated on the original uncompressed video and not manually annotated data, since our intention is not to measure the generic performance of the segmentation and tracking algorithm.

For each frame, we compute the recall R and precision P in pixel-level segmentation as follows:

$$R = \frac{TP}{TP + FN} \quad ; \quad P = \frac{TP}{TP + FP}$$

where TP indicates the true positives, FP the false positives and FN the false negatives.

The object-level tracking evaluation is provided by comparing the tracking results on original and compressed videos. Specifically, the evaluation of the loss in tracking accuracy due only to video compression and streaming is performed as follows: be TR a generic track of an object and $Length(TR)$ its length in time. In compressed videos, a track can be often split in more sub-tracks with different identifiers. Defining TR^i_{orig} as the i^{th} track on the original video, and $TR^{i,j}_{cmpr}$ (with *j=1,..N*) the N distinct tracks on the compressed video in which the TR^i_{orig} was (mistakenly) split; defining a $\hat{j}$ for each TR^i_{orig} as:

$$\hat{j} \; = \; \arg\max_j Length(TR^{i,j}_{cmpr}) \tag{9.1}$$

then the accuracy ACC can be measured as follows:

$$ACC \; = \; \frac{\sum\limits_{\forall i} TR^{i,\hat{j}}_{cmpr}}{\sum\limits_{\forall i} TR^i_{orig}} \tag{9.2}$$

Since frames are lost during the transmission, we need a way to align the two videos (original and compressed) for having a correct comparison. The embedding of the frame number used to measure the latency (described in Section 5.1) is exploited also for video alignment.

The performance of the tracking with freely moving camera is obtained on the comparison of the Graph-Based (GB) approach proposed in Chapter 8 against Camshift (CS) and a *particle filtering* tracking based on color features (PF) similar to that proposed in [92]. For the sake of a fair comparison the three approaches make use of the same color space (see eq. 8.1).

In contrast to our approach, CS and PF do not exploit structural models. Therefore, we issue several independent instances of those algorithms on each single feature of the object model. They work well in standard conditions, but for the intrinsic limitation

	Video 1	**Video 2**	**Video 3 - a and b**	
Generic info	Outdoor, moving cam, 1 person	Outdoor, moving cam, 2 persons	Indoor, static cam 3 occluding people	
Model	F,T,P,LA,RA	F,T,P	F,T,H,P	F,T,LA,RA
Challenges	Severe scale vars and rotations, camouflaging bkg	Scene cuts, total obj. disapp., scale vars, rotations, camouflaging bkg	severe occlusions several color distractors	

Table 9.2: Benchmark (F=face, T=torso, H=hands, P=pants, LA,RA= left/right arm).

due to the lack of a structure model, they are likely to fail in challenging conditions, especially in the case of occlusions.

The test bed consists in this case of 3 videos and one of them (Video 3) is taken from AVSS 2007 dataset[1]. In this latter video, the tracking is applied twice, on two different target objects. Tab. 9.2 summarizes the main characteristics of the benchmark videos. In all cases the target objects are persons.

Differently from the ground truth of the first scenario, in this case we manually extracted the ground truth (with the support of the VIPER-GT tool [2]), consisting of several oriented bounding boxes, each containing a single part of the object to be tracked. Given the ground truth and the output of the tracking algorithms, it is possible to automatically compute the performance based on a set of metrics. Specifically, using the VIPER-PE tool [2], we obtained true positives, false negatives, false positives and, from them, *recall* and *precision*; all these measures were extracted both at object and pixel level. For object wise evaluation, the provided measure is a discrete value, and it is obtained comparing the boolean value "present vs. not-present" of the ground truth of an object part with the boolean value "detected vs. not-detected" of the algorithm response for the corresponding object part. For pixel wise evaluation, the provided measure is the *F-measure* defined as an aggregation of recall R and precision P:

$$F = \frac{2 \cdot R \cdot P}{R + P}$$

$F = 1$ reveals either a perfect matching or the correct tracking suspension when the object is absent from the scene. Conversely, $F = 0$ reveals either a total failure or the detection of an object when this is not present. The pixel-wise recall and precision, which constitute the F-measure, are computed aggregating together the pixel measures performed separately on each single model class.

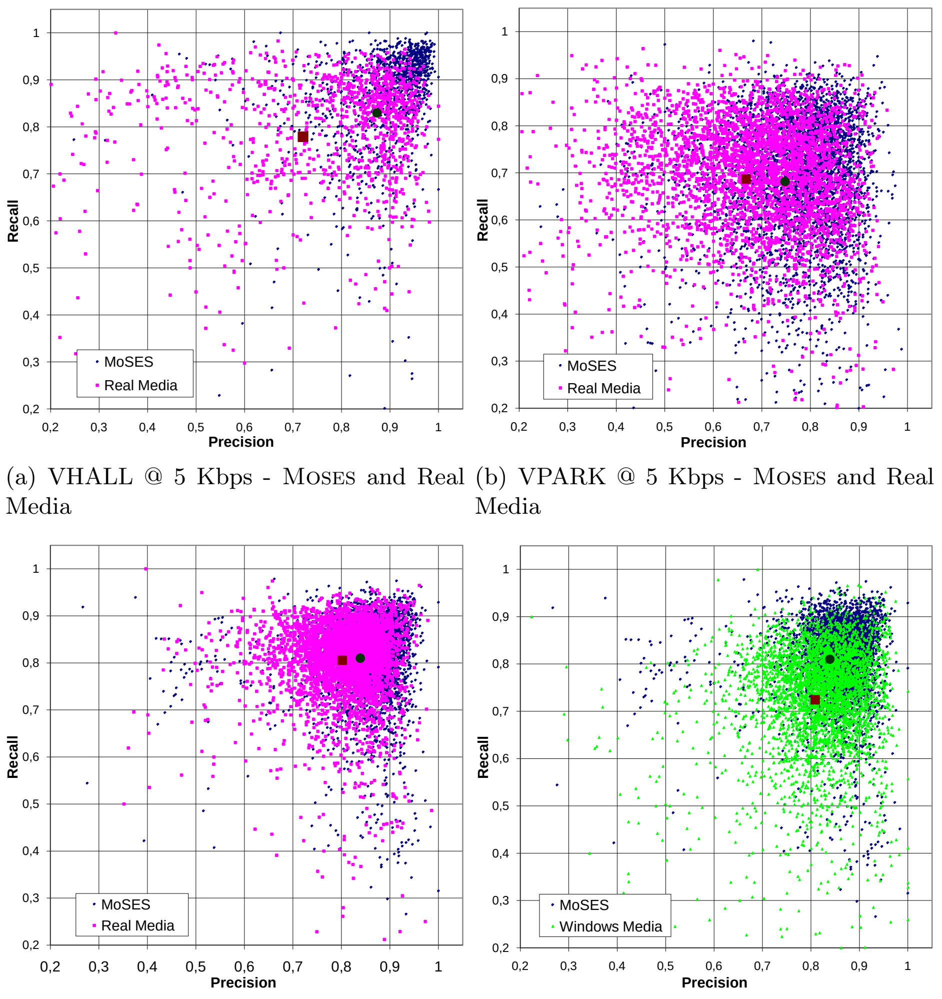
9.2 Results and Evaluations

The pixel-wise evaluation of segmentation on low-bandwidth video streaming from steady camera are presented in Fig. 9.1. Each dot represents the recall-precision of a single video frame. Obviously, the closer the points are to the upper-right corner (corresponding to $R = 1$ and $P = 1$) the higher the accuracy is. The graphs also report the average recall and precision, represented by a green circle (Moses) and brown square (Real Media and Windows Media). The average and variance of recall and precision,

[1]URL ftp://motinas.elec.qmul.ac.uk/pub/multi_face

[2]URL: viper-toolkit.sourceforge.net/

computed on the correctly received frames only, are summarized in Tab. 9.3. This table also shows the percentage of frame losses due to the strong compression rates.



(a) VHALL @ 5 Kbps - MoSES and Real Media

(b) VPARK @ 5 Kbps - MoSES and Real Media

(c) VPARK @ 20 Kbps - MoSES and Real Media

(d) VPARK @ 20 Kbps - MoSES and Windows Media

Figure 9.1: Recall vs Precision for pixel-level segmentation over VHALL and VPARK at QVGA resolution.

We initially considered the hardest case in terms of bandwidth, by supposing to send four video streams over GPRS, coding each video at 5 Kbps. As a comparison, we tested both MoSES and Real Media. Windows Media and VLC were unable to encode QVGA video at such a low bandwidth.

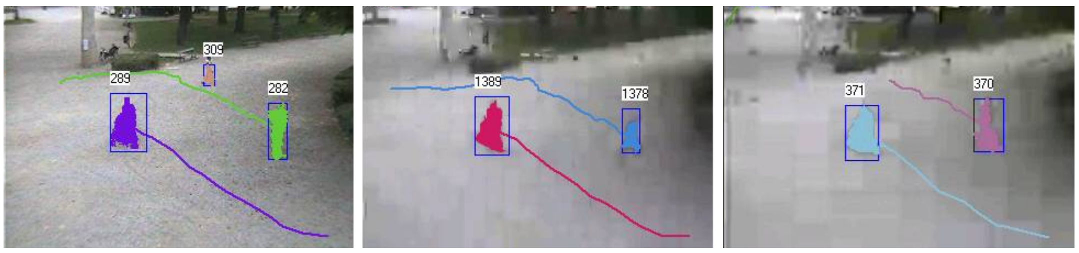
The graph in Fig. 9.1(a) shows precision and recall for the VHALL: even with such a limited bandwidth, the segmentation based on MoSES streaming is very close to the one obtained on the original video. This result does not hold in the case of the outdoor sequence of the VPARK (Fig. 9.1(b)): in fact, the extensive presence of moving objects and the frequent illumination changes make the compression less effective; in this case the average recall of MoSES is less than 70% and the precision only approximately 75% (see Tab. 9.3). Thus, we also performed a test over EDGE-GPRS using 20 Kbps for

	% of lost frames	Segmentation				Tracking	
		Recall		Precision		# of Objs	Tracking Accuracy
		avg.	std.dev.	avg.	std.dev.		
Moses, VHALL@5Kbps	0.41%	82.95%	1.99%	87.31%	1.40%	29	96.51%
RealM, VHALL@5Kbps	8.87%	77.90%	1.63%	72.0%	2.54%	29	81.32%
Moses, VPARK@5Kbps	0.11%	68.15%	1.46%	74.8%	1.68%	49	89.11%
RealM, VPARK@5Kbps	14.60%	68.67%	1.35%	66.8%	1.91%	49	67.95%
Moses, VPARK@20Kbps	0.34%	81.00%	0.98%	83.9%	1.40%	49	91.91%
RealM, VPARK@20Kbps	0.02%	80.53%	0.93%	80.2%	1.08%	49	91.83%
WindM, VPARK@20Kbps	1.81%	72.42%	1.32%	80.9%	1.18%	49	89.87%

Table 9.3: Segmentation and tracking accuracy over VHALL and VPARK at QVGA resolution.

each video stream. Windows Media supports this bitrate (VLC still does not), and it is then added to the comparative tests (Fig. 9.1(c) and Fig. 9.1(d)). Real Media shows a better recall on Windows Media, but has similar precision. However, Moses performs better than both the compared systems in almost all working conditions, on recall, precision and frame losses.

Tab. 9.3 summarizes also the results for object-level tracking. It is straightforward that with 20 Kbps the performance of the tracking (whatever system is used) is not strongly affected, having approximately 90% of accuracy compared to the tracking on the original video. Instead, when the bitrate falls to 5 Kbps, only Moses is able to maintain reasonable performances. The tracking accuracy on the compressed video depends not only on segmentation accuracy but also on the frame loss rate. The tracking of Real Media on the VPARK at 5 kpbs strongly suffers from the high frame loss rate (14.60%), that is concentrated in the portion of the video with higher motion. A sample of the tracking in the park video sequence at 5 Kbps is shown in Fig. 9.2: the tracking consistency is visually represented by the superimposed trajectory of the objects.



(a) (b) (c)

Figure 9.2: Snapshots showing the video tracking obtained with Sakbot on the Park sequence: (a) original, (b) Moses @ 5 Kbps, (c) Real Media @ 5 Kbps. The tracking numbers on the objects are just sequential IDs and do not need to be consistent between one video and the others.

Regarding the tracking with freely moving camera, the pixel-wise evaluation is shown in Fig. 9.4, where the frame-by-frame F-measure is plotted. On the top part of each measurement plot, a colored bar gives a schematic representation of the challenges along the time (e.g. zooming in/out, rotation, occlusion). The legend is reported in Fig. 9.3.

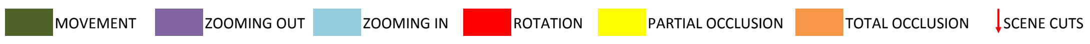


Figure 9.3: Legend of the timeline shown on the top of each video measurement in Fig. 9.4 and 9.5.

The object-wise evaluation is shown in Fig. 9.5 (refer again to time-line legend of Fig. 9.3). Each object model class (face, torso, etc.) is represented by a block of 4 colored lines: the black line represents the ground truth, showing the presence/absence of the object class; the colored lines represent the detections produced with the three approaches (cyan for CS, green for PF and magenta for GB). In case the model class is present and the tracking algorithm detects a wrong match for that class, the colored line is absent (e.g. Fig. 9.5(a) in GB, frame 43 of Video 1, right arm). In case the model class is absent and the tracking algorithms still detects a match (false positive), a colored line is present (e.g. Fig. 9.5(a) in CS and PF, frame 200 on Video 1, face). Tab. 9.4 reports the summary of the performance on the benchmark videos. The three original video sequences, the four post-processed videos with our graph based tracking and the four ground truths in VIPER XML meta-data can be downloaded from the author's home page [3].

Since Video 1 does not contain severe occlusions, scene cuts or object disappearances, the structural model of GB does not significantly increases the performance compared to CS or PF, with exception of frames 199 and 231, when the face exits from the view: our approach correctly suspends the face tracking to resume it when it reappears, whereas the other approaches fail (see Fig. 9.5(a), Video 1, 5th block of measurements ("Face - Model 0"), around frame 200: GB tracking (magenta bar) behaves coherently with the ground truth (black bar), while CS and PF completely fail). The correct suspension of the tracking of the only class "face" is an example of the advantage of using dominant sets instead of greedy clique search, that would have generated instead a face false-positive detection in order to force a complete subgraph isomorphism.

In Video 2, the sharp scene cuts (frames 156 and 231) and the full object disappearance (frame 156) make the performance of CS and PF drop severely. Our approach instead is not affected at all, suspending the tracking when necessary and resuming it as soon as the structure of the model is found again (in Fig. 9.5(a) the detection colored bars of GB are correctly suspended at the scene cut and resumed a few frames after the re-appearance of the object; in Fig. 9.4(b) the F measure to 1 demonstrates the correct tracking suspension. As soon as the object reappears, the F measure of GB goes to 0 for a short time since the algorithm takes a few frames to locate the structure and resume the tracking.

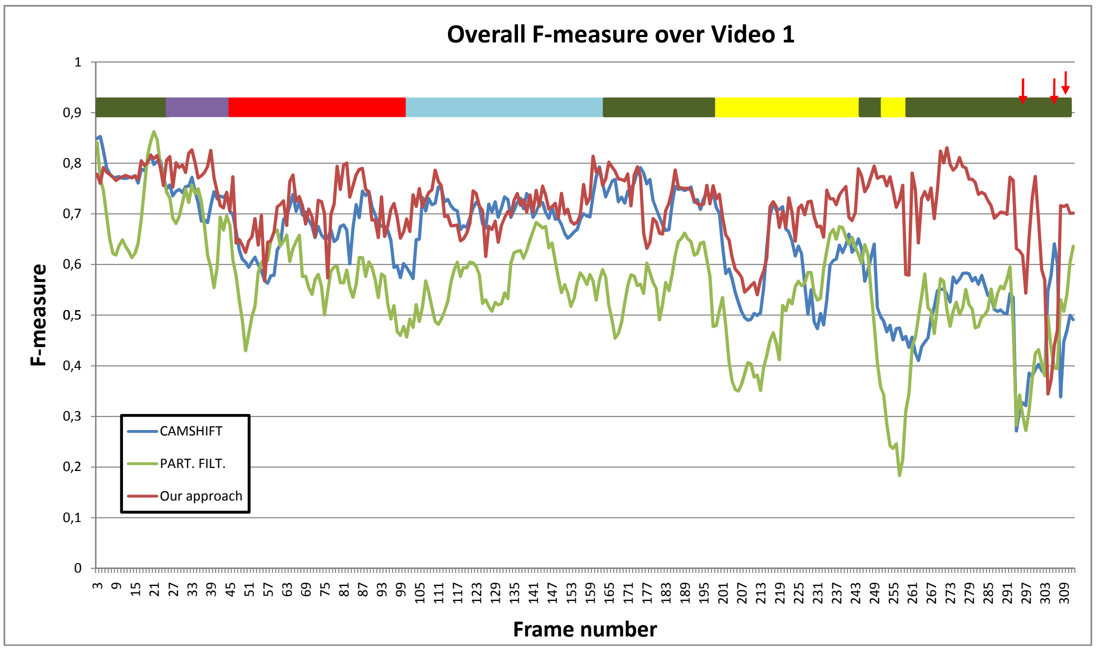
Video 3 is a static camera sequence but the persons occlude each other several times and the scene is full of color distractors (e.g. the several skin-colored regions of faces and arms, the two blue jeans, the dark t-shirt of the person on the right and the dark cupboard on the back wall). In such conditions the use of a structural model is determinant to have a successful tracking. In the specific, from Fig. 9.5(b) it is possible to appreciate that, in Video 3-a, GB approach suspends the tracking during the several total occlusions, while PF and (especially) CS suffer (see F-measure in Fig. 9.4(c)). In Video 3-b, a severe occlusion around frame 245 hides 3 object classes on 4: the GB approach suspends the tracking of the whole object because the structural coherency

[3]URL: imagelab.ing.unimore.it/imagelab/~gualdi/

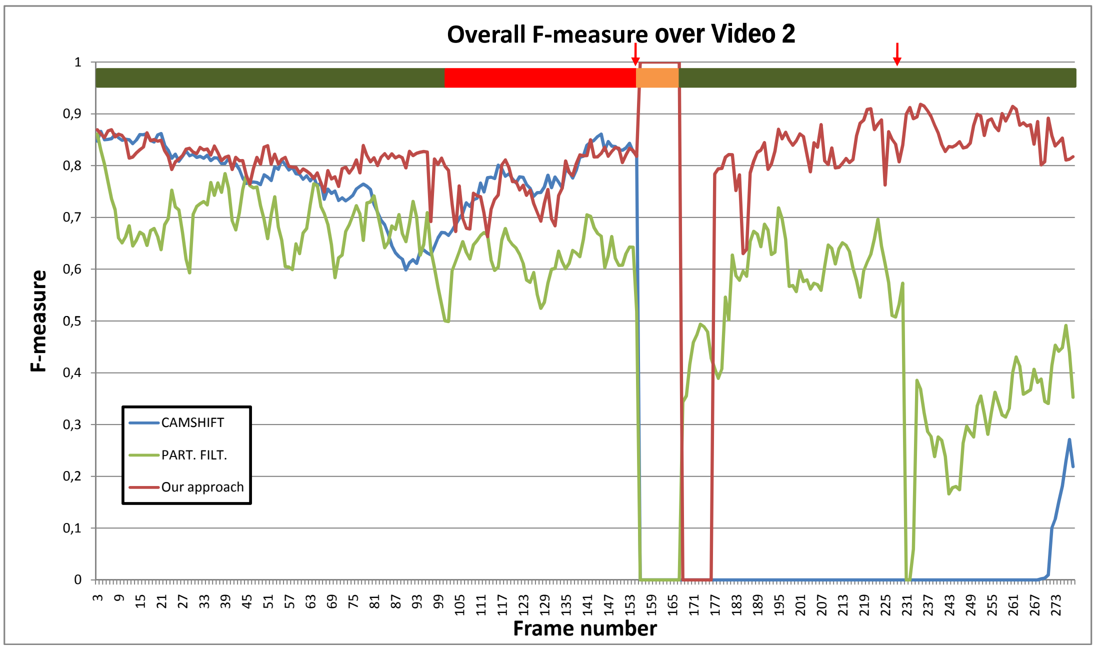
Object level	**Recall (%)**			**Precision (%)**			**F-measure (%)**		
	CS	PF	GB	CS	PF	GB	CS	PF	GB
Video 1	96,72	96,41	99,24	87,66	74,52	95,31	91,97	84,07	97,24
Video 2	92,83	95,99	100,00	66,86	89,00	97,20	77,73	92,36	98,58
Video 3-a	30,25	72,46	97,87	12,94	69,89	89,58	18,13	71,15	93,54
Video 3-b	88,59	86,28	98,13	91,55	79,39	87,65	90,05	82,69	92,59
Pixel level	**Recall (%)**			**Precision (%)**			**F-measure (%)**		
	CS	PF	GB	CS	PF	GB	CS	PF	GB
Video 1	84,54	66,92	85,71	52,77	49,80	62,14	64,09	55,41	71,16
Video 2	53,20	49,42	84,87	36,85	64,64	76,20	43,34	55,50	79,78
Video 3-a	7,93	30,68	65,26	4,76	36,93	53,42	5,67	32,07	57,67
Video 3-b	65,06	44,20	71,95	69,67	74,30	63,65	64,66	52,74	66,90

Table 9.4: Summary of the performance.

is not found. Conversely CS and PF do not account for structure and continue the tracking of the only class that is still visible, yielding higher F-measures.

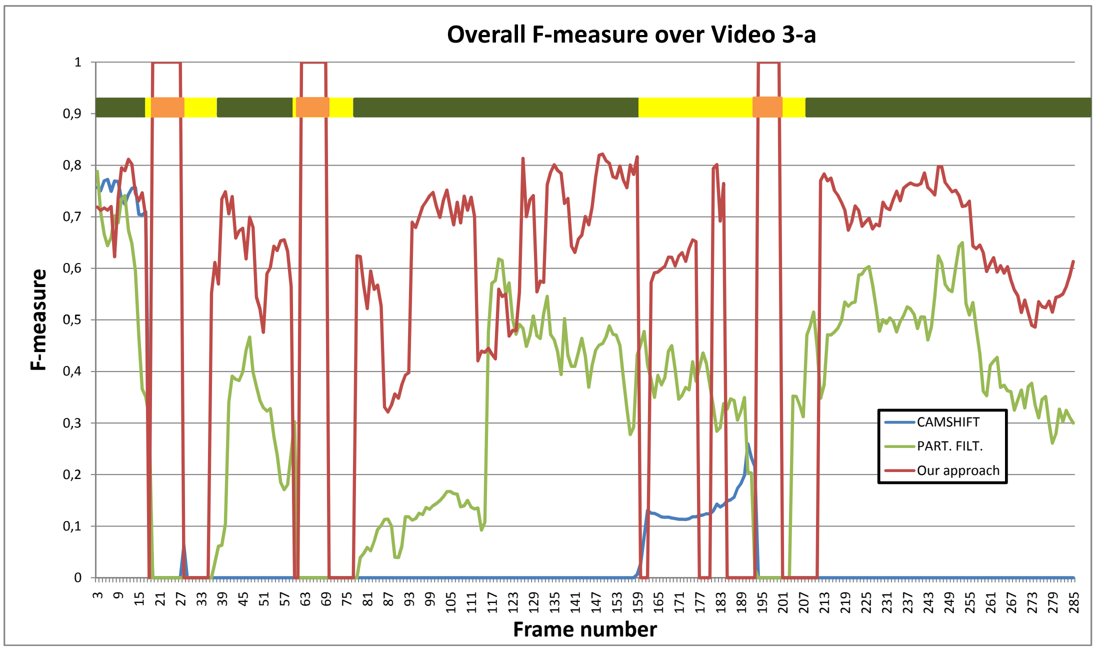


(a)

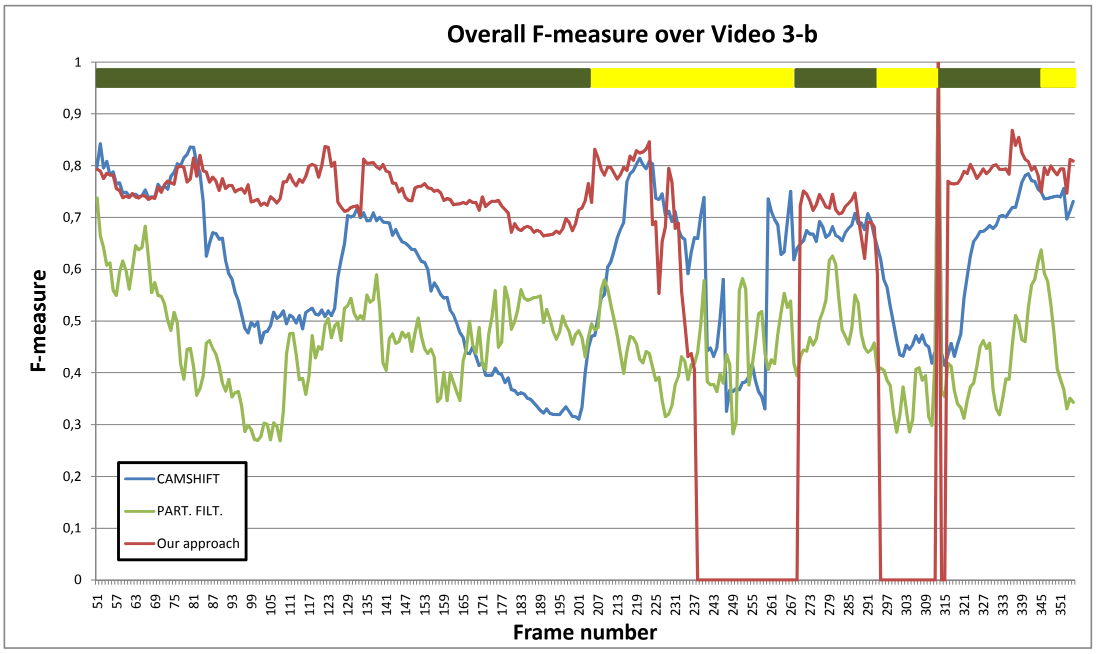


(b)

Figure 9.4: Pixel-level measure of performance.

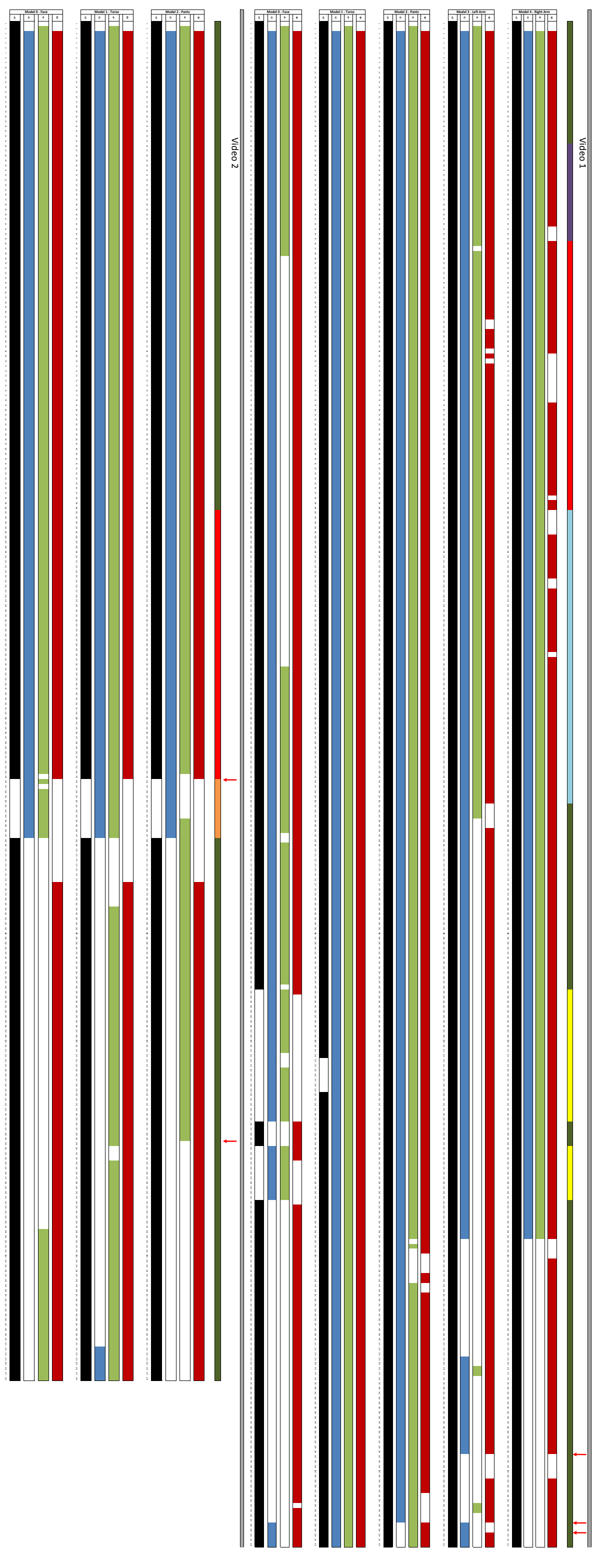


(c)



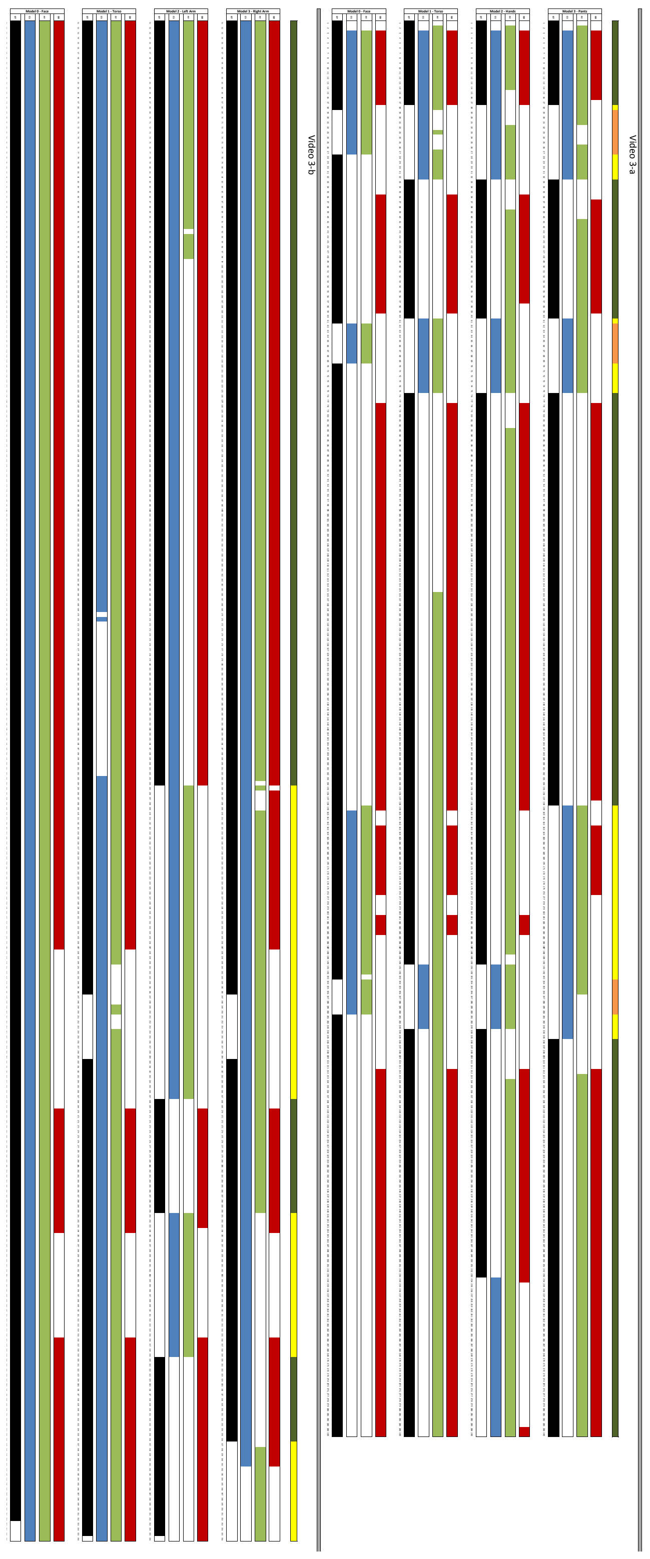
(d)

Figure 9.4: Pixel-level measure of performance.



(a)

Figure 9.5: Object-level measure of performance on Video 1 and 2. The figure is best viewed in the electronic version.



(b)

Figure 9.5: Object-level measure of performance on Video 3a and 3b. The figure is best viewed in the electronic version.

Part III

People Surveillance in Mobile Contexts

Chapter 10

Introduction

There are many scenarios of mobile video surveillance where obtaining object tracking can be not only extremely challenging (due to type and degree of camera motion, number of objects to track, visual occlusions and clutter, etc.), but also not necessary. Tracking is indeed a very rich piece of information, however for many surveillance purposes just object detection is needed: this is the case of thesis part III, that deals with appearance-based object detection in mobile surveillance scenarios. The appearance is a feature that can be used regardless of the state of motion that lies behind the generating object, and this condition is very useful in mobile contexts. In the specific, we focus our attention on the detection of pedestrians (i.e. standing people), for two reasons: on one side, pedestrian detection is actually a very active research topic and on the other, people are the main objects of interest in surveillance and security. Nevertheless, since the pedestrian is an extremely articulated object equipped with a significant number of degrees of freedom, it is clear that what is proposed for pedestrians can be easily extended to other classes of objects.

Real-world outdoor scenes are typically very challenging even to modern state of the art pedestrian classifiers. For this reason, for example, a complete benchmark infrastructure for training and testing of pedestrian classifiers, provided with hours of annotated video data recorded in real-world scenarios and with off-the-shelf procedures for assessing pedestrian detection performance has been recently proposed [2]. Given this challenging context, after having chosen and implemented a state of the art classifier, we propose a set of innovative methods to tackle two different subjects related to pedestrian detection: classification performance and detection speed. Regarding the first, we propose improvements over the original proposal and then introduce additional modifications to make it suitable also for the detection of sub-parts of pedestrian, specifically tackling, but not being limited to, the head detection. On the second, since high accuracy classifiers can be quite demanding on CPU cycles, we propose methods to reduce the computational load of the detection step.

The methods proposed in this thesis part are effectively used in the prototype described in Appendix A for video security aimed at the automatic detection of construction workers that do not wear the protective helmet.

Publications related to Part III: [j,m]; see the list of author's publications, page 147.

10.1 On Pedestrian Classifiers

Pedestrian detection in images and videos is a problem that has been strongly addressed in computer vision within the past five years. The task is made harder in complex environments where standard approaches based on foreground segmentation and appearance-based tracking become unreliable because of occlusions, moving cameras or challenging illumination conditions. It is therefore necessary to adopt probabilistic models and sophisticated pattern recognition techniques to handle uncertainty and noise.

The approaches proposed in the literature can be summarized in two main classes [93]. The first one makes use of a model of the human body by looking for body parts in the image and then imposing certain geometrical constraints on them [35, 94–96]. One relevant limitation of these approaches is that they require a sufficiently-high image resolution for detecting body parts, and this is not appropriate in many surveillance contexts, especially when the camera sensors overlook long views.

The second class of proposals, often called holistic approaches, is based on applying a full-body human classifier over all possible sub windows of a given image [1,97–103]. A wide range of features has been proposed; among them, Haar wavelets [104], histogram of gradients (HoG) [105], a combination of the two [106], histograms of differential optical flows [102], shapelet [107], covariance descriptors [1], etc. Over these features, the classifiers most typically used are SVMs (typically linear, as in [102] or histogram intersection kernels [108] SVMs) or boosting algorithms (like AdaBoost [101], LogitBoost [1, 109], MPL Boost [110], etc.) usually combined in cascades.

A lot of efforts have been spent to propose classifiers that minimize the computational load, maintaining similar accuracy of state of the art methods. The compelling reason is the following: even if the single classification step is by itself fairly fast (i.e. a millisecond of CPU time, or a fraction of it), the procedure is to be repeated several times (typically tens of thousands or more) in order to obtain object detection over a whole image. Some of the optimizations propose to use cascade classifiers with features that were originally proposed to work on slower classifiers (e.g. boosting with HOG features [103]); a recent paper [98] proposes to employ both intensity-based (those used in [101]) and gradient-based features (called edge orientation histogram, EOG) with Real AdaBoost: instead of using the standard boosted cascade, a novel cascaded structure is proposed in which both stage-wise classification and inter-stage cross-reference information are used. Yao *et al.*in [97] speeds up the covariance descriptor classifier of [1] through a smart use of covariance descriptors (i.e. considering subsets of the matrices) and embedding motion information in the descriptor itself. Another optimized version of it is in [111], where the inverse of an exponential mapping, used to map covariance matrices into the euclidean space of symmetric matrices, is avoided.

10.2 Pedestrian Classification with Covariance Descriptors

As founding block of our studies we adopt the pedestrian classifier based on covariance descriptors proposed by Tuzel *et al.* [1], for three main reasons: first, it is demonstrated to perform better in per-window classification with respect to the popular HoG SVMs [105]; second, it is based on a rejection cascade of boosting classifier: this architecture benefits of the property that a very reduced portion of the rejection cascade is used when classifying those patches whose appearance strongly differs from the trained model (reducing therefore the computational load); third, the covariance descriptor is very flexible and can be modeled according to the different application contexts: for example, it can be successfully used for region matching and texture classification [112]).

Without digging in details, the classifier is based on a rejection cascade of Logit-Boost classifiers (the strong classifiers), each composed of a sequence of logistic regressors (the weak classifiers). The original Logit-Boost classifier [109] is modified to account for the fact that covariance matrices do not lie on euclidean space but on Riemannian manifolds. Each logistic regressor is trained to best separate the covariance descriptors that are computed on *randomly sampled sub-windows* of the positive (pedestrian) or negative (non-pedestrian) training images.

Given an input image I and the following 8-dimensional set F of features (defined over each pixel of I):

$$F = \left[x, y, |I_x|, |I_y|, \sqrt{I_x^2 + I_y^2}, |I_{xx}|, |I_{yy}|, \arctan \frac{|I_y|}{|I_x|} \right]^T \tag{10.1}$$

where x and y are the pixel coordinates, I_x,I_y and I_{xx},I_{yy} are respectively the first and the second-order derivatives of the image, it is then possible to compute the covariance matrix of the set of features F for any axis-oriented rectangular patch of I. Regardless of the specific composition of F, this matrix is referred as “covariance descriptor”: as aforementioned, it is proved to be a very informative descriptor for several computer vision tasks and, moreover, extremely well suited for pedestrian classification. Furthermore, since the sub-windows associated with the logistic regressors are rectangular and axis-oriented, the covariance descriptor can be efficiently computed using integral images [112].

The main issue related with covariance matrix approaches is that they lie on a Riemannian manifold, and in order to apply any traditional classifier in a successful manner, it is necessary to map the manifold over an Euclidean space. A detection procedure over a single patch involves the mapping of several covariance matrices (approx. 350) onto the Euclidean space via the inverse of the exponential map [1]:

$$\log_\mu(Y) = \mu^{\frac{1}{2}} \log \left(\mu^{-\frac{1}{2}} Y \mu^{\frac{1}{2}} \right) \mu^{\frac{1}{2}} \tag{10.2}$$

This manifold specific operator maps a covariance matrix from the Riemannian manifold to the Euclidean space of symmetric matrices, defined as the space tangent to the Riemannian manifold in μ, that is the weighted mean of the covariance matrix of the positive training samples. Each matrix logarithm operation (equation 10.2) requires at least one SVD of an 8x8 matrix, and such operation is known to be computationally demanding.

The original algorithm in [1] proposes to train a cascade of LogitBoost classifier made of 30 stages, and exploiting logistic regressors as weak classifiers. They also suggest to use the well known INRIA pedestrian database [105] for training data. We implemented, trained and tested this precise configuration, that we name as “INRIA-based detector” for the rest of the thesis part.

10.3 Improving Object Classification Accuracy

Although the classifier introduced in the last section provides remarkable human detection performance in per-window measurements, when applied to pedestrian detection in images and video, it is likely to produce false positives and negatives, due to visual distractors, clutter, occlusions or unusual poses: actually, this is common to all classifiers.

Therefore in Section 11.1, we propose to take into account the complexity of the observed scenes to boost the classification accuracy. This is obtained replacing the final stages of the rejection cascade, that is trained for generic human detection, with specific and dedicated cascades trained on positive and negative samples (semi) automatically acquired from that specific view only.

When the covariance descriptor classifier is applied to objects with non-rectangular shape (e.g. holes, heads, wheels, etc.), the performances in terms of classification accuracy degrade due to the inclusion of non-discriminative pixels within the rectangular patches associated to the logistic regressors. Hence, in Section 11.2 we propose to make the classifier suitable to the case of circular and concave features by using a polar representation which unrolls the slice of an annulus in a rectangular patch. This change of representation allows the direct exploitation of covariance descriptors for concave circular patches and makes the classifier suitable, for instance, for the detection of the head of the pedestrians.

Locating circles in images has been deeply explored in the literature, for both robotic or industrial applications [113], and 3D object reconstruction or traffic sign recognition [114]. All the proposed methods present two main shortcomings for our purposes: first, they rely on fitting the pixel values or edge points with a certain parametric function. This can be difficult to generalize and is heavily affected by the unfavorable correlation between strong false positives and weak true positives (weak signal problem). This is a typical limit of parametric approaches such as Hough transforms. For this reason, [114] proposes to measures a curve's distinctiveness through a one-parameter family of curves, in order to gain in accuracy. The second shortcoming regards the computational complexity. Most of these methods are highly time consuming, tackling the problem as an optimized search in highly dimensional spaces. In [113] the problem is formulated as a maximum likelihood estimator and the method is proved to be fast and accurate also in the case of occlusions, but it relies on the good extraction of the points describing the curves. Differently from all these works, the approach we propose exploits appearance-based features, making it suitable also in case where the objects to classify are not easily modeled by parametric curves or precise edges cannot be extracted due to the complexity of the scenes.

In Section 11.3 we show that the computation of covariance descriptors using multi-spectral (color) image derivatives yields more accurate results than using plain gray-level derivatives, as proposed in [1]. Several works demonstrate that exploiting color in addition to luminance yields favorable results for image derivatives and for other related tasks, like edge detection or texture classification. A seminal work is [115], where the author proposes a method for computing a single image derivative from a RGB image. In [116], color is exploited for (sub-pixel) edge detection defining a local directional measure of multi-spectral contrast. In [117] an operator called compass finds edges without the assumptions that regions on either side have constant color. Eventually, in [118] describes the extension of the Mumford-Shah functional [119] to color images. Without penetrating in the details of this research topic, we just verify that the use of simple multi-dimensional gradient methods can improve performance. The suitability of the proposed solutions on polar images and on multi-spectral gradients are verified in Chapter 14 on two case studies, namely head detection and polymer classification.

10.4 Optimizing Object Detection Approaches

The two most exploited approaches to perform object detections within images are sliding window (e.g. [1, 101, 105]) and Hough paradigm (e.g. [120]). The first paradigm is based on the idea of passing to the window-based classifier all possible sub windows of an image; since all classifiers tend to trigger multiple detections over a single object, the detection step is typically followed by a non-maximal suppression. In the second, instead, each feature is first extracted over the image, and then it casts votes, so that the objects to detect will accumulate votes from all the features, in a Hough fashion.

Focusing on sliding window, the approach has the drawback of brute force methods, that is the high computational load due to the number of windows to check, that grows quadratically in each dimension to span over [121]; the state space of the sliding window typically consists of image coordinates and scale, but may also contain aspect ratio and rotation. The proposals to decrease its computational burden are basically the following:

1. prune the set of sliding windows exploiting other cues (e.g. motion [97], depth [122], geometry and perspective [123], or whatever cue that is different from the appearance cue used by the detector itself),
2. hardware optimized implementation using GPUs [124],
3. efficient sub-window space exploration [121, 125].

In Chapters 12 and 13, we propose new methods to speed up the detection and we purposely avoid approaches (2) and (3): the former, because we think that the load of object detection can be reduced as an algorithm itself, beyond the use of massive multi processors architectures; the latter, because those methods search for the global maximum of the function at a time, but we do not know a priori how many objects are going to be present in the image.

Specifically in Chapter 12 we propose a two-fold method that belongs to the class (1), and that exploits motion and perspective to reduce the computational burden of the sliding window approach. *As a first contribution*, we use motion information as focus of attention for human detection; differently from other approaches we avoid to use motion-based people segmentation since it is not possible to rely on clear segmentation in presence of strong visual clutter or with moving cameras. For this same reason, we purposely avoid the approach proposed in [97], that describes how to include motion information as a feature in the covariance descriptors for classification purposes: this approach implies a robust detection of the motion. Moreover, motion observation can be view-dependent, making the training set less expandable to different scenes and view points. Our proposal instead is to drive the search for humans on the areas where motion is present or was present in the recent past. This provides a good trade-off between searching all over the image (that has the drawback of heavy computational load) and limiting the search to current moving regions only (that hinders to detect still people in the scene). Additionally, even if the motion information is not extremely accurate, the detection recall is not going to be affected. *As a second contribution*, we use a rough estimation of the scene perspective to reduce false detections. More specifically, since we consider mobile video surveillance scenarios where the geometry of the observed scene is rather dynamic, we developed an automatic calibration procedure which estimates (through RANSAC) the reasonable height of standing people in function of their position in the scene. This estimate is used to discard the windows that are significantly out of scale. The use of RANSAC allows the system to use an unsupervised approach for the perspective learning. The paper in [123] presents a very nice statistical framework to model the relationship between objects and scene geometry, by modeling the interdependences between objects, surface orientations, and camera viewpoints. This framework effectively evaluates the correct perspective of objects in the scene, in a way similar to our proposal. Both the aforementioned approach can be effectively used to make the sliding window approach more suitable for real-time applications. Moreover, they have the positive effect to increase detection performance in terms of precision. Detailed experimental results will be provided in Chapter 14.

Beside the exploitation of additional cues (motion, perspective, etc.) to speed up the computation of the sliding window detection, in Chapter 13 we introduce an innovative technique that does not belong to any of the three afore-listed classes, since it *replaces* the sliding window paradigm: indeed we claim that, by exploiting only the features used by the classifier itself, it is possible to drive a more efficient exploration of the state space of the sliding window, in order to effectively detect pedestrians with a much lower computational effort. Additionally, we demonstrate that the data obtained by such method can be easily plugged into a Bayesian recursive filter, in order to exploit the temporal coherency of the pedestrians in videos. It is not our intention to challenge pedestrian classification techniques, neither to confront with pedestrian detectors techniques which exploit multiple cues (i.e. motion, depth or geometry) in order to boost the performance of the underlying classifier.

Our method uses Monte Carlo sampling to provide an incremental estimation of a likelihood function and our innovative contribution is the use of the response of the rejection cascade to build such function. In practice, this response (given by the number of boosting classifiers that successfully detect the object) is used to increasingly draw samples on the areas where the pedestrians are potentially present, distributing the new samples among the potential targets and avoiding to spend search time over other image regions; this procedure is known as *density interpolation* [126]. When working with videos, this likelihood function is then plugged into a Bayesian recursive context, through a particle filter. Several works [69,126] have used particle filter to track a single object also in challenging situations. However, one important shortcoming of particle filters is that they are poor when the target distribution is multi-modal. Multiple modes can be generated either by ambiguity on the object state due to insufficient measurements or clutter, or by measurements coming from multiple objects. Standard particle filtering is not suitable for tracking multiple objects since it may happens that all the particles quickly migrate to one of the modes, subsequently discarding all other modes (problem of *sample depletion*). For this reason specific particle filtering techniques to handle multi-target tracking are proposed [127–129]. Besides this, another important feature is the ability to handle the entrance and exit of objects, i.e. to manage a varying number of objects. This is usually done by first issuing a detection algorithm and then assigning multiple particle filters to the obtained detections. The method has two shortcomings: first, it requires to periodically perform detection and deciding when to do it is a critical trade-off between performance and possible loss of targets; second, this approach can quickly degrade performance with the increase of the number of targets (the number of particles linearly increase with the number of objects). An alternative approach is to use a particle filter capable to work with multi-modal posteriori, such as the *mixture particle filter* (MPF) proposed by Vermaak *et al.* [130] in which the different targets corresponds to the modes of the mixture pdf. This approach has been further refined in the Boosted Particle Filter (BPF) introduced by Okuma *et al.* [129], where a cascaded AdaBoost algorithm are used to guide the particle filter. The proposal distribution is a mixture model that incorporates information from both the AdaBoost and the dynamical model of the tracked objects. Our proposal, based on [126], is able to deal with multiple targets maintaining a fixed number of particles, but in addition it is able to handle the entrance and the exit of objects without using additional detection algorithms.

Chapter 11

Improving Covariance Descriptor Classification

In this chapter we propose three methods, namely relevance feedback, polar image transformation and multi spectral image derivatives, to improve the performance of the LogitBoost classifier proposed in [1]. The methods were developed within the scenario of constriction working sites, that embodies an example of mobile video surveillance system since it is distributed and composed of multiple cameras that are moved from time to time, to account for the morphological changes in the structure of the construction site. This environment is typically very cluttered, with several people and machineries moving all around and undoubtedly constitutes a very challenging scenario. Details of a case study are provided in Appendix A. Regardless of the specificity of the test bed, the proposed methods are generic and can be easily extended to many other mobile scenarios.

11.1 Additional Learning with Relevance Feedback

We applied the INRIA-based detector (see Section 10.2) in construction working scenarios; confirming the robustness of the approach, the detection rates are good, especially on positive detections; in fact the appearance of human silhouettes in this specific context does not differ much from the one coming from a more generic scenario: at the low resolution used to observe the humans in our working system (just a few pixels in width and height), the only noticeable difference is the presence of a protective helmet: being the features mainly based on image derivatives (neither color, nor luminance), the visual appearance of a helmet does not differ from human hair or just a cap. For this reason, the miss rate reported in [1] is approximately confirmed in our context also.

On the opposite, the performance on negative samples is seriously challenged: as a matter of facts, the extremely cluttered scenarios of construction sites produce a rate of false positives that is higher than the one produced on the INRIA test bed (the false positives per window, FPPW, are increased by a factor of 6 approximately). We tackle this problem training a few extra cascades in a relevance feedback (RF from now on) fashion [131], that will replace a portion of the rejection cascade in the INRIA-based detector; it is important to notice here that we do not want to replace the whole training

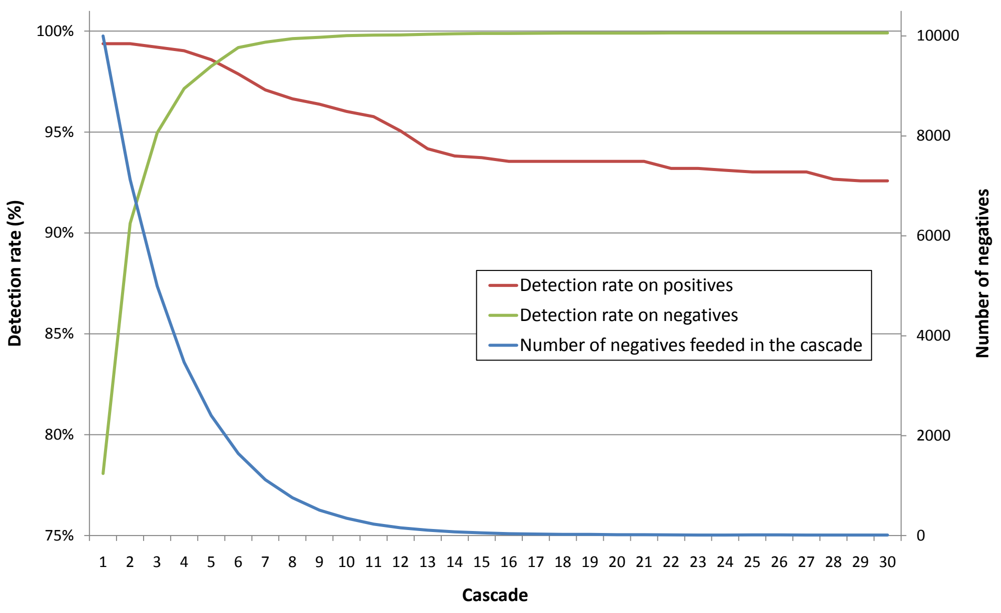


Figure 11.1: Asymptotic behavior of the last cascades of the LogitBoost classifier based on covariance descriptors, trained on INRIA pedestrian dataset. The rates are expressed as rate of detection (rejection) for positive (negative) test samples.

of the INRIA-based detector, since it provides by itself remarkable performance even in our challenging scenario, but we adapt and improve a few stages of the cascaded detector with additional scenario- and view-dependent video data in a manner that will be compliant with the requirements of mobile surveillance systems.

The performance of the 30-cascades of the INRIA-based detector is asymptotic (see Fig. 11.1), resulting in limited rejecting abilities of the last cascades; therefore it is reasonable re-train them, using additional RF training data, that is generated using a twofold procedure:

1. *implicit (assessment free) RF feeding*: the pool of negative examples is enriched with background images from our construction working site that are automatically extracted using Sakbot (see Chapter 7). In order to be sure of the absence of humans inside the scene, we accept background images only when no motion is detected during a time gap of a few minutes (typically 10). In our test bed of videos from construction working sites, we verified that it is feasible to extract several person-free images on daily basis;

2. *explicit (with assessment) RF feeding*: from the pool of detections obtained with the INRIA-based detector (run on a few videos from a specific view), an user provides an assessment, extracting some true positives and false positives, that will respectively enrich the positive and negative training data.

Differently from the implicit feeding, that produces only generic negative training data, it is very important to note that the explicit feeding enriches also the positive training data: the additional negative samples extracted here have a strongly-informative content, being all of them (false) positive detections according to the INRIA-based detector. Therefore, any additional cascade learnt using these specific data as negative training data will generate a hard split inside the space of positive detections modeled so far. By reinforcing the positive training data with samples (snapshots of workers) taken from the same context of the false positives, the cascade learner can better model the logistic function of the LogitBoost.

These re-trained cascades using RF training data are placed at the final stages of the rejection cascade process of the INRIA-based detector; therefore it is not possible, at this point, to raise the performance over the true-detections (what was wrongly rejected by the first cascades cannot be recovered then, because of the nature of rejection cascades), but conversely it is still possible to act in a strong manner on the reduction of the false-detections, that is exactly our goal.

The learning time of a complete 30-cascades LogitBoost detector on Riemannian manifolds is very long (in the order of days of computation); since the cardinality of the miss-classified negative samples decreases exponentially at each cascade (see Fig. 11.1), the longest time is spent on learning the first cascades. For this same reason, the re-training of the last stages of the LogitBoost classifier using RF training data is computed in a sufficiently short time (in the order of a couple hours of computation), so that it can be considered as a symptomatic operation: in other words the additional cascades based on relevance feedback data could be removed and recomputed whenever the performance of the system begins to degrade (e.g. when the viewed scene is subject to remarkable changes, that is a common situation in construction working sites).

In case it is not possible to provide explicit assessment, additional learning on implicit data only can be performed, making the system totally autonomous and suitable for scalable multi-camera systems.

11.2 Polar Representation of Covariance Descriptors for Circular Features

After having detected the pedestrians in an image or a video footage, it is often useful for surveillance purposes to identify the precise head position, in order to extract further information (person gender, ethnicity, presence of headgears, cues for face recognition, etc.).

As introduced in Section 10.3, it is possible to apply the generic covariance-descriptor classifier to head detection as well, but when this algorithm is applied to objects with non-rectangular shape (e.g. holes, heads, wheels, etc.), the performances in terms of classification accuracy degrade due to the inclusion of non-discriminative pixels within the rectangular patches associated to the logistic regressors.

Aiming to classify circular features, the use of patches with generic circular shapes would catch variations more accurately than just using axis-oriented rectangular shapes. Indeed, using circles or annulus would exclude from the covariance matrix computation all the pixels that do not strictly belong to the circular shape to recognize. Even if this technique would yield more accurate classification results, the use of non-rectangular or non-axis-oriented patches would hinder the use of integral images, that are strongly exploited by the classifiers for fast covariance matrix computations [112]]. This limitation can be solved by the use of polar images; defining a reference point C (x_C, y_C) and:

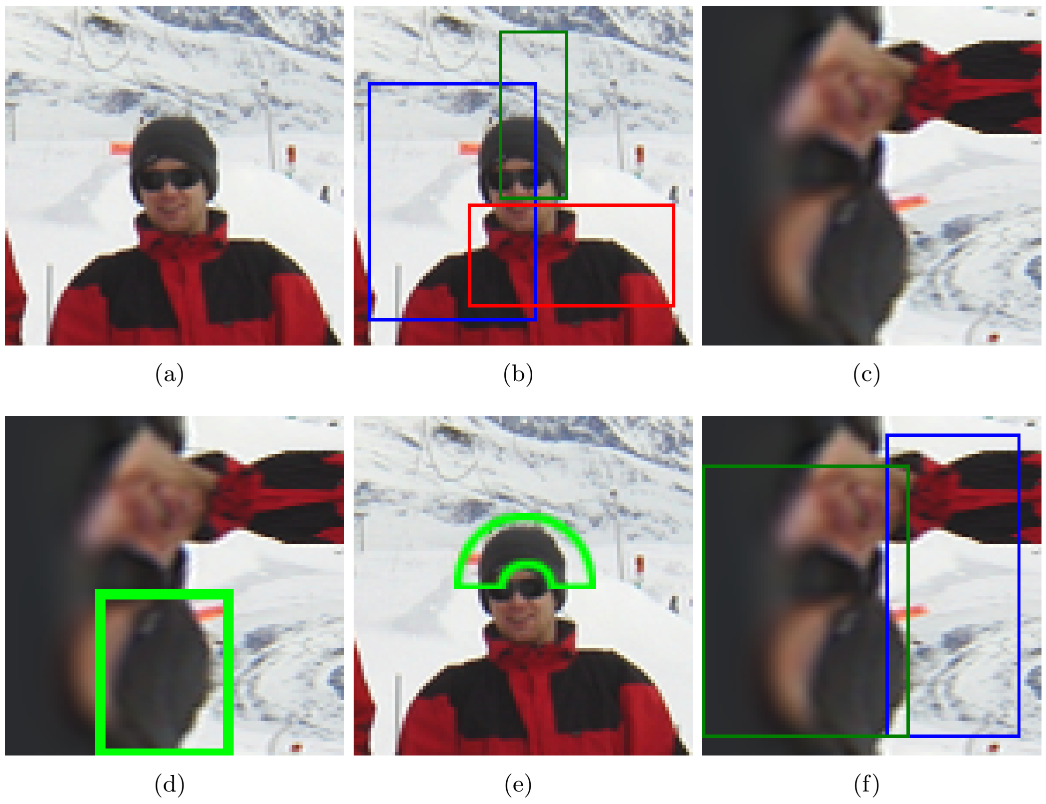


Figure 11.2: (a) a patch used for head classification; (b) examples of rectangular patches used by weak classifiers according to the original proposal [1]; (c) polar transformation of patch w.r.t. the center of image (a); (d) rectangular patch on polar image; (e) transformation of the patch in (d) onto the original image; (f) examples of rectangular patches used by weak classifiers learnt over polar images.

$$\begin{aligned} \rho &= \sqrt{(x - x_C)^2 + (y - y_C)^2}, \\ \vartheta &= \arctan\left(\frac{y - y_C}{x - x_C}\right) \end{aligned} \tag{11.1}$$

the polar image of $I(x, y)$ w.r.t. to point C is $I_p(\rho, \vartheta)$ (see Fig. 11.2(a) and 11.2(c)).

Indeed, given an image and its polar transformation w.r.t. the image center, any slice of annulus on the original image (centered in the image center) can be represented as an axis-oriented rectangular patch in the polar transformation. Therefore, the polar transformation creates a bridge between the circular patches (useful for classification purposes) and the rectangular patches (needed by the intrinsic classifier architecture); given an image to classify, as first step the polar image transformation is computed and then the weak classifiers are applied on it: specifically, each of them operates on a rectangular sub-window over the polar image, that represents a slice of annulus over the original image (see Fig. 11.2(d) and 11.2(e)): this procedure generates a classifier more suited to circular shape classification.

11.3 Multi-Spectral Image Derivatives for Covariance Descriptors

In appearance-based object classification, it is common to avoid the use of chrominance since in most cases color does not convey any discriminative information (e.g. in the classification of pedestrians, vehicles, textures, etc.). Instead, since color can be successfully used to compute more accurate edges w.r.t. luminance images [115–118], we claim that the use of chrominance for image derivative computation can improve the classification results also.

In order to compute covariance descriptors sensitive to luminance and chrominance, we exploit multi-dimensional gradient methods and define the following directional derivatives for the RGB color space:

$$\begin{aligned} I_x^{RGB} &= \sqrt{\left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2};\ I_y^{RGB} = \sqrt{\left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2} \\ I_{xx}^{RGB} &= \sqrt{\left|\frac{\partial^2 R}{\partial x^2}\right|^2 + \left|\frac{\partial^2 G}{\partial x^2}\right|^2 + \left|\frac{\partial^2 B}{\partial x^2}\right|^2};\ I_{yy}^{RGB} = \sqrt{\left|\frac{\partial^2 R}{\partial y^2}\right|^2 + \left|\frac{\partial^2 G}{\partial y^2}\right|^2 + \left|\frac{\partial^2 B}{\partial y^2}\right|^2} \end{aligned} \tag{11.2}$$

and similarly for Lab color space:

$$\begin{aligned} I_x^{Lab} &= \sqrt{\left|\frac{\partial L}{\partial x}\right|^2 + \left|\frac{\partial a}{\partial x}\right|^2 + \left|\frac{\partial b}{\partial x}\right|^2};\ I_y^{Lab} = \sqrt{\left|\frac{\partial L}{\partial y}\right|^2 + \left|\frac{\partial a}{\partial y}\right|^2 + \left|\frac{\partial b}{\partial y}\right|^2} \\ I_{xx}^{Lab} &= \sqrt{\left|\frac{\partial^2 L}{\partial x^2}\right|^2 + \left|\frac{\partial^2 a}{\partial x^2}\right|^2 + \left|\frac{\partial^2 b}{\partial x^2}\right|^2};\ I_{yy}^{Lab} = \sqrt{\left|\frac{\partial^2 L}{\partial y^2}\right|^2 + \left|\frac{\partial^2 a}{\partial y^2}\right|^2 + \left|\frac{\partial^2 b}{\partial y^2}\right|^2} \end{aligned} \tag{11.3}$$

At this point, it is straightforward to extend equation 10.1 to RGB and Lab color spaces:

$$\begin{aligned} F^{RGB} &= \left[x, y, \left|I_x^{RGB}\right|, \left|I_y^{RGB}\right|, \sqrt{(I_x^{RGB})^2 + \left(I_y^{RGB}\right)^2}, \left|I_{xx}^{RGB}\right|, \left|I_{yy}^{RGB}\right|, \arctan\frac{\left|I_y^{RGB}\right|}{\left|I_x^{RGB}\right|}\right]; \\ F^{Lab} &= \left[x, y, \left|I_x^{Lab}\right|, \left|I_y^{Lab}\right|, \sqrt{(I_x^{Lab})^2 + \left(I_y^{Lab}\right)^2}, \left|I_{xx}^{Lab}\right|, \left|I_{yy}^{Lab}\right|, \arctan\frac{\left|I_y^{Lab}\right|}{\left|I_x^{Lab}\right|}\right] \end{aligned} \tag{11.4}$$

Chapter 12

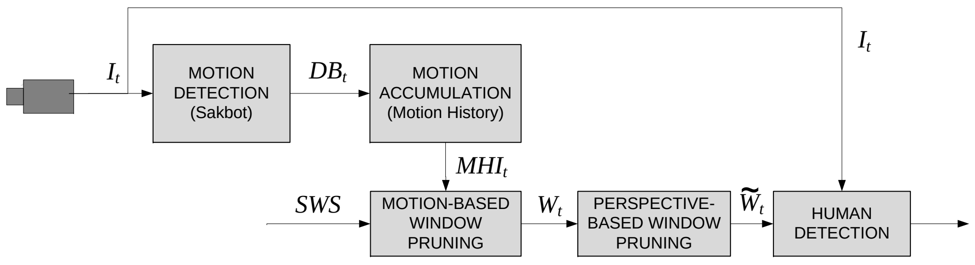
Exploit Motion and Perspective to Speed Up Sliding Window

As introduced in Section 10.4, pedestrian detection at frame level is usually performed with a sliding window approach, that is a brute force search of the learned pattern in the whole space of possible window states. This section propose methods to reduce the number of the windows to classify in order to obtain a real-time response from the system. Actually if it were possible to obtain reliable object tracking in the scene, this exhaustive search could be avoided, performing focused human classification over the bounding box of the tracked objects; however in Chapter 10 we stated the intention to study appearance-based methods assuming to work on scenarios where tracking algorithms are not available.

Let's name the set of all possible windows as "Sliding Windows Set", or SWS: this set spans over the whole space of window states: typically position and scale, $w = (w_x, w_y, w_s)$ (we do not consider aspect ratio, nor rotation). The cardinality of the SWS depends on the size of the image, on the range of scales to check and on the stride of scattering of the windows: regarding this latter parameter, to obtain a successful detection process, the SWS must be rich enough so that at least one window targets each pedestrian in the image. To be more precise, every classifier has a degree of sensitivity to small translations and scale variations, i.e. the response of the classifier in the close neighborhood of the window encompassing a pedestrian, remains positive ("region of support" of a positive detection). It is then straightforward that having a sufficiently wide region of support can be a very desirable property for the sliding window detection process, since the SWS could be uniformly pruned, up to the point of having at least one window targeting the region of support of each pedestrian in the frame. Vice versa a too wide region of support could generate de-localized detections [121].

From this point of view, an important advantage of the covariance descriptors is its relatively low degree of sensitivity to small translations and scale variations, i.e. its region of support over the positive detections was demonstrated to be higher with respect to many other descriptors (especially w.r.t. HoG). The size of the region of support depends on the training data, and the INRIA-based detector (see Section 10.2) shows a radius of such region of approximately 15% of the window size and 20%.

Since we make no assumption on the observed scene, the size of humans is totally unknown and the range of the searched scales is fairly wide; differently from Tuzel's



I_t
MOTION DETECTION (Sakbot)
DB_t
MOTION ACCUMULATION (Motion History)
I_t
MHI_t
SWS
MOTION-BASED WINDOW PRUNING
W_t
PERSPECTIVE-BASED WINDOW PRUNING
$\widetilde{W}_t$
HUMAN DETECTION

Figure 12.1: Scheme of the approach proposed to reduce computational load of sliding-window object detection exploiting motion and perspective.

approach [1], that performs sliding window at 5 different scales only (increasing the size of 20% from step to step), denoting the assumption of a prior knowledge about the human size inside the image, we search at very different scales, employing 21 steps, with an increase of 10% in size from step to step. At any given scale, we define a window displacement stride (both in x and y) that is not static (typical values are from 2 to 4 pixels), but is function of the size of the window, generating smaller (bigger) displacements for small (big) windows. Such loose conditions over the sliding window generation have the drawback to generate a very redundant window set that becomes critical considering that for each window a classification procedure must be issued and that the process is fairly CPU consuming (recall the exponential mapping of Eq. 10.2).

Therefore we employ a two-fold technique that prune the sliding-window set (SWS in Fig. 12.1): the first fold exploits the motion history image MHI_t, i.e. retaining a number windows is proportional to the amount of motion present in the area (W_t in fig. 12.1, and Section 12.1); the second fold uses perspective constraints, i.e. removing all the windows whose size is not compliant with the perspective model ($\widetilde{W_t}$ in fig. 12.1, and Section 12.2). After these procedures, the human detection algorithm is issued over the survived windows only. The experimental results presented in Chapter 14 will demonstrate the significant reduction of the SWS size and the further advantage of the increase in detection precision.

12.1 Motion-based pruning

Sakbot application (Section 7) is exploited to build a background model and to segment moving objects and to remove other artifacts. Given pixel p and time t, the instantaneous motion $MD_t\left(p\right)$ (MD as motion detection) is computed by thresholding the difference with the background model. Then, to account for the accumulation of motion in time (and, thus, considering also regions where the motion was present in the recent past) we exploit the *Motion History Image* (MHI) introduced by Bobick and Davis in [132], defined as

$$MHI_t\left(p\right) = \begin{cases} \tau & \text{if } MD_t\left(p\right) = 1 \\ \max\left(0, MHI_{t-1}\left(p\right) - 1\right) & \text{otherwise} \end{cases} \tag{12.1}$$

where the parameter τ represents the duration period over which the motion is integrated. $MHI_t\left(p\right) \in [0, \tau]$: 0 means no motion in the history of the pixel, τ means motion in the latest frame.

A straightforward approach for pruning useless windows would exploit the motion detection MD_t, removing all the windows that contain no (or low degree of) motion. This sharp window rejection has the drawback to prune windows in areas where the motion is fragmented (because of background camouflage) or where there is absence of motion just in the very last frame. Employing a more conservative approach based on the motion history image MHI_t solves the problem; however, if the τ (see eq. 12.1) is set from moderate to high values (i.e. 5-30 seconds), and if there are objects that move all around the image frame, the reduction of the number of windows is very limited. As a solution to this, it is reasonable to relate the number of retained windows in an area with the amount of motion history present in that area, by increasing the amount of discarded windows together with the "age" of the motion recorded within the window (i.e. windows containing "older" motion will be discarded more easily than windows containing "fresher" motion). The precise procedure is presented in Alg. 1. At first, the motion ratio MR_t^i for the window w^i at frame t is computed as the ratio of pixels with motion ($MHI_t(p) > 0$ and $p \in w^i$) with respect to its area. If the motion ratio is too low, the window is discarded; otherwise there is a further verification, that depends on the maximum (i.e. most recent) motion found inside the window: a random pruning technique is performed depending on such value. As mentioned in Section 10.2, the LogitBoost detector based on covariance features is quite insensitive to small translations and scales, thanks to the region of support. Therefore, pruning a (limited) number of windows around a human silhouette will not affect the overall detection performance, since from the many windows encompassing the pedestrian, some of them will very likely survive from the pruning of Alg. 1 and trigger a true-detection over the human detector.

Algorithm 1 Motion-based window pruning

```
1:  Require: Sliding-window set $SWS$, Frame $I_t$, Motion History Image $MHI_t$

2:  $W_t = 0$
3:  for all windows $w^i \in SWS$ do
4:      $MR_t^i = \frac{\#\{pixel\ p|p \in w^i \wedge MHI_t(p) > 0\}}{\#\{pixel\ q|q \in w^i\}}$
5:      if $MR_t^i < 0.5$ then
6:          prune $w^i$
7:      else
8:          $v = \max_{p \in w^i} MHI_t(p)$
9:          Sample $u$ from uniform distribution $U(u|[0, \tau])$
10:         if $u < v$ then
11:             $W_t \leftarrow w^i$
12:         else
13:             $w^i$ is discarded
14:         end if
15:     end if
16: end for
```

12.2 Perspective-based pruning

As mentioned before, the SWS contains a very wide range of scales since we do not make any prior assumption on the size of pedestrians in the frame. Hoiem *et al.* [123] propose a statistical framework to automatically retrieve the scene perspective in order to focus the detection tasks at the right scales. With a similar approach, we make the following hypotheses:

1. all the people move on the same ground plane;
2. people are in standing position;
3. camera tilt is small to moderate;
4. camera roll is zero or image is rectified;
5. camera intrinsic parameters are typical of rectilinear cameras (zero skew, unit aspect ratio, typical focal length);
6. all the observed people are assumed to have consistent physical height.

Hypothesis (2) comes with the definition of pedestrians; hypothesis (3) is satisfied because in our context the cameras are installed with very low tilt in order to observe wide views. Hypothesis (4) is fulfilled through initial system configuration. By employing cameras with fixed focal length and by compensating the other camera parameters with an intrinsic calibration hypothesis (5) is satisfied too. By focusing our attention to adult people detection we can assume without loss of generality that the difference on people height is negligible (hypothesis (6)).

Finally, assuming for the sake of simplicity that hypothesis (1) is a-priori satisfied, it is correct to approximate the height (in pixels) of the human silhouette with a linear function in the image coordinates (x, y) of the point of contact with the ground plane (confirmed also by eq. 7 in [123]). By estimating the parameters of this function, we can further prune the sliding-window set W_t by discarding all the windows whose height significantly differs from the estimated function and obtain the set $\widetilde{W_t}$ (see Fig. 12.1). In case the hypothesis (1) is violated (for instance workers on scaffoldings move on multiple parallel planes), it is still possible to perform perspective pruning by partitioning the image in areas and accept the rougher assumption that the height (in pixels) of the people inside each area is almost constant.

Differently from [123], that recovers the perspective using a probabilistic framework, we use a LSQ (Least SQuare) estimator with outliers rejection based on RANSAC. During the training phase, the motion-based pruning (see Section 12.1) and the pedestrian detector are run over a video that must contain, among other objects, also some people: all the windows that the human detector classify as positives are passed to a LSQ estimator and to the RANSAC iterative method, that is capable to discard the outliers (due to out-of-scale false detections) and retain only the windows which contribute to the correct parameter estimation. Detailed results are provided in Chapter 14.

Chapter 13

Multi-Stage Sampling with Boosting Cascades

As we mentioned in Section 10.2, one of the advantages of the rejection cascade of classifiers covariance is computational: given the task of pedestrian detection on real world images and defined the set of windows to test, only a small portion of them will run through the whole cascade; in fact, most of the patches are typically very dissimilar to the trained pedestrian model and will be rejected at its earlier stages, reducing therefore the overall load of detection process. We introduce the detection response R as

$$R(w) = \frac{P(w)}{M} \tag{13.1}$$

where w is a window, defined by the 3-dimensional vector (w_x, w_y, w_s), being respectively coordinates of the window center and window scale; we assume a constant aspect ratio of pedestrian ($width/height = 1/3$) and no rotation; P is the index of the last cascade which provides a positive classification for w and M is the total number of cascades. Given the structure of rejection cascades, the higher the degree of response $R(w)$ is, the further w reached the end of the cascade, the more similar it is to the pedestrian model (up to the extreme of $R = 1$, that means successful classification). The cascade of LogitBoost classifiers with covariance descriptors rejects 80% of the negative samples within the first $\frac{1}{5}$ of the cascade (i.e. $R(w) < 0.2$ for 80% of generic negative patches).

Having defined the response of the classifier through eq. 13.1, it is now possible to provide a representation of the *region of support* (see Fig. 13.1) that has been introduced in Chapter 12, where we proposed a method to reduce the cardinality of the SWS. Indeed, the original sliding window paradigm quantizes uniformly the state space, incurring in a two-fold problem: large waste of detections (i.e., computational time) over areas where pedestrians are not present and need of a redundant SWS to find every pedestrian in the scene. In addition to our proposal, which exploits motion and perspective, many other works use additional visual cues (depth, geometric inferences, etc.) to reduce the SWS [97,122,123].

Such cues typically have a nature that is orthogonal to the *appearance based* nature of the detection. What we claim in this chapter is that there is space for improvement of efficiency *within the domain of appearance*; exploiting the architecture of the boosting cascade and the shape of the region of support of covariance descriptors, we propose

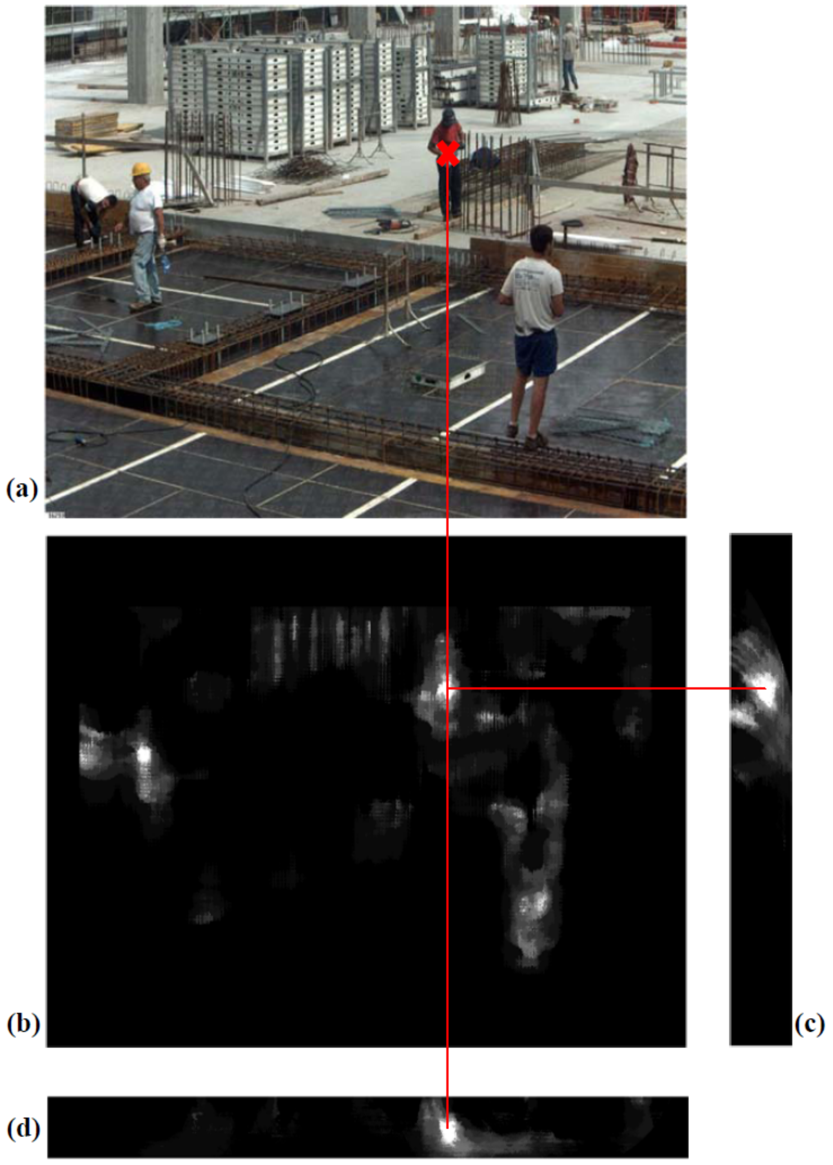


Figure 13.1: Example of the "region of support" for the cascade of LogitBoost classifiers trained on INRIA pedestrian dataset. (a) original frame. (b) shows the normalized response of the classifier by keeping w_s fixed - at the scale corresponding to the red-shirt man - and changing w_x and w_y. (c) is obtained by fixing w_x at the value corresponding to the center of the man and spanning w_y and w_s values, while (d) fixing w_y and spanning w_x and w_s.

to apply a sampling-based framework to speed up the detection process w.r.t. the sliding window approach. Such proposal can easily co-exist with any additional (and orthogonal) cue to further improve performances.

Our objective is to detect pedestrians as an estimation of the states given the observations, i.e. estimate the modes of the continuous density function $p(\boldsymbol{X}|\boldsymbol{Z})$, where $\boldsymbol{X} = (w_x, w_y, w_s)$ is the state and $\boldsymbol{Z}$ corresponds to the image. In section 13.1 we introduce an approximation of the likelihood function, through a multi-stage sampling-based process, that progressively improves the approximation of the likelihood. Such procedure has the advantage to provide a global view of the landscape of the likelihood function and, at the same time, to support efficient sample placement. The likelihood can then be used to detect the presence of the pedestrians within the single image. In section 13.2 we deal with pedestrian detection in videos, plugging the likelihood approximation method into a Bayesian-recursive filter. We have studied this method on the LogitBoost pedestrian classifier proposed by Tuzel *et al.* [1], but regardless of our specific choice, the method can be easily applied to any classifier which is able to provide a confidence measure over the detections.

13.1 Multi-Stage Kernel-Based Density Estimation on Single Images

In the context of single images, we purposely avoid to consider any prior information (such as motion, scene geometry, etc.) in order to provide a more general solution. Consequently, the state pdf can be assumed proportional to the measurement likelihood function, i.e. $p\left(\boldsymbol{X}|\boldsymbol{Z}\right) \propto p\left(\boldsymbol{Z}|\boldsymbol{X}\right)$.

The measurement likelihood function is estimated by iteratively refining through m stages (m fixed) the proposal function based on the observations. Algorithm 2 shows the complete procedure. The initial proposal distribution $q_0\left(\boldsymbol{X}\right)$, from which we sample the first set S_1 made of N_1 samples, is a uniform distribution on the state space (see line 1 of Alg. 2 and yellow points in the example image of Fig. 13.2). Each sample s represents a state (w_x, w_y, w_s) in the domain of the windows. Scattering samples according to a uniform distribution is somehow similar to the sliding window strategy (though the samples are not equally distributed and their locations are not deterministically defined), but N_1 is significantly lower than the cardinality of a typical SWS (numeric examples will be provided in Chapter 14). The rationale is that part of these samples will fall in the *basin of attraction* of each region of support of the pedestrians in the image and will provide an initial rough estimation of the measurement function. Being driven by the previous measurements, at any stage i we progressively refine the proposal distribution q_i that will be used to perform new sampling. This growing confidence over the proposal makes it possible to decrease, from stage to stage, the number of N_i to sample (see Fig. 13.2), differently from [126], where N_i is constant over stages.

The N_1 samples drawn from $q_0\left(\boldsymbol{X}\right)$ (line 5) will be used to compute a first approximation of the measurement density function p_1, through a Kernel Density Estimation (KDE) approach with Gaussian kernel, generating a mixture of N_1 Gaussians: for each j-th component, mean, covariance and weight are defined as follows: the mean $\mu_i^{(j)}$ is set to the sample value $s_i^{(j)} = \left(w_{x,i}^{(j)}, w_{y,i}^{(j)}, w_{s,i}^{(j)}\right)$; the covariance matrix $\Sigma_i^{(j)}$ is set to a covariance Σ_i (line 7), which, at any given stage i, is constant for all samples. The work in [126] proposed to determine the Σ for each sample as a function of its k-nearest neighbors; this strategy yielded fairly unstable covariance estimations when applied to our context: in fact, working with a very the low number of samples, k is to be kept pretty low (to maintain a significance over the covariance estimation), and this makes the estimation quite dependent on the specific randomized sample extraction. We preferred to assign an initial Σ_1 proportional to the size of the region of support of the

Algorithm 2 Measurement Step

```
1: Set $q_0(\boldsymbol{X}) = U(\boldsymbol{X})$
2: Set $S = \emptyset$
3: for i=1..m do
4:     Draw $N_i$ samples from $q_{i-1}(\boldsymbol{X})$:
5:         $S_i = \left\{ s_i^{(j)} | s_i^{(j)} \sim q_{i-1}(\boldsymbol{X}),\ \ j = 1, \dots, N_i \right\}$
6:     Assign a Gaussian kernel to each sample:
7:         $\mu_i^{(j)} = s_i^{(j)} \quad ; \quad \Sigma_i^{(j)} = \Sigma_i$
8:     Compute the measurement on each sample $s_i^{(j)}$:
9:         $l_i^{(j)} = R^{\lambda_i}\left(\mu_i^{(j)}\right)$ with $R^{\lambda_i} \in [0,1]$
10:    Obtain the measurement density function at step $i$:
11:        $p_i(\boldsymbol{Z}|\boldsymbol{X}) = \sum_{\pi_i^{(j)} \neq 0} \pi_i^{(j)} \cdot \mathcal{N}\left(\mu_i^{(j)}, \Sigma_i^{(j)}\right)$
12:    where:        $\pi_i^{(j)} = \frac{l_i^{(j)}}{\sum_{k=1}^{N_i} l_i^{(k)}}$
13:    Compute the new proposal distribution:
14:        $q_i(\boldsymbol{X}) = (1-\alpha_i)\, q_{i-1}(\boldsymbol{X}) + \alpha_i \frac{p_i(\boldsymbol{Z}|\boldsymbol{X})}{\int p_i(\boldsymbol{Z}|\boldsymbol{X}) d\boldsymbol{X}}$
15:    Retain only the samples with measurement value 1:
16:        $\widetilde{S}_i = \left\{ s_i^{(j)} \in S_i | R\left(\mu_i^{(j)}\right) = 1,\ \ j = 1, \dots, N_i \right\}$
17:        $S = S \bigcup \widetilde{S}_i$
18: end for
19: Run variable-bandwidth meanshift (Non-Maximal-Suppression) over $S$. Obtain the
    set of modes $\mathcal{M}_1$
20: Prune the modes in $\mathcal{M}_1$ that do not represent reliable detection (details in Sect.
    13.1). Obtain the new set of modes $\mathcal{M}_2$
21: Assign a Gaussian Kernel to each modes $\omega^{(j)} \in \mathcal{M}_2$ and compute the final likelihood
    function:
22:        $p(\boldsymbol{Z}|\boldsymbol{X}) \propto \sum_{\forall \omega^{(j)} \in \mathcal{M}_2} \mathcal{N}\left(\omega^{(j)}, \overline{\Sigma}\right)$
```

classifier, and decrease the Σ_i of the following stages: this has the effect of incrementally narrowing the samples scattering, obtaining a more and more focused search over the state space.

Finally, the response R of the classifier (eq. 13.1) is exploited, in a novel way, to determine the weight $\pi_i^{(j)}$ of the j-th component. The intention is that those samples falling close to the center of any region of support (i.e., close to the mode/peak of the distribution) might receive higher weight with respect to the others, so that the proposal distribution q_i, that is partly determined by p_i, will drive the sampling of the next stage more toward portion of the state space where the classifier yielded high responses. Conversely, sampling must not be wasted over areas with low response of the classifier. In other words, these weights must act as attractors which guide the samples toward the peaks. This is accomplished by connecting the weights $\pi_i^{(j)}$ to the response R of the pedestrian detector in the sample location $\mu_i^{(j)}$ (line 9). The exponent λ_i used

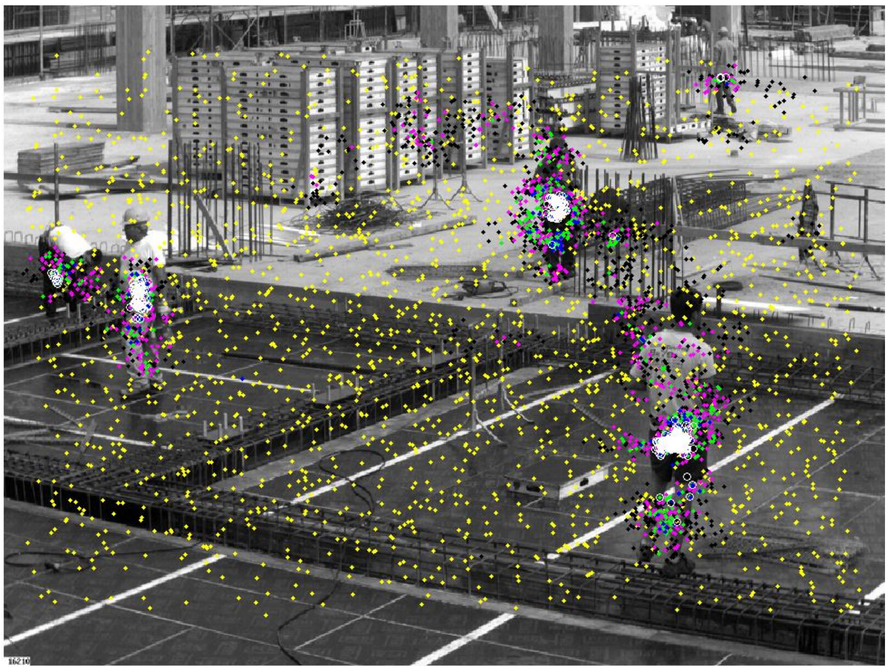


Figure 13.2: Distribution of samples in the stages with $m = 5$ and $(2000, 1288, 829, 534, 349) = 5000$ samples. Stage order is yellow, black, magenta, green and blue. White circles represents the samples that triggered a successful pedestrian classification.

to compute the measurement is positive and increases at every stage: at early stages, $\lambda_i \in (0; 1)$, therefore the response of the samples is quite flattened, in order to treat fairly equally all range of non zero responses; at later stages λ_i grows beyond 1, so that only the best responses will be held in account, while the others will be nullified. This behavior is clearly depicted in Fig. 13.2 where the samples at subsequent stages (even if less numerous) are concentrated in the peaks of the distribution (i.e. where the response of the pedestrian detector is higher).

The Gaussian mixture of line 11 is used as a partial estimation $p_i(\boldsymbol{Z}|\boldsymbol{X})$ of the likelihood function. This estimation is linearly combined with the previous proposal distribution $q_{i-1}(\boldsymbol{X})$ to obtain the new proposal distribution (line 14), where α_i is called *adaptation rate*.

The process is iterated for m stages and at the end of each stage only the samples of S^i that triggered a successful human detection (i.e. $R = 1$) are retained (line 16) and added to the final set of samples S (line 17). The samples retained in S are shown with white circles in Fig. 13.2.

The non-maximal suppression is accomplished using a *variable-bandwidth mean-shift* suited to work on Gaussian mixtures [126], that provides a mixture of Gaussians representing in a compact way the modes of the distribution generated by samples S (line 19). All those modes that do not contain a minimum number of detections (threshold τ_1), or that contain less than $\tau_1/2$ of strong detections (given the classification confidences provided by each LogitBoost classifiers of the cascade, a detection is considered strong if the minimum confidence is higher than a threshold τ_2) are suppressed (numerical values are provided in Chapter 14). The survived modes are considered definitive pedestrian detections and the corresponding mixture is used as the final likelihood function $p(\boldsymbol{Z}|\boldsymbol{X})$ (line 22).

13.2 Kernel-based Bayesian Filtering on Videos

We extend here the previous method to the context of videos, by propagating the modes in a Bayesian-recursive filter. It is crucial to highlight that the propagation of the conditional density among frames (observations in time) is not used here to solve *data association*, i.e. we are not aiming at people tracking in a strict sense. The recursive nature of particle filtering is used here to exploit temporal coherence of pedestrians. In the sequential Bayesian filtering framework, the conditional density of the state variable given the measurements is propagated through prediction and update stages as:

$$p(\boldsymbol{X}_t|\boldsymbol{Z}_{1:t-1}) = \int p(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})\, p(\boldsymbol{X}_{t-1}|\boldsymbol{Z}_{1:t-1})\, d\boldsymbol{X}_{t-1} \tag{13.2}$$

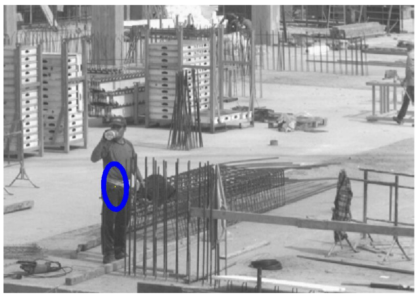
$$p(\boldsymbol{X}_t|\boldsymbol{Z}_{1:t}) = \frac{p(\boldsymbol{Z}_t|\boldsymbol{X}_t)\, p(\boldsymbol{X}_t|\boldsymbol{Z}_{1:t-1})}{\int p(\boldsymbol{Z}_t|\boldsymbol{X}_t)\, p(\boldsymbol{X}_t|\boldsymbol{Z}_{1:t-1})\, d\boldsymbol{X}_t} \tag{13.3}$$

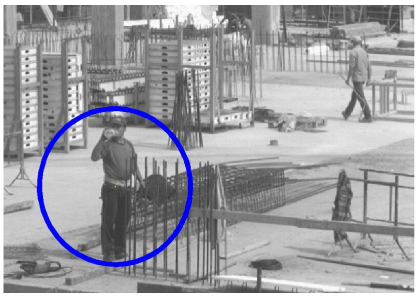
The priori $p(\boldsymbol{X}_{t-1}|\boldsymbol{Z}_{1:t-1})$ is propagated from the posteriori at the previous frame; for the first frame only $p(\boldsymbol{X}_0|\boldsymbol{Z}_0)$ no prior assumptions are made and uniform distribution is used similarly to what has been previously described in the procedure for single images (Alg. 2, line 1). The predicted pdf is obtained (eq. 13.2) as the product of the priori with the motion model and then marginalizing on $\boldsymbol{X}_{t-1}$. Since in complex scenes is difficult to identify a correct motion model [133], we applied a zero-order function with Gaussian noise of fixed covariance.

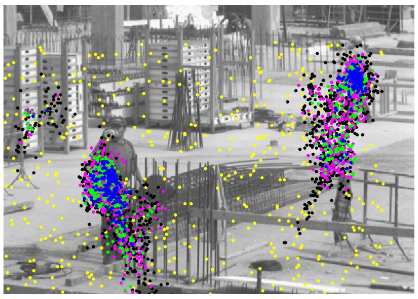
Fig. 13.3 depicts the different steps of this procedure. The priori is convolved with white noise which has the only effect of increasing its covariance (producing the predicted pdf - Fig. 13.3(b)). Differently from the case of single images, where q_0 is uniform, in videos, at each time t (i.e. frame), $q_0(\boldsymbol{X}_t)$ is obtained by applying a *quasi-random sampling* [134] to the predicted distribution p:

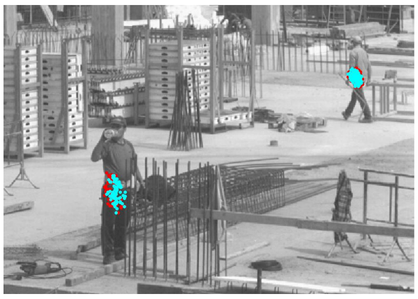
$$q_0(\boldsymbol{X}_t) = \beta \cdot p(\boldsymbol{X}_t|\boldsymbol{Z}_{1:t-1}) + (1-\beta)\cdot U(\boldsymbol{X}_t) \tag{13.4}$$

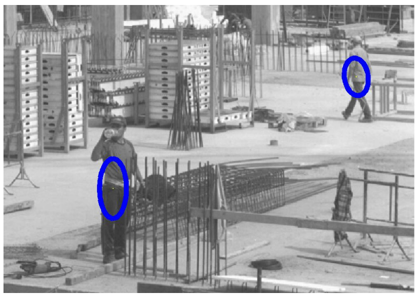
where β is used to decide the amount of random sampling. The random sampling is crucial to detect new pedestrians entering the scene (Fig. 13.3(c)). Given q_0, Algorithm 2 is used to iteratively estimate the likelihood $p(\boldsymbol{Z}_t|\boldsymbol{X}_t)$ (Fig. 13.3(e)). Any newly detected likelihood mode is confirmed as a new-entry pedestrian detection, verifying the same conditions defined in Alg. 2, line 20. Note that the quasi-random sampling is applied only to the proposal distribution q_0 (the proposal of the first stage of the multi-stage sampling). The likelihood and the predicted are multiplied to obtain (unless a normalization factor) the posterior pdf (see eq. 13.3).

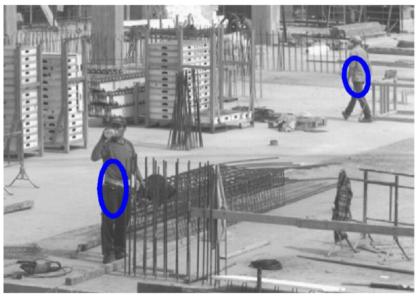

(a) Priori


(b) Predicted


(c) Resampling


(d) Measurements


(e) Likelihood


(f) Posterior

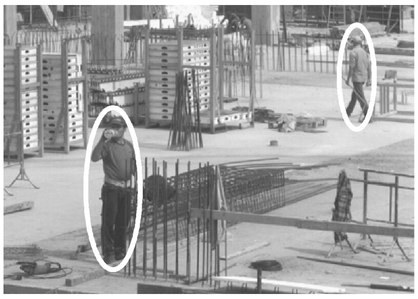

(g) Detections

Figure 13.3: Multi-stage sampling in the context of Bayesian recursive filtering. In (c) the yellow dots represents the quasi-random sampling. The coloring is consistent with the one proposed in Fig. 13.2. The man on the upper-right corner is out of the influence of the predicted pdf, but the uniform component of eq. 13.4 allows some samples to fall within the region of support of that person and to act as attractors for the samples in the next stages. In (d), red dots represents successful detections, cyan dots are successful detections with high detection confidence.

Chapter 14

Experimental Results

14.1 Test Beds and Operational Conditions

The additional learning with relevance feedback (Section 11.1) and the detection with sliding-window sets pruned by means of motion (Section 12.1) and perspective (Section 12.2), are tested on a test bed made of 4 videos recorded in Construction Working Site (CWS). Details are provided in Tab. 14.1.

Video Id	**Number of Frames**	**Type of Perspective**	**Notes**
CWS1	1740 352x288, 3 fps	planar	Even presence of moving peds. and other moving objs.
CWS2	2760 352x288, 3 fps	planar	More other moving objs. w.r.t moving peds.
CWS3	1740 352x288, 3 fps	multiple planes (scaffoldings)	More moving peds. w.r.t. other moving objs. and significant ped. occlusions.
CWS4	1000 352x288, 3 fps	planar	same scenario as CWS1; used for perspective learning only

Table 14.1: Construction Working Site benchmark. By "other moving objects" we mean typical construction working site machineries: cranes, bulldozers, trucks, etc.

The system configurations used in these tests are the following: (a) the INRIA-based detector, a 30-cascades LogitBoost classifier, trained on the INRIA pedestrian dataset [105], according to the method and the directions provided in [1]; (b) the first 24 cascades of the INRIA-based detector, followed by 6 cascades learned with explicit and implicit RF training data; (c) same as the previous, but using only implicit RF data. The explicit feeding was composed of 500 true-positive detections and 1000 false-positive detections; the implicit feeding was composed of 6 negative background images. In both cases, the RF data is extracted from videos recorded in the same day and at the same camera position used in videos CWS1, CWS2 and CWS3.

The use of polar representation (Section 11.2) and of multi-spectral image derivatives (Section 11.3) for covariance descriptors are tested in two different classification contexts of objects with circular features: head and polymer detection. In the first case, the goal is to find the exact position of the head of pedestrians, after having located them within an image through a pedestrian detector. Candidate head locations are searched (with a sliding window approach) in the upper part of the bounding box identifying the person. In this scenario, we require to record video with a high-resolution camera (1MPixel or more): pedestrian detection is performed at low resolution, but head detection exploits the full resolution of the camera over the detected pedestrian box. This condition becomes mandatory when the detected pedestrians have sizes of approximately 15 to 30 pixels in height (and a third of it in width), and the head corresponds to only 20 to 50 pixels: these resolutions are too low to obtain meaningful covariance descriptors. In the second case, the goal is to examine photomicrography image libraries and automatically extract the images that contain polymers.

The tests are based on per-window performance, that is more suitable to measure classifier performance, instead of per-frame performances, more suited to detection performance. We defined two datasets, both for training and for testing (see Tab. 14.2), partially extracted from two public datasets: the INRIA pedestrian and the MicroLab Gallery[1]. The positive set is made of patches of fixed size containing heads and polymer bubbles respectively, placed in the patch center. The negatives set instead is made of images containing all but heads and polymers: the negative patches are extracted from these images with randomly generated windows with size equal to the positive patches.

	Training		**Testing**		**Patch Sizes**	
	Pos.	**Neg.**	**Pos.**	**Neg.**	**Pos.**	**Neg.**
Head Image DataSet	1162	2438	266	906	98x98	320x240, 640x480
Micrography Image DataSet	2996	750	1360	106	50x50	640x480

Table 14.2: Head Images Dataset and Micrography Images Dataset.

			# Imgs	**Size of Images**	**# People**	**Size of People**	**# People per Image (avg)**
Tests on Images	Graz02 [135]		310	640x480	620	55px-410px	2.00
	INRIA [105]		288	333x531-1280x960	582	80px-800px	2.02
Tests on Videos	CWS	CWS5	148	800x600@1fps	340	55px-350px	2.30
		CWS6	114		398		3.49
		CWS7	68		83		1.22
	Caltech [2]		63	640x480	55	55px-260px	0.87

Table 14.3: Benchmark of videos used for testing the multi-stage sampling based detection. Videos CWS5,CWS6,CWS7 from construction working sites are taken from similar scenarios of Videos CWS1,CWS2,CWS3,CWS4 proposed in Tab. 14.1

Regarding the head detection, we trained six different classifiers: three with the traditional rectangular patches (called *straight* solutions), which compute image derivatives on gray, RGB and Lab values, exploiting respectively the eq. 10.1 and 11.4; three which compute the same image derivatives but over the polar transformation (called *polar* solutions). Each classifier is composed of a rejection cascade with 18 LogitBoost classifiers. In polymer classification, since the dataset contains only gray level images,

[1]URL: www.microlabgallery.com

we do not exploit any multi-spectral image derivative but only the polar transformation.

Regarding the multi-stage sampling based detection (Chapter 13), we performed extensive experimentation on images and videos. In both cases, the frame size is fairly large (rarely less than 640x480, up to 1280x960) in order to better stress on our main advantage, i.e. the use of less samples with respect to sliding window strategy which can be very expensive for such large images. Additionally, we are also considering a large range of scales (typically a factor of 8 from the smallest to the biggest) since the considered images contain people of quite diverse sizes. Finally, tests have been carried out considering no other information than appearance (neither motion nor scene geometry).

Experimental results are obtained on the benchmark reported in Tab. 14.3. In order to compare with the state of the art we used publicly available datasets which also provide ground-truth annotations. In the case of images, we have used the Graz02 dataset [135] which contains more than 300 images at the fixed size of 640x480, both in landscape and portrait layout. Moreover, we tested our approach on the well-known INRIA pedestrian testing set [105] which contains very diverse images, both in size, number of people and complexity. Regarding the videos, we tested our approach on a video taken by a publicly available dataset (provided by Caltech [2]) and on 3 videos taken from construction working sites (CWS).

The accuracy of pedestrian detection is measured at object level in terms of the matching of the bounding box found by the detector (BB_{dt}) with the bounding box in the ground truth (BB_{gt}). A matching is found using the measure defined in the PASCAL object detection challenges [136] which states that the ratio between the area of overlap of BB_{dt} with BB_{gt} and the area of merge of the two BBs must be greater than 0.5. Moreover, multiple detections of the same ground-truthed person, as well as a single detection matching multiple ground-truthed people, are affecting the performance in terms of recall and precision.

Regarding our approach, most of the tests have been performed using a total number of 5000 particles, divided over $m = 5$ stages as follows: $N_i = NP \cdot e^{\gamma \cdot (i-1)}$, where $NP = 2000$ represents the initial number of particles (i.e., N_1), whereas γ is a constant factor (equal to 0.44 in our tests) which ensures that the number of particles diminishes over the stages in an exponential way. A similar approach is followed also for λ_i and Σ_i, which are the exponent for the measurement (Alg. 2, line 9) and the covariance for the Gaussian kernels, respectively. The starting values are 0.1 and $diag(7, 14, 0.125)$ (obtained considering the region of support, and with normalized scales) and the exponential constant are 1 and -0.66, respectively. Finally, the thresholds for the non-maximal suppression (see Section 13.1) have been set to $\tau_1 = 4$ and $\tau_2 = 4.0$.

14.2 Results and Evaluations

In Tab. 14.4 we present a qualitative summarization of the effect of motion-based and perspective-based pruning, where it is possible to appreciate how the two pruning techniques improve both performance (the number of windows to analyze per frame gets strongly reduced) and precision of the detection (since pruning basically reduces the number of false positives coming out from the human detector). As explained in section 12.1, the use of pruning based on the instant motion detection might negatively

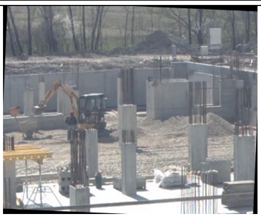
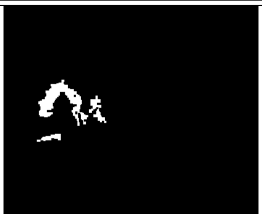
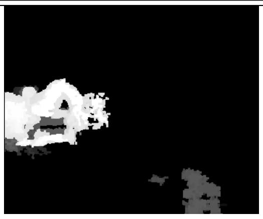
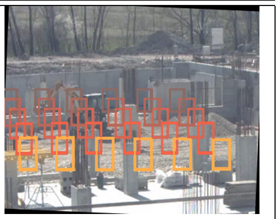
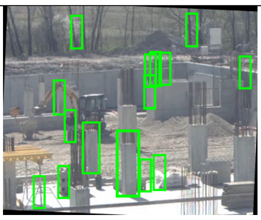
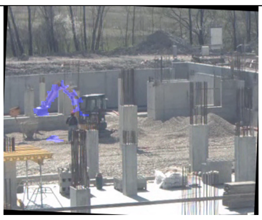
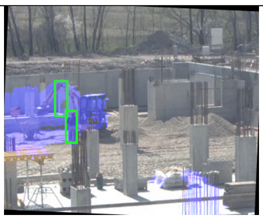
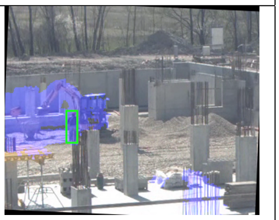
Input image I_t	**Motion Detection** MD_t	**Motion history image** MHI_t	**Motion history** MHI_t+**Perspective**
			
			
283798	425	35012	3127

Table 14.4: Qualitative evaluation of human detection, using motion-based and perspective-based pruning. Top row, from left to right: original snapshot; instant motion detection; motion history; perspective depiction. Middle row: human detection results, making use of the corresponding mask; from left to right: use of no mask, of instant motion, of motion history; of motion history and perspective. Bottom row: number of windows that survived the pruning and were fed to the human classifier.

affects the recall, and the example here gives a clear depiction of the problem: the man close to the bulldozer is steady in the latest frame (MD_t is zero over him), but moved in a recent past (MHI_t is non zero over him); this is reflected in the corresponding detections. The number on analyzed windows presented in the third row of Tab. 14.4 refers to the mean computed over several frames and the order of magnitude is confirmed all along the three videos CWS1,CWS2,CWS3: pruning based on MHI_t reduces sliding windows by approximately a factor 10, pruning on perspective by a further factor 10.

	False Positives		**True Positives**
	Out of Scale	**In Scale**	**In Scale**
# of Windows	110	40	310
Consensus Set	0	20	270
Error (Std. Dev.)	21.57	8.7	1.5

Table 14.5: Results of LSQ and RANSAC for perspective learning over video CWS4. The last row refers to the standard deviation of the differences between the estimated and the actual heights of the detection windows.

Regarding the perspective-based pruning, Tab. 14.5 reports the accuracy of the RANSAC-based method during the perspective-learning phase (Section 12.2) trained using video CWS4. The first row reports the number of windows that successfully passed the human classifier and that were used to feed the perspective learner; please notice that some of the false positive windows were showing a very similar scale to what a human would have shown in that same position (column "in scale" false positives, Tab. 14.5). The consensus set provided by the RANSAC, that is then used by the LSQ to estimate the model parameters, excludes all the false detections that are out of scale; this fact, together with the standard deviation of the error (low for in-scale detections, much higher for out-of-scale detections) gives proof that a linear model for perspective estimation is correct and it can be successfully learned by means of the classification output only, avoiding any camera calibration. Fig. 14.1 depicts the same experimental data in graphic manner.

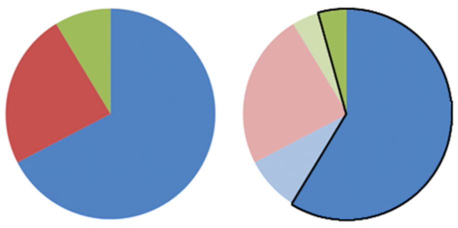


Figure 14.1: (a) pedestrian detections passed to the perspective learner; they are separated in three classes: (blue) true positives, that are in scale by definition; (green) false positives in scale; (red) false positives out of scale. (b) portion of the detections that is included in the consensus set obtained as a result of the perspective learnt with LSQ and RANSAC: out of scale false positives are completely eliminated.

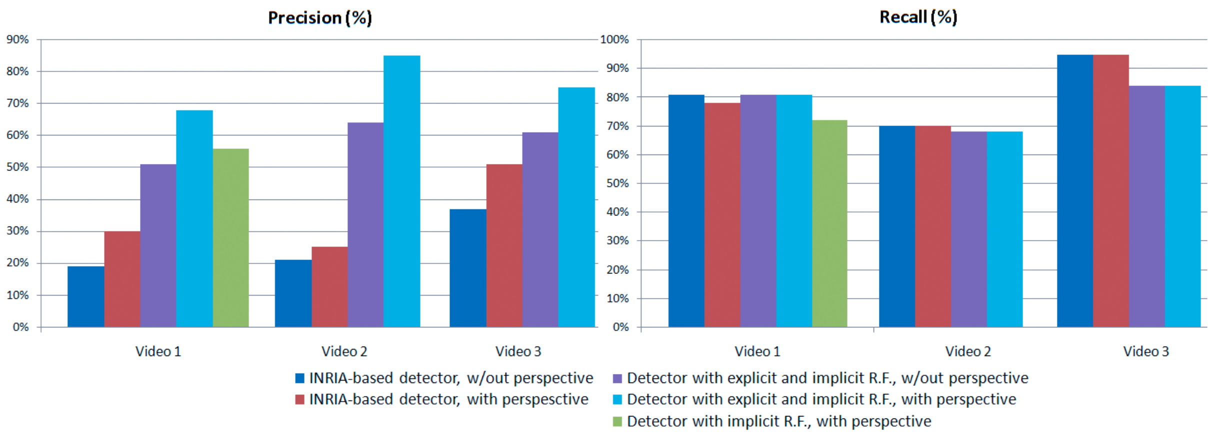


Figure 14.2: Precision and recall on pedestrian detection comparing INRIA-based detector with the relevance feedback additional learning and the perspective estimation on videos CSW1,CSW2,CSW3.

The results of the measurements of human detection performance on the three videos are shown in Fig. 14.2, that depicts the *precision* and the *recall* of the system with several setups: these two metrics were calculated measuring true/false positives and false negatives pedestrian detection. The use of perspective-based pruning reduces the number of false detections, therefore improving the precision. Please consider that in video 3 the humans are moving on scaffoldings, therefore the hypothesis (1) of section 12.2 is not satisfied and we model the perspective not as a function of the image coordinates but as an automatically estimated *constant*: regardless of the roughness of the approach, the precision improves significantly (+14%). The perspective pruning could rarely cause the removal of true-detection windows also: it happens when the window to be detected is slightly out of proportion and falls out of the range admitted by the perspective: this explains the minimal decrease of recall (video CSW1, first two bins of the recall histogram of Fig. 14.2).

The additional learning with RF gives a strong improvement over precision: when using both explicit and implicit RF data, the detection gains approximately +35% in video CSW1, +60% in video CSW2, +23% in video CSW3 w.r.t. the INRIA-based detector. Notice that the stronger is the presence of moving (or recently moved) objects different from humans in the scene (from highest to lowest: video CSW2,CSW1,CSW3), the higher is the gain in precision thanks to additional RF learning: this can be explained because, as we mentioned in Section 11.1, the RF cascades are placed at the final stages of a rejection cascade process and they will improve the performance over (the removal of) false-positives and not over (the inclusion) of false-negatives. For this same reason, our approach does not affect the recall, that remains basically the same of the INRIA-based detector. Only in video CSW3 there is a slight decrease of recall, due to the fact that the implicit feeding trained the additional cascades in a very strong way over the scaffolding, and in such video the humans are often integrated into or partially hidden by the scaffoldings.

Regarding the implicit learning performance, the additional learning based on implicit feeding only, increases the precision, even if with a lower degree w.r.t. the additional learning with both explicit and implicit feeding (experiments were performed on video CWS1 only). On the opposite there is a slight negative effect on recall: as

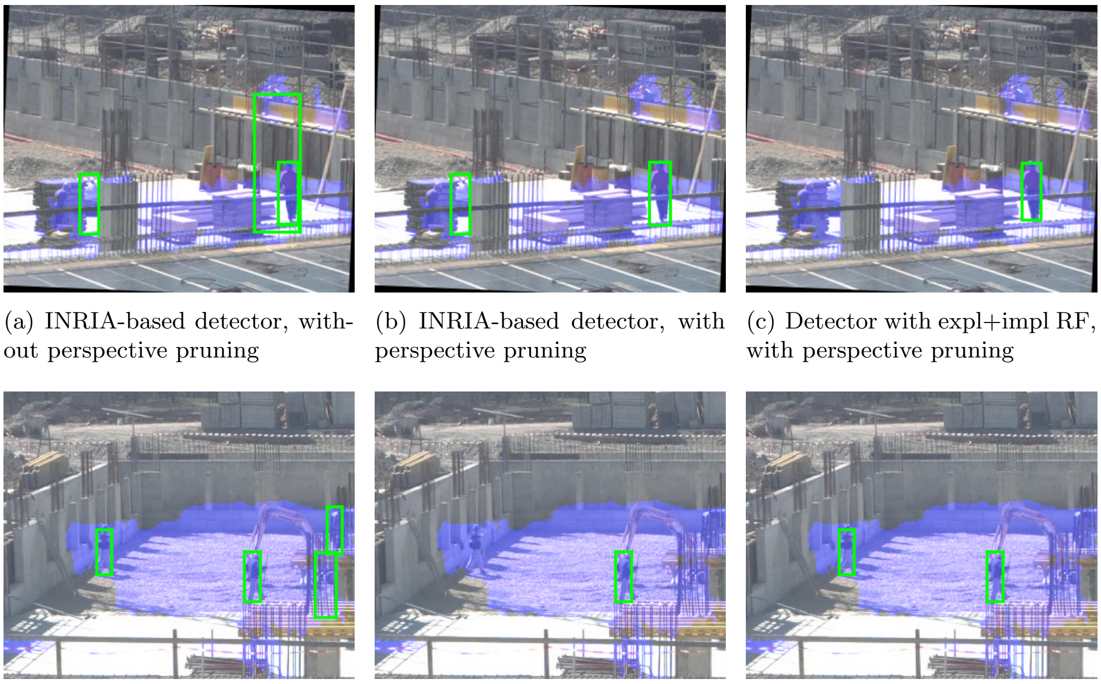

(a) INRIA-based detector, without perspective pruning
(b) INRIA-based detector, with perspective pruning
(c) Detector with expl+impl RF, with perspective pruning
(d) INRIA-based detector, with perspective pruning
(e) Detector with impl only RF, with perspective pruning
(f) Detector with expl+impl RF, with persperspective pruning

Figure 14.3: Example of human detection in video CWS3 (a,b,c) and in video CWS1 (d,e,f). The blue shadow highlights the pixels of the image where $MHI_t > 0$.

explained in section 11.1, the additional cascades learnt with implicit RF cannot count on the positive samples taken from the construction working site, that would help in shaping the positive-detection space.

Fig. 14.3 provides a visual outcome of the detections over video CSW3 and CSW1. Sub-figures 14.3-a,b,c depict the advantage of perspective-pruning first, and of false-positive removal thanks to the RF cascades then. Sub-figures 14.3-d,e,f show the behavior of the RF cascades with respect to the INRIA-based detector: the implicit RF cascades (Fig. 14.3-e) remove the false positives together with a true-positive; the explicit and implicit RF cascades instead (fig. 14.3-f) are able to remove false positives without affecting the true positives.

Regarding the classification of circular features by means of polar transformation and multi-spectral derivatives, Fig. 14.4 plots the results of the classifiers applied over the Head Images DataSet. Regardless of the chosen image derivative, the last cascades of the polar classifiers always yield better results w.r.t. the straight classifiers. Moreover, the proposed method generates lighter classifiers that will benefit the detection process with a lower computational load (on average, over the three color spaces, polar classifiers use 23% less weak classifiers; see Fig. 14.5). The use of color brings further increase in performance: overall best performances are obtained with the polar classifier using Lab image derivatives, that has a *miss rate* (MR) of 4.5% (w.r.t. 7.5% of the straight classifier over gray values), a *False Positives Per Window* (FPPW) of 0.037% (w.r.t. 0.135%) and 33.5% less weak classifiers.

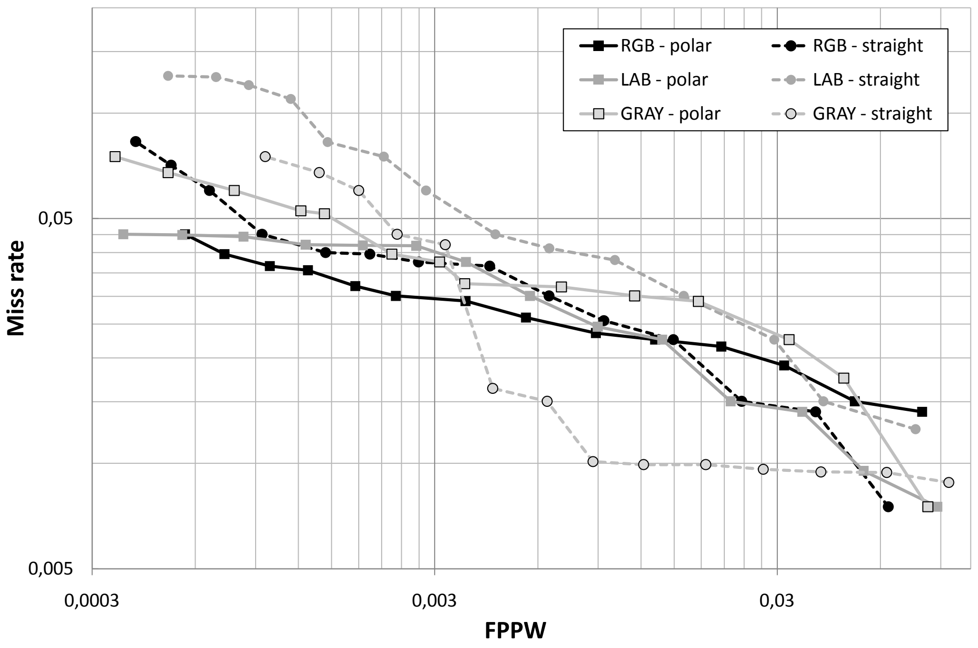
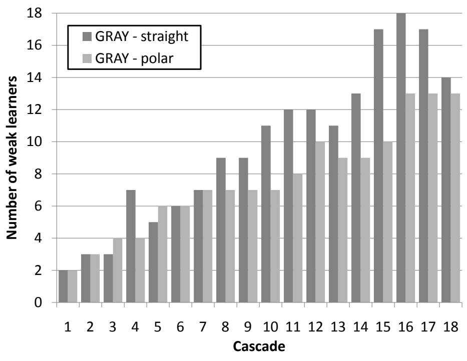
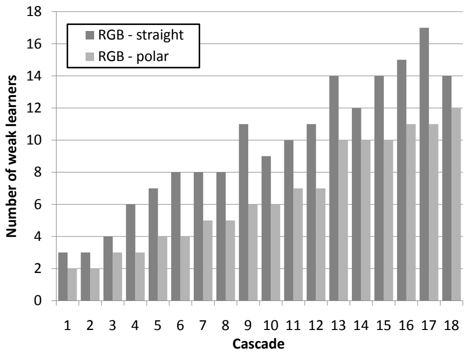


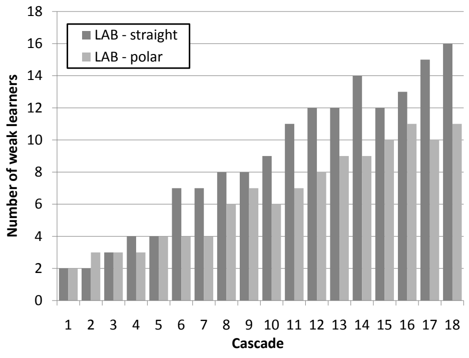
Figure 14.4: Miss rate versus false positives per window on the Head Image Dataset. Each marker represents the performance up to a cascade level. The markers at the bottom-right corner represent the results using up to 5^{th} cascade; adding more cascades, the markers move toward the upper-left corner, up to the 18^{th} cascade. The more cascades are introduced, the lower the FPPW, the higher the miss rate.



(a) Gray



(b) RGB



(c) Lab

Figure 14.5: Number of weak classifiers per cascade for the 6 classifiers: (a) gray (straight/polar), (b) RGB (straight/polar), (c) Lab (straight/polar).

Tab. 14.6 shows the MR of straight and polar classifiers vs FPPW on the Micrography Images DataSet. On average the miss rate of the polar classifier is 2 order of magnitude lower than the straight classifier. This strong advantage in performance is due to the clear circular shape of the polymer bubbles that strongly benefits from the polar representation. Fig. 14.6 shows some visual examples of detection of heads in complex scenarios and of polymer bubbles in the photomicrography dataset.

Regarding multi-stage sampling based detection, the tests on the two single image datasets (Graz02 and INRIA) are made in a slightly different way. The test on Graz02 aims at demonstrating the advantage of our method with respect to a sliding window strategy. Therefore, we compared the former with the latter at different number of windows. Specifically, the scale stride is always set to 1.2, while the number of windows

Miss Rate	10^{-2}	10^{-3}	10^{-4}
	FPPW		
Straight	$1.32 \cdot 10^{-2}$	$29.12 \cdot 10^{-2}$	$33.68 \cdot 10^{-2}$
Polar	$0.05 \cdot 10^{-2}$	$0.15 \cdot 10^{-2}$	$0.29 \cdot 10^{-2}$

Table 14.6: Results on the Micrography Images Dataset. FPPW at fixed miss rate on gray values in comparison between straight and polar classifiers.

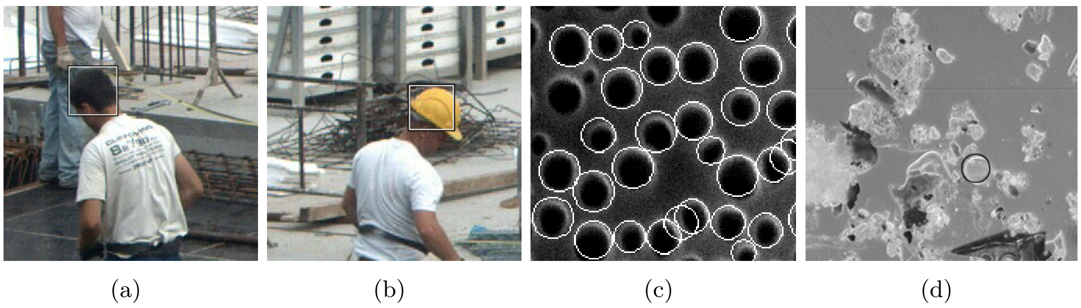


Figure 14.6: Examples of correct head detection in complex scenes (a) and (b) and polymer bubbles detection: (c) detection over polymer image (some missed detections are present); (d) detection over non-polymeric image (just one false positive).

has been tested to 5000, 10000, 20000 and 50000 (which respectively generate a window displacement stride of 15.6%, 10.9%, 8.2% and 5% of the window widths and heights). The non-maximal suppression is performed with mean shift and just for this test $\tau_2 = 2$. Tab. 14.6(a) shows the results achieved in terms of *False Positives Per Image* (FPPI) and MR as suggested in [2]. Please note that our approach has been indicated as *MSBoost* and it is capable to achieve approx the same FPPI of sliding window with 10000 windows (twice as many), and as a good miss rate as the sliding window with 20000 windows (four times as many).

(a)

Graz02 DataSet	**FPPI**	**MR**
SW 5000 wnd	0.39	0.76
SW 10000 wnd	0.66	0.57
SW 20000 wnd	1.08	0.46
SW 50000 wnd	1.66	0.37
MSBoost 5000 p.	0.74	0.43

(b)

Video CWS5	**FPPI**	**MR**
Non-rec. 2500 p.	0.29	0.34
Rec. 5000 p.	0.56	0.13
Rec. 2500 p.	0.45	0.14
Rec. 1250 p.	0.30	0.29

(c)

Other Videos		**FPPI**	**MR**
CWS	Video CWS5 (5000 p.)	0.56	0.13
	Video CWS6 (5000 p.)	0.98	0.55
	Video CWS7 (5000 p.)	0.42	0.78
Caltech video (15000 p.)		1.35	0.58

Table 14.7: Summary of results of multi-stage sampling based detection.

The second test (on INRIA dataset), instead, aims at comparing our approach with existing methods, thanks to the tool provided in [2]. Being unknown the number of windows used in their tests, we are unable to provide a fair comparison. Therefore, in Fig. 14.7 we report the results obtained with *MSBoost* employing a total number of 15000 particles, which are (probably) much less than those used in the compared algorithms. It is worth noting that we set our parameters in order to have no more than one FPPI.

Also the experiments on videos have been divided in two types of tests. Firstly, we

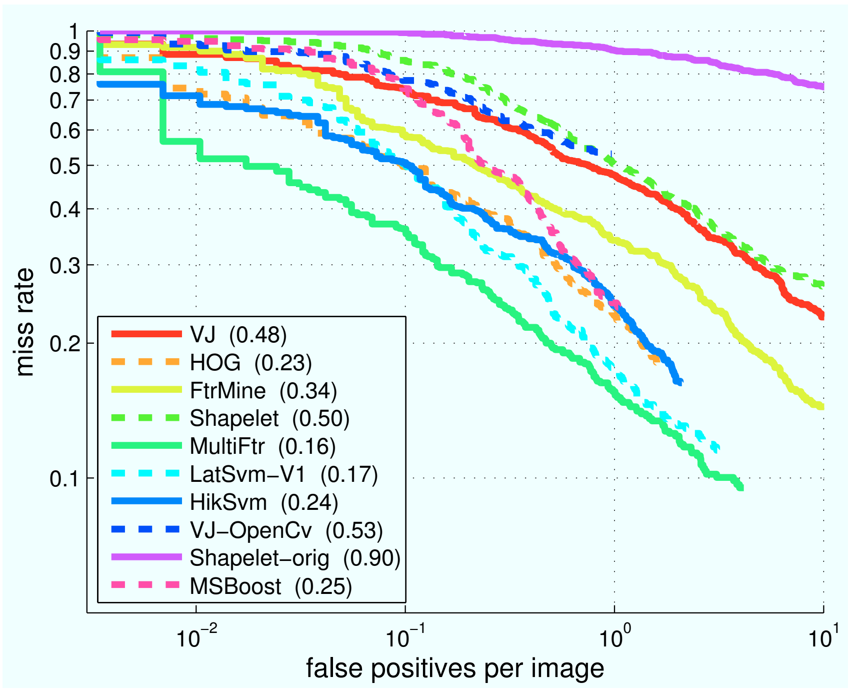


Figure 14.7: Results on INRIA dataset. Number in brackets represent the MR when FPPI=1. The plots are automatically generated by the tool described in [2]. Please refer to that paper for the other quoted methods.

aim at validating the usefulness of the Bayesian-recursive approach on video CSW5. Therefore, we compare the FPPI and MR achieved by using a non-recursive approach (Section 13.1) with 2500 particles with the Bayesian-recursive (Section 13.2) with varying number of particles (5000, 2500 and 1250). From Tab. 14.6(b) it is evident that the Bayesian recursive approach achieves with half of the particles the same performance (even slightly better) than the non-recursive approach.

Then, to further validate our approach we tested on two other videos from the CWS dataset which contain several heavy occlusions of the pedestrians (summary in Tab. 14.6(c)). Additionally, we briefly tested our method on a video taken from a public dataset (Caltech [2]). This dataset is very complete and challenging: the frames are heavily compressed, which makes our approach, based on covariance of image derivatives quite unreliable. Nevertheless, preliminary results on this dataset (which we are eager to test further) are rather good.

Regarding the time complexity of our approach, on average it takes about 1 second to perform 5000 detections using a C++ implementation on a dual-core off-the-shelf PC, also by exploiting the intrinsic parallelization of the algorithm. The complete approach can process about 0.75 frames per second (fps) with 5000 particles, which can be proportionally increased by reducing the number of particles (e.g., it becomes about 3 fps with 1250 particles which give good results on video CSW5).

Part IV

Conclusions and Appendices

Chapter 15

Conclusions

The thesis describes the research efforts spent on three topics that are tightly related to *mobile video surveillance*, namely video streaming, object tracking and object detection (specifically modeled for pedestrians): each of them corresponds to a thesis part. Our research has been very focused to identify the issues that mobile surveillance contexts arise and to search and design tailored solutions. Each thesis part encompass thorough reviews of state of the art literature, detailed description of the proposed methods and eventually exhaustive experimental results; in order to provide fair evaluations, wherever possible, we have compared the proposed methods by means of repeatable procedures, exploiting, as term of comparison, either well-known systems (from commercial or research applications) or public and annotated video datasets.

In part I, we report the efforts for building a complete streaming system for mobile video surveillance called Moses. The two sides of such system, specifically Movie (Mobile Video Encoder) and Movide (Mobile Video Decoder), are implemented with suitable optimization of open source modules to obtain efficient video streaming over GPRS/EDGE-GPRS networks. Measurements of image quality, video latency, frame loss and video fluidity on recorded videos are collected over Moses and other three well-known solution (i.e. Windows Media, Real Media and VLC) and then a thorough comparative evaluation is provided. A part from these off-line tests, Moses has been tested in challenging real-time live-video scenarios. Our extensive set of experiments demonstrates the effectiveness of the proposed solution. Specifically, we can draw the following brief conclusions:

1. for computer-based video surveillance, where low latency is crucial and fluidity is unnecessary, the Moses system is to be configured with the adaptive playback control disabled. In these conditions, the latency introduced in our system is much lower than in all the compared solutions;

2. for human-based video surveillance, the adaptive frame rate control dramatically improves fluidity, at the cost of a slight raise of latency, which still remains much lower than in the compared solutions;

3. for the trade-off quality vs compression bitrate, in terms of both PSNR and frame losses, the proposed system outperforms the others.

In part II, after having connected Moses to a motion detection and tracking system (Sakbot), we measure the performance degradation of the video analysis due to video compression and streaming only; we tackle here the specific case of mobile video surveillance where the source modules are made of a mobile and minimalist hardware configuration able to grab video, compress it and eventually stream it over the network; in such conditions the video analysis is performed remotely. We evaluate pixel-level segmentation and object-level tracking obtained on compressed videos w.r.t. their original uncompressed version, demonstrating that it is possible to perform successful fixed-camera object tracking on highly compressed video, up to a minimum of 5 Kbps in indoor scenarios and 20 Kbps in outdoor ones. These bandwidths are supported by GPRS or Tetra radio-mobile networks.

We then envision a different video surveillance scenario, where the source module is moving with totally unconstrained motion. In this case, we propose a joint feature-structure approach for object tracking, that increases robustness in presence of free camera motion, varying focal length, scene cuts, severe occlusions and distractors. The proposed coherence measure used for weighting the association graph is also demonstrated to be a valid metric for the reliability of the tracking, allowing it to be suspended in case the object is not found in the scene. Moreover, the exclusion of low-coherence nodes from the extracted dominant set allows to reject false positive detections, often due to distractors. It is worth noting that the use of color features presented in this work is not a limitation, since the framework is flexible and open to be extended to different types of features. Regarding the graph matching step, other search heuristics can be plugged into the framework in substitution of dominant sets. It should be noted that the current implementation of this heuristic does not fit a real-time tracking as the processing time for a single frame can span from one to several seconds, whereas discrete matching techniques could be much faster. We propose a set of experimental results that evaluate the proposed tracking algorithm on non-compressed recorded videos, envisioning a monolithic-moving surveillance system. Nevertheless, given the reported performances of Moses on bitrate compression typical of UMTS or HSPA bandwidths, it is plausible to deploy this tracking algorithm in a moving-distributed system.

In part III, we tackle appearance based methods for mobile video surveillance: indeed appearance features are very suitable for mobile scenarios of video analysis since they are not strongly dependent on the motion status of the originating object (object tracking is dramatically more dependent on motion conditions). In the specific, we propose to modify a state of the art pedestrian classifier based on boosting classifiers and covariance descriptors using a relevance feedback approach to (semi) automatically enrich the training data: this allows to re-train the last stages of the boosting cascade, increasing the accuracy of the system on the false positive detections that are generated by the visual clutter of the observed scene; we then propose to use polar image transformations and multi-spectral images for obtaining a modified version of the classifier more suited to the detection of circular features.

Since the classification step is fairly CPU-consuming because of the inverse of the exponential mapping, we propose two methods to reduce the load of the detection: at first we tackle the reduction of cardinality of the windows in the sliding window approach, by means of motion and perspective. Regarding the first, we employ motion history images, purposely avoiding to count on precise motion segmentation. For the second, we exploit the response of the pedestrian classifier to learn the perspective of the scene in a totally autonomous way. Experimental results prove the efficacy of the proposed approach.

Eventually, we introduce a novel method suitable for both images and videos, to avoid the brute force strategy of sliding window paradigm; the proposed method works within the domain of appearance used by the classifier itself, exploiting the response of the boosting cascade to drive an efficient estimation of the measurement function; multi-stage sampling based strategy is used. The derived measurement function can be plugged in a kernel-based Bayesian filtering to exploit temporal coherence of pedestrian in videos. Experimental results show a gain in computational load maintaining same accuracy of sliding window approach. It is our intention to test our approach with larger public datasets to further validate the promising results described in this thesis.

15.1 Author's Publications

The research activity performed during the PhD has produced the following publications (publications prior to PhD are not listed here).

Journals and Book Chapters

(a) **G. Gualdi**, A. Prati, R. Cucchiara, "Video Streaming for Mobile Video Surveillance" in IEEE Transactions on Multimedia, vol. 10, n. 6, pp. 1142-1154, October, 2008.

(b) R. Cucchiara, **G. Gualdi**, "Mobile Video Surveillance Systems: an Architectural Overview", in Mobile Multimedia Processing: Fundamentals, Methods, and Applications; LNCS State-of-the-Art Surveys, edited by Xiaoyi Jiang, Matthew Ma, Changwen Chen (in press).

International Conferences

(c) **G. Gualdi**, R. Cucchiara, A. Prati, "Low-latency Live Video Streaming over Low-Capacity Networks" in Proceedings of the 8th IEEE International Symposium on Multimedia (ISM2006), San Diego, CA - USA, pp. 449-456, Dec. 11-13, 2006.

(d) **G. Gualdi**, A. Prati, R. Cucchiara, "Video Transcoding and Streaming for Mobile Applications" in Lecture Notes in Computer Science (LNCS), Digital Libraries: Research and Development, Revised Selected Papers, vol. 4877, Tirrenia, Pisa, Italy, pp. 262-267, Feb. 13-14, 2007.

(e) **G. Gualdi**, A. Prati, R. Cucchiara, "Mobile Video Surveillance with Low-Bandwidth Low-Latency Video Streaming" in Proceedings of ACM Workshop on Mobile Video (in conjunction with ACM Multimedia 2007), Augsburg, Germany, pp. 67-72, Sept. 28, 2007.

(f) **G. Gualdi**, A. Prati, R. Cucchiara, "An Open Source Architecture for Low-Latency Video Streaming on PDAs" in Proceedings of the 9th IEEE International Symposium on Multimedia (ISM2007), Taichung, Taiwan, pp. 302-309, Dec. 10-12, 2007.

(g) **G. Gualdi**, Albarelli, A. Prati, Torsello, Pelillo, R. Cucchiara, “Using Dominant Sets for Object Tracking with Freely Moving Camera” in Proceedings of 8th Int’l Workshop on Visual Surveillance, (colocated with ECCV 2008), Marseille, France, October 17, 2008.

(h) **G. Gualdi**, A. Prati, R. Cucchiara, E. Ardizzone, M. La Cascia, L. Lo Presti, M. Morana, “Enabling technologies on hybrid camera networks for behavioral analysis of unattended indoor environments and their surroundings” in Proceedings of 1st ACM Workshop on Vision Networks for Behavior Analysis, (colocated with ACM Multimedia 2008), Vancouver, British Columbia, Canada, pp. 101-108, October 31, 2008.

(i) R. Cucchiara, **G. Gualdi**, “Mobile surveillance: video analytics from and to mobile devices” in Proceedings of 1st International Workshop on Mobile Multimedia Processing, (colocated with ICPR 2008), Tampa, Florida, USA, December 7, 2008.

(j) **G. Gualdi**, A. Prati, R. Cucchiara, “Covariance Descriptors on Moving Regions for Human Detection in Very Complex Outdoor Scenes” in Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras (ACM/IEEE ICDSC 2009), Como, Italy, Aug. 30, Sept. 1-2, 2009.

Internal Technical Reports

(k) **G.Gualdi**, A. Prati, R. Cucchiara, “Design and Deployment for video communication and mobile video surveillance for patrols”, Dip. di Ingegneria dell’Informazione, University of Modena and Reggio Emilia, Nov. 2006.

(l) **G.Gualdi**, A. Prati, R. Cucchiara, “Analisi della qualit video dei prodotti Wti Alexys e Wavestore”, Dip. di Ingegneria dell’Informazione, University of Modena and Reggio Emilia, Jul. 2008.

(m) **G.Gualdi**, A. Prati, R. Cucchiara, “Sicurezza nei Cantieri - Sistema di Videosorveglianza per il Rilevamento Automatico di Operai Senza Elmetto di Sicurezza”, Dip. di Ingegneria dell’Informazione, University of Modena and Reggio Emilia, Oct. 2009.

Submitted Publications

(n) **G. Gualdi**, Andrea Prati, Rita Cucchiara, “Polar Representation of Covariance Descriptors for Circular Features”, submitted to IET Electronics Letters.

(o) **G. Gualdi**, A. Prati, R. Cucchiara, “Multi-stage Sampling with Boosting Cascades for Pedestrian Detection in Images and Videos”, submitted to International Conference on Computer Vision and Pattern Recognition (IEEE-CS CVPR 2010).

(p) C. Grana, D. Borghesani, **G. Gualdi**, R. Cucchiara, “Bag-of-Words Classification of Miniature Illustrations”, submitted to 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS).

Chapter 2, "*What is video surveillance?*" is covered by the following publications: 1 Book Chapter [b], 1 International Conference [i].

Thesis Part I, "*Video Streaming Architecture for Mobile Video Surveillance*" is covered by the following publications: 1 Journal [a], 1 Book Chapter [b], 4 International Conferences [c,d,f,i], 1 Technical Report [k].

Thesis Part II, "*Object Tracking for Mobile Video Surveillance*" is covered by the following publications: 1 Journal [a] and 2 International Conferences [e,g].

Thesis Part III, "*People Surveillance in Mobile Contexts*" is covered by the following publications: 1 International Conference [j, 1 Technical Report [m] and 2 submitted publications: [n,o].

Appendices

Appendix A

Video Security In Construction Working Sites

The present chapter describes the prototype for the *automatic detection of workers in construction working sites that do not wear protective helmets.* The prototype, partially funded by Bridge 129 S.p.A, was designed and developed between October 2008 and October 2009. The work passed through different stages, namely: the study and review of construction working site scenarios (Section A.1), the definition of a prototype (Section A.2), the collection of video and images datasets and eventually the evaluation of the prototype (Section A.3).

A.1 The Construction Working Site Scenarios

The term *construction working sites* (CWS) include a very wide range of different scenarios that likewise generate a very wide range of visual conditions. There are infrastructure CWS, new building CWS, building renovation CWS, night CWS, underground CWS, etc. Each scenario can be extremely challenging for automatic video analysis for very different reasons. Instead of tackling the problem from a very broad perspective, we have decided to focus on a specific type of construction sites, i.e. new building constructions, and to dissect, with the assistance of construction engineers, the several phases that it passes through; this precise analysis has provided the solid support to tailor the prototype on a well defined set of requirements and constraints.

The new building CWS basically passes through three macro stages:

1. *preliminary stage*: the CWS is at the very beginning and it is mainly represented by an open space, where preliminary works are performed (measurements of ground, material stacking, etc.) or the initial construction frameworks are built. There is very limited presence of vertical structures of any kind, typically just outer walls of the building, scaffoldings and material cages on the outer perimeter. At the very beginning of this stage the CWS gets surrounded by a perimetric border (either wire, fence or just a marker). There is intense machinery activity (e.g. cranes, bulldozers, trucks, etc.) that takes place all over the CSW; on the

Publications related to Appendix A: [j,m]; see the list of author's publications, page 147.

opposite there is a low degree of presence of lone workers (they typically work inside or on the side of machineries). The motion is mainly planar, on the ground plane. This CWS stage is depicted in Fig. A.1(a),A.1(b),A.1(c). Video analysis is challenged by the massive and chaotic motion of machineries or induced by them (e.g. smoke, debris, cranes moving big pale of materials, etc.), and this condition typically brings to failure object tracking algorithms.

2. *main building construction stage*: the CWS enters in its core building activity, that takes place inside the outer walls, and it transitions from being an open space to a well-defined building with indoor spaces. At this time the machineries are typically left out of the building and the workers are the main actors; their motion is not constrained to the ground plane anymore, since scaffoldings and flat surfaces are present at several levels of the construction. This stage is depicted in Fig. A.1(d),A.1(e): the degree of motion is reduced w.r.t. the preliminary stage. Now the real challenge is due to severe occlusions. Furthermore, the cameras have to be repositioned from time to time since new pieces of construction could cover their field of view; for this reason video analysis cannot rely on static geometrical models.

3. *inner works construction stage*: the building has reached its final structure and the main construction works are finished. This is the time for details and finishings (e.g. paintings, electrical and hydraulic works, etc.), that take place either on external scaffoldings or in indoor environments. The perimetric border is now removed and there is a very reduced number of machineries around. This phase is depicted in Fig. A.1(f). The video analysis in this stage is either typical indoor video surveillance (the challenge could be the lack of lightning) or outdoor, where the workers are partially occluded by scaffoldings.

Given this introduction, it is straightforward to observe that the CWS scenario is a typical case study for mobile video surveillance. In the specific, we designed a prototype equipped with mobile and embedded source modules, to facilitate repositioning, and with radio-mobile UMTS network communication, to reduce as much as possible the quantity of wirings. These conditions draw the outline of a mobile-distributed system, where the video analytics is to be performed remotely in case of limited size of the front-end devices.

A.2 Prototype description

Having realized that video surveillance based on motion segmentation and object tracking cannot provide reliable results in the CWS scenario, we have designed an application (scheme in Fig. A.2) that exploits appearance cues only.

The cameras (either fixed or PTZ cameras) are equipped with high resolution CCDs (i.e. 2 MPixels, 1600x1200), and the video is streamed to the remote processing unit through Moses . As we will see, the processing frame rate is fairly low, therefore the video streaming is configured to dispatch a slow sequence of high-resolution images and does not saturate the radio-mobile network. Upon video decoding, a sub-sampled version of the video (800x600 or less, depending on the scene view) is provided to the pedestrian detector. This module is based on the covariance descriptor classifier based

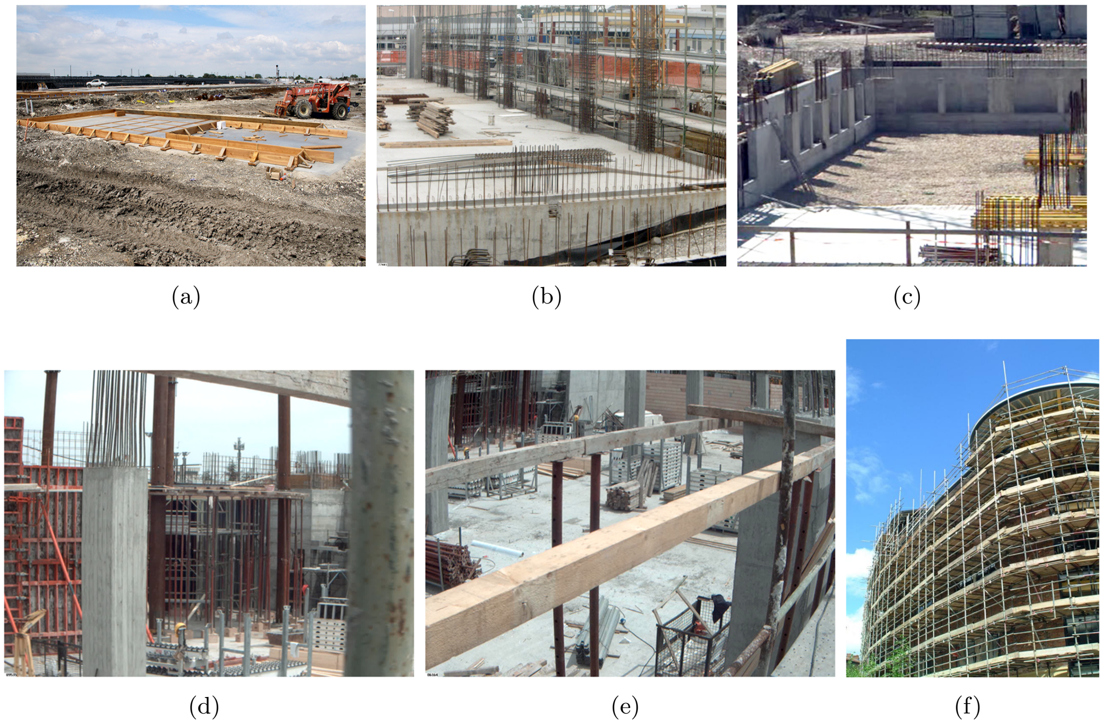


Figure A.1: Examples of the construction working site in its preliminary stage (a,b,c), in its main building stage (d,e) and during the inner works stage (e)

on LogitBoost [1], with a configuration similar to the INRIA-based detector (Section 10.2). The last 6 stages of the rejection cascade are removed and re-trained every night with the relevance feedback additional learning (Section 11.1), exploiting the implicit feeding on a sub-set of background images detected the day before. When user assessment is available, it is possible to recompute the additional cascades using explicit feeding also.

The pedestrian detection is performed with the multi-stage sampling strategy (Section 13). Given the reduced processing frame rate, we do not exploit the recursive Bayesian filtering, but at each detection step we perform sampling of the first sample set from a uniform proposal distribution (i.e. the method proposed for the detection on single images, Section 13.1). In case the camera operates in motion, the uniform distribution spans over the three dimensions of the window state space, namely (x, y) coordinates and scale; conversely, if the camera is fixed, it is possible to exploit motion and perspective (Section 12) in order to perform sampling over a portion of the state space. This is equivalent to use a non-uniform proposal pdf that takes into account motion and perspective: perspective gives a contribution that remains static until it is recomputed, while motion contributions changes frame by frame. The use of non-uniform proposal distribution further reduce the computational load of the detection step. Fig. A.3 depicts a few examples of pedestrian detection.

The head detector module receives full-resolution frames from MOVIDE and a set of bounding boxes from the pedestrian detector; using the covariance descriptor enhanced for circular features (Section 11.2) and exploiting the Lab color space derivatives (Section 11.3), this module performs the precise detection of the head on each full-resolution

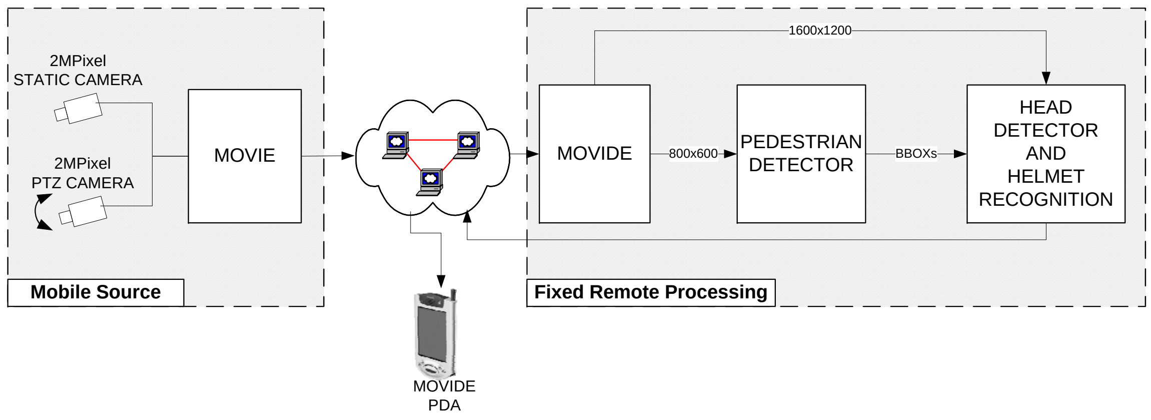


Figure A.2: Schema of the prototype for the automatic detection of workers in construction working sites that do not wear protective helmets.

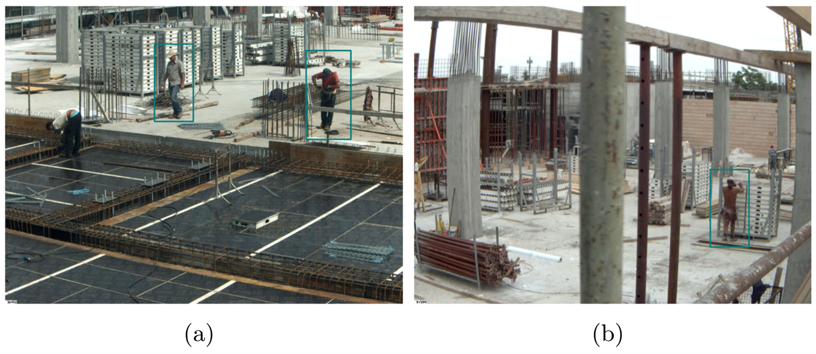


Figure A.3: Pedestrians detected in CSW in two different scenarios: (a) preliminary phase of CSW; (b) main building phase of CSW.

pedestrian patch: specifically the module runs a multi-stage sampling-based detection on the upper part of the body and in case the detector finds no potential heads (e.g. in case of false positive from the pedestrian detection) or more than one (e.g. in case of circular distractors), the patch is discarded. Otherwise the head detection is considered reliable and passed to the helmet recognition.

The main peculiarity that separates bare heads (or heads with headgears, like hats, caps, bandanas, etc.) from heads with helmets is the color. Indeed, security or protective helmets purposely have very bright and vivid colors that make them very visible. We trained a binary tree classifier on the Lab color space. The color is computed as the average on a small patch in the upper part of the detected head; Fig. A.4(a) shows the scattering of the training dataset for the helmet recognition over the Lab color opponent space; *a* and *b* are the most clearly discriminating features (Fig. A.4(b)), but white helmets can be distinguished from heads only using luminance (Fig. A.4(c),A.4(d)). However, if the prototype is deployed in scenarios where white helmets are not used, the binary tree classifier should be retrained using the two chromatic components only, since avoiding luminance makes the classifier more robust to lightning changes. We tested also other color spaces (RGB and HSV), verifying that Lab best separates the two classes.

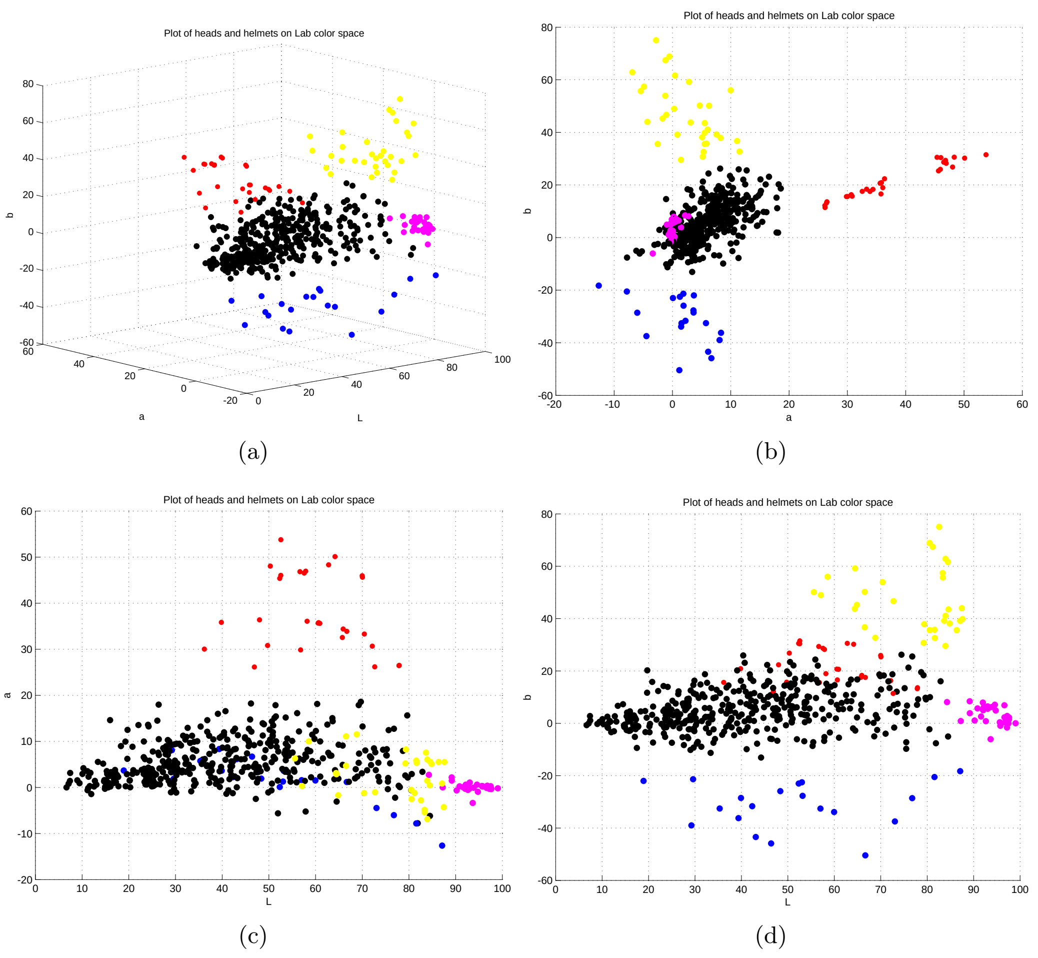


Figure A.4: Scattering of the color of the 527 patches (399 heads and 128 helmets) on the Lab color space; (a) 3d-view; (b) 2d-view of ab; (c) 2d-view of La; (c) 2d-view of Lb; black dots = bare heads or heads with headgears; blue,red,yellow dots = heads with helmet of corresponding color; magenta = white helmets.

A set of PDAs are connected via UMTS to the remote processing unit and receive real-time warnings on workers without protective helmets. Moreover they can down stream live VGA or QVGA video directly from the cameras, using Moses architecture.

A.3 DataSets and Performance Evaluation

The pedestrian classifier is made of a 26-stage rejection cascade trained on images taken from two datasets: the INRIA pedestrian training dataset and the *CWS dataset*, that is a pedestrian dataset that we generated using only images taken from construction working site scenarios: the training set is made of 1343 positive patches (i.e. 96x160 patches containing a centered worker of 64x128) and 639 person free images; the testing dataset is made of 323 positive patches and 173 person free images. We also generated a head dataset composed of 534 positive patches (i.e. 98x98 patches containing a centered head of 32x32) for training and 134 for testing. For the negative images we use the CSW dataset. The training dataset for the helmet recognition is made of 527 patches: 399 heads and 128 helmets.

We recorded approximately 20 hours of videos with 2 MPixel camera (Basler BIP-1600c) in a wide construction working site along the three stages of its development. The prototype has been configured with very conservative parameters over the two classifiers (pedestrian and head), in order to reduce as much as possible the number of false detections even at the cost of higher miss rates and heavier computational load.

The pedestrian detector has a recall of 56.1% and a precision of 95.2%. The head detector has a recall of 87% and a precision of 95.7% (see Fig. A.5). The helmet recognition misclassify approximately 10% of helmets with heads and vice versa. Removing the white helmets, misclassification drops to approximately 3%. Examples of the final results are provided in Fig. A.6. The prototype runs on a Quad-Core processor with 4GB RAM. The processing frame rate is approximately from 3 to 20 frames per minute, depending on the number of samples used in the detectors, on the use of priors, on the complexity of the visual scene and on the quantity of motion. It performs pedestrian detection on a 800x600 frame in approx. 7 seconds and each head detection in approx. 5 seconds. Given this working performances, the prototype is to be configured a sample checking tool and not as an exhaustive or pervasive detector.

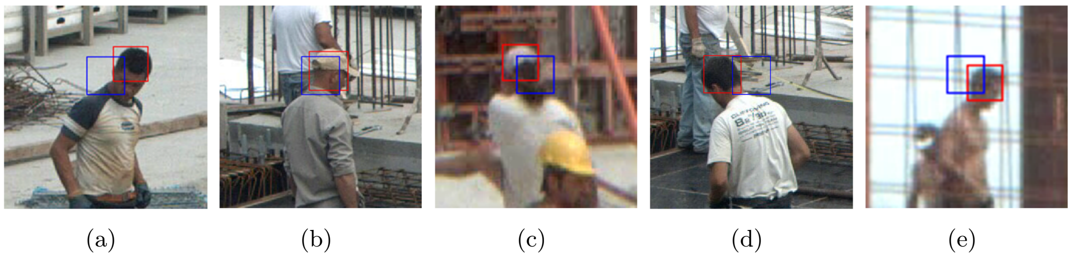


Figure A.5: Examples of head detection using the classifier with polar transformation and multi-spectral derivatives. The blue box represent the head position blindly estimated in a fixed position of the pedestrian detection bounding box; the red box shows the head position obtained with the head detector.

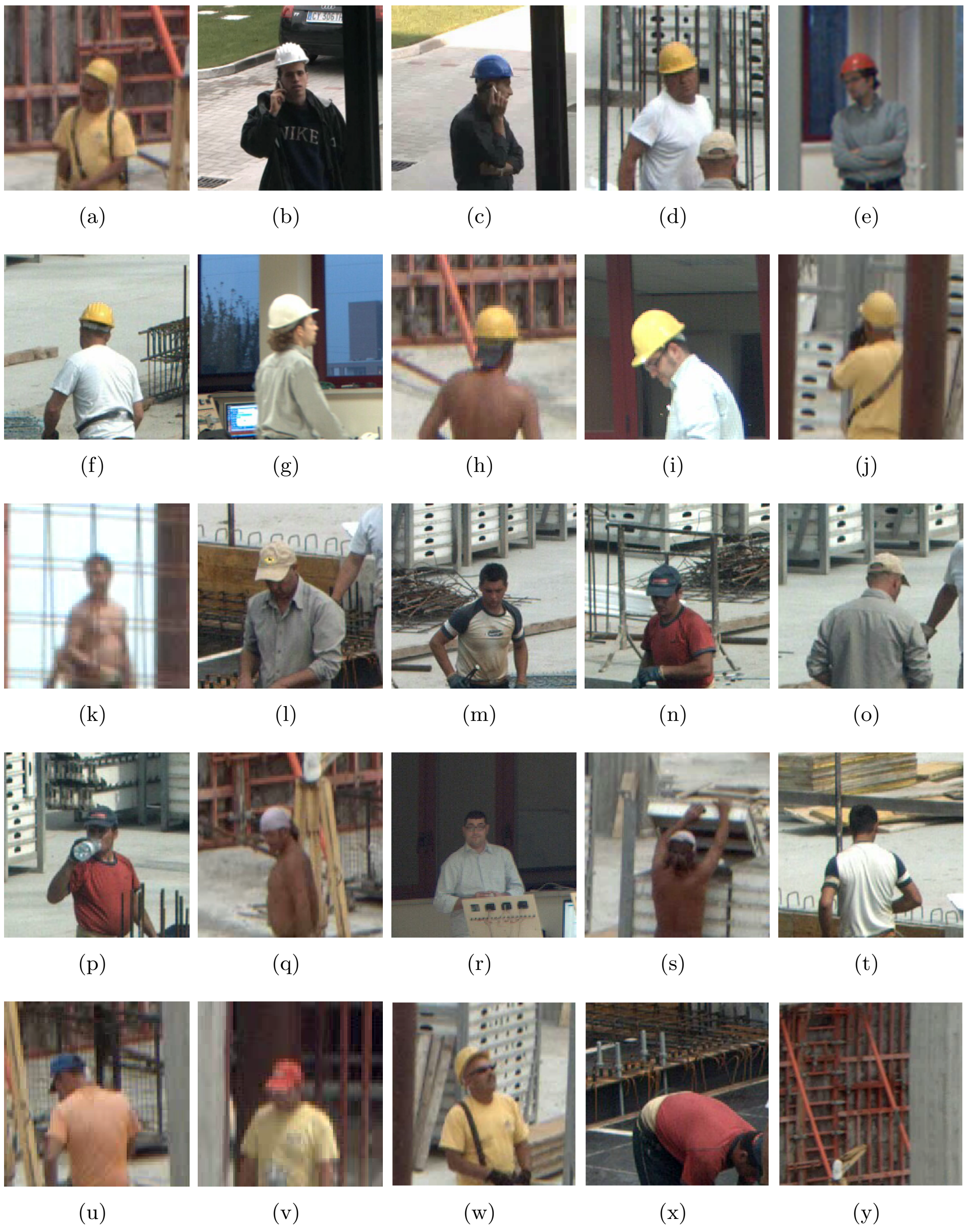


Figure A.6: Examples of final results (only upper body is shown): (a-j) correct detection and recognition of workers with helmet; (k-t) correct detection and recognition of workers with no helmet; (u-w) wrong helmet recognition (u,v are recognized as helmets, w is recognized as head); (x-y) wrong pedestrian detection that passes also through the head detection.

Appendix **B**

Open Source Video Encoding, Streaming and Decoding

In Chapter 4 we introduced the MOSES streaming system, which is divided in two applications: the encoding part, called MOVIE (Mobile Video Encoder) and the decoding part called MOVIDE (Mobile Video Decoder, see logos in Fig. B.1). We propose here a brief instruction manual for both applications that can be downloaded from the author's home page[1].

(a) (b)

Figure B.1: Movie and Movide logos.

B.1 MOVIE, Mobile Video Encoder

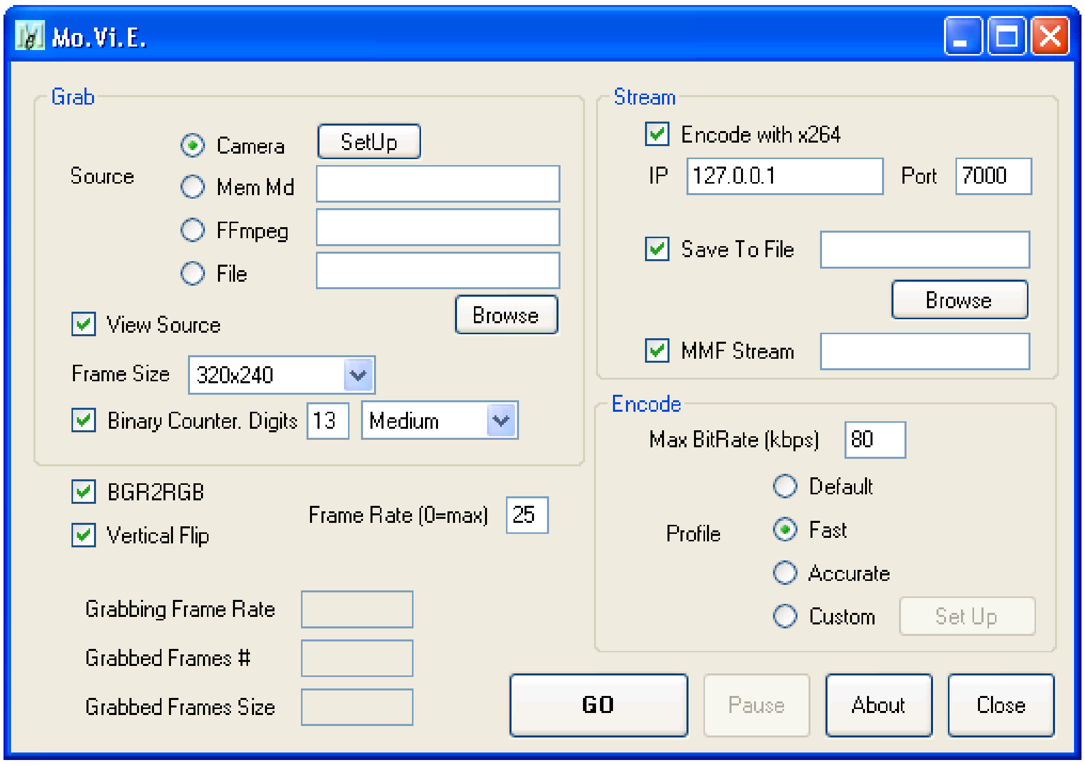
MOVIE is devoted to video grabbing, compression, streaming and file storage, plus minor video operations. We review here the configuration details that can be used in each of these four functionalities. Refer to Fig. B.2.

Video grabbing: the video grabbing functions can be controlled from the group *Grab* in the main window. The program can grab video from four different sources:

- USB WebCams (radio button *Camera*): the webcam driver must be properly installed in the system. Using the button *SetUp*, it is possible to configure the video parameters of the camera with the control panel provided by the vendor. In

Publications related to Appendix B: [a,c,d,f,k]; see the list of author's publications, page 147.

[1] imagelab.ing.unimore.it/imagelab/˜gualdi/


Mo.Vi.E.

Grab

Source: [✓ Camera] SetUp; Mem Md; FFmpeg; File; Browse

☑ View Source

Frame Size: 320x240

☑ Binary Counter. Digits: 13, Medium

☑ BGR2RGB

☑ Vertical Flip

Frame Rate (0=max): 25

Grabbing Frame Rate

Grabbed Frames #

Grabbed Frames Size

Stream

☑ Encode with x264

IP: 127.0.0.1 Port: 7000

☑ Save To File

Browse

☑ MMF Stream

Encode

Max BitRate (kbps): 80

Profile: Default; [✓ Fast]; Accurate; Custom; Set Up

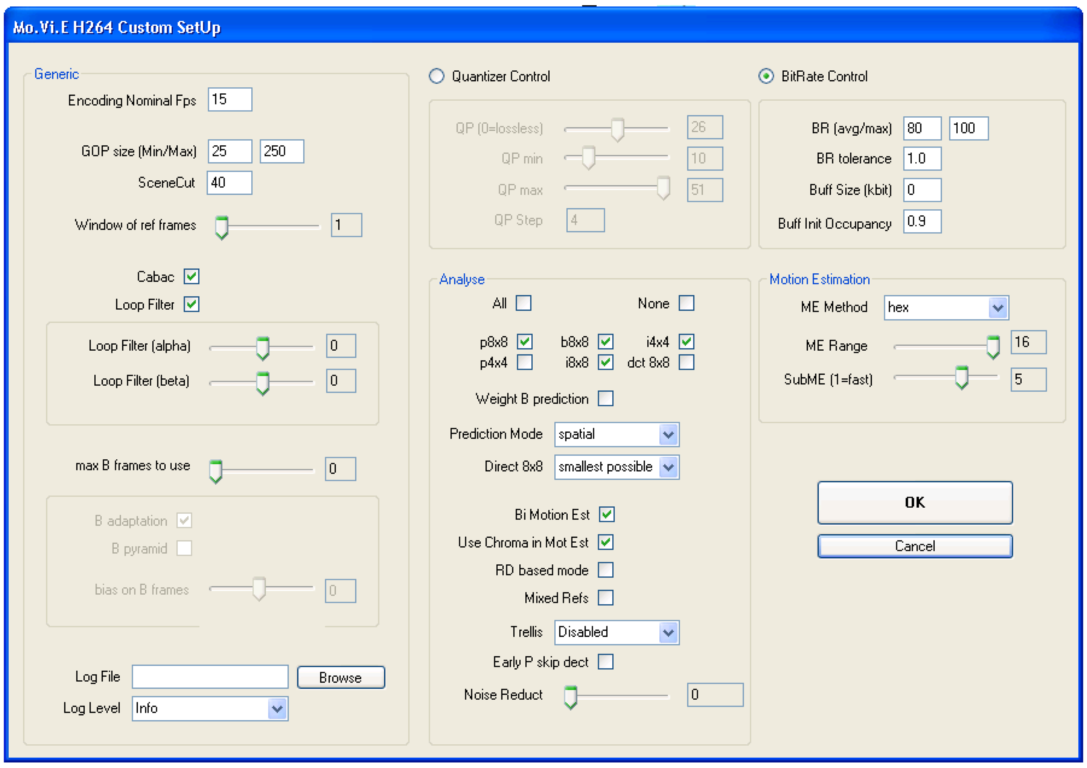
GO Pause About Close

Figure B.2: Movie main window.

case more than one usb camera is plugged in the system, it is possible to choose which one to use.

- Memory Mapped Files (radio button *Mem Md*): memory mapped files (MMFs) are the easiest interface offered by Moses for external applications to use its video encoding, streaming and decoding. In the side text box, type the name of the memory file to read video from. For details see Section B.3.
- FFMpeg Library (radio button *FFmpeg*): Movie is linked to FFMpeg and therefore can read any video from file or network stream that is supported by that library. In the text box it is possible to specify the file name (with absolute path), or the network URL, that can also contain authentication data (e.g. `rtsp://myIP/mpeg4` or `http://username:password@myIP/cgi-bin/mjpeg?buffer=0`). It is also possible to force FFMpeg to use a specific video format with the *-f* option that must be placed before the URL (e.g. `-f mjpeg http://myIP/mjpeg`).
- AVI or MPEG files (radio button *File*): this feature is supported by OpenCV file reading functions and uses the codecs installed in the Windows operating system.

Video compression: the video compression parameters can be controlled from the group *Encode* in the main window. It is possible to define a desired maximum bit rate, implicitly requiring Constant Bit Rate (CBR) coding; moreover four different encoding profiles can be used:


Mo.Vi.E H264 Custom SetUp

Generic
Encoding Nominal Fps: 15
GOP size (Min/Max): 25 250
SceneCut: 40
Window of ref frames: 1
Cabac ✓
Loop Filter ✓
Loop Filter (alpha): 0
Loop Filter (beta): 0
max B frames to use: 0
B adaptation ✓
B pyramid
bias on B frames: 0
Log File:
Log Level: Info

Quantizer Control [✓ BitRate Control]
QP (0=lossless): 26
QP min: 10
QP max: 51
QP Step: 4

BR (avg/max): 80 100
BR tolerance: 1.0
Buff Size (kbit): 0
Buff Init Occupancy: 0.9

Analyse
All ☐ None ☐
p8x8 ✓ b8x8 ✓ i4x4 ✓
p4x4 ☐ i8x8 ✓ dct 8x8 ☐
Weight B prediction ☐
Prediction Mode: spatial
Direct 8x8: smallest possible
Bi Motion Est ✓
Use Chroma in Mot Est ✓
RD based mode ☐
Mixed Refs ☐
Trellis: Disabled
Early P skip dect ☐
Noise Reduct: 0

Motion Estimation
ME Method: hex
ME Range: 16
SubME (1=fast): 5

Figure B.3: Movie x264 configuration panel.

- the Default profile performs encoding with the default options offered by the encoding library x264.
- the Fast profile corresponds to the baseline profile introduced in Chapter 5: it minimizes the CPU load at the cost of low video quality compression.
- the High profile is the opposite of baseline, i.e. highest video quality compression at the cost of higher CPU burden.
- the Custom profile let's the user freely choose most of the x264 encoding parameters with the panel control shown in Fig. B.3. It is possible to configure quantization control instead of bit rate control, motion estimation, P and B frames, filters, etc. Please refer to H.264 [47] standard and X.264 web site [2] for more details.

Video streaming and storage: the grabbed video can be relayed to MOVIDE or to other systems in three different ways:

- Enabling the check box *Encode with h264* it is possible to encode the grabbed video with x264 and dispatch it via UDP streaming to a specific IP address (and port); the details of this network streaming are reported in Section 4.1.
- Enabling the check box *Save to file* it is possible to save the grabbed video to file system. AVI or H264 file extensions can be chosen: in the first case OpenCV file

[2]URL: www.videolan.org/developers/x264.html

writing functions are exploited, using one of the codec installed in the Windows operating system; in the second case the X.264 encoder is silently enabled and a plain (i.e. free of any encapsulation) video stream is written to file system.

- Enabling the check box *MMF Stream* it is possible to forward the grabbed video in its RGB uncompressed version on a MMF: provide its name on the side text box. For details see Section B.3.

Other operations and informative boxes: regardless of the chosen video source, check box *Video Source* enables video browsing; check box *Binary Counter* enables the drawing of the binary coded frame numbers on each frame (see Section 5.1 for details); it is also possible to define the number and the size of the binary digits to print. Eventually, check box *BGR2RGB* changes the RGB/BGR coding and check box *Vertical Flip* performs vertical image flip; it is possible to change the grabbing frame rate through check box and text box *Frame Rate*: the value 0 means that the system processes at the highest possible frame rate. The system reports the grabbing frame rate (i.e. live update), frame size and frame number in the three respective boxes.

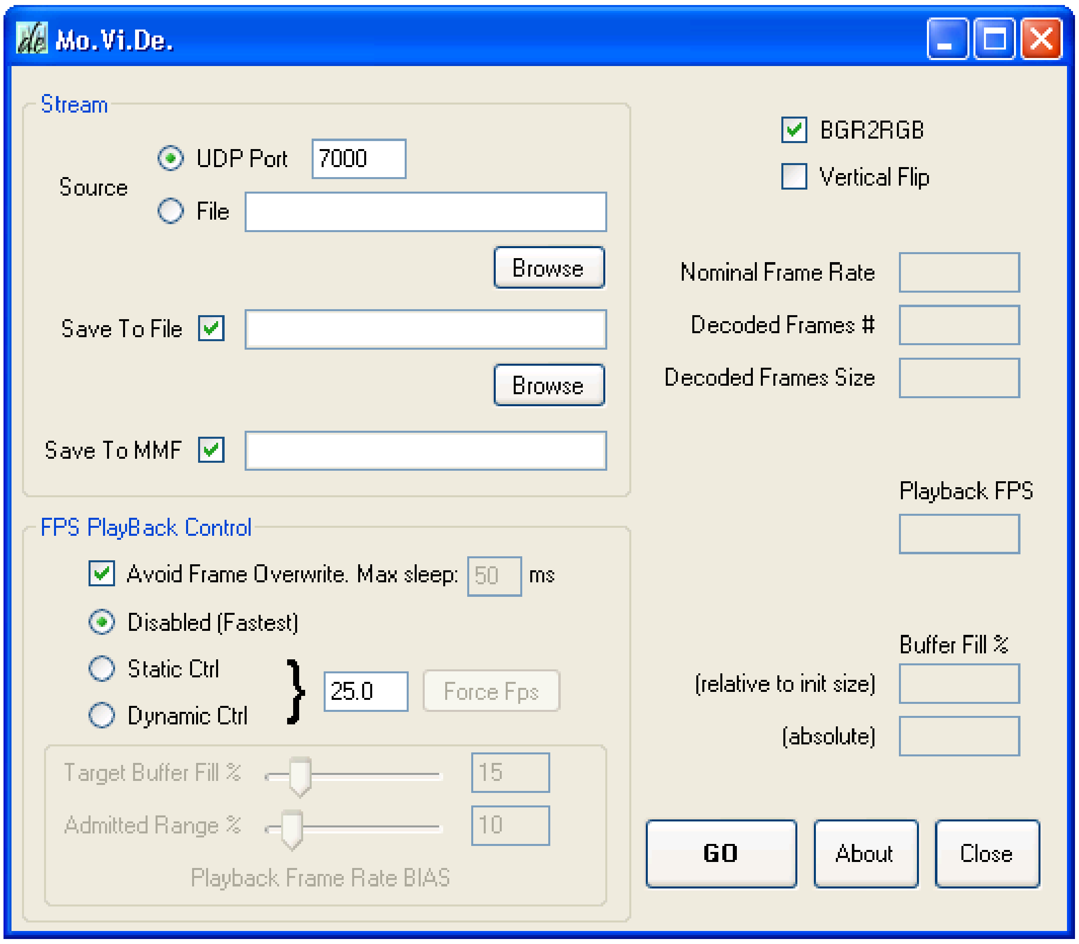
B.2 MOVIDE, Mobile Video Decoder

MOVIDE is devoted to H.264 video down streaming, file reading and optimized playback, plus minor video operations. Refer to Fig. B.4.

H.264 video down streaming and file reading: it is possible to down stream video from a UDP port (radio button *UDP*) or to read from a file (radio button *File*): any H.264 compliant video stream is supported, provided the fact that no encapsulation method is added to the streamed data. In the group *Stream* there are two more options, that are used for exporting the video stream: specifically it is possible to save the video stream to file (check box *Save To File*), typically used when the stream is downloaded from the network and needs to be recorded, or to forward it to other applications using MMF (check box *Save To MMF*). For details see Section B.3.

Optimized playback: the options for controlling the playback frame rate are inside the group *FPS PlayBack Control*; the theory behind them are thoroughly reported in Section 4.2. It is possible to enable or disable the decoder-display coupling (check box *Avoid Frame Overwrite*), defining also the maximum waiting time (in milliseconds) for enabling eventual overwriting (text box *Max Sleep*). The adaptive playback frame rate is controlled through the three radio buttons: *Disabled* plays the video at the maximum possible speed; *Static Ctrl* configures a static frame rate (traditional playback); *Dynamic Ctrl* enables the adaptive play back frame rate: the system will smoothly adapt the frame rate in order to reach the target buffer occupancy defined with the slide bar *Target Buffer Fill*; the slide bar *Admitted Range* defines its tolerated range; in other words these two values define T_L and T_H of Eq. 4.1. Using the button *Force Fps* it is possible to force an instantaneous change of playback frame rate.

Other operations and informative boxes: it is possible to change RGB/BGR coding (check box *BGR2RGB*) and to perform vertical image flip (check box *Vertical Flip*).



Mo.Vi.De.

Stream

Source: [✓ UDP Port] 7000 / File

Save To File ✓

Save To MMF ✓

FPS PlayBack Control

✓ Avoid Frame Overwrite. Max sleep: 50 ms

[✓ Disabled (Fastest)] / Static Ctrl / Dynamic Ctrl } 25.0 Force Fps

Target Buffer Fill % 15

Admitted Range % 10

Playback Frame Rate BIAS

✓ BGR2RGB

Vertical Flip

Nominal Frame Rate

Decoded Frames #

Decoded Frames Size

Playback FPS

Buffer Fill %

(relative to init size)

(absolute)

Figure B.4: Movide main window.

The text box *Nominal Frame Rate* reports the video frame rate reported inside the H.264 stream, while the *Playback FPS* is the real actual playback frame rate. The two boxes *Buffer Fill, relative to initial size* and *Buffer Fill, absolute* report the actual buffer occupancy in percentage: the first w.r.t. the size of the buffer at the beginning of the streaming session, the second w.r.t. the actual buffer size.

B.3 Video Sharing through Memory Mapped Files

A memory mapped file (MMF) is a segment of virtual memory which has been assigned a direct byte-for-byte correlation with a file; it is possible in this way to perform very efficient inter process communication, with performance comparable to shared memory among threads. From the author's web site it is possible to download the DLL that creates a MMF mono directional communication channel for sharing uncompressed RGB frames. The DLL supports a front-end (video provider) and a back-end (video receiver). The application that provides the video creates the MMF with a specific name, writes a sequence of frames and eventually closes its side of the MMF. The video receiver instead, opens the MMF (it has to be previously created by the provider), reads frames from it, and eventually closes its own side of the MMF. Upon receiving a frame, the application has the possibility to know from the DLL if the frame is new or has been downloaded from MMF already.

Appendix C

Proofs for Dominant Set Framework

Definition 1 *A* structural model *of an object is a connected graph $G_m = (P, S)$ where P is the set of distinct parts we want to use to represent the object and $S \subseteq P \times P$ are their structural relations, where $(p_a, p_b) \in S$ if and only if p_a and p_b are joined in the object.*

Definition 2 *Given a structural model $G_m = (P, S)$, a set of features clusters C assigned to $|P|$ classes by a surjective labelling function $l : C \to P$ and their attributes A, we define the* labeled graph *as the $|P|$-partite graph $G = (C, E, A, l)$ where C is the vertex set, $E = \{(u, v) \in C \times C | (l(u), l(v)) \in S\}$ the edge set, A the vertex attributes and l the vertex labelling function.*

Definition 3 *Given labeled graphs $G_1 = (C_1, E_1, A_1, l_1)$ and $G_2 = (C_2, E_2, A_2, l_2)$ a* labeled isomorphism *between them is a relation $M \subseteq C_1 \times C_2$ such that for each $(u_1, u_2), (v_1, v_2) \in M$, with $u_1, v_1 \in C_1$ and $u_2, v_2 \in C_2$, the following properties hold:*

$$l_1(u_1) = l_2(u_2) \wedge l_1(v_1) = l_2(v_2) \tag{C.1}$$

and

$$u_1 = v_1 \Leftrightarrow u_2 = v_2 \tag{C.2}$$

Definition 4 *Given the set of edges matches of a labeled isomorphism M:*

$$e(M) = \{[(u_1, v_1), (u_2, v_2)] \in E_1 \times E_2 | (u_1, u_2) \in M \wedge (v_1, v_2) \in M\}$$

and let $\omega : (E_1 \times E_2) \times (E_1 \times E_2) \to \mathbb{R}^+$ be a measure of coherence between pairs of edges matches, then the total weight of M is defined as:

$$\Omega(M) = \sum_{a \in e(M)} \sum_{b \in e(M) \setminus \{a\}} \omega(a, b) \,. \tag{C.3}$$

Definition 5 *Given labeled graphs $G_1 = (C_1, E_1, A_1, l_1)$ and $G_2 = (C_2, E_2, A_2, l_2)$ and a function $\omega : (E_1 \times E_2) \times (E_1 \times E_2) \to \mathbb{R}^+$ that measures the coherence between pairs of edge associations, we define an association graph between them as an edge weighted*

Publications related to Appendix C: [g]; see the list of author's publications, page 147.

graph $Ga = (Va, Ea, \omega)$ where $Va = E_1 \times E_2$, $Ea \subset Va \times Va$ and, $\forall u_1, v_1, w_1, z_1 \in C_1$ and $u_2, v_2, w_2, z_2 \in C_2$, the element:

$$\{[(u_1, v_1), (u_2, v_2)], [(w_1, z_1), (w_2, z_2)]\}$$

belongs to Ea if and only if the following three conditions hold:

$$l_1(u_1) = l_2(u_2) \wedge l_1(v_1) = l_2(v_2) \wedge l_1(w_1) = l_2(w_2) \wedge l_1(z_1) = l_2(z_2), \tag{C.4}$$
$$u_1 = w_1 \Leftrightarrow u_2 = w_2, \tag{C.5}$$
$$v_1 = z_1 \Leftrightarrow v_2 = z_2 \tag{C.6}$$

Definition 6 *In order to obtain a relation between vertices in V_1 and V_2 we define a natural map $v : \mathcal{P}(V_a) \to \mathcal{P}(V_1 \times V_2)$ as:*

$$v(X) = \{(u_1, u_2) \in V_1 \times V_2 | [(u_1, v_1), (u_2, v_2)] \in X \vee [(v_1, u_1), (v_2, u_2)] \in X\}$$

Lemma 3 *Given labeled graphs G_1, G_2 and their association graph G_a, $X \subseteq V_a$ is a clique if and only if $v(X)$ is a labeled isomorphism between G_1 and G_2.*

Proof We will prove both the implications by contrapposition. First we will prove that if X is not a clique then $v(X)$ is not a labeled isomorphism:

Suppose that X is not a clique. This means that it contains at least a pair of vertices $[(u_1, v_1), (u_2, v_2)] \in X$ and $[(w_1, z_1), (w_2, z_2)] \in X$ that do not satisfy either (C.4) or (C.6). Those vertices induce associations (u_1, u_2), (v_1, v_2), (w_1, w_2) and (z_1, z_2) in $v(X)$. Suppose that (C.4) is not satisfied, this means that some label correspondence is not satisfied. For instance, this happens if $l(u_1) \neq l(u_2)$, thus in this case (C.1) is not satisfied by the induced association (u_1, u_2). The same happens if $l(v_1) \neq l(v_2)$, $l(w_1) \neq l(w_2)$ or $l(z_1) \neq l(z_2)$. Suppose that (C.6) is not satisfied, this means that some injection condition is not satisfiend. For instance, this happens if $u_1 = w_1$ and $u_2 \neq w_2$, this in this case (C.2) is not satisfied by the induced associations (u_1, u_2) and (w_1, w_2). The same happens for the others conditions. In both cases $v(X)$ is not a label isomorphism, which proves our claim.

Now we will prove that if $v(X)$ is not a labeled isomorphism between G_1 and G_2 then X is not a clique:

Suppose that $v(X)$ is not a label isomorphism between G_1 and G_2. This means that it contains a pair of associations $(u_1, u_2) \in v(X)$ and $(w_1, w_2) \in v(X)$ that do not satisfy either (C.1) or (C.2). For such matches to be in $v(X)$ it means (by Def. 6) that either $[(u_1, v_1), (u_2, v_2)] \in X$ or $[(v_1, u_1), (v_2, u_2)] \in X$ for some $v_1 \in V_1$ and $v_2 \in V_2$ and that either $[(w_1, z_1), (w_2, z_2)] \in X$ or $[(w_1, z_1), (w_2, z_2)] \in X$ for some $z_1 \in V_1$ and $z_2 \in V_2$. Suppose that (C.1) is not satisfied because $l(u_1) \neq l(u_2)$ and that $((u_1, v_1), (u_2, v_2)) \in X$, in this case neither (C.4) is satisfied and $((u_1, v_1), (u_2, v_2))$ is not adjacent to any other vertex in G_a, thus X is not a clique in G_a. Other cases that do not satisfy (C.1) are similar. Similar observations show that if (C.2) is not satisfied a pair of vertices not connected by an edge exist in X. In both cases X is not a clique in G_a, which proves our claim.

Lemma 4 *If $X \subseteq V_a$ is a maximal clique in G_a, then $v(X)$ is a maximal labeled isomorphism between G_1 and G_2. Conversely, if M is a maximal labeled isomorphism between G_1 and G_2 then $e(M)$ is a maximal clique in G_a.*

Proof We will prove both the implications by contrapposition. First we will prove that if $v(X)$ is not a labeled isomorphism then X is not a maximal clique:

Suppose that $v(X)$ is a labeled isomorphism between G_1 and G_2 but it is not maximal. This means that the association (u_1, u_2) can be added to $v(X)$ without invalidating the conditions (C.1) or (C.2). We know from Def. 1 that the model is connected and from Def. 2 that for each part in the model at least one vertex in G_1 and G_2 is associated to the corresponding label by l. This guarantees that at least $(u_1, v_1) \in E_1$, $(u_2, v_2) \in E_2$ or $(v_1, u_1) \in E_1$, $(v_2, u_2) \in E_2$ for some v_1 and v_2 with $l(v_1) = l(v_2)$. We know form Lemma 1 that X is a clique in G_a. In addition, in either cases the associations between those edges can be added to X without invalidating (C.4), as the labelling correspondence for u_1,u_2 is guaranteed and v_1, v_2 can be choosen of the same class. Also (C.6) will not be invalidated by the new associations, as if v_1 is not yet in X any compatible v_2 will not invalidate it, otherwise the already present v_2 can be choosen. This shows that X is not maximal, which prove our claim.

Now we will prove that if $e(M)$ is not a maximal clique then M is not a maximal labeled isomorphism.

Suppose that $e(M)$ is a clique in G_a but it is not maximal. Since by Def. 4 $e(M)$ contains all the edge associations between correspondent vertices in M if a new edge association can be added to it without invalidating (C.4) and (C.6), then it induces new vertices associations in M that do not invalidate (C.1) and (C.2). Thus M is not maximal, which prove our claim.

Theorem 2 *Given two feature graphs G_1 and G_2, each maximal(maximum) weight labeled isomorphism M between them induces a maximal(maximum) edge weight clique in $Ga(G_1, G_2)$ and vice versa.*

Proof Lemma 2 proves the maximality correspondence. In addition, Def. 4 states that the weight of the label isomorphism M is exactly the sum of the weights assigned to the edges of $e(M)$ by the similarity function ω. Thus, for each maximal(maximum) labeled isomorphism M a maximal(maximum) edge weight clique $e(M)$ of exactly the same weight exists. In addition, it is easy to see that if X is maximal then $X = e(v(X))$, thus for each maximal(maximum) edge weight clique X in Ga a maximal(maximum) labeled isomorphism $v(X)$ of exactly the same weight exists.

Bibliography

[1] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian Detection via Classification on Riemannian Manifolds," *IEEE Trans. on PAMI*, vol. 30, no. 10, pp. 1713–1727, Oct 2008.

[2] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 304–311.

[3] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Trans. on Systems, Man, and Cybernetics - Part C*, vol. 34, 2004.

[4] S. Li, X. Wang, M. Li, and X. Liao, "Using Cable-Based Mobile Sensors to Assist Environment Surveillance," *Parallel and Distributed Systems. 14th IEEE International Conference on*, pp. 623–630, 2008.

[5] F. Bashir and F. Porikli, "Collaborative tracking of objects in EPTZ cameras," *Visual Communications and Image Processing 2007*, vol. 6508, no. 1, 2007.

[6] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A System for Video Surveillance and Monitoring," Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, 2000.

[7] C. Stauffer and W.E.L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 747–757, 2000.

[8] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. on PAMI*, vol. 22, pp. 809–830, 2000.

[9] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting Moving Objects, Ghosts and Shadows in Video Streams," *IEEE Trans. on PAMI*, vol. 25, no. 10, pp. 1337–1342, 2003.

[10] P. Smith, T. Drummond, and R. Cipolla, "Layered Motion Segmentation and Depth Ordering by Tracking Edges.," *IEEE Trans. on PAMI*, vol. 26, no. 4, pp. 479–494, 2004.

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

[12] R. Vezzani and R. Cucchiara, "AD-HOC: Appearance Driven Human tracking with Occlusion Handling," in *Tracking Humans for the Evaluation of their Motion in Image Sequences. First Int'l Workshop*, 2008.

[13] A. Senior, A. Hampapur, Y.L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *Image and Vision Computing*, vol. 24, no. 11, pp. 1233–1243, 2006.

[14] S. Calderara, R. Cucchiara, and A. Prati, "Bayesian-competitive Consistent Labeling for People Surveillance," *IEEE Trans. on PAMI*, vol. 30, no. 2, pp. 354–360, 2008.

[15] R. Cucchiara, A. Prati, and R. Vezzani, "Advanced Video Surveillance with Pan Tilt Zoom Cameras," in *Workshop on Visual Surveillance*, 2006.

[16] J.W. Davis, A.M. Morison, and D.D. Woods, "An adaptive focus-of-attention model for video surveillance and monitoring," *Mach. Vision Appl.*, vol. 18, no. 1, pp. 41–64, 2007.

[17] A. Del Bimbo and F. Pernici, "Uncalibrated 3D Human Tracking With A Ptz-Camera Viewing A Plane," in *Proc. 3DTV International Conference: Capture, Transmission and Display of 3D Video*, 2008.

[18] G.L. Foresti, "Object recognition and tracking for remote video surveillance," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 9, no. 7, pp. 1045–1062, 1999.

[19] M.M. Trivedi, T.L. Gandhi, and K.S. Huang, "Distributed interactive video arrays for event capture and enhanced situational awareness," *Intelligent Systems, IEEE*, vol. 20, no. 5, pp. 58–66, 2005.

[20] G.T. Kogut and M.M. Trivedi, "Maintaining the Identity of Multiple Vehicles as They Travel Through a Video Network," in *Proceedings of IEEE Workshop on Multi-Object Tracking*, 2001, p. 29.

[21] B.A. Stancil, C. Zhang, and T. Chen, "Active Multicamera Networks: From Rendering to Surveillance," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 4, pp. 597–605, 2008.

[22] K. Janusz, "Videopsy: dissecting visual data in space-time," *Communications Magazine, IEEE*, vol. 45, no. 1, pp. 34–42, 2007.

[23] Z. Zhang, P. Venetianer, and A. Lipton, "A Robust Human Detection and Tracking System Using a Human-Model-Based Camera Calibration," in *Proceedings of 8th Int'l Workshop on Visual Surveillance*, 2008.

[24] F. Porikli and O. Tuzel, "Object tracking in low-frame-rate video," *SPIE Image and Video Comm. and Processing*, vol. 5685, pp. 72–79, 2005.

[25] G. Tsagkatakis and A. Savakis, "Face detection in power constrained distributed environments," in *Proceedings of 1st Int'l Workshop on Mobile Multimedia Processing*, 2008.

[26] X. Liu, D. Doermann, and H. Li, “Camera phone based tools for the visually impaired,” in *Proceedings of 1st Int’l Workshop on Mobile Multimedia Processing*, 2008.

[27] W.S. Leoputra, S. Venkatesh, and T. Tan, “Pedestrian detection for mobile bus surveillance,” *Control, Automation, Robotics and Vision. 10th International Conference on*, pp. 726–732, 2008.

[28] M. Andriluka, S. Roth, and B. Schiele, “People-tracking-by-detection and people-detection-by-tracking,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, June 2008, pp. 1–8.

[29] C. Bibby and I. Reid, “Robust Real-Time Visual Tracking Using Pixel-Wise Posteriors,” in *Proc. of European Conference on Computer Vision*, 2008, pp. 831–844.

[30] R. Kaucic, A.G. Amitha Perera, G. Brooksby, J. Kaufhold, and A. Hoogs, “A unified framework for tracking through occlusions and across sensor gaps,” in *Proc. of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 990–997.

[31] G. Klein and D. Murray, “Improving the Agility of Keyframe-Based SLAM,” in *Proc. of European Conference on Computer Vision*, 2008, pp. 802–815.

[32] E. Silverman, R. Simmons, F. Gelhaus, and J. Lewis, “Surveyor: A remotely operated mobile surveillance system,” *Robotics and Automation. Proceedings. IEEE International Conference on*, vol. 3, pp. 1936–1940, 1986.

[33] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, “Distributed Embedded Smart Cameras for Surveillance Applications,” *Computer*, vol. 39, no. 2, pp. 68, 2006.

[34] D.C. Silva, T.F. Pereira, and V.V. Moreira, “ERS-210 Mobile Video Surveillance System,” *Artificial intelligence, Portuguese Conference on*, pp. 262–265, 2005.

[35] D. Ramanan, D.A. Forsyth, and A. Zisserman, “Tracking People by Learning Their Appearance,” *IEEE Trans. on PAMI*, vol. 29, no. 1, pp. 65–81, 2007.

[36] A.G.A. Perera, R. Collins, and A. Hoogs, “Evaluation of compression schemes for wide area video,” in *Applied Imagery Pattern Recognition. IEEE Workshop on*, 2008, pp. 1–6.

[37] O. Al-Baker, R. Benlamri, and A. Al-Qayedi, “A GPRS-based remote human face identification system for handheld devices,” *Wireless and Optical Communications Networks. IFIP International Conference on*, 2005.

[38] L. Marcenaro, F. Oberti, G.L. Foresti, and C.S. Regazzoni, “Distributed architectures and logical-task decomposition in multimedia surveillance systems,” *Proceedings of the IEEE*, vol. 89, no. 10, 2001.

[39] Y.K. Wang, L.Y. Wang, and Y.H. Hu, “A mobile video surveillance system with intelligent object analysis,” *Multimedia on Mobile Devices*, vol. 6821, no. 1, 2008.

[40] Y. Imai, Y. Hori, and S. Masuda, “Development and a brief evaluation of a web-based surveillance system for cellular phones and other mobile computing clients,” *Human System Interactions. Conference on*, pp. 526–531, 2008.

[41] T. Raty, L. Lehikoinen, and F. Bremond, “Scalable Video Transmission for a Surveillance System,” *Information Technology: New Generations. Int’l Conference on*, pp. 1011–1016, 2008.

[42] O. Steiger, T. Ebrahimi, and A. Cavallaro, “Surveillance video for mobile devices,” *Advanced Video and Signal Based Surveillance. IEEE Conference on*, pp. 620–625, 2005.

[43] Y.C. Tseng, Y.C. Wang, K.Y. Cheng, and Y.Y. Hsieh, “iMouse: An Integrated Mobile Surveillance and Wireless Sensor System,” *Computer*, vol. 40, no. 6, pp. 60–66, 2007.

[44] Bing-Fei Wu, Hsin-Yuan Peng, Chao-Jung Chen, and Yi-Huang Chan, “An encrypted mobile embedded surveillance system,” *Intelligent Vehicles Symposium, Proceedings. IEEE*, pp. 502–507, 2005.

[45] G. Meng, M.H. Ammar, and E.W. Zegura, “V3: a vehicle-to-vehicle live video streaming architecture,” in *Proc. of IEEE Intl Conf on Pervasive Computing and Communications*, 2005, pp. 171–180.

[46] K.P. Lim, D. Wu, S. Wu, R. Susanto, X. Lin, L. Jiang, R. Yu, F. Pan, Z. Li, S. Yao, G. Feng, and C.C. Ko, “Video streaming on embedded devices through GPRS network,” in *Proc. of IEEE Intl Conference on Multimedia and Expo*, 2003, vol. 2, pp. 169–172.

[47] “Advanced Video Coding for Generic Audiovisual Services,” Tech. Rep., ITU Rec. H624/ISO IEC 14996-10 AVC, 2003.

[48] T. Wiegand, G.J. Sullivan, G. Bjntegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, July 2003.

[49] A. Puri, X. Chen, and A. Luthra, “Video coding using the H.264/MPEG-4 AVC compression standard,” *Signal Processing: Image Communication*, vol. 19, pp. 793–849, 2004.

[50] A. MacAulay, B. Felts, and Y. Fisher, “WHITEPAPER IP Streaming of MPEG-4: Native RTP vs MPEG-2 Transport Stream,” Tech. Rep., Envivio, Inc., Oct. 2005.

[51] J. Lu, “Signal processing for Internet video streaming - A review,” in *Proc. of Conf on Image and video communications and processing*, 2000, pp. 246–259.

[52] Meng-Ting Lu, Chang-Kuan Lin, J. Yao, and H. Chen, “Complexity-aware live streaming system,” in *Proc. of IEEE Int’l Conference on Image Processing*, 2005, vol. 1, pp. 193–196.

[53] Z. He, "Resource allocation and performance analysis of wireless video sensors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 5, pp. 590–599, 2006.

[54] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik, "Video coding for streaming media delivery on the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 269–281, Mar. 2001.

[55] L. Zhixiong and H. Guiming, "An embedded adaptive live video transmission system over GPRS/CDMA network," in *Proc. of Intl Conf. on Embedded Software and Systems*, 2005.

[56] H.C. Chuang, C. Huang, and T. Chiang, "Content-Aware Adaptive Media Playout Controls for Wireless Video Streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 6, pp. 1273–1283, 2007.

[57] P. Mahonen, "Wireless Video Surveillance: System Concepts," in *Proceedings of International Conference on Image Analysis and Processing*, 1999, pp. 1090–1095.

[58] D. Agrafiotis, T.K. Chiew, P. Ferre, D.R. Bull, A.R. Nix, A. Doufexi, J. Chung-How, and D. Nicholson, "Seamless wireless networking for video surveillance applications," in *Proceedings of SPIE - The International Society for Optical Engineering, 5685 (PART 1)*, 2005, pp. 39–53.

[59] X. Cai, F.H. Ali, and E. Stipidis, "MPEG4 over local area mobile surveillance system," *IEE Colloquium (Digest)*, vol. 3-10062, pp. 81–83, 2003.

[60] C.F. Wong, W.L. Fung, C.F.J. Tang, and S.H.G. Chan, "TCP streaming for low-delay wireless video," in *Second International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks*, 2005.

[61] K.Y. Lam and C.K.H. Chiu, "The Design of a Wireless Real-time Visual Surveillance System," *Multimedia Tools and Applications*, vol. 33, no. 2, pp. 175–199, 2007.

[62] W. Ajib and P. Godlewski, "Acknowledgment procedures at radio link control level in GPRS," *Wirel. Netw.*, vol. 7, no. 3, pp. 237–247, 2001.

[63] N. Damera-Venkata, T.D. Kite, W.S. Geisler, B.L. Evans, and A.C. Bovik, "Image quality assessment based on a degradation model," *Image Processing, IEEE Transactions on*, vol. 9, no. 4, pp. 636–650, 2000.

[64] J. Brunheroto, R. Chernock, P. Dettori, X. Dong, J. Paraszczak, F. Schaffa, and D. Seidman, "Issues in data embedding and synchronization for digital television," in *Proc. of IEEE Intl Conf. on Multimedia and Expo*, 2000, vol. 3, pp. 1233–1236.

[65] B.K. Schmidt, J.D. Northcutt, and M.S. Lam, "A Method and Apparatus for Measuring Media Synchronization," *Lecture Notes in Computer Science*, vol. 1018, pp. 190–202, 1995.

[66] C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Trans. on PAMI*, vol. 23, no. 1, pp. 54–72, 2001.

[67] Y. Bar-Shalom and T. Foreman, *Tracking and Data Association*, Acad.Pr.Inc, 1988.

[68] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 2000, vol. 2, pp. 142–149.

[69] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," *Int'l Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[70] M. Bartalmio, G. Sapiro, and G. Randall, "Morphing active contours," *IEEE Trans. on PAMI*, vol. 22, no. 7, pp. 733–737, 2000.

[71] P.F. Felzenszwalb and D.P. Huttenlocher, "Pictorial Structures for Object Recognition," *Int'l Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[72] M. Isard, "PAMPAS: real-valued graphical models for computer vision," *Proc. of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 613–620, 2003.

[73] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky, "Nonparametric belief propagation," *Proc. of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 605–612, 2003.

[74] L. Sigal, S. Bhatia, S. Roth, M.J. Black, and M. Isard, "Tracking loose-limbed people," *Proc. of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 421–428, 2004.

[75] C. Gomila and F. Meyer, "Graph-based object tracking," *Proc. ICIP*, vol. 2, pp. 41–44, 2003.

[76] D. Conte, P. Foggia, J.M. Jolion, and M. Vento, "A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions," *Pattern Recognition*, vol. 39, no. 4, pp. 562–572, 2006.

[77] E. L. Andrade Neto, E. Khan, J.C. Woods, and M. Ghanbari, "Segmentation and tracking for interactive sport scenes using region adjacency graphs, picture trees and prior information," in *Picture Coding Symposium (PCS) 2003*, 2003, pp. 353–358.

[78] A. Graciano, R. Cesar-Jr, and I. Bloch, "Graph-based Object Tracking Using Structural Pattern Recognition," *Proc. of SIBGRAPI*, pp. 179–186, 2007.

[79] D.S. Jang, S.W. Jang, and H.I. Choi, "2D human body tracking with Structural Kalman filter," *Pattern Recognition*, vol. 35, no. 10, pp. 2041–2049, 2002.

[80] G. Schindler and F. Dellaert, "A Rao-Blackwellized Parts-Constellation Tracker," in *WDV*, 2006, pp. 178–189.

[81] F. Tang and H. Tao, "Object tracking with dynamic feature graph," *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 25–32, 15-16 Oct. 2005.

[82] H. Borotschnig, D. Sinclair, and A. Pinz, "Fuzzy Graph Tracking," in *5th International Symposium on Intelligent Robotic Systems*, 1997, pp. 91–101.

[83] D. Terzopoulos, A.P. Witkin, and M. Kass, "Energy Constraints on Deformable Models: Recovering Shape and Non-Rigid Motion," in *AAAI*, 1987, pp. 755–760.

[84] D. Terzopoulos, J.C Platt, A.H. Barr, and K.W. Fleischer, "Elastically deformable models," in *SIGGRAPH*, 1987, pp. 205–214.

[85] M. Pavan and M. Pelillo, "Dominant Sets and Pairwise Clustering," *IEEE Trans. on PAMI*, vol. 29, no. 1, pp. 167–172, 2007.

[86] A. Prati, I. Mikic, M.M. Trivedi, and R. Cucchiara, "Detecting Moving Shadows: Algorithms and Evaluation," *IEEE Trans. on PAMI*, vol. 25, no. 7, pp. 918–923, July 2003.

[87] R. Cucchiara, C. Grana, G. Tardini, and R. Vezzani, "Probabilistic People Tracking for Occlusion Handling," in *Proc. of Int'l Conference on Pattern Recognition (ICPR 2004)*, Aug. 2004, vol. 1, pp. 132–135.

[88] G.R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," in *Proc. of IEEE Workshop on Applications of Computer Vision*, 1998, pp. 214 – 219.

[89] M. Pelillo, "Replicator Equations, Maximal Cliques, and Graph Isomorphism," *Neural Comput.*, vol. 11, no. 8, pp. 1933–1955, 1999.

[90] S. Rota Bulò, A. Torsello, and M. Pelillo, "A Continuous-Based Approach for Partial Clique Enumeration," in *GbRPR*, 2007, pp. 61–70.

[91] K.V. Mardia and P.E. Jupp, *Directional Statistics*, Wiley, 2000.

[92] K. Nummiaro, E. Koller-Meier, and L.J. Van Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99–110, 2003.

[93] D. M. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, Jan 1999.

[94] S. Ioffe and D.A. Forsyth, "Probabilistic Methods for Finding People," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 45–68, 2001.

[95] K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human Detection Based on a Probabilistic Assembly of Robust Part Detectors," in *ECCV04*, 2004, pp. Vol I: 69–82.

[96] P.F. Felzenszwalb and D.P. Huttenlocher, "Pictorial Structures for Object Recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[97] J. Tao and J.M. Odobez, "Fast Human Detection from Videos Using Covariance Features," in *Workshop on Visual Surveillance (VS) at ECCV 2008*, 2008.

[98] Y.T. Chen and C.S. Chen, "Fast Human Detection Using a Novel Boosted Cascading Structure With Meta Stages," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1452–1464, 2008.

[99] A. Utsumi and N. Tetsutani, "Human Detection using Geometrical Pixel Value Structures," in *FGR*, 2002, pp. 39–44.

[100] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 4, pp. 349–361, 2001.

[101] P.A. Viola, M.J. Jones, and D. Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[102] N. Dalal, B. Triggs, and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance," in *ECCV (2)*, 2006, pp. 428–441.

[103] Q. Zhu, M.C. Yeh, and S. Cheng, K.T.and Avidan, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients," in *CVPR (2)*, 2006, pp. 1491–1498.

[104] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection," *IJCV*, vol. 38, no. 1, pp. 15–33, 2000.

[105] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, June 2005, vol. 1, pp. 886–893 vol. 1.

[106] C. Wojek and B. Schiele, "A Performance Evaluation of Single and Multi-feature People Detection," in *DAGM Symp. on Patt Rec*, 2008, pp. 82–91.

[107] P. Sabzmeydani and G. Mori, "Detecting Pedestrians by Learning Shapelet Features," *CVPR*, pp. 1–8, 2007.

[108] S. Maji, A.C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," *CVPR*, pp. 1–8, 2008.

[109] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[110] B. Babenko, P. Dollár, Z. Tu, and S. Belongie, "Simultaneous Learning and Alignment: Multi-Instance and Multi-Pose Learning," in *Faces in Real-Life Images*, October 2008.

[111] S. Paisitkriangkrai, C. Shen, and J. Zhang, "Fast Pedestrian Detection Using a Cascade of Boosted Covariance Features," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 8, pp. 1140–1151, Aug. 2008.

[112] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Proc. of European Conference on Computer Vision*, 2006, pp. 589–600.

[113] I. Frosio and N.A. Borghese, “Real-time accurate circle fitting with occlusions,” *Pattern Recogn.*, vol. 41, no. 3, pp. 1041–1055, 2008.

[114] Y.C. Cheng, “The Distinctiveness of a Curve in a Parameterized Neighborhood: Extraction and Applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 8, pp. 1215–1222, 2006.

[115] S. Di Zenzo, “A note on the gradient of a multi-image,” *Comput. Vision Graph. Image Process.*, vol. 33, no. 1, pp. 116–125, 1986.

[116] A. Cumani, “Edge detection in multispectral images,” *CVGIP: Graph. Models Image Process.*, vol. 53, no. 1, pp. 40–51, 1991.

[117] M.A. Ruzon and C. Tomasi, “Color edge detection with the compass operator,” in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999, vol. 2.

[118] A. Brook, R. Kimmel, and N.A. Sochen, “Variational Restoration and Edge Detection for Color Images,” *J. Math. Imaging Vis.*, vol. 18, no. 3, pp. 247–268, May 2003.

[119] D. Mumford and J. Shah, “Boundary detection by minimizing functionals,” in *Proc. of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 1985, vol. 1, pp. 22–26.

[120] B. Leibe, A. Leonardis, and B. Schiele, “Robust Object Detection with Interleaved Categorization and Segmentation,” *IJCV*, vol. 77, no. 1, pp. 259–289, May 2008.

[121] C.H. Lampert, M.B. Blaschko, and T. Hofmann, “Efficient Subwindow Search: A Branch and Bound Framework for Object Localization,” *IEEE T-PAMI*, vol. 31, pp. 2129–2142, 2009.

[122] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, “Robust Multiperson Tracking from a Mobile Platform,” *IEEE T-PAMI*, vol. 31, no. 10, pp. 1831–1846, 2009.

[123] D. Hoiem, A.A. Efros, and M. Hebert, “Putting Objects in Perspective,” *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3–15, 2008.

[124] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele, “Sliding-Windows for Rapid Object Class Localization: A Parallel Technique,” in *DAGM Symp. on Patt Rec*, 2008, pp. 71–81.

[125] A. Lehmann, B. Leibe, and L. Van Gool, “Feature-Centric Efficient Subwindow Search,” in *ICCV*, October 2009.

[126] B. Han, Y. Zhu, D. Comaniciu, and L.S. Davis, “Visual Tracking by Continuous Density Propagation in Sequential Bayesian Filtering Framework,” *IEEE T-PAMI*, vol. 31, no. 5, pp. 919–930, 2009.

[127] C. Hue, J.P. Le Cadre, and P. Perez, “Tracking multiple objects with particle filtering,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 791– 812, July 2002.

[128] M. Isard and J. MacCormick, “BraMBLe: A Bayesian Multiple-Blob Tracker,” in *ICCV*, 2001, pp. 34–41.

[129] K. Okuma, A. Taleghani, N. De Freitas, J.J. Little, and D.G. Lowe, “A Boosted Particle Filter: Multitarget Detection and Tracking,” in *ECCV*, 2004, pp. 28–39.

[130] J. Vermaak, A. Doucet, and P. Pérez, “Maintaining Multi-Modality through Mixture Tracking,” in *ICCV*, 2003, pp. 1110–1116.

[131] G. Salton and C. Buckley, “Improving retrieval performance by relevance feedback,” *Journal of the American Society for Information Sctence*, vol. 41, no. 4, pp. 288297, 1990.

[132] A.F. Bobick and J.W. Davis, “The Recognition of Human Movement Using Temporal Templates,” *IEEE Trans. on PAMI*, vol. 23, no. 3, pp. 257–267, Mar. 2001.

[133] B. Han, D. Comaniciu, Y. Zhu, and L. Davis, “Incremental density approximation and kernel-based Bayesian filtering for object tracking,” in *CVPR*, 2004, pp. I–638–I–644 Vol.1.

[134] V. Philomin, R. Duraiswami, and L. Davis, “Quasi-Random Sampling for Condensation,” in *ECCV*, ’00, pp. 134–49.

[135] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, “Generic Object Recognition with Boosting,” *IEEE T-PAMI*, vol. 28, no. 3, pp. 416–431, 2006.

[136] J. Ponce, T. Berg, M. Everingham, D. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, C. Russell, A. Torralba, C. Williams, J. Zhang, and A. Zisserman, *Dataset issues in object recognition*, p. 2948, Springer, 2006.