DOTTORATO DI RICERCA IN
INGEGNERIA DELL'INFORMAZIONE
XIX CICLO

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Ingegneria dell'Informazione

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Computer Vision for People Video Surveillance

Tesi di:
Dott. Ing. Roberto Vezzani

Relatore:
Prof. Ing. Rita Cucchiara

Contents

List of Figures

List of Tables

Symbols and Definitions

Table 1: Symbols and Conventions

Symbol	Definition
$\mathbf{I}^t$	Image frame at time t
$\mathbf{I}(\mathbf{x})$ $\mathbf{I}(x,y)$	Point $\mathbf{x} = (x,y)$ of the image $\mathbf{I}$
$\mathbf{B}$	Background image
$\mathbf{A}$	Appearance image
$\mathbf{F}^t$	Foreground image
$\mathbf{DB}^t$	Distance image
VO	Visual Object: foreground object extracted for example with a background suppression algorithm
$\overrightarrow{\mathbf{mv}}(x,y)$	motion vector computed at the coordinate (x,y), where ρ and α are the magnitude and the angle of the vectors expressed using the polar coordinates
DH	Direction Histogram, i.e. the histogram of the motion vector directions
ρ_R	motion reliability of a region R
O	Single real object tracked in the scene, described by the state vector: $O = \{\{o_1, \dots, o_N\}, \vec{c}, \vec{e}, \Pi\}$
$\mathcal{O}^t$	set of objects at time t
$\{o_i\}$	set of the N points which constitute the object O
$\vec{x}(o)$	gives the coordinates vector (x,y) of the point o
$(\bar{o})$	gives the RGB color vector of the point o
$\alpha(o)$	gives the alpha component of the point o
$\vec{c}$	position with respect to the image coordinate system of the object centroid
$\vec{e}$	velocity of the object centroid
$\hat{c}$	estimated position with respect to the image coordinate system of the object centroid
Π	*probability of non-occlusion*, i.e. probability of being the foremost object
g_O	matching function between the points of F and the point of the object O

Symbol	Definition
$\tilde{F}$	set of foreground's points which match at least one object
$\tilde{O}$	codomain of the function g_O, that includes the points of O which have a correspondence in $\tilde{F}$
R_{DO}	region with a *dynamic occlusion* due to overlap of another object
R_{SO}	region with a *scene occlusion* due to (still) objects included in the scene
R_{AO}	region with an *apparent occlusions*, i.e. region not visible because of shape changes, silhouette's motion, shadows, or self-occlusions
$B = \{b(x,y)\}$	Blob mask of an object
θ_B	horizontal projection histogram of a blob B
π_B	vertical projection histogram of a blob B
$Ph_B \triangleq (\theta_B, \pi_B)$	Projection histogram set computed on the blob B
Γ_M	Set of the four main detected postures: standing, sitting, crawling, laying
Γ_{VB}	Set of view based postures, where the four main postures are specialized taking into account the view: frontal, left side, right side
SP	support point position, i.e. the point of contact between the person feet and the floor
$\tilde{\Xi}$	head model, with $\tilde{\Xi} = \left\{(\tilde{X}_c, \tilde{Y}_c), \tilde{W}, \tilde{\mathbf{H}}\right\}$
Ξ	head candidate
$L^{i,s_h}, h = 1..4$	h-th 3D Field of View of the camera C_i, with s_h corresponding to one of the four image borders $x = 0$, $x = x_{max}$, $y = 0$, and $y = y_{max}$
$L_j^{i,s}$	the 3DFoV line corresponding to s of the Camera C^i seen by the camera C^j

Chapter 1

Computer Vision based Surveillance Systems

1.1 Introduction

After September 11, 2001, preempting terrorist acts,and providing for the security of citizens at home and abroad have become top priorities not only for the United States but for many other nations around the globe. To this aim, a huge amount of information needs to be captured, processed, interpreted and analyzed. Various computer based technologies can provide a great help in addressing these challenges. Traditional Close Circuits TeleVision (CCTV) networks are a well established off the shelf product with well defined specifications and a mature market [2]. However, this kind of surveillance brings with it the problem of managing the large volume of information that can be generated by such a network of cameras. The video streams are transmitted to a central location, displayed on one or several video monitors and recorded. Security personnel observe the video to determine if there is ongoing activity that warrants a response. Given that such events may occur infrequently, detection of salient events requires focused observation by the user for extended periods of time.

Commercially available video surveillance systems attempt to reduce the burden on the user by employing video motion detectors to detect changes in a given scene [3]. Video motion detectors can be programmed to signal alarms for a variety of reasonably complex situations, but the false alarm rate for most systems in typical environments is unacceptable yet.

Ideally, a video surveillance system should only require the user to specify the objectives of the surveillance mission and the context necessary to interpret the video in a simple, intuitive manner. For many scenarios real-time interpretation is required for the information produced by the system to be valuable. Therefore the challenge is to provide robust real-time video surveillance systems that are easy to use and are composed of inexpensive, commercial off-the-shelf hardware for sensing and computation. Given the capability to interpret activity in video streams in real-time, the utility of a video surveillance system increases dramatically and extends to a larger spectrum of missions. With such a system, a single user can observe the environment using a much larger collection of sensors. In addition, continuous, focused observation of activity for extended periods of time becomes possible. As such capabilities mature, the roles of video surveillance systems will encompass activities such as peace treaty verification, border monitoring, surveillance of facilities in denied areas, hazard detection in industrial facilities and automated home security.

In-house safety is a key problem because deaths or injuries for domestic incidents grow each year. This is particularly true for people with limited autonomy, such as visually impaired, elderly or disabled. Currently, most of these people are aided by either a one-to-one assistance with an operator or an active alarming system in which a button is pressed by the person in case of an emergency. Also, the continuous monitoring of the state of the person by an operator is provided by using tele-viewing by means of video surveillance or monitoring worn sensors. However, these solutions are still not fully satisfactory, since they are both too expensive or require an explicit action by the user (e.g., to press a button), that is not always possible in emergency situations.

Furthermore, in order to allow ubiquitous coverage of the persons movements, indoor environments often require a distributed setup of sensors, increasing costs and/or required level of attention (since, for example, the operator must look at different monitors). To improve efficiency and reduce costs, on the one hand the hardware used must be as cheap as possible and, on the other hand, the system should ideally be fully automated to avoid both the use of human operators and explicit sensors. Again, standard CCTV surveillance systems are not so widespread in domestic applications since people do not like to be continuously controlled by an operator. Privacy issues and the “big brother” syndrome prevent their capillary distribution, even if the technology is now cheaper (cameras, storage, and so on). Conversely, new solutions, fully automated and without the need of a continuous monitoring of human operators, are not invasive and can be acceptable in a home system.

Automating the detection of significant events by means of cameras requires the use of computer vision techniques able to extract objects from the scene, characterize them and their behavior, and detect the occurrence of significant events. Peoples safety at home can be monitored by computer vision systems that, using a single static camera for each room, detect human presence, track peoples motion, interpret behavior (e.g. recognizing postures), assess dangerous situations completely automatically and allow efficient on-demand remote connection. Since the features required by these systems are always evolving, general purpose techniques are preferable to ad-hoc solutions.

1.2 Related Work in Video Surveillance

The set of challenges outlined above span several domains of research. Some of that are faced and described in this thesis. We will review the majority of relevant work directly in the correspondent chapter. In this section, we will focus on famous generic video surveillance systems proposed in the literature only.

Some noteworthy prototypes of CV-based people tracking systems have been developed in the last decade, especially in the U.S.A., and funded by DARPA (Defences Advances Research Projects Agency) programs. One of the pioneering systems of people tracking is Pfinder ("Person Finder") [4], developed at MIT Media Labs, that employs the Maximum A Posteriori (MAP) probability models to detect human body in 2D image planes, especially in indoor scene. The famous W^4 ("What, Where, When, Who") system developed at University of Maryland, is able to detect multiple people in the outdoors and to analyze body silhouette for inferring people's activity [5]. VSAM (Video Surveillance And Monitoring), developed at Carnegie Mellon University, was a big project of cooperative multi-camera surveillance applied in the University campus [6]. Similar research has been carried out in private US research Labs: at IBM, the group of People Vision Project [7] proposed new solutions for appearance-based tracking, also in cluttered indoor environments; at the Siemens labs, in the Imaging and Visualization Department [8] the first formulation of tracking based on mean-shift techniques was defined, in order to follow also body parts in crowded environments. In Europe, since 1996, the group of Prof. Blake at Oxford university proposed *Condensation* (Conditional Density Propagation) [9] approach to track moving objects also from moving cameras. Many European projects were funded for video surveillance which include *Advisor* and *Avitrack*. At the ImageLab Laboratory in Italy the *Sakbot* (Statistical And Knowledge-Based Object Tracker) system [1] has been developed to detect and track people and vehicles using an approach which is robust to occlusions and shadows. It has been used in projects in collaboration with University of California at San Diego [10] for security and with European companies in the area of intelligent transportation systems [11].

Nowadays, many consolidated techniques have been tested for tracking single people from single fixed cameras, and possibly extracting body information, if the camera is placed in an adequate position to have enough resolution for the body shape. Therefore, many researches worldwide are now focusing on distributed cameras and multi-modal acquisition, such as fixed and moving pan-tilt-zoom (PTZ) cameras. Hu et al. [12] report a good survey of multi-camera surveillance systems. Mubarak Shah at University of South Florida [13] proposed an approach for learning geometrical information for consistent labeling or spatially coherent labeling [14], i.e. to maintain the identification of a person and its trajectory when he/she is moving from the field of view of a camera to the one of another camera, by means of homographic geometrical reconstruction. An improved approach has been defined also by the NPD partner. This approach exploits both homography on the ground plane and epipolar geometry, by using the automatically-extracted feet and the head position, respectively [15]. This allows the tracking and disambiguation of groups of people in areas covered by multiple cameras.

The use of active (PTZ) cameras to acquire high-resolution images of portions of the scene or to follow (and "keep in the scene") interesting people has been proposed recently in the literature [16]. On this topic, the NPD partner has been on the frontier by being first to propose a system based on a single PTZ camera [17].

1.3 Thesis Overview

In this thesis a description of most of the components of a videosurveillance system will be given. Fig. 1.1 shows a scheme of the overall architecture of our system. The blocks colored in green are not described in this treatment. We have considered different kinds of input devices: static cameras, PTZ camera, and handled cameras. For each of these input frame sources, a suitable foreground segmentation algorithm have been studied and developed, respectively, a traditional Background Suppression (Chapter 2), a background mosaicing for PTZ cameras (Chapter 3), and a MRF framework for object detection with moving cameras (Chapter 4). The foreground image should be analyzed in order to separate and follow the single objects along time; to this aim, an appearance based object tracking have been proposed (Chapter 5).

The detected objects can be classified to distinguish among people, vehicles, furniture, animals, and so on. This task is strongly dependent on the application and the point of view; thus, in this dissertation we skip the classification phase introducing some empirical and context based rules to identify the interesting targets (e.g., people).

People Video surveillance can be carried out through several kinds of information extractors; the two most frequently used will be described, i.e. the Posture Classification (Chapter 6) and the Face Detection (Chapter 7). These two chapter can be considered the main contribution of this thesis as well as my research activity of these three years.

The previously described task sequence can be seen as a high level feature extraction from a single video source. These features can be combined in a reasoning system, for example to detect alarm situation, events of interest, or to produce and store video annotations. Furthermore, a new video stream can be generated keeping the semantic content of the input video and, at the same time, meeting the requirements and constraints imposed by the remote device (e.g., bandwidth, resolution, number of colors, etc ...) (Section 11).

Minimum resolution and maximum frame size are two constraints usually imposed to assure satisfactory performance and a real time computation. This bring to reduce the field of view of each camera, requiring a multicamera system to cover the entire scene of interest. An integration between different video sources and among cameras and other sensors can be carried out, aiming at covering a wider area or improving the system performance. In Chapter 8 details about the consis-

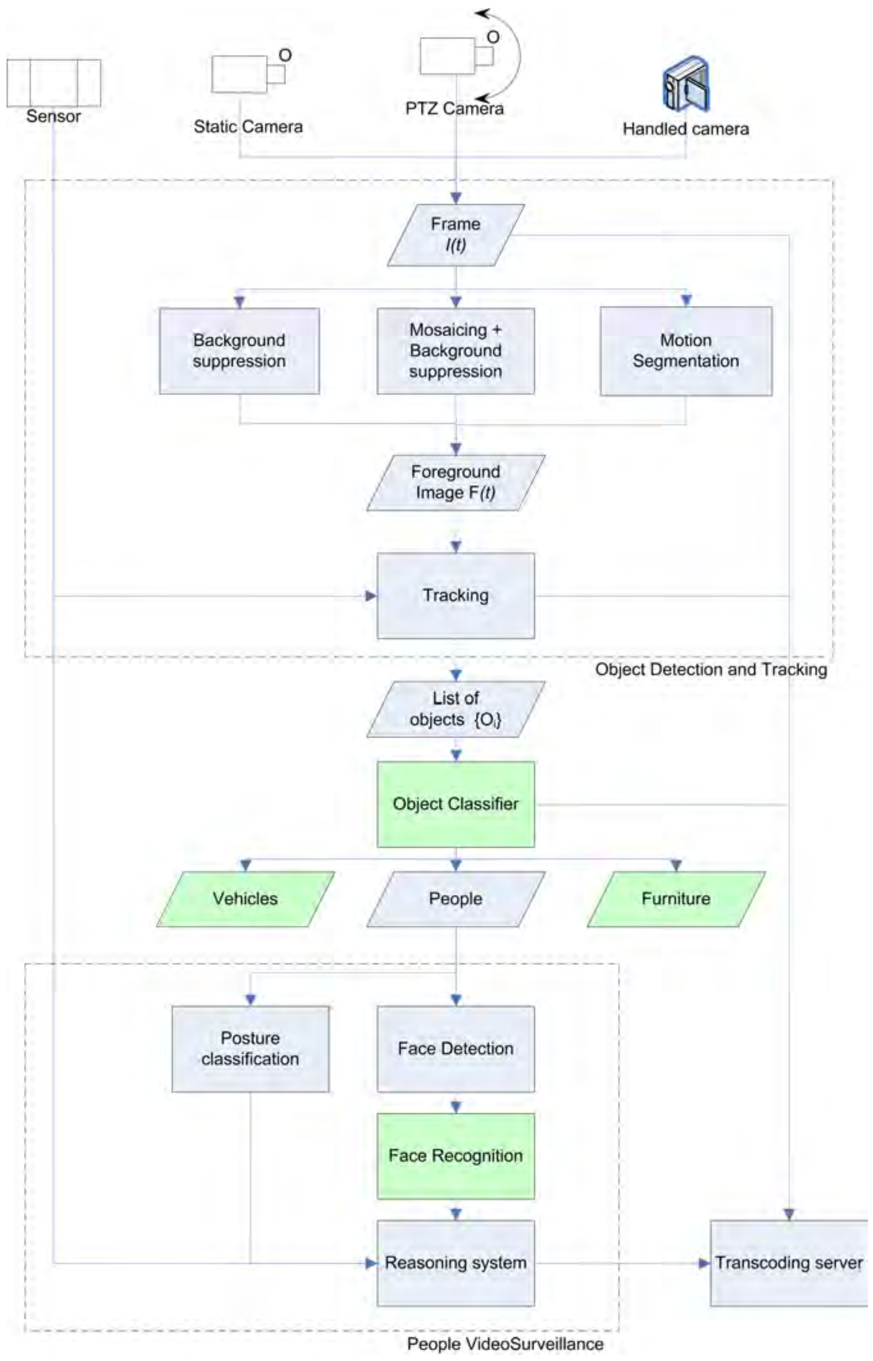


Figure 1.1: Overall Scheme of a People Video surveillance System

tent labeling algorithm we proposed are reported. Goal of the consistent labeling module is the detection of correspondences between different views of the same object (person). Chapter 9 describes some techniques useful to recover the trajectory of a moving handled camera by means of reference image acquired by static and geo-referenced cameras. Finally, Chapter 10 presents some possible employments of hardware sensor, e.g. proximity PIR sensors, to improve video based surveillance systems.

Part I

Object Detection and Tracking

Chapter 2

Fixed Cameras: the Sakbot system

2.1 Introduction

Detection of moving objects in video streams is the first relevant step of information extraction in many computer vision applications, including video surveillance, as well as people tracking, traffic monitoring and semantic annotation of videos. In these applications, robust tracking of objects in the scene calls for a reliable and effective moving object detection that should be characterized by some important features: high precision, with the two meanings of accuracy in shape detection and reactivity to changes in time; flexibility in different scenarios (indoor, outdoor) or different light conditions; and efficiency, in order for detection to be provided in real-time as well as allow further elaboration and reasoning steps.

In this task, we assume that the models of the target objects and their motion are unknown, so as to achieve maximum application independence. In the absence of any *a priori* knowledge about target and environment, the most widely adopted approach for moving object detection with **fixed camera** is based on *background subtraction* [4, 18–25]. An estimate of the background (often called a *background model*) is computed and evolved frame by frame: moving objects in the scene are detected by the difference between the current frame and the current background model. It is well known that background subtraction carries two problems: the first is that the model should reflect the real background as accurately as possible, to allow the system accurate shape detection of moving objects. The second problem is that the background model should immediately reflect sudden scene changes such as the start or stop of objects, so as to allow detection of only the actual moving objects with high reactivity (the "transient background" case). If the background model is neither accurate nor reactive, background subtraction causes the detection of false objects, often referred to as "ghosts" [18, 20]. In addition, moving object segmentation with background suppression is affected by the problem of *shadows* [21, 26]. Indeed, we would like the moving object detection to not classify shadows as belonging to foreground objects, since the appearance and geometrical properties of the object can be distorted and delocalized, which in turn affects many subsequent tasks such as posture classification. Moreover, the probability of object undersegmentation (where more than one object is detected as a single object) increases due to connectivity via shadows between different objects.

The approach adopted in this work has been defined at Imagelab in 2001 [1,27] and is called **Sakbot** (Statistical And Knowledge-Based ObjecT detection) since it exploits statistics and knowledge of the segmented objects to improve both background modeling and moving object detection. Sakbot has been used in different projects of indoor and outdoor surveillance. In [28] has been compared with MOG [20]. Sakbot's processing is the first step for most of the processes described in the following chapters, such as people tracking, posture classification, and so on.

Feature	**Systems**
Statistics	• Minimum and maximum values [18] • Median [29,30], **[1]** • Single Gaussian [4,21,31] • Multiple Gaussians [20,26,32] • Eigenbackground approximation [22,33] • Minimization of Gaussian differences [23]
Adaptivity	[4,18,19,22,24,34], **[1]**
Selectivity	[18,19,24,26], **[1]**
Shadow	[21,26], **[1]**
Ghost	[18,20], **[1]**
High-frequency illumination changes	• Temporal filtering [22,32,33] • Size filtering **[1]**
Sudden global illumination changes	[18], **[1]**

Table 2.1: Compared background subtraction approaches. Our approach is referred with [1].

Many works have been proposed in the literature as a solution to an efficient and reliable background subtraction. Table 2.1 is a classification of the most relevant papers based on the features used. Most of the approaches use a statistical combination of frames to compute the background model (see Table 2.1). Some of these approaches propose to combine the current frame and previous models with recursive filtering (adaptivity in Table 2.1) to update the background model. Moreover, many authors propose to use pixel selectivity by excluding from the background update process those pixels detected as in motion. Finally, problems carried by shadows have been addressed [21,26,35]. The adopted method proves to be accurate and reactive, and at the same time fast and flexible in the applications. Details on comparison between Mode, Median and statistic functions are in [36].

2.2 Background suppression

Let us call x a point of the video frame I at time t ($x \in \mathbf{I}^t$). $\mathbf{I^t}(x)$ is the value of point x in the color space. Since images are acquired by standard color cameras or decompressed from videos with standard formats, the basic color space is RGB. Thus $\mathbf{I^t}(x)$ is a vector with R,G,B components. The goal is to compute, at each time t, both the set VO^t of visual objects and the background model $\mathbf{B^t}$.

$\mathbf{B^t}$ is the background model at time t and is defined for each point of the image. If x is a point of the uncovered background then $\mathbf{B^t}(x)$ should correspond to its value in the current frame; however, if x is a point of a visual object (i.e. that has been segmented and classified), $\mathbf{B^t}(x)$ is an estimation of the value of background covered by the object itself.

If point x does not belong to any object, the background value in x is predicted using only statistical information ($\mathbf{B^{t+\Delta t}}(x)$) on the following set S of elements:

$$S = \{\mathbf{I^t}(p), \mathbf{I^{t-\Delta t}}(p), ..., \mathbf{I^{t-n\Delta t}}(p)\} \cup w_b\{\mathbf{B^t}(p)\} \tag{2.1}$$

As it is possible to note from Eq. 2.1, in order to improve the stability of the model we exploited *adaptivity* too. 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 is computed by using the median function (as in [29, 30]) as follows:

$$\mathbf{B^{t+\Delta t}}(p) = \underset{i=1,...,k}{arg\,min} \sum_{j=1}^{k} Distance(\mathbf{x_i}, \mathbf{x_j}) \qquad \mathbf{x_i}, \mathbf{x_j} \in S \tag{2.2}$$

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

$$Distance(\mathbf{x_i}, \mathbf{x_j}) = max(|x_i.c - x_j.c|) \quad with\ c = R, G, B. \tag{2.3}$$

In the experiments, the median function has proven effective while, at the same time, of less computational cost than the Gaussian or other complex statistics.

Foreground points resulting from the background subtraction could be used for the selective background update; nevertheless, in this case, all the errors made during background subtraction will consequently affect the selective background update. A particularly critical situation occurs whenever moving objects are stopped for a long time and become part of the background. When these objects start again, a ghost is detected in the area where they were stopped. This will persist for all the following frames, preventing the area to be updated in the background image forever, causing *deadlock* [26]. Our approach substantially overcomes this problem since it performs selectivity not by reasoning on single moving points, but on detected and recognized moving objects. This object-level reasoning proved much more reliable and less sensitive to noise than point-based selectivity.

Therefore, we use a knowledge-based background model, i.e., a selectively updated background, in the sense that a different background model is selected whether the point belongs to an object or not. Differently from other proposals ([18,19,24,26]), selectivity is at *object-level* and not at pixel-level only, in order to modify the background in accordance with the knowledge of the objects detected in the scene. The advantage is that the background model is not "corrupted" by moving objects and thus it is possible to use a short Δt and a small n so as to also achieve reactivity.

In our approach, after background subtraction, a set of points called foreground points is detected and then merged into labeled blobs according to their connectivity. An initial camera motion compensation might have been performed previously, should the application require it (for example, to compensate small camera vibrations due to non-ideal operational conditions). This step is based on the choice of a fixed reference in the scene assumed to be never occluded at run time. In order to improve detection, background subtraction is computed by taking into account not only a point's brightness, but also its chromaticity, as in Eq. 2.3.

$$\mathbf{DB^t}(x) = Distance(\mathbf{I^t}(x), \mathbf{B^t}(x)) \tag{2.4}$$

The L-inf distance has proven effective in our experiments, while at the same time being less computationally expensive than other distances. In fact, other metrics can be used as the Euclidean distance, or the Mahalanobis distance used in [4], but this last is computationally more severe since it associates the correlation between parameters using the covariance matrix.

The selection of the initial set of foreground points is carried out by selecting the distance image $\mathbf{DB^t}$ defined in Eq. 2.4 with an adequately low threshold T_L. Among the selected points, some are discarded as noise, by applying morphological operators. Then, the shadow detection process is applied (as described in Section 2.3) and the detected points are labeled as shadow points. A region-based labeling is then performed to obtain connected blobs of candidate moving objects and shadows. Eventually, blob analysis validates the blobs of candidate moving objects as either moving objects or ghosts. VOs are validated by applying a set of rules on *area*, *saliency* and *motion* as follows:

- The blob VO must be large enough (greater than a threshold T_A that depends on the scene and on the signal-to-noise ratio of the acquisition system); with this validation, blobs of a few pixels (due, for instance, to high frequency background motion, like movements of tree leaves) can be removed;

- The blob VO must be a "salient" foreground blob, as ascertained by a hysteresis thresholding. The low threshold T_L set on the difference image $\mathbf{DB^t}$ inevitably selects noise together with all the actual foreground points. A high threshold T_H selects only those points with a large difference from the background and validates the blobs which contain at least one of these points.

- The blob VO must have non negligible motion. To measure motion, for each pixel belonging to an object we compute the spatio-temporal differential equations for optical flow approximation, in accordance with [37]. The *average optical flow* computed over all the pixels of a VO blob is the figure we use to discriminate between VOs and ghosts: in fact, VOs should have significant motion, while ghosts should have a near-to-zero average optical flow since their motion is only apparent.

Optical flow computation is a highly time-consuming process; however, we compute it only when and where necessary, that is only on the blobs resulting from background subtraction (thus a small percentage of image points). In [28] this part has been modified in order to reduce the computational cost and to improve the bootstrap phase. The same validation process should also be carried out for shadow points, in order to select those corresponding to the set of *VO shadows* and those belonging to *ghost shadows*. However, computing the optical flow is not reliable on uniform areas such as shadows. In fact, the spatial differences in the optical flow equation is nearly null because shadows smooth and make uniform the luminance values of the underlying background. Therefore, in order to discriminate VO shadows from ghost shadows, we use information about connectivity between objects and shadows. Shadow blobs connected to VOs are classified as shadows, whereas remaining ones are considered as ghost shadows. The box of Fig. 2.1 reports the rules adopted for classifying the objects after blob segmentation. All foreground objects not matching any of the rules in Fig. 2.1 are considered background and used for background update.

$< VO >$	$\longleftarrow$	(foreground blob) $\wedge\ \neg$ (shadow) $\wedge$ (large area) $\wedge$ (high saliency) $\wedge$ (high average optical flow)
$< Ghost >$	$\longleftarrow$	(foreground blob) $\wedge\ \neg$ (shadow) $\wedge$ (large area) $\wedge$ (high saliency) $\wedge\ \neg$ (high average optical flow)
$< VO\ shadow >$	$\longleftarrow$	(foreground blob) $\wedge$ (shadow) $\wedge$ (connected with VO)
$< Ghost\ shadow >$	$\longleftarrow$	(foreground blob) $\wedge$ (shadow) $\wedge\ \neg$ (connected with VO)

Figure 2.1: Validation rules

In conclusion, *the background model remains unchanged for those points that belong to detected VOs or their shadow*. Instead, points belonging to a ghost or ghost shadow are considered potential background points and their background model is updated by use of the statistic function.

2.3 Shadow detection

By *shadow detection* we mean the process of classification of foreground pixels as “shadow points” based on their appearance with respect to the reference frame, the background. The shadow detection algorithm we have defined in Sakbot aims to prevent moving cast shadows being misclassified as moving objects (or parts of them), thus improving the background update and reducing the undersegmentation problem. The major problem is how to distinguish between moving cast shadows and moving object points. In fact, points belonging to both moving objects and shadows are detected by background subtraction by means of Eq. 2.4. To this aim, we analyze pixels in the Hue-Saturation-Value (HSV) color space. The main reason is that the HSV color space explicitly separates chromaticity and luminosity and has proved easier than the RGB space to set a mathematical formulation for shadow detection.

For each pixel belonging to the objects resulting from the segmentation step, we check if it is a shadow according to the following considerations. First, if a shadow is cast on a background, the hue component changes, but within a certain limit. In addition, we considered also the saturation component, which was also proven experimentally to change within a certain limit. The difference in saturation must be an *absolute* difference, while the difference in hue is an *angular* difference.

We define a shadow mask SP^t for each point p resulting from motion segmentation based on the following three conditions:

$$SP^t(p) = \begin{cases} 1 \quad if & \begin{array}{c} \alpha \leq \frac{I^t(p).V}{B^t(p).V} \leq \beta \wedge \\ |I^t(p).S - B^t(p).S| \leq \tau_S \wedge \\ D_H \leq \tau_H \end{array} ; \quad \alpha \in [0,1],\ \beta \in [0,1] \\ 0 \quad otherwise & \end{cases} \tag{2.5}$$

where the $.H$ denotes the hue component of a vector in the HSV space and is computed as:

$$D_H^t(p) = min(\,|I^t(p).H - B^t(p).H|,\ 360 - |I^t(p).H - B^t(p).H|\,) \tag{2.6}$$

The lower bound α is used to define a maximum value for the darkening effect of shadows on the background, and is approximately proportional to the light source intensity. Instead the upper bound β prevents the system from identifying as shadows those points where the background was darkened too little with respect to the expected effect of shadows. Approximated values for these parameters are also available based on empirical dependence on scene luminance parameters such as the average image luminance and gradient which can be measured directly. A sensitivity analysis for α, β, τ_H and τ_S is reported in [25]. A detailed comparison of this method with others proposed in the literature is reported in [10].

Fig. 2.2 shows the efficacy of the shadow removal system. Fig. 2.2(a) shows the input frame, Fig. 2.2(b) and Fig. 2.2(c) report the silhouette of the foreground

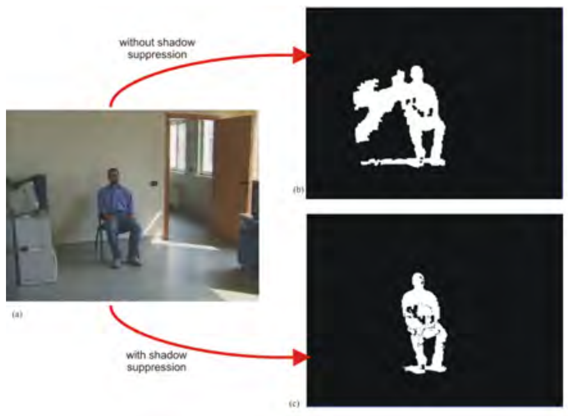


Figure 2.2: Comparison of the results achieved by preserving or removing shadows: (a) is the input frame, (b) the extracted VO by including shadows and (c) that obtained by removing shadows.

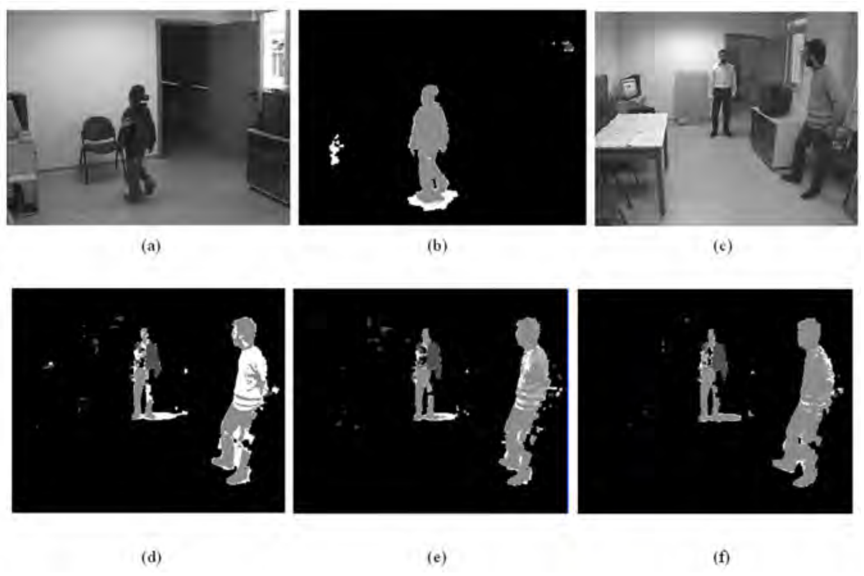
object without or with shadow suppression, respectively. The empirically determined parameters used in our experiments are reported in Table 2.2.

Parameter	Description	Value
Δt	Frame Subsampling factor for the background update	10
n	Number of past values stored for the background statistic	7
T_L	Low threshold for the background difference	15
T_H	High threshold for the object validation	40
T_A	Area threshold for the object validation	40
	Shadow parameters	
α	Low Value threshold	0.77
β	High Value threshold	0.97
τ_H	Hue threshold	120
τ_S	Saturation threshold	0.3

Table 2.2: Empirically determined parameters

2.4 Conclusions

In this chapter the first step of a traditional video surveillance system has been described. In particular, a motion detection based on a background suppression technique is proposed in order to segment the foreground pixels in a video captured by a fixed camera. The *Sakbot* approach here proposed estimates the background image with a median function, extracts the foreground image by means of a distance function thresholded with an hysteresis. Additional functionalities have been added in order to detect shadow regions and ghosts, i.e. wrong foreground regions



Shadow Parameters				
Figure	α	β	τ_S	τ_H
b	0,97	0,77	0,3	120
d	0,97	0,77	0,3	120
e	0,97	0,70	0,3	120
f	0,97	0,60	0,3	180

Figure 2.3: Results of the shadow removal algorithm obtained with different parameters. a) and c) are the input frames, while b), d), e), and f) are the outputs of the shadow detector: white pixels correspond to shadows.

due to the inclusion into the background of previously still objects. The good performance, both in term of computational cost and efficacy, allow this module to be used as a first step for real time indoor and outdoor surveillance. Besides a fixed camera, requirements of *Sakbot* are almost constant illumination condition and a “still” background: waving leafs, flags, or clouds can seriously jeopardize the right working of the system.

Chapter 3

Pan Tilt Cameras

3.1 Introduction

This chapter describes a solution of advanced video surveillance for moving people segmentation from a **Pan Tilt moving camera**. The method is designed to work in real time for creating a mosaic image of the whole scene (by registering overlapped images provided by successive frames of the active camera) and to detect moving people very quickly.

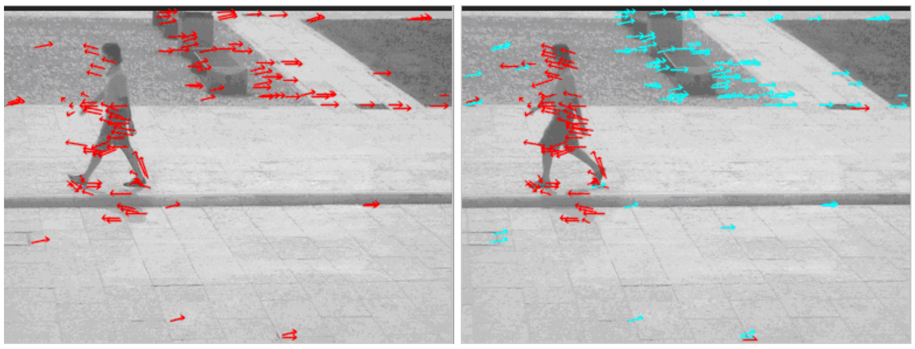
We propose a new method for fast ego-motion computation based on the so-called *direction histograms*. The method works with an uncalibrated camera that moves with an unknown path and it is based on the compensation of the camera motion (i.e., the *ego-motion*) to create the mosaic image and on the frame differencing to extract moving objects. Successive steps eliminate the noise and extract the complete shape of the moving objects in order to exploit a tracking algorithm.

3.2 Background mosaiking

Our approach for moving object segmentation from moving camera consists in two basic steps: first, the ego-motion is estimated and compensated to build a mosaic image and, second, frame differencing and post-processing are applied to extract the single moving objects.

The motion vectors of the current frame are extracted using a pyramidal implementation of the Lucas-Kanade algorithm (Fig. 3.1(a)). Then, they are clustered to find the dominant motion, that corresponds to the ego-motion assuming that the background is dominant over the moving objects.

The clustering is performed with an innovative and fast process. It can be demonstrated that, for small pan and tilt angles the camera motion model can be approximated with a pure translational model. With this hypothesis, a *direction histogram* containing all the directions of the extracted motion vectors is built (see Fig. 3.2(a)).



(a) Motion vectors with the Lucas-Kanade algorithm

(b) Clustered motion vectors

Figure 3.1: Example of extraction and clustering of the motion vectors

Let $\overrightarrow{\mathbf{mv}}(x, y) = (\rho(x, y), \alpha(x, y))$ be the motion vector computed at the coordinate (x, y), where ρ and α are the magnitude and the angle of the vectors expressed using the polar coordinates. We define the direction histogram $DH(\beta)$ as in Equation 3.1.

$$DH(\beta) = \# \left\{ \overrightarrow{\mathbf{mv}}(x, y) | \alpha(x, y) = \beta \right\} \tag{3.1}$$

with β ranging from 0 to 2π. A 1-D Gaussian filter $G(\mu, \sigma)$ centered on the histogram peak μ is applied on the histogram to eliminate motions different from the dominant one as in Equation 3.2.

$$\widetilde{DH}(\beta) = DH(\beta) \cdot G(\mu, \sigma) \tag{3.2}$$

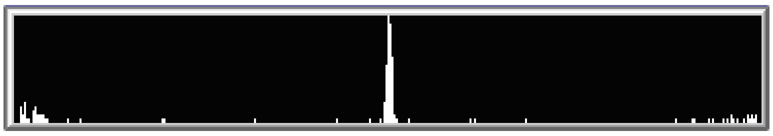
where $\mu = \arg\max_{\beta} DH(\beta)$ and σ is a parameter set to 1 for most of the experiments.

The resulting histogram $\widetilde{DH}(\beta)$ of Equation 3.2 (shown, for instance, in Fig. 3.2(b)) allows to divide the motion vectors into two groups, one due to the camera motion (in cyan in Fig. 3.1(b)) and one due to moving objects (in red in Fig. 3.1(b)), and to compute the direction $\overline{\alpha}$ and amplitude $\overline{\rho}$ of the ego-motion by averaging the vectors retained by the Guassian filter as in Equation 3.3.

$$\overline{\alpha} = \arg\max_{\beta} \widetilde{DH}(\beta) \quad ; \quad \overline{\rho} = \frac{\sum\limits_{\rho \in R} \rho}{|R|} \tag{3.3}$$

where $R = \left\{ \rho(x, y) | \widetilde{DH}\left(\alpha(x, y)\right) \neq 0 \right\}$.

This approach, though intuitive and simple, has proven to act very well, given that the above-mentioned hypothesis holds. For example, Fig. 3.3 reports the result in the case of a person moving with motion concordant with the camera. It is worth


(a) Direction histogram

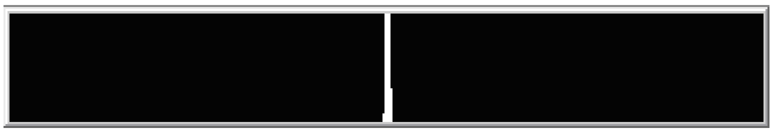

(b) Filtered direction histogram

Figure 3.2: Direction histograms before and after the application of gaussian filter

noting that the direction histogram (Fig. 3.3(a)) contains two peaks corresponding to the ego-motion and the person, respectively. However, Fig. 3.3(b) shows that, though some errors are present, accuracy is still acceptable.

Once the ego-motion is estimated, the current frame is registered (assuming a translational motion model) by compensating for the camera motion given by the vector $(\overline{\rho}, \overline{\alpha})$. The difference in the results performing frame differencing before and after the compensation is shown in Fig. 3.4. Moving pixels are indicated in black.

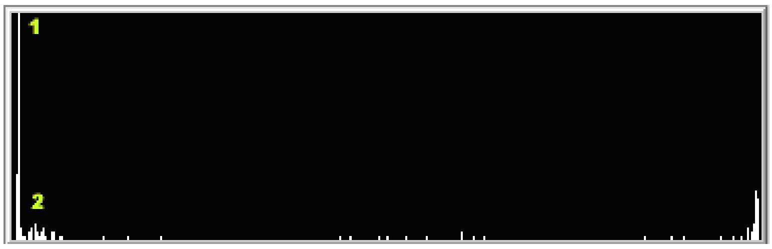
As evident in Fig. 3.4(b), the result provided by frame differencing are still far from being optimal, for both the noise due to imprecise image registration and the ghost of the moving objects. For this reason, post-processing steps must be used. Noise and small areas are removed by morphological operations, whereas ghosts are eliminated by merging information provided by a connected-components analysis and by a Canny edge detector: only edges with at least one point (in the 3x3 neighborhood) detected as moving are retained. Fig. 3.5(a) shows an example of retained edges.

Based on these information, the single moving objects are located. Their shape, however, is imprecisely extracted. Since the performance of the tracking algorithm heavily depends on the precision of the object's shape, a successive step is required. Since standard background suppression techniques are not suitable with our requirements (unknown path, uncalibrated camera, and real-time constraints), we employ a variant of the classical *active contours*, in which the energy is obtained with the following equation:

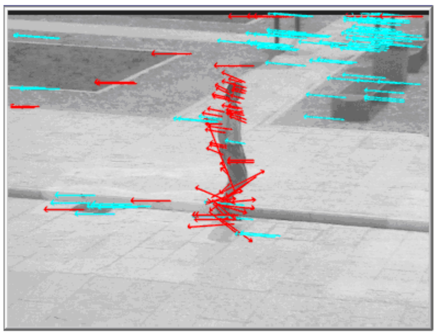
$$E_i = E_{cont,i} + \frac{E_{curv,i}}{2} + E_{dist,i} \tag{3.4}$$

Given $p_1, ..., p_n$ a discrete representation of the contour/shape to be modelled, $E_{cont,i}$ represents the contour continuity energy and is set to:

$$E_{cont,i} = \left| \overline{d} - |p_i - p_{i-1}| \right| \tag{3.5}$$



(a) Direction histogram



(b) Motion vectors with LK

Figure 3.3: Result of the motion vector clustering in the case of a person moving in the same direction of the camera.

where $\overline{d}$ is the average distance between each consecutive pair of points. Minimizing this energy means to have the points more equidistant.

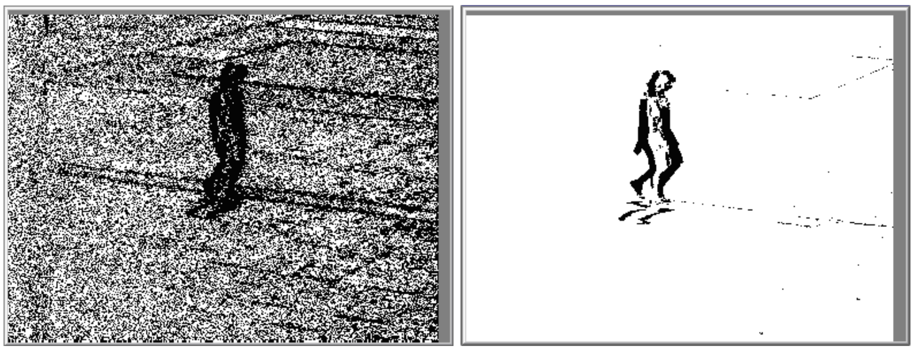
$E_{curv,i}$ is the contour curvature energy (the smoother the contour is, the lower the energy) and is defined as:

$$E_{curv,i} = ||p_{i-1} - 2p_i + p_{i+1}||^2 \tag{3.6}$$

As external energy, we modify the original proposal by considering the image obtained by applying the Distance transform to the image containing the edges retained by the post-processing. Examples of input edge image, external energy image and resulting snake are reported in Fig. 3.5. Finally, contour filling is employed to obtained a rough segmentation of the person's shape to be provided to the appearance-based probabilistic tracking proposed in [38], that is meant to be robust to occlusions.

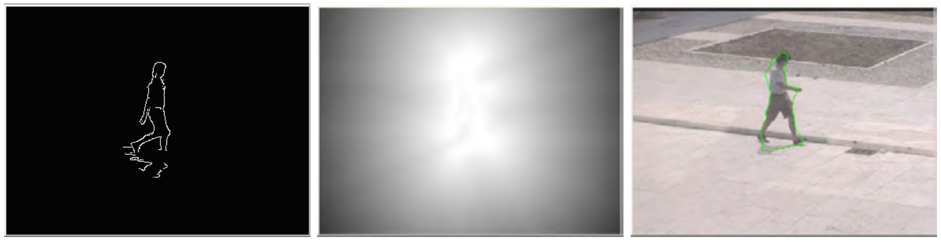
Final mosaic image is constructed by superimposing the registered image on the mosaic and applying a simple alpha blending algorithm. Moreover, moving objects are not pasted onto the mosaic.

Once moving people are detected the system allows to select a single object (hoping it is a person) to be followed, by moving the camera to keep him framed. In the current implementation of the system the "youngest" object(in the sense of that tracked by less time) in the scene is followed until he is visible. When he



(a) Without compensation (b) With compensation

Figure 3.4: Frame differencing (a) without and (b) with ego-motion compensation



(a) Edges (b) Energy (c) Snakes

Figure 3.5: Active contours: input edge image, external energy image and the resulting snake

exits from the area the camera either follows the next "youngest" object, if any, or goes to a predefined position. Person following is achieved by moving the camera towards the person as soon as he approaches the limit of the current field of view.

3.3 Experimental Results

We carried out several tests to check the performances of the described system. As described in Section 3.2, we implemented and tested a dynamic mosaicing algorithm able to extract the foreground region on a moving (Pan-Tilt-Zoom) camera. This evaluation has been carried out by means of both qualitative and quantitative analysis. For qualitative analysis, a large set of videos has been taken with different illumination conditions, different number of people (from none to 5-6 simultaneously moving people), and different movements of the camera (only pan, only tilt, both pan and tilt). This analysis has demonstrated that, if the hypotheses hold, the system produces very good mosaic images, like those shown in Fig. 3.6. The only distortions appear at the top of the image where they do not affect moving object segmentation. Mosaic images have been also evaluated using a quantitative (i.e., objective) measure such as the PSNR. For example, the PSNR of the mosaic images reported in Fig. 3.6 with respect to ground truths (generated by exhaustively trying all the possible displacements and choosing that minimizing the error) is 40.82 dB.

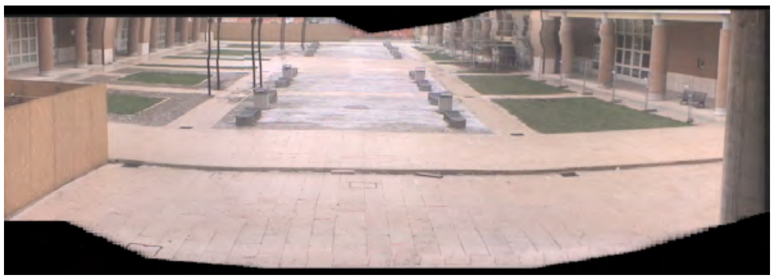


Figure 3.6: An example of mosaic image.

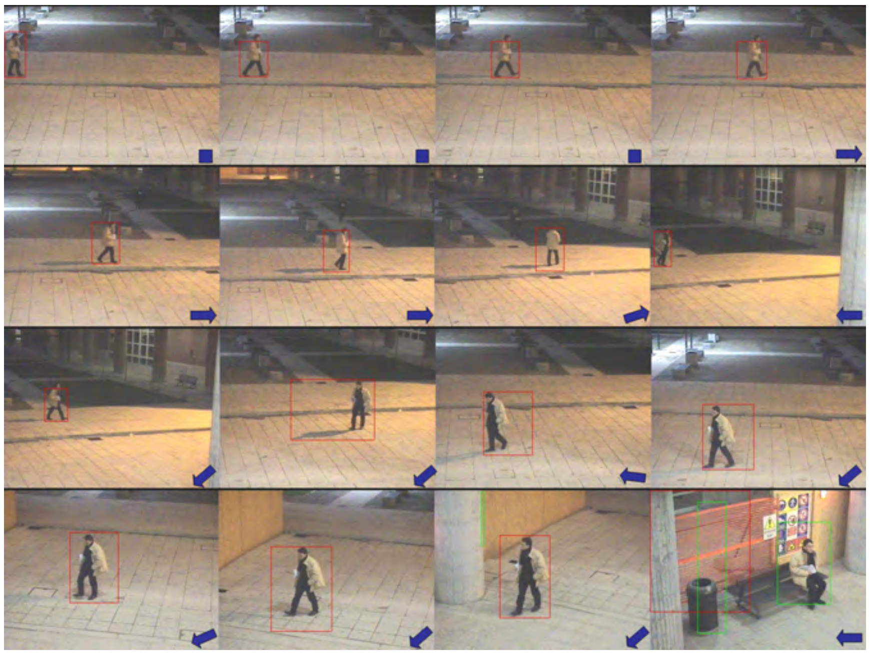


Figure 3.7: Snapshots from a live sequence with person following.

Fig. 3.7 shows a sequence reporting some snapshots of the results for object (person) following. The red bounding box identifies the person followed, while green ones identify other moving objects. The drawings on the bottom right corner of each image show the actual movement of the camera. It is worth noting that these results have been obtained with a completely unsupervised system working on live camera. It is evident that there are some imprecisions: for instance, on row 2, column 3, the second person is not segmented since it is very dark; on row 3, column 2, shadows are connected to the moving person; erroneous moving objects are detected on the column in the last row, columns 3 and 4. In particular, the last snapshot reports a wrong segmentation due to the presence, in the background, of much texture and to the closeup of the scene.

From the computational point of view, the system works in real time with a standard PC at 3Ghz, with an average frame rate from live camera of about 5 fps, including also the person following task and the following face detection. Considering that the current acquisition device releases 12.5 frames per second, we can properly speak of “real time”.

3.4 Conclusions

The method here proposed for detection of people in videos from a moving camera is conceived to cope with Pan and Tilt movements only, since the motion model used is strictly translational. For zoom changes and other free camera movements a more complex motion model must be introduced. Despite that, pan-tilt cameras are wildly used for surveillance, so this approach assumes an outstanding interest. Like the *Sakbot* system, it requires some environmental conditions in order to be applied, such as a “still” background and slow illumination changes.

Chapter 4

Moving Cameras

4.1 Introduction

As above mentioned, moving object segmentation in videos is a key process in video-surveillance applications. Several difficulties associated to this task can arise if there is a relative motion of the observer with respect to the objects and background. When the background is stationary and videos are acquired by fixed cameras, moving object detection can be faced thorugh background suppression or frame differencing (see Chapter 2). Even when background is moving with a known path, for instance when the camera's motion is constrained (for example in surveillance Pan-Tilt-Zoom cameras with a programmed camera path) modified methods of background suppression and frame differencing have been proposed (see Chapter 3). Instead, the segmentation of moving objects becomes more critical when the video is acquired by a **free moving camera** with an unconstrained and a priori unknown motion. More in general, we consider the framework of videos with moving foreground objects on a moving background. In this case, segmentation cannot be accomplished computing visual motion only, but other features must be exploited such as color, shape, texture and so on. In addition, our aim is to develop a system able to compute dominant motions in real-time, suitable for applications of indoor/outdoor surveillance.

Many works propose the integration of multiple features for detecting and tracking moving objects on video with a moving background. Among them, a work of Gelgon and Bouthemy [39] proposes an approach based on a color segmentation followed by a motion-based region merging. The authors adopt the affine motion models and the merging of regions is performed with Markov Random Fields (MRF, hereafter). An implicit tracking of the objects is also included, and it is obtained with the initialization of the MRF according with a label prediction. This approach is based on two basic assumptions on the objects: they must have different colors (the segmentation of the scene in objects is based only on color information) and rigid motion (expressible with affine equations). This solution produces good results, but the computational complexity is too high to allow real-time implementation: authors in [39] state to be able to process a frame every 80 seconds[1].

In this chapter we present an approach inspired by the one of [39] substantially modified in order to abate the computational time. To this aim our algorithm assumes some simplifications with respect to [39] but also some important new features. The basic simplification is the hypothesis of a translational motion model instead of an affine one: this hypothesis limits the applicability to the (very frequent) cases of pure translational movements or that can be approximated as translational between two consecutive frames.

However we have introduced some important novelties:

- the "Partitioned Region Matching": we propose a new motion estimation algorithm based on region matching, but emphasizing advantages of both region matching and block matching;
- a measure of "motion reliability": we use in the MRF model a factor which takes into account the reliability of the motion estimation phase;
- a new minimization algorithm of the MRF function that approximates the classical approach with a search based on arcs instead that on nodes. This method abates the computational costs, by preserving most of the efficacy of the algorithm. In this way we have defined a technique for very fast segmentation reaching up to 10 frames per second[2] in frames of 100x100 pixels.

4.2 Related Work

Several researchers have approached the problem of motion detection on video acquired by moving camera. Table 4.1 tries to summarize the most relevant contributions to this field reported in the literature. We tried to classify them by means of several features such as whether they explicitly compute camera motion or not, which motion model (translational, affine, quadratic, etc.) is assumed, which features are used to segment the video (visual features as color, or motion, or both), and whether they achieve real-time performance or not.

In practice, we classify the approaches by means of eight factors (see Table 4.1):

1) Type of camera motion (II[8]):

[1]On a ULTRASPARC Machine

[2]On a dual Pentium III 1000 MHz with 512 MB of RAM

[3]U: unconstrained, C: constrained, F:Fixed

[4]Feature of the first and basic segmentation; VF: Visual features, M: motion, Mix: mixed

[5]SPHS: Real time on a special purpose hardware system.

[6]E: Estimation, D: Detection; OF: Optical flow, FD: frame difference, BGS: background suppression, BC: block correlation

[7]Only if motion estimation is adopted; A: Affine, T: Translational, 3D: six 3D-parameters, P: perspective

Ref.	**Camera motion**^{3}	**Capture System**	**Egomotion**		**Objects motion**		**Segmentation**^{4}	**Tracking**	**Real time**^{5}
			Y/N	**Model**^{7}	**D/E**^{6}	**Model**^{7}			
[40]	U	3D stereo	No		E: n/a	3D	Mix: Disparity and motion	No	n/a
[41]	U	2D estimation	No		E: iterative	A.	VF: Color/gray level	No	No
[42]	U	2D	Yes	A.	D: FD		M	Yes	SPHS
[43]	U	2D	No		E: dominant components retrieval	A.	M	No	No
[44]	U	2D	Yes	A.	D: FD		Mix	No	No
[45]	C (only pan-tilt)	2D	No		D: BGS		M	Yes	Yes
[46]	U	2D	No		E: OF	P. or A.	M: simultaneously with E	No	No
[47]	U	2D	Yes	quad.	E: RMR	A.	M	No	No
[48]	U	2D	Yes	n/a	D: FD		M	Yes	Yes
[49]	U	2D	Yes	n/a	E: OF	A.	VF: luminance	No	No
[50]	U	2D	Yes	A.	D: FD		VF: Color	No	No
[39]	U	2D	No		E: n/a	A.	VF: Color	Yes	No
[51]	U	2D	No		E: region matching	T.	M	Yes	No
[52]	U	2D	No		E: OF	A.	M	No	No
[53]	C (only pan-tilt)	2D	Yes	T.	E: temporal derivative	n/a	M	With only 1 moving object	Yes

Ref.	Camera motion[3]	Capture System	Egomotion		Objects motion		Segmentation[4]	Tracking	Real time[5]
			Y/N	Model[7]	D/E[6]	Model[7]			
[54]	U	2D	Yes	A.	D: FD		Mix: motion confidence	Yes	Yes
[55]	U	2D	No		E: feature tracking	A.	Mix: some image cues	No	No
[56]	F	2D	No		D: FD		M	No	SPHS
[57]	U	3D stereo	Yes	3D	E: BC	3D	M	Yes	n/a
[58]	U	2D	No		E: n/a	A.	Presegmentation required	No	No
[59]	U	3D stereo	No		E: OF	n/a	Mix: Edge, depth and optical flow	Yes	SPHS
[60]	U	2D	Yes	A.	D: FD		M: through multiscale MRF	No	No
[61]	U	2D	Yes	A.	D: FD		M: through multiscale MRF	No	No
[62]	U	2D	Yes	A. & 3D	D: FD and BGS		M	No	No
[63]	U	2D	No		E: edge tracking		VF: Edges	Yes	No
[64]	U	2D	No		E: corner based	A.	Mix: clustering of edges	Yes	Yes

Ref.	**Camera motion**[3]	**Capture System**	**Egomotion**		**Objects motion**		**Segmentation**[4]	**Tracking**	**Real time**[5]
			Y/N	**Model**[7]	**D/E**[6]	**Model**[7]			
[65]	U	2D omni	Yes	n/a	E: OF		M	Yes	No
[66]	U	2D	No		E: edge tracking	A.	User guided initialization	Yes	Yes
[67]	U	2D	No		E: BC	T.	VF: gray level	No	No
[68]	U	3D with stereo	No		E: OF	A.	M	Yes	SPHS
[69]	U	2D	No		E: OF	A.	M	No	No
[70]	U	2D	No		E: OF		Mix: form	No	No
[71] Our prop.	U	2D	No		E: partitioned region matching	T.	VF: color	No	Yes

Table 4.1: A review of related work.

The first assumption copes with the camera's motion model, namely:

- *fixed* camera (camera not in motion);
- *constrained moving* camera (e.g.: pan-tilt camera, camera with a fixed path);
- *unconstrained moving* camera (when the motion of the camera is a priori unknown).

In the paper we analyze only approaches with a relative motion between camera (i.e. the observer) and the background, and in particular videos acquired with unconstrained moving camera or with an unconstrained moving background.
2) Acquisition system (III): It refers to the number and the type of cameras involved in the acquisition process and the model of the scene.

- *2D* system (single camera);
- *3D* system with *stereoscopic* vision (two normal cameras);
- *3D* system (e.g., with a normal camera and a range camera.

Our proposal assumes a 2D system with a single moving camera.
3) Computation of camera motion (IV and V):
The camera motion can be estimated through the evaluation of the dominant motion with different techniques and models. Some approaches (labeled with "yes" in column IV) exploit camera motion computation to produce compensated videos from original ones in order to apply algorithms developed for fixed camera. Other approaches do not distinguish camera motion from the motions of foreground objects.
4) Motion of objects (VI and VII):
A very relevant difference among proposals is the adoption of either a *motion detection* or a *motion estimation* approach. In the former, each pixel/region is simply classified as "fixed" or "in motion", while with motion estimation a quantitative measure of motion is also provided.
5) Motion model (V and VII):
To describe the motion of a pixel in an image, the two components (along axis x and y) of the shift vector are enough. Instead, the motion of a region is more complex and could require more parameters to describe it. In this case, it is possible to adopt different models to represent the motion of the regions: *translational*, *affine*, *quadratic*, etc ... The more the parameters, the higher the quality of the estimation is, but, at the same time, the more complex and computationally expensive the algorithm becomes.
6) Segmentation features (VIII):
The extraction of moving objects from the background is performed through image segmentation exploiting features. Thus, approaches are characterized by the feature used as:

[8]Roman numbers between brackets are referred to the columns of the Table 4.1

- segmentation based on *visual features* (e.g. color, edge, texture)
- segmentation based on *motion*
- *mixed* segmentation (motion and visual features together)

In the first case, the objects are separated independently from their motion; this typically brings to an over-segmentation. For example, a segmentation based only on color information can separate a person with shirt and pants of different colors into several objects. In this case the use of another feature (i.e., motion) helps in merging different regions of the same object. Another possible approach is to compute the segmentation by considering only the motion information, but in this case the aperture problem can be critical, as in the case of non-textured moving objects. The most reliable solution to this problem is, typically, to implement a segmentation that takes into account visual features and motion at the same time in a mixed segmentation.
7) Tracking (IX):
Not all the approaches include a *tracking phase* that allows the system to trace objects' positions over time. In some cases an implicit tracking is integrated in the motion segmentation process (e.g., in the system proposed in [39] the tracking is performed through the initialization of the labels based on prediction: objects that confirm the prediction, maintain the same label over time).
8) Real time implementation (X):
As stated before, our aim is to meet *real-time* requirements[9]. For this reason, we are interested in evaluating whether an algorithm works in real-time or not. In Table 4.1 we reported only if the time requirements are considered and if the real-time constraints are satisfied or not (as stated by the respective authors).

In conclusion, we can group the proposals into three classes:

a) *based on camera motion computation*: these methods compute camera motion and, after its compensation, they apply an algorithm defined for fixed-camera; (an example is the proposal of Chapter 3)

b) *based on motion segmentation*: the objects are mainly segmented by using the motion vector computed at pixel level;

c) *based on region merging with motion*: the objects are obtained with a segmentation based on visual features, and next merged on motion parameters computed on a region-level.

Among approaches based on the computation of the camera motion followed by the application of a fixed-camera technique, some of them use frame differencing ([42, 44, 48, 53, 60, 61]) and some other uses background suppression with

[9]We use the term *frame-rate* to say 25/30fps (European/American acquisition standards) and the term *real-time* as a more "soft" constraint that could be considered acceptable for many applications (for instance in surveillance systems 10 fps are often assumed as a real-time).

background obtained through mosaicing techniques ([45] and [62]). Refer also to [72] for background mosaicing construction and to [73] for a comparison of approaches to camera motion computation.

Systems described in [43,46,51,52,59,64,68,69] belong to class b). The algorithm used for segmentation is the characterizing factor for these works. Also the proposals reported in [40] and [55] could be included into this class; unlike the others, they use motion and some visual features simultaneously in the segmentation process.

The approaches described in [41,50,67] belongs to the third class, that use color as visual feature for initial segmentation, [39] that assumes a possible exploitation of gray-level, texture or color, [63] that makes use of edges. In [58] and [66], the authors present only the merge phase and suppose the existence of a previous region segmentation step. Our proposal belongs to this class.

Finally, [47] and [49] are two mixed approaches: [47] performs an initial camera motion computation to remove camera motion similarly to the approaches of class a) and obtains the region in motion by frame differencing. Then, to separate single objects, it exploits a motion based segmentation only with a "split and merge" algorithm on the regions extracted by frame differencing. [49] computes camera motion for compensation and then uses an approach similar to [39], composed by a color segmentation process followed by motion based region merging.

Our proposal (the one labeled with **[71]** in Table 4.1) follows the same guidelines of [39] aiming to a real time implementation with general-purpose platforms. By the analysis of the Table 4.1, we could note that few proposals take into account the time requirements; among these, many use a special purpose hardware [42,56,59,68], or a manual initialization [66] or are oriented to constrained motion model [45,74] or a specific application (as [48] for periodic model). [53] and [64] are general purpose systems, but they don't assure the extraction of the entire moving objects, because they make use only of the motion information in the segmentation process.

4.3 Description of the proposed approach

The approach can be summarized as follows: each frame of the sequence is segmented into regions by using *color*, with the assumption that each region contains only one (or part of one) object. For this task a *region growing* algorithm is applied. According with color segmentation, a *Spatial Region Graph (SRG)*, also called adjacency graph, is created. In SRGs, nodes represent regions, whereas arcs represent topological adjacencies. We consider an attributed graph: area size, the coordinate of the bounding-box and the centroid's position are associated with each node. Then, *motion estimation* is computed with the adoption of the translational motion model. To accomplish this task there are basically two ways: the first requires *pixel-level motion estimation* (e.g., with optical flow or block matching) followed by a statistical function (e.g., mean or mode) over regions; the second (adopted in our system) directly exploits computation of the *region-level motion vectors* through region matching (and in particular with *Partitioned Region Matching*, described in the following). In addition to motion vector a measure of *motion reliability* is defined and computed for each region, based on the presence of "strong" gradients (that allows correct estimation of the motion) and a good match. Computed information are then stored in the spatial graph. Then SRG is the starting point of a Markov Random Field (MRF) framework, where regions are merged according with an energy function influenced by the motion parameters (velocity and reliability). The largest region is assumed as background and its motion as the motion of the camera, while other regions are classified as moving objects. Temporal continuity of motion field is supposed to perform a prediction of the next frame. This prediction is used to initialize the region-level MRF framework during the analysis of the next frame, with two advantages: the reduction of merging phase duration and the implicit tracking of the objects.

In Fig. 4.1 the complete algorithm is schematically reported; on the left are presented the functional blocks and on the right the partial outputs obtained with a synthetic sequence are shown. Fig. 4.1.b and Fig. 4.1.c are the color segmentation output and the spatial region graph, respectively. Fig.4.1.d and Fig. 4.1.e describe the segmented regions and the graph after motion computation and MRF estimation, respectively. Finally Fig. 4.1.f shows detected moving objects.

As above-mentioned, our proposal has been initially inspired by the work of [39], and initial description reported in [75]. According with that, we mix color feature and motion and exploit MRF to optimize the spatial region graph in order to extract objects as regions with similar motion parameters. Nevertheless, many substantial differences have been introduced in all sub-tasks and in particular in:

- *Color segmentation*: in accordance with [39] we believe that color is the most salient feature to be exploited. In [39] color segmentation is performed at the first frame only and then refined frame by frame by means of the MRF. Instead, focusing on speed, MRF optimization has been substituted by a color (re-)segmentation at each frame. In particular in our system we apply a region growing algorithm to every frame. This last solution has proved to be less accurate but much faster. Indeed, the construction of the pixel level MRF prediction has been tested, as defined in [39], and has resulted useless if color segmentation on every frames is adopted; thus, the pixel level prediction and the optimization of the segmentation with MRF are not implemented in our version;

- *Motion model and estimation*: we have adopted the translational motion model, less realistic than the affine one, but characterized by a much lower complexity. However, when camera and objects are moving slowly the simplification results acceptable. Correlation based algorithms (such as block matching) could be sufficient for motion estimation; in particular, taking advantage from the presence of a region segmentation, we propose an original

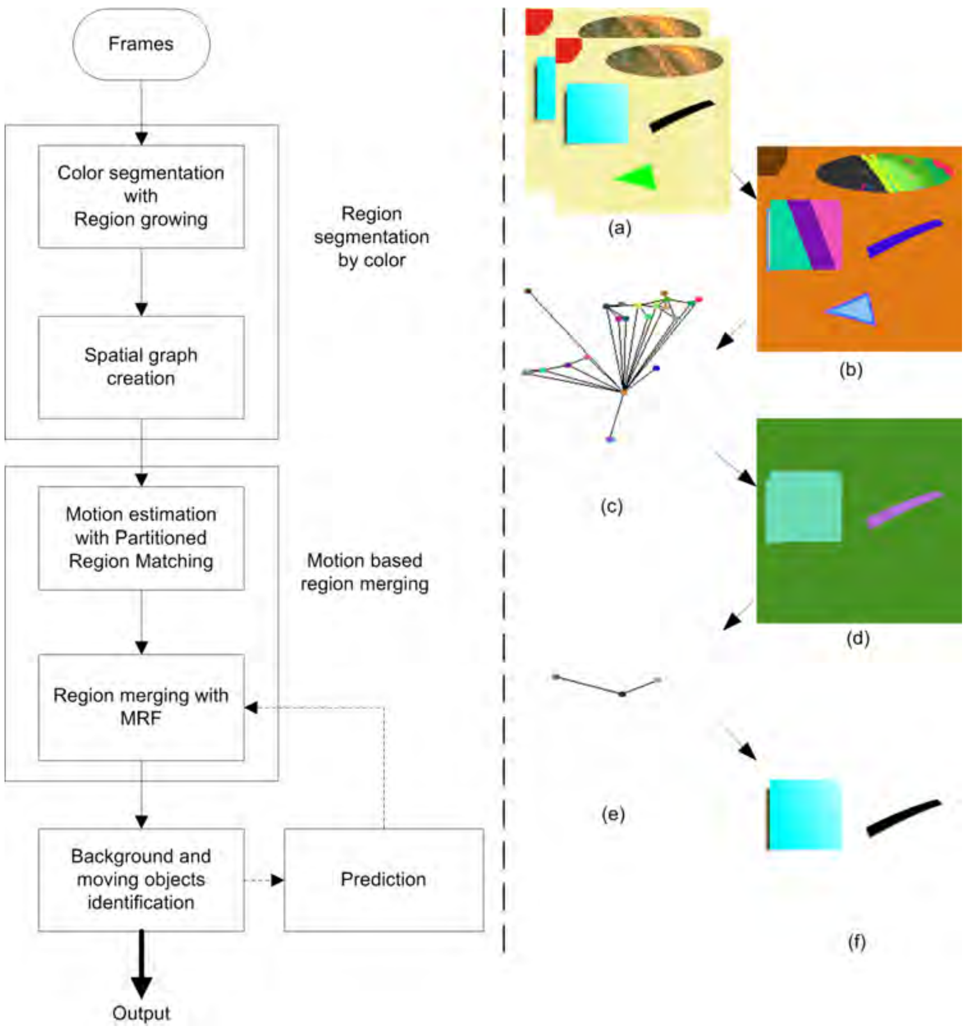

Frames
Color segmentation with Region growing
Spatial graph creation
Region segmentation by color
Motion estimation with Partitioned Region Matching
Region merging with MRF
Motion based region merging
Background and moving objects identification
Prediction
Output
(a)
(b)
(c)
(d)
(e)
(f)

Figure 4.1: The block diagram of the proposal

region matching method, the Partitioned Region Matching (PRM) (see section 4.3.2). Lastly, we define a motion reliability term exploited in MRF;

- *MRF optimization*: the energy function used is changed due to the different motion model adopted. In addition, section 5 describes a new optimized algorithm that allows a real time processing.

4.3.1 Color segmentation

Color segmentation is implemented with a region growing algorithm. Since our goal is to detect moving objects, we work principally on an object-level and a perfect color segmentation at pixel level is not required. Thus, we defined a simple iterative approach inspired to [76], with suitable modifications. At each iteration an unlabelled point (or seed) is extracted from the image: a new label is assigned in order to create a region with a single pixel. Then, the 4-connected neighbor points are evaluated. A point x is merged to the current region S if it is unlabelled and if it has a color close enough to the mean color of the region. To check if the sum of the absolute difference between xcolor components of the point x and the correspondent mean values evaluated on the current region is lower than a fixed threshold α, we use the following equation:

$$dist(x, S) = |x_R - \overline{R}_S| + |x_G - \overline{G}_S| + |x_B - \overline{B}_S| \leq \alpha \tag{4.1}$$

where x_R, x_G, x_B are the three RGB color components of the evaluated point x, $\overline{R}_S$, $\overline{G}_S$, $\overline{B}_S$ the mean values of RGB color components computed over the region S and α a fixed threshold.

A possible limitation of this implementation is the dependency on initial seeds and scanning order. Only for computational reasons, seeds are not extracted with a particular algorithm that could extrapolate significant points (e.g., points of local minimum/maximum of the intensity) but the first unlabeled pixel starting from the top-left corner of the image is selected.

To reduce segmentation dependency on the visiting order of the neighbor pixels we have adopted a propagative search with the following strategy. First, the pixels adjacent to the seed are pushed into a stack A. Then, we pop one by one pixels from stack A and insert their neighbors into a stack B. We evaluate pixels from stack B and push new pixels in stack A. In this way the regions grow on all directions with homogeneity.

The presence of strong edges or noise implies the creation of little regions, some of that with only one pixel. To overcome this problem we have introduced a second threshold β: regions with a dimension lower than β are merged with an adjacent one. Thresholds α and β are less critical than expected, but depend on the tradeoff between precision and speed. If α or β are "low" more regions are created. Nonetheless, the following MRF phase will solve a possible initial over-segmentation.

4.3.2 Partitioned Region Matching (PRM)

The aim of the motion estimation phase is the computation of a motion vector for each region produced with color segmentation. Several variants of block matching are available in the literature (e.g. [77] and [78]) to evaluate the motion of a block with a translational model; moreover, a statistical function is needed to integrate the blocks motion measures over the whole region. When a previous color segmentation is available, as in this case, region-matching techniques produce often better results than block matching in a relative short time. To this aim we present here a variation of the classic region matching.

Differently from the block matching algorithms, that confront fixed sized blocks, region matching exploits the whole regions extracted with segmentation as comparing patterns and a function defined over the entire regions as matching value (i.e. distortion rate). For example, we could adopt the measure expressed in Eq. 4.2, that is an adapted version of the classical Sum of Absolute Differences (SAD) over a region:

$$SAD_R(\mathbf{v}) = \sum_{(x,y)\in R} \left(\sum_{i\in\{R,G,B\}} |I_t(x,y)_i - I_{t+1}(x+v_x, y+v_y)_i| \right) \tag{4.2}$$

where $I_t(x,y)_i$ is the i^{th} color component of the point of the image I at the (x,y) coordinates at the frame $t$and $\mathbf{v} = (v_x, v_y)$ is the displacement to evaluate.

The most important advantage of region matching with respect to block matching is the possibility to estimate motion of uniform regions without texture, by using the shape. In fact, the presence of at least two not-parallel gradients is required to correctly compute a motion vector (also known as the “aperture problem”, [79]). The adoption of regions in substitution of blocks makes more probable that the previous requirement will be satisfied, both for the bigger size and for the presence of the object border inside the region. At the same time, problems with region matching based on Eq. 4.2 can arise when the motion is not strictly translational or when the shape varies over time (for example in presence of occlusions). If the affine model is assumed, the Eq. 4.3 can be modified as follows:

$$\begin{aligned} SAD'_R(\mathbf{v}) &= \textstyle\sum_{(x,y)\in R} \left(\sum_{i\in\{R,G,B\}} |I(x,y)_i - I(\tilde{x},\tilde{y})_i| \right) \\ \tilde{x} &= a_1(x - x_G^R) + a_2(y - y_G^R) + a_3 \\ \tilde{y} &= a_4(x - x_G^R) + a_5(y - y_G^R) + a_6 \end{aligned} \tag{4.3}$$

where (x_G^R,y_G^R) are the centroid coordinates of the region R and $a_1 ... a_6$ are the six affine motion parameters to evaluate. The introduction of six parameters instead of two makes more complex and heavy the motion estimation. In presence of occlusions, the variation of the occluded object's shape can introduce errors. To reduce this problem, we define a new matching criterion that we call *Partitioned Region Matching* (PRM in the following), that ensembles features of both block and region matching.

PRM is defined as follows:

1) a region R is partitioned in a number n of disjoint sub-regions SR^k, so that

$$R = \bigcup_{k=1}^{n} SR^k \quad \text{and} \quad SR^i \cap SR^j = \varnothing | i, j \in 1...n \tag{4.4}$$

2) the integer, finite set **V** of shifts in all directions is defined as follows:

$$\mathbf{V} = \{\mathbf{V}_j = (v_{xj}, v_{yj}) | v_{xj}, v_{yj} \in [-s, +s]\} \tag{4.5}$$

where s is the maximum shift that the system can evaluate; please note that this integer set does not allow sub-pixel accuracy in motion estimation; once again, we prefer to limit the complexity of the system to be able to meet real-time constraints;

3) for each sub-region SR^k the function $SAD_{SR^k}(\mathbf{v}_j)$ is evaluated, according with Eq. 4.2; the motion vector $\mathbf{v}^k \triangleq (v_x^k, v_y^k)$ of the sub-region SR^k is assumed equal to:

$$\mathbf{v}^k = \arg \min_{\mathbf{v}_j \in \mathbf{V}} \{SAD_{SR^k}(\mathbf{v}_j)\} \tag{4.6}$$

4) a probability function $P(\mathbf{v})$ is associated to the region and it is computed as the product of the *a posteriori* probability (i.e., number of occurrences of each motion vector) with the *a priori* probability.

$$P(\mathbf{v}) = \frac{\frac{\sum_{k=1}^{n} \delta(\mathbf{v}, \mathbf{v}^k)}{n} \cdot P_{priori}(\mathbf{v})}{z} \tag{4.7}$$

where

$$\delta(\mathbf{v}, \mathbf{v}^k) = \begin{cases} 1 & if \;\; \mathbf{v} = \mathbf{v}^k \\ 0 & if \;\; \mathbf{v} \neq \mathbf{v}^k \end{cases} \tag{4.8}$$

and z is the normalization factor. The *a priori* probability could be computed taking into account the motion in the previous frame, physical constraints, and knowledge on the scene or application. To reduce computational complexity, in the experiments reported in this paper the *a priori* probability has been set identically to $\frac{1}{|\mathbf{V}|}$. In this case, $P(v)$ is computed simply as

$$P(\mathbf{v}) = \frac{\sum_{k=1}^{n} \delta(\mathbf{v}, \mathbf{v}^k)}{n} \tag{4.9}$$

5) finally, the motion vector $\mathbf{MV}_R = (MVx_R, MVy_R)$ of the whole region R is assumed to be the vector $\mathbf{v}_j$ that maximizes $P(\mathbf{v})$:

$$\mathbf{MV}_R = \arg \max_{\mathbf{v}_j \in \mathbf{V}} \{P(\mathbf{v}_j)\} \tag{4.10}$$

Using Eq. 4.9 to compute the probability values, the motion vector $\mathbf{MV}_R$ results to be the mode of the shift vectors $\mathbf{v}^k$.

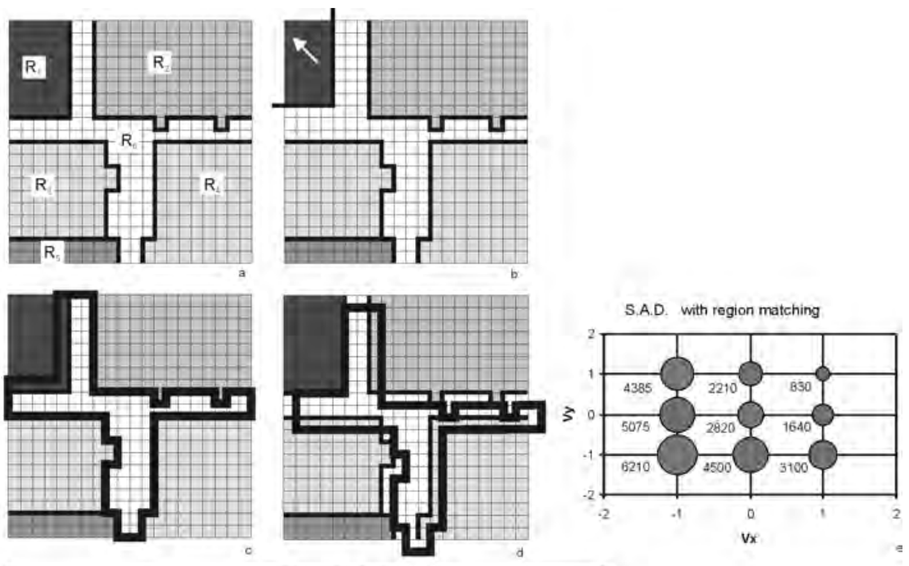
However, we must define how to compute the sub-regions as in 1). The region partitioning should present some characteristics: a) the sub-regions SR^k should not be too small, (e.g. 8x8, 16x16,), otherwise PRM turns into block matching; b) the sub-regions SR^k should have almost the same area; in this manner all the SAD_{SR} are comparable; otherwise, a suitable factor coping with different area should be included in the *a posteriori* probability; c) the edges border should be fragmented into more than one part, to reduce the influence of occlusions.

The region partitioning adopted in this work, compatible with the previous criteria, is the following: let BB_R be the bounding-box (or extent) of the region R and define a partitioning window W_L of LxL pixels. For instance, L could be 8,16, so that $Area(W_L) \ll Area(BB_R)$. Thus, BB_R is decomposed in windowed bounding-blocks WBB_i, i.e. sub-squares LxL as large as the partitioning window is ($BB_R = \bigcup_i WBB_i$). An example is in Fig. 4.2.g. The sub-regions SR_k are the not null intersections between the windowed bounding blocks and the region (see Fig. 4.2.h): $R = \bigcup_{k=1}^{n} SR_k$, $SR_k = WBB_k \cap R$, $SR_k \neq \varnothing$.

In this manner, the criteria a) and c) are always satisfied, while the b) criterium is not, because some border sub-regions could be smaller than central ones. In spite of this, the central sub-regions give the same contribution in the statistical function of the smaller border sub-regions. This means to assume that the border motion is more significant than the central one.

The defined PRM reduces the problems of occlusion typical of region matching, since occlusions affect only a limited number of sub-regions that are not significant due to the equations 4.7, 4.9, and 4.10; moreover it resolves the drawback of motion vector locality of block matching, working on larger patterns.

To better understand and appreciate the power of this solution we can consider the two consecutive frames in Fig. 4.2 (a and b); we suppose that only R_1 is in motion, while the other regions are fixed. To compute motion of R_6 (i.e., the background), we search the best match between the second and the first frame. In Fig. 4.2 (c) and (d) are represented two possible matches, obtained with $(v_x, v_y) = (0, 0)$ and $(v_x, v_y) = (1, 1)$, respectively. Obviously, the first is the correct match; Fig. 4.2 (f) shows that is also the case with the maximum number of matching pixels (the pattern extracted from the second frame covers 88 pixel of R_6 and 12 of R_1 in the first frame), but the SAD_R, if computed over the whole region, is smaller in the second case (see Fig. 4.2 (e)), where the pixel of the region R_6 covered are only 68. Therefore, a classical region matching based on SAD_R fails. This problem often occurs in real sequences, when the foreground moving objects have very different colors from the background. In this case the background's motion may be wrongly evaluated as equal to the motion of a foreground occluded object. With PRM, instead, we can estimate the correct motion vector. The partitioning of R_6 is made as Fig. 4.2 (g): for graphical reasons the size of each window is 4x4 pixels, but in the implemented system partitioning windows are larger (e.g., L=8, 16, ...). The entire region results segmented in ten sub-regions. In Fig. 4.2 (h) we



Vx	Vy	Region:	R_1	R_2	R_3	R_4	R_5	R_6	SAD_R
		Gray level:	20	200	230	240	50	255	
		$d(R_x,R_6)$:	235	55	25	15	205	0	
-1	-1		22	12	7	0	1	58	6210
-1	0		19	0	8	0	2	71	5075
-1	1		14	0	15	7	3	61	4385
0	-1		16	13	1	0	0	70	4500
0	**0**		**12**	**0**	**0**	**0**	**0**	**88**	**2820**
0	1		7	0	9	9	1	74	2210
1	-1		8	20	0	8	0	64	3100
1	0		4	10	0	10	0	76	1640
1	**1**		**0**	**7**	**7**	**18**	**0**	**68**	**830**

Composition of the match

f

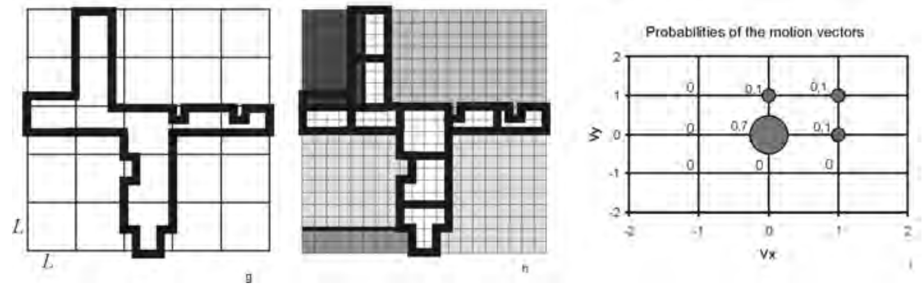


Figure 4.2: *Example of partitioned region matching (PRM). (a)(b) two consecutive frames where R_1 is the only moving object, (c)(d) two possible matches of R_6, obtained with $(v_x, v_y) = (0,0)$ and $(v_x, v_y) = (1,1)$ respectively, (e)(f) SAD_R values with Eq. 4.2, the columns R_i in (f) are the number of pixel matching $R_6 - R_i$ that must weighted with the correspondent distance* d *in color (e.g., first row: $6210 = 22 \cdot 235 + 12 \cdot 55 + 7 \cdot 25 + 1 \cdot 205$), (g) parts creation for PRM; (h) best match with PRM, (i) probabilities of the motion vectors*

can see that only three of ten sub-regions of the R_6 are affected by the occlusion of R_1, while most of sub-regions can correctly estimate the motion vector. Fig. 4.2 (i) shows the probabilities $P(v_x, v_y)$ associated to the motion vectors (v_x, v_y): the vector with the maximum value is the right: $(v_x, v_y) = (0, 0)$.

For each region R we also define a *motion reliability* as:

$$\rho_R = \frac{\sum_{(x,y)\in R} \left[\sum_{i\in\{R,G,B\}} \left(\left| \frac{\partial I(x,y)_i}{\partial x} \right| + \left| \frac{\partial I(x,y)_i}{\partial y} \right| \right) \right]}{SAD_R(\mathbf{MV}_R) + 1} \tag{4.11}$$

ρ_R takes into account two factors: the gradient in the color image and the goodness of the best match found, evaluated as the SAD (obtained with a shift of $\mathbf{MV}$) increased by 1 to prevent division by 0. If a region presents a low texture, the choice of the best match could be ambiguous, therefore the reliability of the motion estimation is low too. Reliability is low also if the best match presents an high distortion and thus an high SAD (e.g., in presence of occlusions and deformations).

Moreover, the motion reliability value resolves another problem: region matching (as all the other motion estimation techniques) can not evaluate the motion in the image border. If a region has a bounding-box entirely contained in the image border (large as s of Eq. 4.5) , we set ρ_R to 0.

4.3.3 Region level MRF

The Markov Random Fields result very powerful to resolve segmentations and classifications. In this kind of problem, the undirected graph is the typical representation of the MRFs, with nodes corresponding to a set of variables and the arcs reporting their interactions. The nodes contain both variables based on the observations of the system modeled (e.g., color, motion, etc...) and the output values (e.g., the labels). Moreover, a function of this variables (i.e., the energy function), is defined over the whole graph. The main property of the MRFs is the independency between the variables of nodes not connected with an arc; consequently, the energy function could be decomposed as the sum of local terms defined over the *two-site cliques*[11] (i.e., couple of nodes connected with an arc). Goal of a MRF framework is the minimization of the energy function and, to this aim, an optimization algorithm should be introduced.

We use a Markov Random Field framework to merge regions with similar motion. As proposed in [39], the energy function is composed by three terms: one based on motion (U_1), one used to perform a geometrical regularization (U_2), and one based on the number of labels assigned (U_3):

$$U(e, o) = U_1(e, o) + U_2(e) + U_3(e) \tag{4.12}$$

where e and o are respectively the labels and the observation fields.

[11]Hereinafter, we will refer to a two-sites clique simply with the term “clique”.

Differently from [39], we exploit motion reliability too, including it in the energy term based on motion:

$$U_1(e,o) = \sum_{(s,t)\in\Gamma} V_1(e_s, o_s, e_t, o_t) \tag{4.13}$$

$$\begin{gathered} V_1(e_s, o_s, e_t, o_t) = \begin{cases} 0 & e_s \neq e_t \\ c_1 \cdot \|MV_s, MV_t\| \cdot \sqrt{\rho_s \cdot \rho_t} & e_s = e_t \end{cases} \\ \|MV_s, MV_t\| = \sqrt{(MVx_s - MVx_t)^2 + (MVy_s - MVy_t)^2} \end{gathered} \tag{4.14}$$

where Γ is the set of two-dimensional cliques and s and t are two sites of a clique (corresponding to regions); MVx and MVy are the two components of the motion vector, ρ is the motion reliability and c_1 is a positive constant. The motion distance is computed with Euclidean formula, while the reliability term is included to make motion-based energy term U_1 more significant than the geometrical term in presence of high values of reliability.

In addition, we can decompose the geometrical regularization term into the sum of local terms defined over the cliques as in [39]:

$$U_2(e) = \sum_{(s,t)\in\Gamma} V_2(e_s, e_t) \tag{4.15}$$

$$V_2(e_s, e_t) = \begin{cases} 0 & e_s \neq e_t \\ -c_2 \frac{\xi_{s,t}}{\xi_{s,t} + \sqrt{(G_x^s - G_x^t)^2 + (G_y^s - G_y^t)^2}} & e_s = e_t \end{cases} \tag{4.16}$$

where c_2 is a positive constant, ξ is the length of the shared border and $\mathbf{G} = (G_x, G_y)$ is the region centroid. This term enhances the fusion of two adjacent regions with a long shared border and a small distance between centroids.

The last term takes into account the number of labels assigned, i.e. the cardinality of e ($\#e$):

$$U_3 = c_3 \cdot \#e \tag{4.17}$$

where c_3 is a positive constant.

In conclusion, the motion based term tries to keep separate regions with different motion, geometrical term decrements the weight of motion distance if there is a strong adjacency and U_3 represents a sort of motion difference threshold under which two regions will be merged.

The optimization of the energy function is performed with a multi-scale iterative algorithm. A label and a binary stability flag (preset to “unstable”) are associated to each node; initial label is assigned according with a prediction computed on the previous frame. Known the labels of regions in the previous frames we predict the label of each node by computing the maximum overlapped area and imposing motion continuity. The algorithm extracts one by one in random order the regions with “unstable” flag and evaluates which label (taken from a set composed by the old label, the labels of the adjacent nodes and an outlier label to allow separation of connected regions) minimizes the energy function. The new label is assigned to the node, the stability flag is set to “stable” and, if the label has been changed, the state of the adjacent nodes is set to “unstable”. When all the nodes become stable, a compression of the graph is performed by grouping nodes with the same label into a single label; then the algorithm is iteratively exploited on the new graph.

4.3.4 Background and moving object identification

After MRF optimization, we should obtain a final graph with each node corresponding to an object (or a blob containing objects with the same motion). The largest one is assumed to be the background. Nevertheless, assuming that all the other regions are moving objects could be incorrect, since the background could be separated in more parts by interposed objects. To prevent this, every region with the same motion of the background is merged with it, even if not adjacent. To be less sensitive to noise, we can do the same with regions that exhibit similar motion (i.e., the Euclidean distance in the velocity space must be under a suitable threshold). Remaining regions are assumed to be moving objects. If small regions are not relevant for the application, a size threshold can be introduced.

4.4 Experimental results

The presented system has been implemented and tested on both synthetic and real sequences[12]. *Ground-truth tests* (i.e. tests with manually segmented frames as ground-truth) have been performed to evaluate the efficacy. Finally, a time analysis has been carried out to study dependencies on parameters and the capability to satisfy real-time requirements. In the following subsections we will analyze the results obtained on different sequences.

4.4.1 “Object sequence”

“Object” sequence (Fig. 4.3) is synthetic video composed of 20 frames of 200x200 pixel each. It has been made properly to test the system and contains five objects in strictly translational motion over a mobile background. In particular, there are:

- a red circle O_1, with homogeneous color but with a vanished border. The object is entering into the scene and its shape and size are varying on the time; this tests the reliability of the region matching in presence of objects that are entering/exiting from the scene;
- an ellipse O_2 with a very strong texture: it represents an object that color segmentation can not entirely extract (see Fig. 4.3) and only the motion field can correctly reconstruct;

[12]The videos are available at the Imagelab Laboratory page http://imagelab.ing.unimo.it/imagelab/benchmark.asp

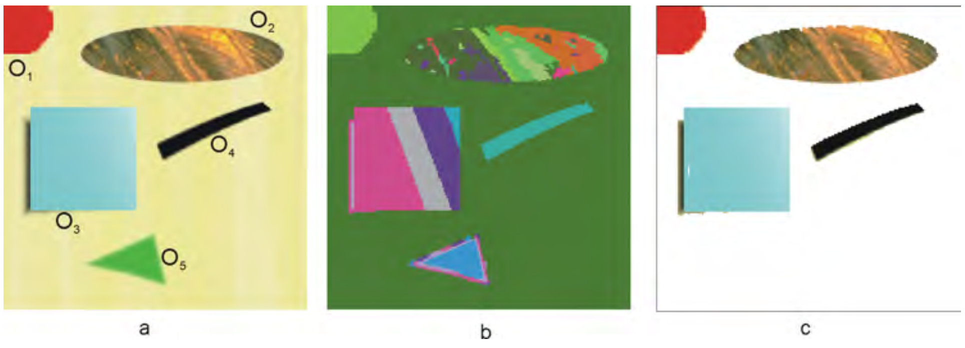


Figure 4.3: Frame 6 of object sequence: (a) original frame, (b) after color segmentation with region growing and (c) moving objects reported on output (in this frame O_5 is not in motion).

Table 4.2: Size and centroid coordinates estimation errors on object sequence. The size mean error of the circle is carried out only in percentage because the circle is entering on the scene and its size is not fixed

- a blue square O_3, with vanished color: darker in the bottom left with a shadow;
- a black bar O_4, with uniform color: is an ideal object for this application;
- a green triangle O_5, with a transparency;
- the background: is yellow, with a weak texture, just sufficient to compute its motion.

The output of the system confirms correctness of the approach: all the moving objects are correctly retrieved in the whole sequence.

The output was also subjected to a ground truth analysis: see Fig. 4.4 with false positives and false negatives at pixel-level. As it can be seen from Table 4.2 the objects with shadows or vanishing borders have some pixels that are not correctly segmented, but are assigned to the background; this causes the false negatives of Fig. 4.4. The false positives at frame 10 correspond to a new region that the circle leaves in the top-left corner of the image while is still entering (the motion of a new region is unknown and it is assumed to be an indefinite value).

The analysis of the centroids' coordinates is more significant for our application, because it consents to evaluate better the efficacy of the motion estimation and the region merging phases (rather than the color segmentation); Table 4.2 shows that the estimation error is null for the bar and irrelevant for the other objects.

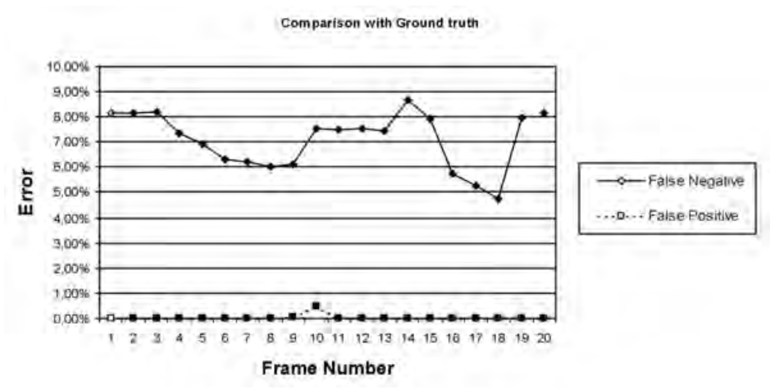


Figure 4.4: Errors expressed in percentage of the total (in pixels) over object sequence. The values refer to objects in motion (different from the background). Thus, false negatives are moving objects' points that are not detected and false positives are points of still objects or of the background labeled as belonging to moving objects.

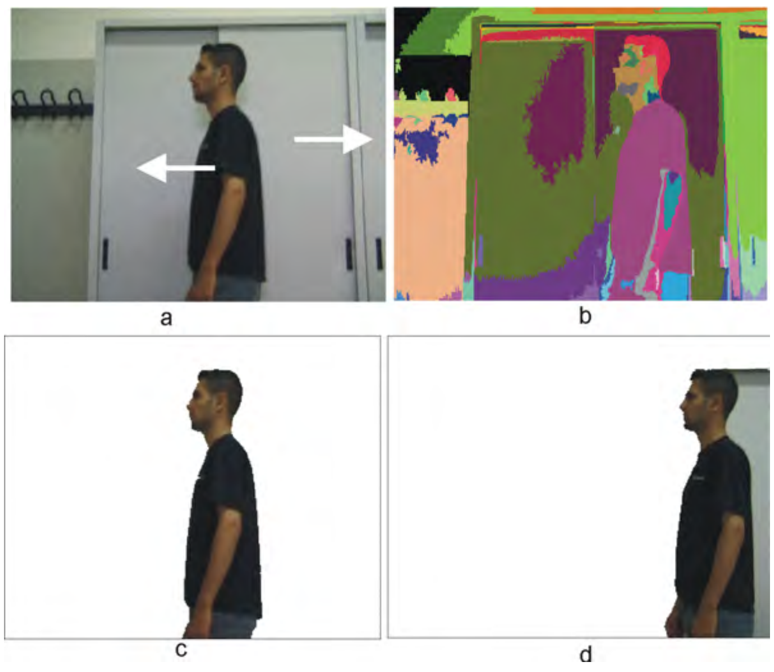


Figure 4.5: Frame 140 from indoor sequence: a) original frame, (b) after color segmentation with region growing, (c) moving objects reported on output. (d) Output of frame 50, with a background's region wrongly merged with the foreground moving object.

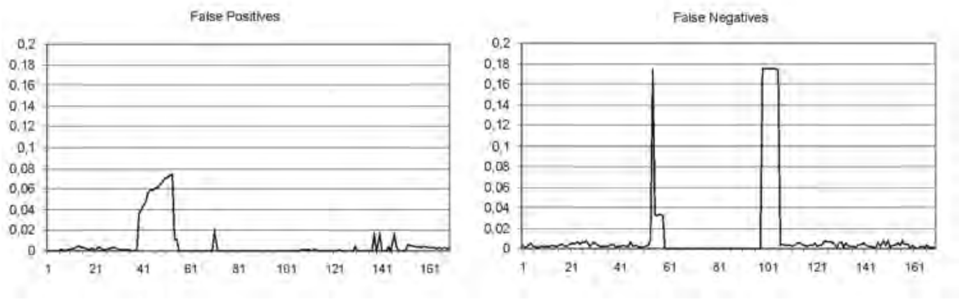


Figure 4.6: Positives and negatives (as in Fig. 4.4) over the indoor sequence

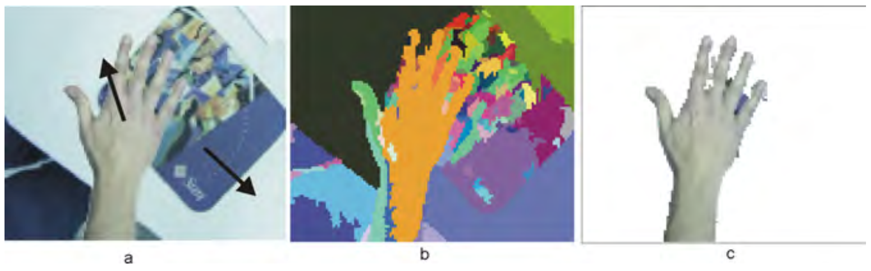


Figure 4.7: Hand sequence: (a) an original frame, (b) after color segmentation, and (c) output

4.4.2 "Indoor sequence"

Fig. 4.5 shows an original frame of an indoor video, the color segmentation, and the final output. The camera and a person are moving in opposite directions (see superimposed arrows).

A ground-truth analysis is performed (Fig. 4.6). The presence of false positives between frames 41 and 61 is caused by a wrong motion estimation. The motion of the background region at the right of the man is equal to the man's motion and the two regions are merged (see Fig. 4.5.d). In this case, neither standard region matching, nor PRM can correctly compute the motion value; in fact, they can not exploit the texture, because the region is homogeneous, nor the shape because the area presents half border affected by the occlusion of the man and the other half corresponding to the image border, always stationary. In the frames from 75 to 100 the person is stopped: the system works correctly, also in presence of the unavoidable camera noise. From frame 100 to 108 the person is not completely detected due to its initial very slow motion, and this is shown in the peak of false negatives reported in the right graph of Fig. 4.6.

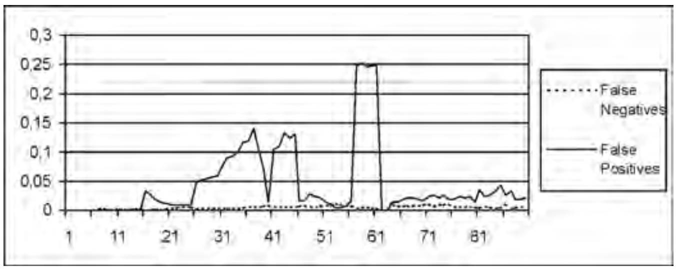


Figure 4.8: False negatives and positives (as in Fig. 4.4) over the hand sequence

Size	Param. of R.G. (α, β)	R.M. Max Shift	# regions	Computational time (seconds) Segm. + SRG	Motion estim.	Region MRF	Total
100x100	70-8	2	43	0.125	0.047	0.031	0.203
200x200	70-25	4	36	0.578	0.531	0.031	1.140
400x400	70-120	8	32	2.562	6.625	0.094	9.281
100x100	70-20	6	23	0.125	0.172	0.016	0.313
200x200	70-20	6	41	0.562	0.938	0.031	1.531
400x400	70-20	6	116	2.594	4.156	0.391	7.141

Table 4.3: Processing time dependencies from parameters. In the first three rows the parameters are scaled with the dimension, while in the other three are fixed.

4.4.3 "Hand sequence"

In hand sequence (see Fig. 4.7) a hand is moving over a desk and the camera is in motion too. This sequence is very hard to analyze, because the background is composed by a lot of small regions (of the mouse pad) with some large and fine-textured ones (of the desk) in addition. Motion estimation is difficult on both: on the small regions when the hand occludes them and on the white desk for the absence of gradients. Fig. 4.8 reports ground truth analysis: in this case, the false positives are some details of mouse pad connected to the hand.

4.4.4 Computational performance analysis

The system proposed on [39] produces satisfactory results in terms of accuracy of segmentation and motion detection (as stated by the authors), but it is computationally too expensive to satisfy strict time requirements (as indicated in [39] and in [80], they needed 80 sec for frame on a ULTRASPARC). Our initial idea was not to develop a more reliable implementation, but to meet real-time requirements and, and the same time, maintain acceptable results.

In this section we present a detailed time analysis that shows the cost of each phase and their possible dependencies on application parameters. In particular:

Max shift (s)	PRM times (seconds)
2	0.0438
3	0.0658
4	0.1022
5	0.1344
6	0.1804
8	0.2814
12	0.5124
15	0.7406
20	1.0606

Table 4.4: Times (in seconds) for motion estimation vs. maximum shift. Frame size: 100x100

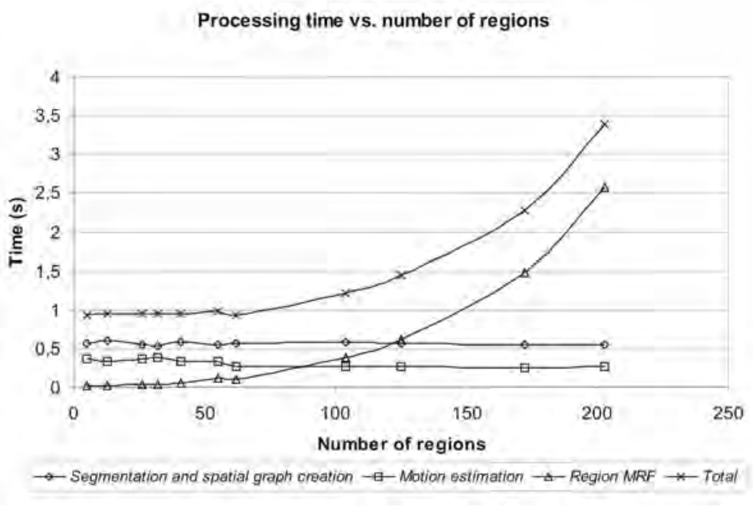


Figure 4.9: Processing time vs. number of regions (on object sequence 200x200)

- *Color segmentation*: the time necessary for this phase depends on the frame size (see Table 4.3); instead, number of created regions and color threshold do not affect it (see also Fig. 4.9).

- *Spatial graph creation*: is very fast and its time cost is negligible.

- *Motion estimation*: is the most critical phase, not only for its algorithmic complexity, but also for the several factors that concur to make difficult the computational time estimation. In fact, it depends on the frame size s, and on the number of regions created by the color segmentation. Table 4.3 and Table 4.4 show the execution time of motion estimation with PRM.

- *Markov Random Field optimization*: for this task, the frame size is obviously not relevant, because the application works only on the graph. On the other hand, the time occurred for energy minimization varies with the number of regions (see Table 4.3 and Fig. 4.9). As it is possible to see, time required for MRF increases not linearly with the number of regions. Table 4.4 shows that PRM is affected by the maximum shift s. If we suppose not to have fast objects (e.g., with $s = 2$) PRM asks for 43.8 msec for frame (it grows up to 1 sec if we shift the region up to 20 pixels).

Calling n the number of nodes of the graph, i.e. number of regions, the minimization algorithm proposed in section 4.3.4 has a complexity of $O(n^4)$. In fact, each iteration analyzes all the n nodes; for each of them it computes the energy value on the n arcs linking it with the others. If the state of the analyzed node changes, all the other n nodes may be re-evaluated. Supposing that each node is evaluated n times, we obtain a complexity of $O(n^3)$. Then a multi-scale optimization is performed, that consists, in the worst case, of n iterations.

As indicated by the total time costs in Table 4.3 and Fig. 4.9, we are far from satisfying real time constraints. In fact, although downscaling the frame size can allow very fast computation of region growing and motion estimation, the MRF optimization requires too high computational times (consider that region growing usually produce almost 30 regions). For this reason, in the next section we propose an innovative algorithm of optimization based on arcs instead of nodes characterized by a quadratic complexity.

4.5 Arc based optimization

4.5.1 Energy function

First, the energy function previously presented has been slightly modified; in particular, the term U_3 is now defined as the number of *broken arcs* (i.e. pairs of regions geometrically adjacent but with different labels). The third energy term is, thus, expressed in a form similar to the other two terms:

$$U_3(e) = c_3 \sum_{(s,t)\in\Gamma} V_3(e_s, e_t) \tag{4.18}$$

$$V_3(e_s, e_t) = \begin{cases} 1 & e_s \neq e_t \\ 0 & e_s = e_t \end{cases} \tag{4.19}$$

where c_3 is a positive constant.

The weak proportionality between number of labels and *broken arcs* justifies this change. The global energy function can be now entirely decomposed into the sum of local terms defined over cliques:

$$\begin{aligned} U(e, o) &= \sum_{(s,t)\in\Gamma} U_{s.t}(e_S, e_t, o_s, o_t) = \\ &= \sum_{(s,t)\in\Gamma} V_1(e_s, e_t, o_s, o_t) + V_2(e_s, e_t) + V_3(e_s, e_t) \end{aligned} \tag{4.20}$$

Analyzing the definitions of the three energy terms we can define a unified local term as:

$$U_{s,t}(e_s, e_t, o_s, o_t) = \begin{cases} V_1(e_s, o_s, e_t, o_t) + V_2(e_s, e_t) & e_s = e_t \\ V_3(e_s, e_t) & e_s \neq e_t \end{cases} \tag{4.21}$$

We can also assign a binary state to each arc that identifies if the two connected regions have the same label or not.

4.5.2 Energy optimization

The main advantage of Eq. 4.21 consists on the possibility of a faster optimization. The algorithm here proposed is composed by two scanning phases: the first on the arcs and the second on the nodes. As above reported, the computational complexity becomes quadratic $O(n^2)$ (evaluation of n^2 arcs plus n nodes).

For each arc (s,t) we evaluate if it is more convenient to break it or not, by considering only the sub-graph composed by the arc and the two connected nodes (s and t). To this aim we define $W_{s,t}$ as in equation 4.22 and the test to perform on each arc is shown in equation 4.23:

$$W_{s,t} = W_{s,t}(e_s, e_t, o_s, o_t) = V_1(e_s, o_s, e_t, o_t) + V_2(e_s, e_t) - V_3(e_s, e_t) \tag{4.22}$$

$$W_{s,t} \geq 0 \Rightarrow (s,t)\ \ broken \tag{4.23}$$

After that, the nodes of the graph are labeled using a modified region growing algorithm. Starting from the first node, all the nodes connected with non-broken arcs are merged with the same label. This optimization is sub-optimal because the graph may present a sort of inconsistence.

In fact, Fig. 4.10.a shows a segmented frame of the indoor sequence and the correspondent SRG. A particular containing three adjacent regions (labeled with

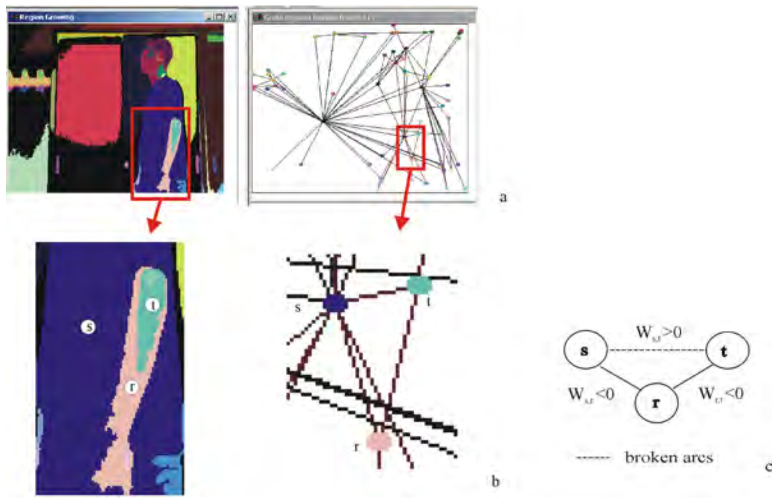


Figure 4.10: Possible graph inconsistence during arc based optimization

r,s,t) is zoomed in Fig. 4.10.b. If we suppose that the signs of $W_{s,t}$, $W_{s,r}$, and $W_{r,t}$ are as in Fig. 4.10.c (positive the first and negative the others two) the graph will become inconsistent. The arc (s,t) is *broken*, but the growing algorithm assigns the same label to nodes s and t walking through arcs (s,r) and (r,t) with the consequent increase of the energy associated to the arc (s,t) and the global energy is not minimized. Instead, an optimum algorithm should evaluate if preserving a unique label for the three nodes is "more convenient" than assign a different label to node s or node t. For this reason, the defined algorithm is not generalizable for any graph, while it is applicable with good results in our application. In fact, the energy function is principally based on motion difference. The situation presented in figure is very infrequent, because it means that r and t have a similar motion, the same for s and r, but not for s and t.

The main task for our purposes is to merge regions with same motion and separate regions with *very different* motion. In other cases, possible errors due to segmentation and motion estimation result to be more relevant than previous simplifications in MRF optimization. Unfortunately, with this new algorithm, we lose the implicit tracking of the objects, because the labeling phase does not consist in an optimization of a prediction as in section 4.3.3 and in [39]. The prediction is useless since labels are given by following the non-broken arks. Indeed, a final tracking phase after the end of motion segmentation could be added, if needed. The performance analysis reported in the next section shows the validity of the method and the advantages in terms of speed.

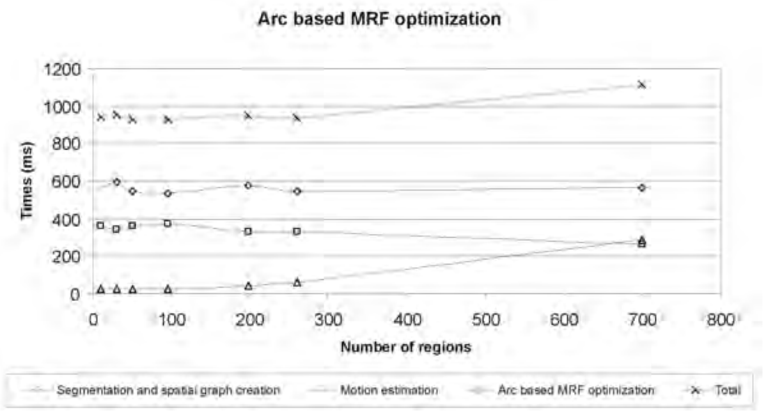


Figure 4.11: Computational time for arc based MRF optimization (on object sequence 200x200)

4.6 Performance analysis on the modified algorithm

4.6.1 Processing times

The main advantage of this new solution is the reduction of computational complexity. The graph in Fig. 4.11 shows time analysis for arc based MRF optimization with respect to the number of initial regions. Comparing it with the one of node-based MRF of Fig. 4.9, the improvement of speed is evident; this phase is not the bottleneck of the system anymore. For instance, for 50, 100, and 200 regions it requires only 23, 26 and 45 ms, respectively, and node-based MRF requires about 970, 1200, and 3300 ms, respectively. In particular, if we analyze "object sequence" and "hand sequence", we downscale them to the size of 100x100 pixel, and assuming s equal to 2, we are able to reach 10 frame per second on a standard dual Pentium III 1000 Mhz with 512 MB of RAM. Obviously, using a more off-the-shelf, powerful PC would allow us to reach better performance.

4.6.2 Qualitative metrics

In remains to prove that the sub-optimal optimization introduced does not affect much the efficacy of our system. To do this, the new system was tested on the same sequences used previously. In the "objects sequence" we obtained identical results: the color segmentation is unchanged and the region motion values are always correctly computed; both the MRF algorithms of sections 4.3.3 and 4.5.2 can extract the objects in motion correctly. Instead, in the case of the "hand sequence" the output is slightly worse with the new optimizer, because wrong motion values are assigned to the parts of small regions affected by occlusion (compare Fig. 4.12 and Fig. 4.8). In Fig. 4.13 the number of pixels in real motion and the ones computed as in motion with the new optimizer are reported. Anyway, as it is possible to see, the results are satisfactory since the difference is limited between the two

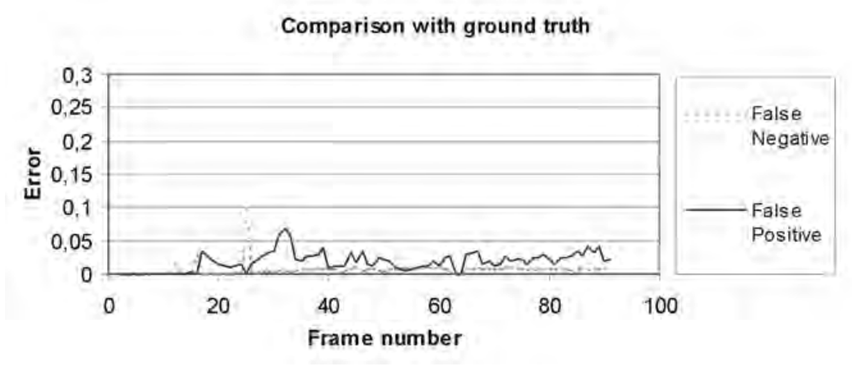


Figure 4.12: False negatives and positives (as in Fig. 4.4) over the hand sequence with arc-based optimization

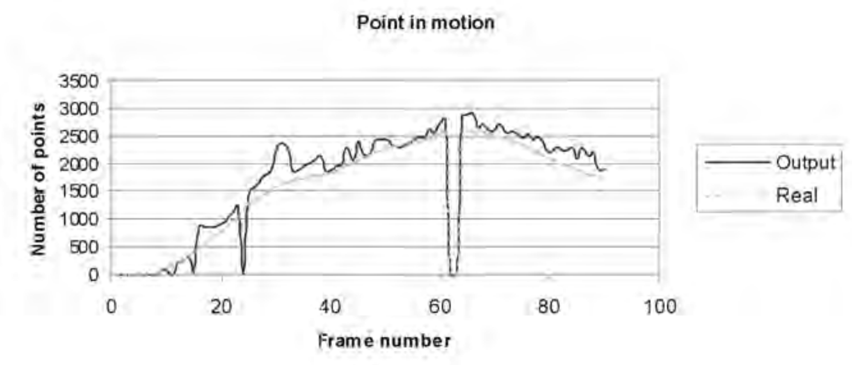


Figure 4.13: Points in motion on hand sequence calculated with arc based optimization

profiles: the “output” curve is higher than the “real” curve because of the merging of small regions of background to the moving hand. Similar performance results in efficacy has been tested in the “indoor sequence” and in other videos

4.7 Conclusions

In this Chapter a method for foreground segmentation on video acquired by a moving and unconstrained camera has been presented. The work on this topic was conducted in 2004, when the PC performances were considerably worse than now. The proposed approach is based on color and motion segmentation with a MRF framework; it achieves satisfactory results when the motion can be approximated with a translational model. A new Partitioned Region Matching has also been proposed to perform a good motion estimation, even in presence of occlusion or shape variation. This motion estimation method, that is a sort of mixing between traditional region matching and block matching, can be used in a lot of different applications.

Chapter 5

Appearance Based Tracking

5.1 Introduction

After a foreground image F has been extracted the real moving objects generating F must be identified, separated, and followed over time. To this aim, several tracking algorithm have been studied and proposed. Among them, appearance-based tracking algorithm are characterized by some peculiarities that make them particularly suitable for video surveillance applications.

Appearance-based tracking is a well established paradigm to predict, match and ensure temporal coherence of detected deformable objects in video streams. These techniques are very often adopted as a valid alternative to approaches based on 3D reconstruction and model matching, by computing the visual appearance of the objects in the image plane only, without the need of defining camera, world and object models. Especially in human motion analysis applications, the exploitation of *appearance models* or *templates* is straightforward. Templates enable the knowledge not only of the location and speed of visible people but also their visual aspect, their silhouette or the body shape at each frame.

Appearance driven tracking is often employed in video surveillance applications, and particularly in people surveillance, action analysis and behavior monitoring, in order to have a precise information about the visible and non visible body aspect at each instant [5, 81–87]. It is used also in human-computer interaction applications [88], in gesture analysis [89] and in many other problems coping with moving objects with deformable shapes.

The appearance model is defined in the image plane as an adaptive model of the visual aspect of an object, or a selected shape or a segmented blob during the time. It is normally represented by a mask A updated as a IIR filter: for each point x of the object, $A(x)$ is computed as a linear function of the previous value of the model and the current observation value $I(x)$:

$$A(x,t) = k_1 \cdot A(x,t-1) + k_2 \cdot I(x,t). \tag{5.1}$$

The parameters k_1 and k_2 take into account the variability of the visual appearance of the points of the object during the time. Eq. 5.1 can be defined in the gray level [5] or in the color space [38, 81–83, 90].

Appearance models are used in tracking algorithms to ensemble the visual status of an object previously tracked. In *discriminative bottom-up* tracking, which searches the best association between a known object and the current image as a result of an optimization problem, the appearance model is embodied in similarity functions to be optimized. These approaches are normally deterministic, in the sense that a single solution at each frame is carried out. As [91] states, bottom-up algorithms are computationally efficient, but it is hard for them to cope with occlusions unless the appearance model itself is robust against occlusions. In *probabilistic top-down* tracking appearance models can be used together with position, speed and other variables to define the status of the object, estimated by a probability distribution function (*pdf*). These methods have a more accurate motion estimation, especially because many solutions can be considered at the same time (e.g. in Particle Filtering) but are more computational expensive. Here occlusion can be modeled from top-down in the same framework, for instance with Bayesian Networks [91]. Many of these methods are called *generative*, since more hypotheses are generated, also by modeling the hidden factors that would affect the observed data. In this second case, appearance models have often a more compact representation, as for instance with color histograms [82, 84] since an appearance model at pixel level can make the pdf dimensionality to high to be computed in time constrained applications. On the other side, when the time is not a constraint but a precise tracking is required (e.g. in medical imaging), more complex appearance models have been proposed: for example, appearance models can be a learned PCA-based framework exploiting both shape and texture, as in Active Appearance Models [90] or Adaptive Active Appearance Models [85].

In deterministic tracking, appearance templates are typically matched against selected regions of the image plane, by means of a previous process of motion or color segmentation. In this case, they are often associated to *probability masks* that contain for each point x a $P(x)$ value between 0 and 1 indicating the probability to have a visible appearance in that point of the reference object [5, 38, 81, 90]. The probabilistic mask is used as a weight in similarity functions to search the best match between object and observation, in order to give a greater importance to stable points in the object appearance.

Appearance-based deterministic tracking offers several advantages in real-time applications: the generality and flexibility, since no object model is required; the speed, since only one solution is maintained at each frame step, and the large amount of information available, since appearance is normally kept at pixel level. Besides, appearance-based tracking of data extracted by a previous segmentation inherits the typical errors caused by the segmentation: partial segmentation, split and merge problems must be solved at tracking level. In addition, the big challenge of appearance-based tracking is the frequent presence of visual occlusions. Occlusions make the current observation totally or partially not available for some time intervals. Therefore, the optimization process of observation-object matching could have more than one solution, produce errors and imprecision. The update process of the appearance models can be incomplete or incorrect with an error propagation and an unavoidable loose of the object identity.

For these reasons, many works address the occlusion handling with appearance-driven tracking. Most of them deal with *dynamic occlusions* only, (sometime called *inter-object occlusions*) i.e. occlusions due to other moving objects. Dynamic occlusions in people tracking generate the so-called “groups of people”: during the occlusions separate people are detected in a segmentation phase as a unique blob. Many tracking proposals are capable to detect the occurrence of an occlusion, associate the unique blob to a set of tracked objects and resume after the occlusion is ended. Other approaches use heuristics or exploit the object models to solve dynamic occlusions, as for instance counting the number of heads in people group tracking [5].

Few works deal explicitly with the problem of *scene occlusions*, i.e. occlusions due to not moving objects that in the scene are closer to the camera w.r.t the observed object. Scene objects can partially hide the observation data. The result is that the appearance model is updated as in the case of a shape deformation. This erroneous appearance model change causes errors in all the measures associated with visual appearance, such as the computation of trajectory, the posture analysis and so on.

Our tracking system focuses on the problem of dealing with different classes of occlusions in the appearance-based deterministic tracking framework to follow deformable shapes and in particular human shapes. The primary goal is to have, at each frame, an appearance-model at pixel level as more accurate as possible, with a very reactive update process copying with frequent shape variations. At the same time, we want to deal with the problem of both dynamic and scene occlusions during their occurrence and not after, updating the appearance model selectively.

This chapter provides a formal definition of our approach, called *Appearance Driven tracking with Occlusion Classification* (*Ad Hoc*). Differently from many other proposals, Ad Hoc tracking works at pixel-level and not at a blob-level, in order to avoid split and merge problems. The problem is formulated in a probabilistic Bayesian model, taking into account both motion and appearance status. The probabilistic estimation is redefined at each frame and optimized as a MAP problem so that a single solution for each frame is provided in a deterministic way. We do not track each object separately but the whole object set is considered in the tracking in a two-step process: a first step, top-down, provide an estimation of the best position of all the objects, predicting their position and optimizing them in a MAP algorithm, according with the pixel appearance and a specifically defined *probability of non-occlusion*. The second step is instead discriminative and bottom-up, since an association of each observation point to the most probable object is done. Thus, the appearance model of each object point is selectively updated at pixel level in the visible part ensuring high reactivity in shape changes.

The novelty of the work is a formal model of *non visible regions* that are non negligible parts of the appearance model not observable in the current frame, where the pixel to object association is not feasible. Non visible regions are classified in three classes, depending on the possible cause: *dynamic occlusions*, *scene occlusions* and *apparent occlusions* (that are just only shape variations). The first are detected in the Bayesian point association; the second and the third ones are discriminated by an analysis of the edges against the background edges. The model is updated differently in the different cases; the appearance is modified with current color value only in the visible pixel; the probability value associated with each pixel is reinforced in visible pixels, smoothed in not visible points due to apparent occlusion and "frozen" in points hidden by a dynamic or static occlusion.

Ad Hoc tracking can be used for whichever shape of objects. Nevertheless we often refer to human tracking since the algorithm is particularly suitable to deal with dynamic shape changes and the non rigid motion of humans.

The context of video surveillance is considered for both outdoor and indoor applications. The proposed tracking is particularly suitable in indoor surveillance when the people body is completely visible in a sufficient large size, and the precise shape tracking is an important task. Despite of the relatively complex model, the derived algorithm is simple, fast and robust and has been applied in many different applications for posture analysis [87], distributed camera people surveillance [92], and also in stopped vehicle surveillance [11].

5.2 Related works

The use of appearance-based tracking with templates for deformable objects, and in particular for humans, is now very widespread, since the pioneer works of Hartiaglou et al. [5], and Bobick and Davis [93]. In these papers, the adaptive templates were initially modeled more for human activity analysis than for tracking.

W4 [5] introduced appearance models and probability maps called "gray-scale textural appearance" and "shape component" respectively. The textural templates keep the adaptive information of the shape visual appearance at gray-level and are included with the probability maps in the "weighed similarity" function. Templates have not been used to cope with people occlusions. The presence of people occluded by other ones (forming a "group of people") was instead addressed with a specific algorithm counting the heads with projective histograms. In [93] the authors proposed a two-component temporal template with a *motion energy image* and a *motion history image* that are exploited to recognize different human actions.

In successive works, appearance models have been inserted directly in the tracking loop. The models are put in correspondence with the current observation composed by the moving blobs obtained with a segmentation process. Typically with fixed camera videos, the observed data are foreground blobs after background suppression. Methods based on Mixture of Gaussian [94], adaptive median [1], a combination of both [95], or other methods can be exploited to extracts foreground points normally grouped into blobs before the tracking algorithm. Senior et al. in PETS2001 [81] defined appearance models and probabilistic maps to track people and vehicles also partially overlapped. In this work, short term occlusions are implicitly taken into account in the adaptive model. Short term occlusions, like shape deformation contribute to smooth the appearance model according with a adaptive coefficient. Tracking fails if the occlusion duration is too high. In case of group of people projective histograms of the probability map are evaluated: blobs are separated vertically in the parts where the probability map is low. It works only if the persons are sufficiently separated. For instance In [38], in order to cope with dynamic occlusions of other people, the segmented blobs have been grouped into macro-objects with potential occlusions and then appearance-based tracking is solved with models against the points of the macro objects. Similar solutions are currently adopted in many video surveillance systems and prototypes. The work [81] has been implemented in IBM S3 [96]. The solution of W4 has been improved in [90], using textural and shape templates at level of body limbs. The definition of body limbs is often adopted if tracking is followed by a posture classification step. In a very recent work [97], an initial limb body model initializes the search of people with HMM and then appearance models of limbs are learned during the tracking to have a precise data about the people posture and action.

One of the most cited solution is the work of Zhao and Nevatia [82]. They state that blobs cannot be used efficiently, since do not incorporate object constraints; blobs have problems of structural changes (merges and splits) that cause the combinatorial search expensive. Therefore they propose the use a simple human model (a normalized ellipsis after homography) instead of blob. In this region they compute T *textural template* (appearance model in rgb space) and *foreground probability* (Fp). Tracking is solved in a discriminative way by estimating the best position with a similarity function with T, Fp and the current image. The motion estimation is provided with a Kalman Filter. Similarly, Ad Hoc is driven by appearance and probability model. Differently form [81, 82] we do not associate the model to a blob, neither to a blob conjunct with a human model like an ellipsis, but we work at pixel level trying to associate each foreground pixel to each object in the tracked object set. This allows a more precise appearance update for each object and a straightforward detection of dynamic occlusions.

We use a first-order model motion prediction, as it is very common in people tracking (see for example [86, 90]), that is refined with a linear regressive method over the trajectory in order to smooth the frequent small errors in motion prediction due to the non rigid body motion. More complex predictive models are not necessary since the prediction is followed by a local search of the best alignment. It can be improved with a Kalman-based prediction in some specific cases.

Our proposal does not exploit generative methods such as Particle Filtering [86] Monte Carlo-based methods [82] or Bayesian networks with HMM [98] that keep the status of each object represented by a pdf, possibly discretized by a set of candidate solutions. In many applications where real-time computation is mandatory a trade-off between number of evaluated solutions and the granularity of each solution must be defined. Generative methods take thousand of solutions (for instance 1000 or 2000 particles in [98]) but are often used in conjunction with compact appearance models, such as color histograms or color distribution [86, 99]. In our case, since the appearance is update at pixel level and many comparisons are necessary, especially in video with frequent and large occlusions, a discriminative methods that consider tracking as an optimization problem is preferred.

Most of the aforementioned approaches address the problem of occlusion. Generative models make it by definition since maintain a large set of hypotheses, and some of them survive and resume after the occlusion. Instead, discriminative appearance-based approaches generally deal with the problem of occlusion detection only. The ratio between the number of observable points and the points of the appearance models provide a measure of possible occlusion: in [82] and in [100] this ratio determines if the object is partially or totally occluded and in that case the model is not updated. In [90] an appearance model for tracking segmented object is proposed: in case of occlusions authors state that adopt a particle filtering with a correlation measure, but there are no information about how detect occlusions and how switch between a deterministic and a particle filtering tracking.

Also in [101] an occlusion is detected when pixels appearance deviates too much from the object model: here Jepson et al. proposed an appearance model which is a mixture of three components: a stable part, learned over long time courses, a "wandering" part which accounts for rapid temporal variations, and an outlier process. Short-term occlusions are solved by not assigning the occluded pixels to the stable part. In [83], the robust Kalman Filter for each point makes the appearance model resistant to short-time partial occlusions, but the authors admit its failure to handle long-time occlusions.

We also perform occlusion detection as in [82, 100], with a confidence measure weighted by the probability, but we add an occluded region classification to update the model selectively copying with both static and dynamic occlusions. In [102] occlusions are classified as "inter-object occlusions" , that are the dynamic ones and the occlusions due to "thin scene structures" and "large structures". The first causes group of people, the second a temporary split and the third the object disappearing. Some heuristic rules are proposed to cope with group, split and temporary disappearing. Also the recent work of Wu and Nevatia [103] aims at coping with both inter-object and scene occlusion. In this case there is no occlusion classification but human tracking is done by parts. A greedy correspondence algorithm is used whenever possible, i.e. when the body parts are visible; otherwise a mean-shift tracker [91] is adopted that allows a robust tracking also if parts of the body are not visible.

5.3 The Tracking Algorthm

Even if *Ad Hoc* tracking will work at a pixel level, the central element in the system is the object O, which is described by its state vector $O = \{\{o_1, \ldots, o_N\}, \vec{c}, \vec{e}, \Pi\}$, where $\{o_i\}$ is the set of the N points which constitute the object O, $\vec{c}$ and $\vec{e}$ are respectively the position with respect to the image coordinate system and the velocity of the centroid, Π is the probability of being the foremost object, thus the *probability of non-occlusion*. Each point o_i of the object is characterized by its position (x, y) with respect to the object centroid, by its color (R, G, B) and by its likelihood α to belong to the object.

The scene at each frame t is described by a set of objects $\mathcal{O}^t = \{O_1, \dots, O_M\}$ which we suppose are generating the foreground image $F^t = \{f_1, \dots, f_L\}$, i.e. the points extracted by any segmentation technique. Each point f_i of the foreground is characterized by its position (x, y) with respect to the image coordinate system and by its color (R, G, B). The tracking aim is to estimate the set of objects $\mathcal{O}^{t+1}$ observed in the scene at time/frame $t + 1$, based on the foregrounds extracted up to now. In a probabilistic framework, this is obtained by maximizing the following probability:

$$P(\mathcal{O}^{t+1}|F^{0:t+1}), \tag{5.2}$$

where the notation $F^{0:t+1} \doteq F^0, \dots, F^{t+1}$. In order to perform this MAP (maximum a posteriori) estimation, we make the assumption of having a first order Markovian model, meaning that

$$P(\mathcal{O}^{t+1}|F^{0:t+1}) = P(\mathcal{O}^{t+1}|F^{t+1}, \mathcal{O}^t). \tag{5.3}$$

Moreover, by using the Bayes theorem, it is possible to write

$$P(\mathcal{O}^{t+1}|F^{t+1}, \mathcal{O}^t) \propto P(F^{t+1}|\mathcal{O}^{t+1})P(\mathcal{O}^{t+1}|\mathcal{O}^t)P(\mathcal{O}^t). \tag{5.4}$$

Optimizing Eq. 5.4 in an analytic way is not possible, so this would require to test all the possible objects sets, by changing their positions, appearances, and probabilities of non-occlusion. This is definitely unfeasible, so we break the optimization process in two steps, by locally optimizing the position, then updating the appearance and the probability of non-occlusion.

5.4 Position optimization

The first task of the algorithm is the optimization of the centroid position for all objects. In Eq. 5.4 the term $P(\mathcal{O}^t)$ may be set to 1, since we just keep the best solution from the previous frame. The term $P(\mathcal{O}^{t+1}|\mathcal{O}^t)$ that is the motion model, is provided by a circular search area of radius r around the estimated position $\hat{c}$ of every object. This is obtained with a first order linear model, that is computed from the previous center and velocity:

$$\hat{c}^{t+1} = \vec{c}^{\,t} + \vec{e}^{\,t} \tag{5.5}$$

$$P(\mathcal{O}^{t+1}|\mathcal{O}^t) = \begin{cases} \frac{1}{\pi r^2} & \text{if } \left\|\vec{c}^{\,t+1} - \hat{c}^{t+1}\right\| < r \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

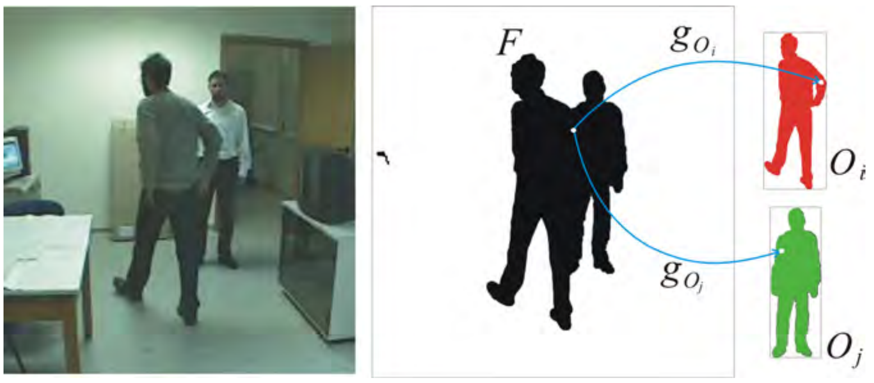


Figure 5.1: Domain and Codomain of the function g_O. g_O transforms the coordinates of a foreground pixel $x \in F$ to the correspondent object coordinates.

In order to measure the likelihood of the foreground to be generated by an object, we need a way to match their point positions, that is defining a relation among the elements of F and O. To this aim we define a function g_O as:

$$\begin{aligned} &g_O : F \to O \\ &g_O(f) = o \in O \mid \vec{x}(f) = \vec{x}(o) + \vec{c}_O. \end{aligned} \tag{5.7}$$

The function $\vec{x}(\cdot)$ gives the coordinates vector (x, y) of the point.

Directly from the definition of the function g_O we can obtain its domain $\tilde{F}_O$ that is the set of foreground points matching object's points. We may then define $\tilde{F} = \bigcup\limits_{O \in \mathcal{O}} \tilde{F}_O$, that is the set of foreground's points which match at least one object. In the same way we call $\tilde{O}$ the codomain of the function g_O, that includes the points of O which have a correspondence in $\tilde{F}$. (See Fig. 5.4).

Since the objects can be overlapped, a foreground point f can be in correspondence with more than one object O. Thus, we can define the set $\mathcal{O}(f)$ as:

$$\mathcal{O}(f) = \left\{ O \in \mathcal{O} : f \in \tilde{F}_O \right\}. \tag{5.8}$$

The term $P(F^{t+1}|\mathcal{O}^{t+1})$ is given by the likelihood of observing the foreground image given the objects positioning, that can be written as:

$$P(F^{t+1}|\mathcal{O}^{t+1}) = \prod_{f \in \tilde{F}} \left[\sum_{O \in \tilde{\mathcal{O}}(f)} P\left(f | g_O(f)\right) \cdot \Pi_O \right] \tag{5.9}$$

obtained by adding for each foreground pixel f the probability of being generated by the corresponding point $o = g_O(f)$ of every matching object $O \in \mathcal{O}(f)$, multiplied by its non-occlusion probability Π_O.

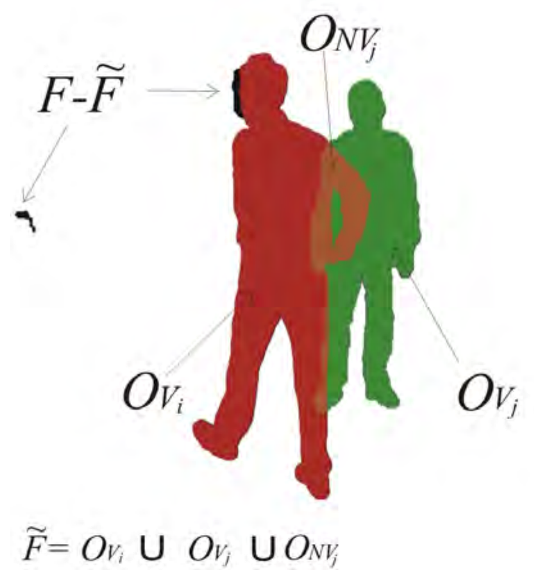


Figure 5.2: Visible and non visible part of an object. $\tilde{F}$ is the foreground part not covered by an object.

The conditional probability of a foreground pixel f, given an object point o is modeled by a Gaussian distribution, centered on the RGB value of the object point:

$$P(f|o) = \frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(\bar{f}-\bar{o})^{\mathrm{T}}\Sigma^{-1}(\bar{f}-\bar{o})} \cdot \alpha(o) \tag{5.10}$$

where $\bar{(\cdot)}$ and $\alpha(\cdot)$ give the RGB color vector and the α component of the point respectively, and $\Sigma = \sigma^2 I_3$ is the covariance matrix in which the three color channels are assumed to be uncorrelated and with fixed variance σ^2. The choice of sigma is related to the amount of noise in the camera. For our experiments we choose $\sigma = 20$.

From a computational point of view, the estimation of the best objects' alignment would require at most $(\pi r^2)^M$ evaluations. It is reasonable to assume that the contribution of the foremost objects in Eq. 5.9 would be predominant, so we locally optimize the function, by considering only the foremost object for every point. The algorithm proceeds as follows:

1) A list with the objects sorted by their probability of non-occlusion (assuming that this is inversely proportional to the depth ordering) is created;

2) The first object O is extracted from the list and its position $\vec{c}$ is estimated by maximizing the probability:

$$P(\tilde{F}|O) \propto \prod_{f \in \tilde{F}_O} P(f|g_O(f)). \tag{5.11}$$

3) After finding the best $\vec{c}$, the matched foreground points are removed and the foreground set F considered in the next step is updated as $F = F \setminus \tilde{F}_O$.

4) The object O is removed from the list as well, and the process continues from 2, until the object list is empty.

The described algorithm may fail for objects which are nearly totally occluded, since a few pixels could force a strong change in the object center positioning. For this reason we introduce a *Confidence* measure for the center estimation, to account for such a situation:

$$\mathit{Conf}(O) = \frac{\sum\limits_{o \in \tilde{O}} \alpha(o)}{\sum\limits_{o \in O} \alpha(o)}. \tag{5.12}$$

If during the tracking the confidence drops under a threshold (set to 0.5 in our experiments), the optimized position is not considered reliable, thus only the prediction is used:

$$\mathbf{c}^{t+1} = \hat{\mathbf{c}}^{t+1} \tag{5.13}$$

5.4.1 Pixel to Track Assignment

This is the second phase of the optimization of Eq. (5.4). Once all the tracks have been aligned, in this top-down approach we aim at adapting the remaining parts of each object state. Even in this case we adopt a sub-optimal optimization. The first assumption we made is that each foreground pixel belongs to only one object. To this aim we perform a bottom-up discriminative pixel to object assignment finding the maximum of the following probability for each point $f \in \tilde{F}$:

$$P(O \to f) \propto P(f|g_O(f)) \cdot P(g_O(f)) = P(f|g_O(f)) \cdot \alpha(g_O(f)), \tag{5.14}$$

where $P(f|g_O(f))$ is the same of (5.10) and we use the symbol $\to$ to indicate that the foreground pixel f is generated by the object O. Directly from the above assignment rule, we can divide the set of object points into visible O_V and non-visible O_{NV} points:

$$\begin{aligned} O_V &= \left\{ o \in O \mid \exists f = g_O^{-1}(o) \wedge \underset{O_i \in \mathcal{O}}{argmax} \left(P(O_i \to f) \right) = O \right\} \\ O_{NV} &= O - O_V \end{aligned} \tag{5.15}$$

In other words, the subset O_V is composed by all the points of O that correspond to a foreground pixel and that have won the pixel assignment. (See Fig. 5.4). The $alpha$ value of each object point is then updated using an exponential formulation:

$$\alpha(o^{t+1}) = \lambda \cdot \alpha(o^t) + (1 - \lambda) \cdot \delta(o, O_V) \tag{5.16}$$

where $\delta(\cdot, \cdot)$ is the membership function:

$$\delta(o, O_V) = \begin{cases} 1 & o \in O_V \\ 0 & o \notin O_V \end{cases} \tag{5.17}$$

The equation (5.16) includes two terms: one proportional to a parameter $\lambda \in [0,1]$ that corresponds to $P(O^{t+1}|O^t)$ and reduces the $alpha$ value at each time step, and one proportional to $1-\lambda$ that increases the α value for the matching visible points $P(F|O)$. Similarly we update the RGB color of each object point:

$$\bar{o}^{t+1} = \lambda \cdot \bar{o}^t + (1-\lambda) \cdot \bar{f} \cdot \delta(o, O_V) \tag{5.18}$$

with $f = g_O^{-1}(o)$.

For computational reasons, if the α value of a point $o \in O$ go below a predefined threshold (set always to $1-\lambda$), the point o is removed from the corresponding object O.

The last step of the object state updating concern the non occlusion probability Π. To this aim we first define the probability Po^{t+1} that on object O_i occludes another object O_j.

$$Po(O_i,O_j)^{t+1} = \begin{cases} 0 & \beta_{ij} < \theta_{occl} \\ (1-\beta_{ij})Po^t_{ij} & a_{ij} = 0 \\ (1-\beta_{ij})Po^t_{ij} + \beta_{ij} e^{\frac{a_{ji}}{a_{ij}}} & a_{ij} \neq 0 \end{cases}, \tag{5.19}$$

where

$$\begin{gathered} a_{ij} = \left\| O_{V,i} \underset{g}{\cap} O_{NV,j} \right\| = \left\| g_{O_i}^{-1}(O_{V,i}) \cap g_{O_j}^{-1}(O_{NV,j}) \right\| \\ \beta_{ij} = \frac{a_{ij}+a_{ji}}{\left\| O_i \underset{g}{\cap} O_j \right\|} \end{gathered} \tag{5.20}$$

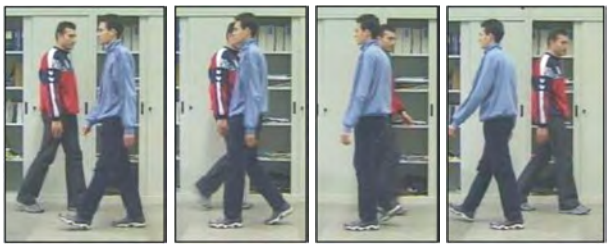
a_{ij} is the number of points shared between O_i and O_j and assigned to O_i; β_{ik} is the percentage of the area shared between O_i and O_j assigned to O_i or O_j, that is less or equal to 1 since some points can be shared among more than two objects.

The value β is used as update coefficient, allowing a faster update when the number of overlapping pixels is high. Vice versa, when the number of those pixel is too low (under a threshold θ_{occl}), we reset the probability value to zero. In the example of Fig. 5.3 the number of assigned pixels and the probabilities of occlusion are depicted for the two objects involved in an occlusion. Note that the probabilities of occlusion update slower at the beginning of the occlusion (i.e., when the number of assigned pixels is low), and then faster. After a certain amount of time, a stable situation is reached, when the foremost object has a high probability of occlusion. The probability of non occlusion for each object can be computed as:

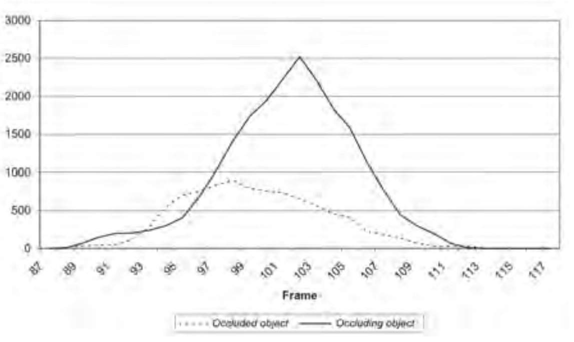
$$\Pi(O_i)^{t+1} = 1 - \max_{O_j \in \mathcal{O}} Po(O_i,O_j)^{t+1}. \tag{5.21}$$

5.4.2 Shape changes and new objects detection

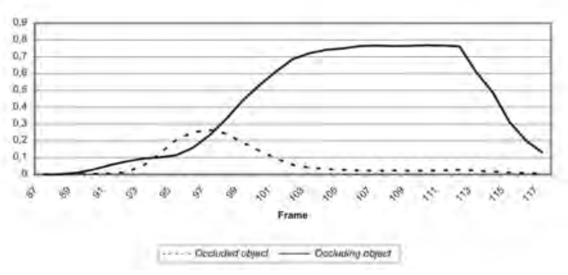
With the probabilistic framework previously described we can ‘assign and track” all the foreground pixels belonging to at least one object. Instead, the foreground



(a) (b) (c) (d)



(e)



(f)

Figure 5.3: (a-d): sample frames; (e) number of assigned point a_{ik} and a_{ki} during the occlusion. (f) Probabilities of occlusion $Po(O_i, O_j)^t$ and $Po(O_j, O_i)^t$.

image contains points $f(\in F - \tilde{F})$ without any corresponding object, due to shape changes or the entrance into the scene of new objects as well. We suppose that a blob of unmatched foreground points is due to a shape change if it is connected (or close to) an object, and in such a situation the considered points are added to the nearest object; otherwise a new object is created. In both cases the α value of each new point is initialized to a predefined constant value (e.g., 0.4). Obviously in this manner we cannot distinguish a new object entering the scene occluded by or connected to a visible object. In such a situation the entire group of connected object will be tracked as a single entity.

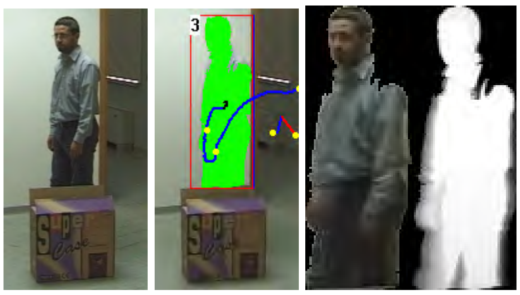


Figure 5.4: An example of loss of part of the model because of an occlusion, if the model does not takes into account occluded pixels and works on assigned pixels only. (a) Input frame; (b) Current segmentation; (c) Appearance Memory model and Probability mask without handling occlusions. The trajectory in (b) changes the direction because of the loss of the memory of the people legs.

5.4.3 Occlusion Detection and Classification

Due to occlusions or shape changes, some points ($o \in O - \tilde{O}$) of an object could have no correspondence with the foreground F due to shape changes or occlusions. Unfortunately, these two reasons require two different and conflicting solutions. To keep a memory of the object shape even during an occlusion the object model need to be slowly updated; at the same time a fast updating can better face shape changes. Other proposed techniques that exploit probabilistic appearance models without coping with occlusions explicitly, use a unique threshold to reach a good trade off between the above mentioned behaviors. An example is shown in Fig.5.4, in which, after a few frames, the track model looses any knowledge about the person's legs. This approach has some drawbacks, such as the shift of the centroid and the erroneous estimation of position and trajectory. In our work, the adaptive update function can be enriched by the knowledge of occlusion regions. In particular, if a point is detected as occluded we freeze the color and α value of the point instead of using Eq. (5.16) and Eq. (5.18).

The introduction of a higher level reasoning is necessary in order to discriminate between occlusions and shape changes. The set of non visible points O_{NV} are the candidate points for occluded regions. After a labeling step over O_{NV}, a set of not visible regions (of connected points) is created; sparse points or too small regions are pruned and a final set of *Non Visible Regions* $\{NVR_j\}$ is created. Each of them can be classified as one of these three classes:

1) *dynamic occlusions* R_{DO}: occlusions due to overlap of another object, closer to the camera; therefore the pixels of this region were assigned to the other object;

2) *scene occlusions* R_{SO}: due to (still) objects, included in the scene and therefore into the background model and thus not extracted by the foreground

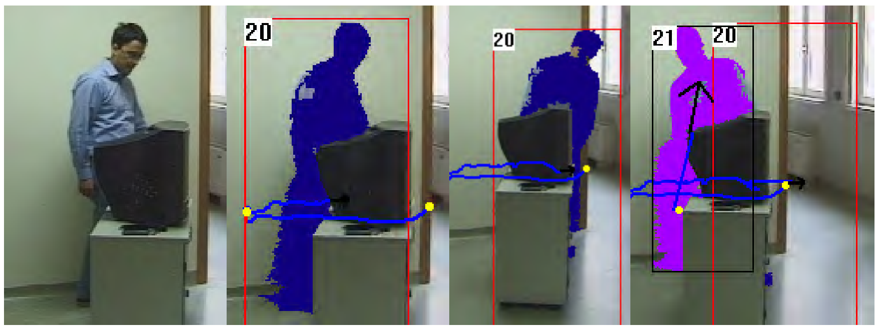


Figure 5.5: Example of erroneous track freezing in case of occlusion. (a) Original image; (b) an occlusion causes the percentage of visible pixels to became too low and the model updated is stopped; (c) the track and VO matching is still maintained; (d) the track 20 is lost and the new track 21 is created because of the change of shape.

segmentation algorithm, but actually positioned closer to the camera;

3) *apparent occlusions* R_{AO}: regions not visible because of shape changes, silhouette's motion, shadows, or self-occlusions.

The presence of an occlusion can be inferred exploiting the *Confidence* value of Eq. 5.12 decreasing below an alerting value, since in case of occlusion the object's shape changes considerably. The occluded points x of the object model ($x \in R_{DO} or x \in R_{SO}$) should not be updated since we do not want to lose the memory of it. Instead, if the *Confidence* decreases due to a sudden shape change (apparent occlusion), not updating the object state would create an error. The solution is a *selective update* according to the region classification.

The detection of the first type of occlusion is straightforward, because we always know the position of the objects, and we can easily detect when two or more of them overlap. R_{DO} regions are composed by the points shared between object O_k and other object O_i but not assigned to O_k. We can mathematically formulate this check as:

$$R_{DO} = \left\{ o \in O_{NV,i} \mid \exists o' \in O_j \wedge g^{-1}_{O_i}(o) = g^{-1}_{O_j}(o') \right\} \tag{5.22}$$

To distinguish between R_{SO} and R_{AO} the position and the shape of the objects in the background can be helpful, but usually they are not provided. However, most of the segmentation algorithms from fixed camera make a background model available. It is computed at each frame in background suppression segmentation techniques, or can be estimated only when needed [5, 82, 94]. Our proposal to discriminate between R_{SO} and R_{AO} exploits the background edges set. This subset of points of the background model contains all points of high color variation, among which the edges of the objects are usually detected. In case of a R_{SO} we would expect to find edge points in correspondence of the boundary between this R_{SO} and the visible part of the track. From the whole set of not visible points O^t_{NV} defined as in Eq. (5.15), we only keep those with a not negligible value of the probability mask in order to get rid of the noise due to motion. The remaining set of points is segmented into connected regions. Then, for each region, the area weighted with the probability values is calculated, and too small regions are pruned. The remaining non visible regions (NVR_j) belonging to an object O, must be discriminated into background object occlusions and apparent occlusions. We call $B(\cdot)$ the set of border points of a region. At the same time, the edges of the background model are computed by a simple edge detector, reinforced by probability density estimation. We could exploit a more robust segmentation technique, e.g. mean shift [91], to extract the border of the objects in the background image, but in our experiments the edge detection has given good results, requiring much less computation. Given the set of edge points $E = \{e_i\}_{i=1..n}$ in the background image a probability density estimate for the background edges can be computed using a kernel $\varphi(\mathbf{x})$ and a window h:

$$p_n\left(\mathbf{x}|\, E\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^2}\varphi\left(\frac{\|\mathbf{x}-\mathbf{e_i}\|}{h}\right) \tag{5.23}$$

The probability density for non-edges $p(\mathbf{x}|\bar{\mathbf{E}})$ can be assumed to be uniform over the same region R. We can then naively compute the a posteriori probability for a pixel $\mathbf{x}$ to be an edge point:

$$P\left(E|\,\mathbf{x}\right) = \frac{P\left(\mathbf{x}|\, E\right)}{P\left(\mathbf{x}|\, E\right) + P\left(\mathbf{x}|\, \bar{E}\right)} \tag{5.24}$$

where we assumed equal *a priori* probability. We can now compute the average a posteriori probability of the set of points $o \in BO_{NV}$ to be generated by the background edges. In particular we are interested on the subset $\tilde{B}(O_{NV}) = B(O_{NV}) \cap B(O_V)$ that is the part of the border of O_{NV} connected to the visible part O_V. The probability estimate allows tolerating a noisy match between BO_{NV} and the edge points. In case this average probability is high enough, meaning that the contour of the occluded region has a good match with the edges, we can infer that another object is hiding a part of the current one, and thus the region is labeled as R_{SO}, otherwise as R_{AO}. In other words, if the visible and the non visible part of an object are separated by an edge, then plausibly we are facing an occlusion between a still object of the scene and the observed moving object. Otherwise, the shape change is more reasonable cause of the no more visible points.

In Fig. 5.6 an example is shown: a person is occluded for a large part by a stack of boxes that are included in the background image. Two parts of its body are not segmented and two candidate occlusion regions are generated (Fig. 5.6(c)): one of them is a shadow included in the object model but now disappeared. In Fig. 5.6(d) the borders of the NVRs are shown, with pixels that have a good match with the edges marked in black. In real occlusion due to a background object the percentage of the points that have a match with respect to the set of bounding points is high;

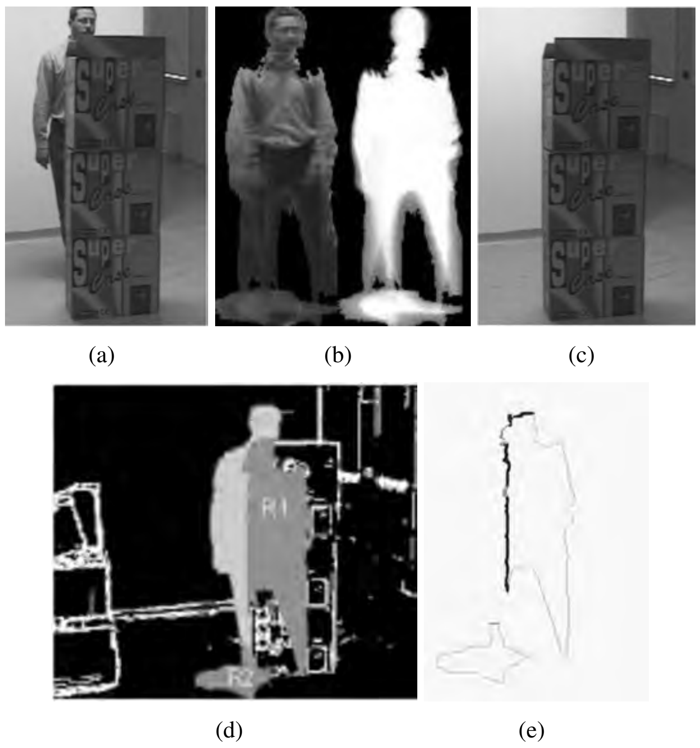


Figure 5.6: Example of R_{SO} regions: (a) The input frame. (b) The color and the α representation of the object points. (c) The current background model (d) The visible part of the track (blue) and the candidate occlusion regions (red).(e) the borders of the Non Visible Regions. Points that have a good match with edge pixels of the background are marked in black. Thus R_1 is classified as a R_{BO} region while R_2 as a R_{AO} region.

thus the region is classified as R_{SO}. On the contrary, for the apparent occlusion (the shadow) we have no matching pixels, and consequently this region is classified as R_{AO}.

In Fig. 5.7 we see another example of apparent occlusion: a person initially standing suddenly falls onto the floor; since his shape considerably changes, a candidate occlusion region appears in correspondence of his legs. The algorithm correctly classifies this region as apparent occlusion, and thus the model is updated.

5.4.4 Refinements

The described model works well in real situations if the initial conditions are ideal, that is if we assume that a single person is entering in the scene at a time, and not occluded by other objects. However, in order to correctly manage all the other conditions, the tracking system has to cope with the well-known problems of merge and split of objects. The same object due to occlusion could initially appear as two different objects: in the example of Fig. 5.8, two of them are associated to the

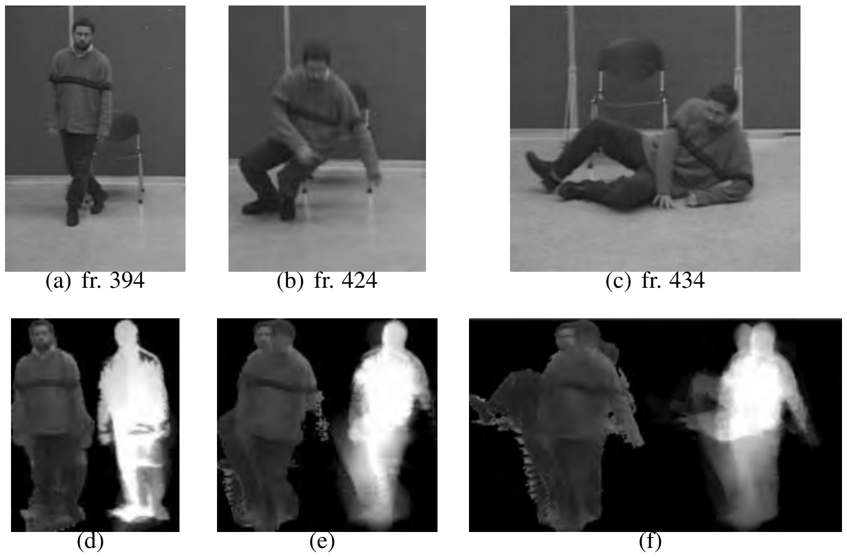


Figure 5.7: (a-c) a person falls on the ground. (d-f) The probability masks show an apparent occlusion in correspondence of the legs.

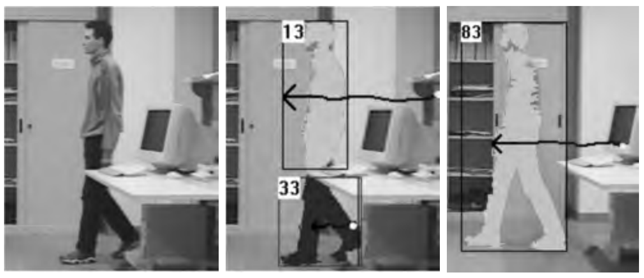


Figure 5.8: Merge. (a) input frame; (b) two objects created; (c) the objects are merged

single person entering the scene, due to the occlusion of the table. In our work, we exploit the motion vectors of the object to detect if a merge is needed. In case the objects are near and have similar motion vectors they are merged together (see Fig. 5.8(c)).

The opposite problem arises when a group of people enters the scene together (see Fig. 5.9). Since they are represented by a single blob, only one object is created. While some authors proposed a splitting technique based on head detection [5,82], we decided to split the objects only in case of group separation. To this aim, the probability mask is periodically analyzed to check the presence of two or more well-separated connected components; in such case, the object is split (Fig. 5.9(c)) and one or more new objects are created. This method is potentially less reactive, since the presence of two or more person is detected only when they are

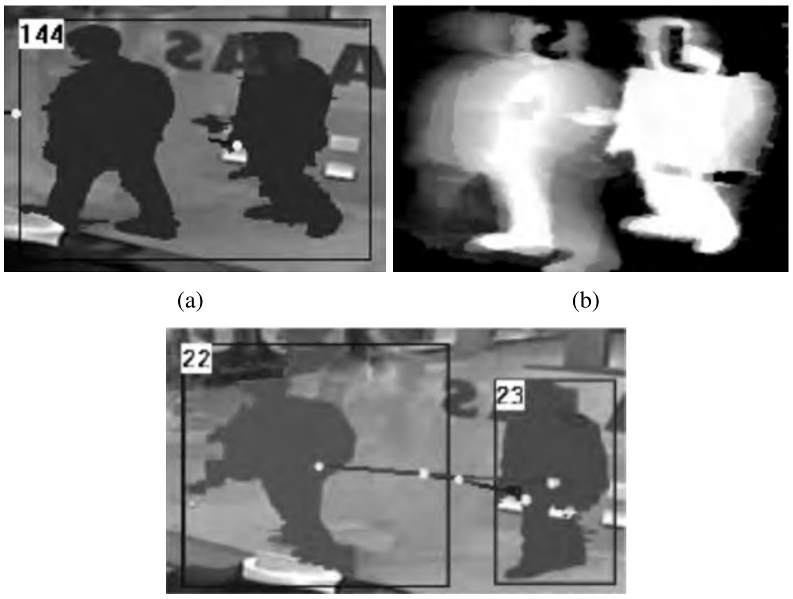


Figure 5.9: Split. (a) a single object for two people separating; (b) the probability mask where two different components are visible; (c) the object is split in two objects

visually separated, but it does not rely on head detection, which is not straightforward in some cases (e.g. a person with an arm higher than the head).

5.5 Experiments

The system has been devised for a project of Indoor Surveillance to control the people behavior at home and detect dangerous situations, as people falling and lying motionless on the floor for a long time. It has been used for video surveillance with a single camera and, in particular, as a basic step for posture detection [87]. In such a situation a frame by frame people behavior control require a complete tracking module with occlusion handling capabilities. In our experiments we exploited a background suppression algorithm called Sakbot (*Statistical Knowledge Based ObjecT Detection*) [1] that detects moving objects, shadows and ghosts (i.e. errors of the background model). SAKBOT exploits selective background update, in order not to update it in the image parts where a visual object is detected. This approach also allows us to segment object that are stopped as VO.

As described in section 5.4.3, the algorithm deal with scene occlusion detection. In the example in Fig. 5.10 a man (with a white t-shirt) is walking behind a table (that is included in the background model) and stopping there. Moreover, he also crosses another person walking in the opposite direction. Fig. 5.10(b) shows the segmented VO. In case of adaptive, but not selective, model update, after a cer-

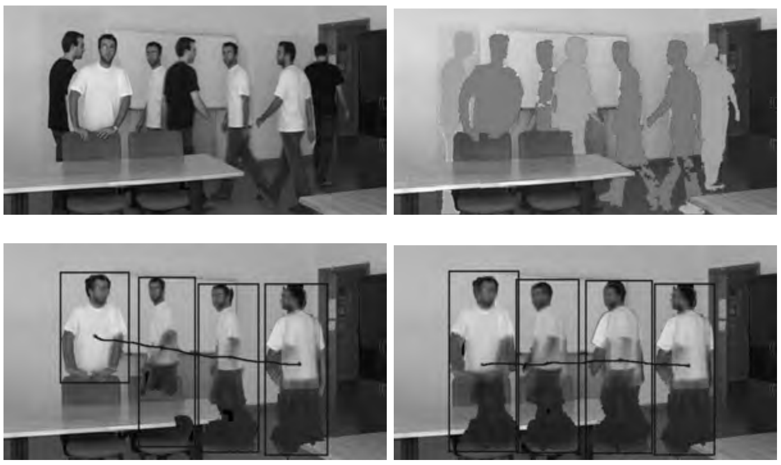


Figure 5.10: Example of background object occlusion. (a) A composition of the input frames. (b) Pixel Assignment. (c) Temporal evolution of the appearance model of one of the tracks in case the background object occlusions are not handled. (d) The same appearance model with the handling of $BOOs$.

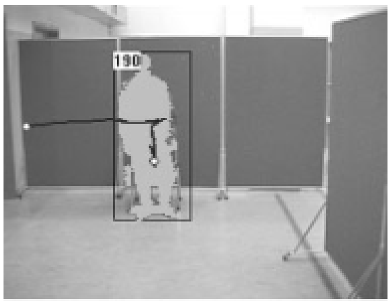
tain amount of time, depending on the coefficient of the update, the memory of his occluded legs is lost, and thus the centroid and the bounding box are not correctly estimated (Fig. 5.10(c)). This is a problem in posture detection since the silhouette of the segmented VO, and all the other features that can be computed, hardly match with a standing-up posture model. The problem is solved with the detection of background object occlusions, where the occluded part of the track is frozen in the probability mask and in the appearance model, leading to a better estimate of track's position. Note that because we don't handle scaling in track alignment, the "frozen" legs in the appearance model appear smaller than their real dimension. Tracking has been extensively tested in a surveillance system mounted in some rooms of the campus and also for distributed surveillance system. As in [104] the appearance memory model and probability mask are very useful to disambiguate and solve consistency labeling problem when a person passes through the field of view of different cameras. In such a context, camera calibration is provided, so that the appearance memory model is warped on the image plane of the new camera. The tracking approach is not too computationally intensive. In our experiment, the indoor video surveillance is able to process about ten frames per second on a standard PC including an initial visual object segmentation module with background suppression and the shadow removal module [1].

Execution times for four benchmark videos are reported in Table 5.1; frames of example for each video are shown in Fig. 5.11. Those videos contain different number of people in the scene, as well as different average percentage of frames occupied by tracks. Video 1 (Fig. 5.11(a)) contains three people walking, sitting and overlapping each other. People shape is often occluded by other people or objects. The size of the tracks is smaller than in the other videos. Video 2 (Fig.

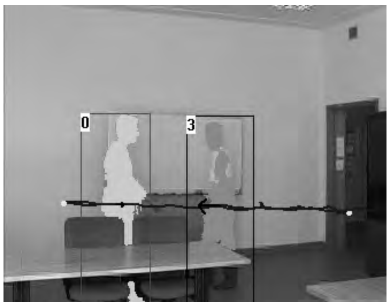
	Video 1	Video 2	Video 3	Video 4
% Average Tracks Area	4%	7%	13%	25%
Number of People	3	1	2	3
1. Sorting by object depth	0.07	0.14	0.14	0.14
2. Position estimation	13.25	23.25	21.10	57.17
3. Pixel to object assignment	0.44	0.27	0.45	1.11
4. Morphological Operations	0.29	0.46	0.49	1.37
5. Occlusion Detection	4.06	8.01	10.18	16.48
6. Object state Updating	2.27	4.11	4.08	10.07
7. Refinements - Merge	0.04	0.04	0.02	0.05
8. Refinements - Split	0.33	0.55	0.75	1.41
TOTAL	20.75	36.83	37.21	87.8

Table 5.1: Execution times in ms for 4 sample videos

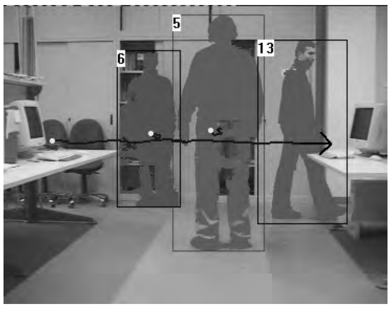
(a) Video 1



(b) Video 2



(c) Video 3



(d) Video 4

Figure 5.11: Sample frames of the videos of Table 5.1

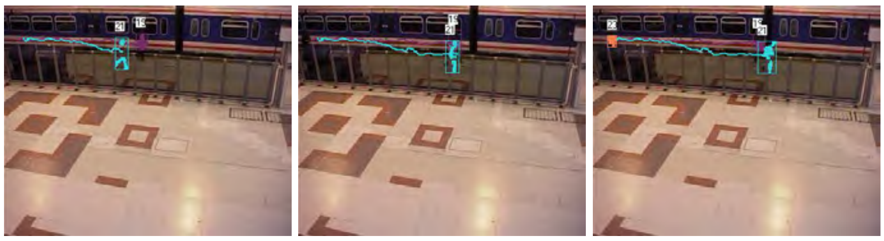

(a) fr 560 (b) fr 580 (c) fr 590

Figure 5.12: Video S1V3; sample output sequence of the tracking system during the crossing of two people behind a plexiglass

5.11(b)) shows a single person that frequently changes the posture (stands up, falls down, lies on the floor, sits). In Video 3 (Fig. 5.11(c)), although two tracks are present, a background object occlusion (a table and two chairs) hides almost half of the tracks for most of the time. One can notice that some of the steps are mainly dependent on the size of the tracks (e.g. merge, split, occlusion detection), while all the other ones depend on the size of the segmented VOs. Consequently, in Video 3 execution times are similar to Video 2, even if the area of the track is almost double. The step 6 (Object state updating) reported at the end includes both visible and non visible pixels respectively updated as in Sec 5.4.1 and in Sec. 5.4.2.

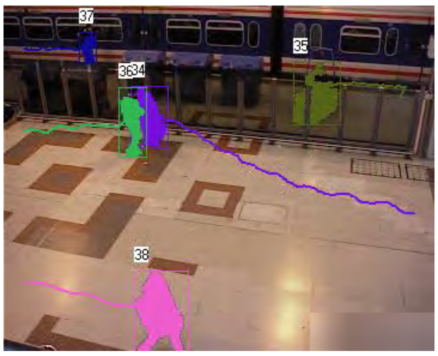
The edge-based method for background object occlusion method is able,on average, to correctly classify the 85% of non visible regions. This approach could be further refined but it is enough precise to allow a good reactivity to silhouette's shape change and, at the same time, a good memory of the appearance memory model also when a person remains occluded by static objects for a long time. Therefore, the proposed tracking module is a general scheme that exploits probability mask and appearance memory model to keep the knowledge of tracked objects even if they are partially hidden. The robustness and the reactivity is based on a selective update process, that manages differently visible pixels, pixels occluded by static or moving regions and pixels that are not visible anymore, due to shape changes self-occlusions or sudden silhouette's motion. This complex but complete process has been tested over days of indoor video surveillance in two rooms equipped with fixed camera, with some actors and indoor furniture. Moreover, it has been tested over the complex videos of PETS 2002 [105], in which people walk and interact behind a shop window.

5.5.1 Results on the PETS 2006 dataset

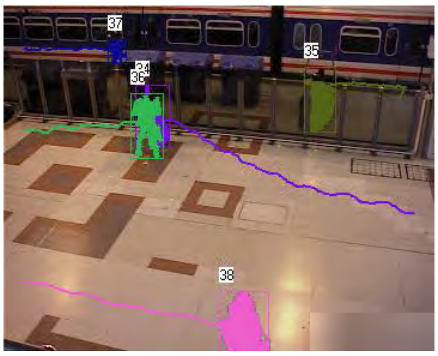
We tested the tracking system on some videos of the PETS2006 dataset. In particular we exploit the seven videos of the third camera, since it's point of view present a slightly changing background. A summary of the obtained results are reported in Table 5.2. In particular we highlight the number of the people walking through the scene and the challenging problems characterizing the videos.

Video (#frames)	**Number of People**	**Solved Issues**	**Errors**
S1V3 (3020)	35	1 left luggage 6 dynamic occlusions 3 groups of people 29 correctly tracked people	1 Identity change 1 split head/feet 2 groups are not correctly split
S2V3 (2550)	31	1 left luggage 1 dynamic occlusions 2 groups of people 27 correctly tracked people	2 groups are not correctly split Re-initialization required after a strong background change
S3V3 (2370)	19	1 left luggage 4 dynamic occlusions 1 groups of people 16 correctly tracked people	1 Identity change 1 group not correctly split
S4V3 (3050)	16	1 left luggage 4 dynamic occlusions 14 correctly tracked people	1 Identity change for a person changing direction during a complete occlusion 1 split head/feet
S5V3 (3400)	30	1 left luggage 6 dynamic occlusions 2 groups of people 27 correctly tracked people	1 split head/feet 1 group not correctly split
S6V3 (2800)	16	1 left luggage 2 dynamic occlusions 1 groups of people 14 correctly tracked people	1 group splits only at the end
S7V3 (3400)	44	1 left luggage 10 dynamic occlusions 3 groups of people 38 correctly tracked people	1 Identity change 2 split head/feet 1 group not correctly split baggage change ID 3 times due to occlusions

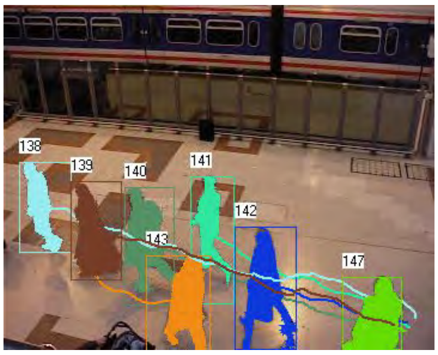
Table 5.2: The PETS 2006 dataset (only the third camera). The number of correctly tracked people and the solved issues are reported. In the last column the errors generated by the tracker are summarized.



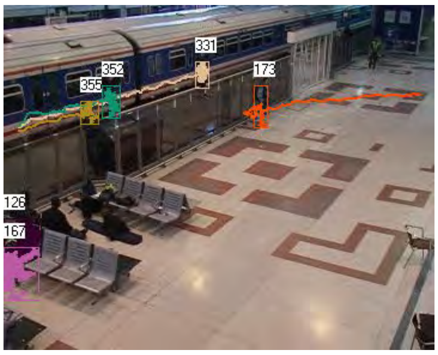
(a) fr 1076



(b) fr 1090

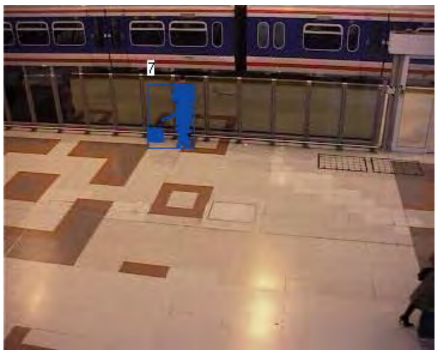


(c) fr 2540

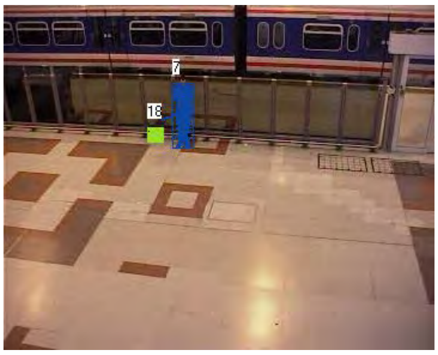


(d) fr 1450

Figure 5.13: Video S2V3 and S2V4; handling of complex situations and interactions

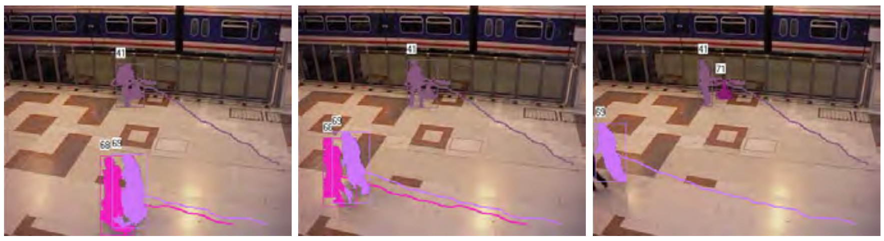


(a) fr 1076



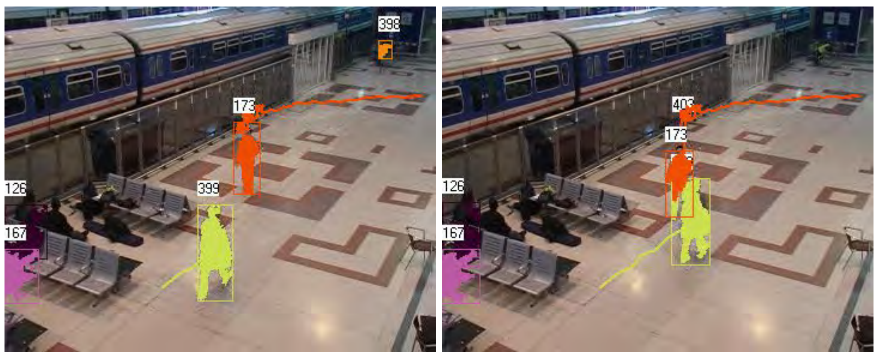
(b) fr 1090

Figure 5.14: Video S3V3; example of object split due to a baggage left



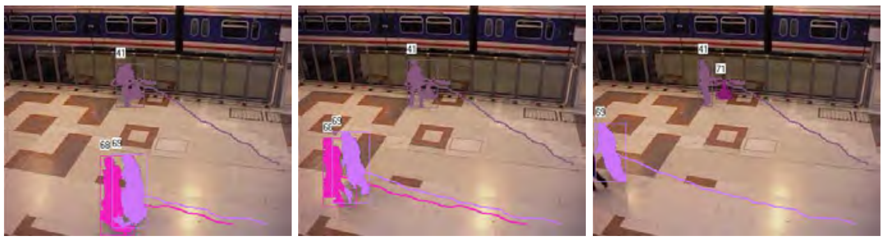
(a) fr 1851 (b) fr 1883 (c) fr 1904

Figure 5.15: Video S1V3; another example of object split due to a baggage left



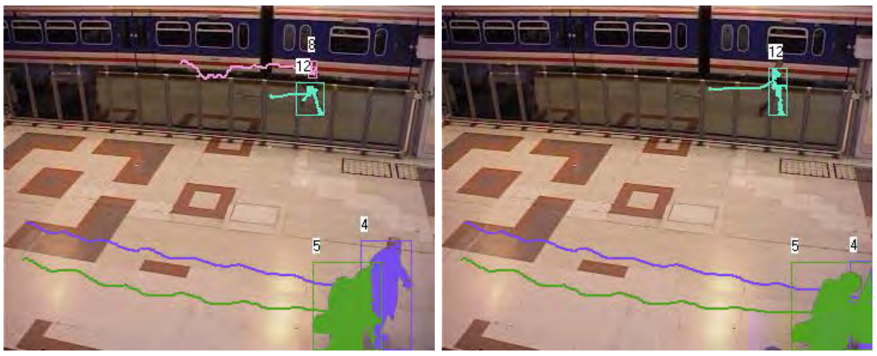
(a) fr 2130 (b) fr 2150

Figure 5.16: Video S1V4; dynamic occlusion correctly managed



(a) fr 1851 (b) fr 1883 (c) fr 1904

Figure 5.17: Video S4V3; object split



(a) fr 325 (b) fr 356

Figure 5.18: Video S1V3; The Merge algorithm used to connect two split parts (torso and feet) of the same person.

In Fig. 5.12 are shown some example frames of the first Video (frames 560, 580, and 590). During this part of the video two people are walking behind the plexiglass, crossing each other 5.16. The labels are correctly assigned and kept. Other example frames showing different kind of interactions between people in the scene are depicted in Fig. 5.13. The aim of the PETS2006 workshop is to use existing systems for the detection of left (i.e. abandoned) luggage in a real-world environment. The luggages are firstly tracked together with their owners; after the leaving, instead, the split algorithm must be invoked in order to assign two different identities to the person and the baggage. Fig. 5.14, Fig. 5.15, and Fig. 5.17 shown the efficacy of our split algorithm over the PETS2006 sequences. Finally, in Fig. 5.18 the merge algorithm has joint together two different parts (feet and torso) of the same person split due to the plexiglass interposed between the person and the camera.

5.6 Conclusions

In this work we defined, developed, and tested the *AD-HOC* tracking, a new approach for multiple people tracking in video surveillance applications. In particular, our effort was focused to overcame large and long-lasting occlusions by using an appearance driven tracking model. Working at a pixel level, it has been possible to define and manage *non visible regions*, i.e. the parts of the objects that are not detected in the current frame, allowing the detection of occlusions. A first effective aspect is the two step algorithm that gives a fast solution to a probabilistic model. The main novelty of the system is a classification of non visible regions into three classes that aims at distinguish between real occlusions (*dynamic occlusions*), occlusions with an object belonging to the background (*scene occlusions*), and shape changes (*apparent occlusions*). Thus, basing on the classification result, a different behavior can be adopted to keep memory of the occluded parts of each object and to recover them once they appear again. The proposed tracking is very robust and fast; it has been adopted in several projects of indoor and outdoor people surveillance, with many people and real operating conditions. Results over the real sequences from PETS 2006 prove the validity of the approach. Since the tracking algorithm is model free, it can be applied to different scenarios, and not only to people.

Part II

People VideoSurveillance

Chapter 6

Human motion capture and behavior analysis

6.1 Introduction

Emerging technologies can offer a very interesting contribution in improving the quality of life of people staying at home or working indoors. Most of these techniques and the related systems are converging in the new discipline of Ambient Intelligence that includes ubiquitous computer systems, intelligent sensor fusion, remote control, tele-healthcare, video surveillance and many other pervasive infrastructure components.

One important goal of these systems is *human behavior analysis*, especially for safety purposes: non invasive techniques, such as those based on processing videos acquired with distributed camera, enable us to extract knowledge about the presence and the behavior of people in a given environment.

Recent research in computer vision on people surveillance jointly with research in efficient remote multimedia access makes feasible a complex framework where people in the home can be monitored in their daily activities in a fully automatic way, therefore in total agreement with privacy policies.

In this context of video surveillance, most of the emphasis is devoted to techniques capable of execution in *real-time* on standard computing platforms and with low cost off-the-shelf cameras. Additionally, in indoor surveillance of people's behavior the techniques must cope with problems of robustness and reliability: for instance, in videos acquired with a fixed camera, the visual appearance of a person is often cluttered and overlapped with home furniture, other people, and so on.

In this chapter, we propose a set of computer vision and motion analysis techniques to detect people and interesting events from the scene, and to classify and recognize them in accordance with a previously defined ontology. For instance, we have defined the event "fallen person" that is recognized whenever an object classified as person changes its posture to "laying" and remains in that posture for a given period.

In particular, people detection is achieved by using the tracking algorithm described in Chapter 5, that provides, frame-by-frame, the list of objects O that can be classified as potential people. For posture classification (main topic of this chpater), the probabilistic classifier we propose exploits both frame-by-frame information of people's silhouettes and past information from the associated tracks in order to overcome overlapping and cluttering problems.

The novelty of the proposed approach is the definition of two steps:

1) a posture classification performed frame-by-frame: this classification exploits simple visual features, i.e. projections of the blob's silhouette (or VO) onto the principal axes, and a machine learning process to create Probabilistic Projection Maps (PPMs) used in a Bayesian classifier.

2) a "temporally integrated" posture classification exploiting tracking information: this is motivated by the concept of "posture state" defined in a *state-transition graph* that takes into account for the classification the reliability of the track and acquired knowledge of the people's average behavior in changing their posture.

This posture classifier is used to detect alarming situations such as a person falling down and laying down for a long time. We will show that the proposed approach is capable of reliably recognizing also postures of people that differ from those used in the training set, provided they are of similar body build, and we will demonstrate that it is quite robust, even in the case of people partially occluded by furniture or other people.

6.2 Related works

There are many reference surveys in the field of human motion capture (HMC) and Human behavior analysis (HBA), for instance, the ones by Cedras and Shah [106], Gavrila [107], Aggarwal and Cai [108], and Moeslund and Granum [109], or more recently, Wang et al. [110].

The basic aim of these models and algorithms is to extract suitable features of the motion and the visual appearance of people (e.g., shape, edges, or texture), in order to classify and recognize their behavior. Many works employ a precise reconstruction of the body model in order to detect the motion of each part of the body. This is normally done for virtual reality and computer graphics application with people moving in structured environments [111].

In surveillance, people are normally not collaborative, they are moving in cluttered scenes interacting each others and with objects (e.g., people carrying packs or sitting on a bench), and the acquisition is usually done with large-FOV cameras (resulting in images acquired with low resolution). In such cases, useful features that can be analyzed are the people's trajectory and the changes in motion status and direction. For this reason, there is a growing research activity in **trajectory**

analysis. For instance, in [112] classification of vehicle trajectory is done in order to extract abnormal trajectories. In [113] Abstract Hidden Markov Models are exploited to recognize special trajectories and monitor special behaviors in indoor areas. In [114] trajectories of people walking in outdoor are represented by graphs, and trajectory comparison is done by means of graph matching. Some works as [33, 115] deal with single interaction between pairs of trajectories and typically refer to very simple interactions (such as the “follow”, or “approach-talk-continue together”) or divergence to typical paths.

All these techniques are employed to classify a given trajectory as being significant (abnormal) or normal and, based on that, describe the people’s behavior. Richer information can be extracted from persons’ appearance, posture and gait. In recognizing behavior based on a person’s shape there are two main approaches. The static one is concerned with spatial data, one frame at a time, and compares pre-stored information (such as templates) with the current image. The goal of static recognition is mainly to recognize various postures, e.g., pointing [116, 117], standing and sitting [118]. The second type of approaches is dynamic recognition where here temporal characteristics of moving target are used to represent its behavior. Typically, simple activities such as walking are used as the test scenarios. Both low and high level information is used. Low-level recognition is based on spatio-temporal data without much processing, for instance, spatio-temporal templates [117, 119] and motion templates [120]. High level recognition are based on pose estimated data and include silhouette matching [121], HMMs [122, 123] and neural networks [124]. In the work of [122] the idea of representing motion data by “movements” (similar to phonemes in speech recognition) is suggested. This enables to compose a complex activity (“word”) out of a simple series of movements (“phonemes”). An HMM is used to classify three different categories: running, walking and skipping. This type of high level symbolic representation is also used in [117] who automatically build a “behavior alphabet” (a behavior is similar to a movement) and model each behavior using an HMM. The alphabet is used to classify different types of actions in a simple virtual reality game and to distinguish between the playing style of different subjects. Another successful application of this symbolic approach is in recognizing signed-language [123].

Recently an increasing number of computer vision projects deal with detection and tracking of human posture as well. An exhaustive review of proposals addressing this field was written by Moeslund and Granum in [109], where about 130 papers are summarized and classified according with several taxonomies.

The posture classification systems proposed in the past can be differentiated by the more or less extensive use of a 2D or 3D model of the human body [109]. In accordance with this, we can classify most of them into two basic approaches to the problem. From one side, some systems (like Pfinder [4] or W^4 [125]) use a direct approach and base the analysis on a detailed human body model: an effective example is the Cardboard Model [126]. In many of these cases, an incremental predict-update method is used, retrieving information from every body part.

Many systems use complex **3D models**, and require special and expensive equipment, such as 3D trinocular systems [127], 3D laser scanners [128], thermal cameras [129], or multiple video cameras to extract 3D voxels [130]. Due to the need for real time performance and low cost systems, we discarded complex and/or 3D expensive solutions. In addition, these are often too constrained to the human body model, resulting in unreliable behaviors in the case of occlusions and perspective distortion, that are very common in cluttered, relatively small, environments like a room.

A second way consists in an indirect approach that, whenever the monitoring of **single body parts** is not necessary, exploits less, but more robust, information about the body. Most of them extract a minimal set of low level features exploited in more or less sophisticated classifiers. One frequent example is the use of neural networks, as in [131, 132]. However, the use of NN presents several drawbacks due to scale dependency and unreliability in the case of occlusions. Another interesting example of this class is the analysis of AC-DCT coefficients in the MPEG compressed domain [133]: this has proven to be also insensitive to illumination changes, but the reported examples only classify different standing postures (with different pointing gestures), while we are interested in classifying very different postures, such as standing up and laying on the floor. Eventually, in [134], a Universal EigenSpace approach is proposed: this presents insensitivity to clothing, but it assumes that most of the possible postures (with most of the possible occlusions) have been learned, and this is far from being realizable.

Another large class of approaches are based on **human silhouette analysis**. The work of Fujiyoshi et. Al. [135] uses a synthetic representation (Star Skeleton) composed by outmost boundary points. A similar approach is proposed in [136] where a skeleton is extracted from the blob by means of morphological operations and then processed using a HMM framework. This approach is very promising and has the unique characteristic of also classifying the motion type, but it is very sensitive to segmentation errors and in particular to occlusions. Moreover, no scaling algorithm to remove perspective distortion is proposed making this approach unfeasible for our target application.

Another approach based on silhouette analysis is reported in [137, 138] where a 2D complex model of the human body is matched with the current silhouette by genetic algorithms. In addition to the problems of segmentation errors and occlusions, this approach also suffers from dependency of the model on the view. In [125], Haritaoglu et al. add to W^4 framework some techniques for human body analysis using only information about the silhouette and its boundary. They first use hierarchical classification in main and secondary postures, processing vertical and horizontal projection histograms from the body's silhouette. Then, they locate body parts on the silhouette boundary's corners.

Our approach is similar to this one, as regards projection features, but, differently from it, is not based on a-priori defined model, but exploits a learning phase to build a probabilistic model of body postures.

6.3 Projection Histograms

The posture classification is based only on the appearance of the person's body and, in particular, on its silhouette. Thus, a reliable blob extraction algorithm must be used to provide this input to the classifier.

Generalzing, we assume to have a lower level module able to provide us, for each frame t, with a list $\mathcal{O}^t = \{O_i^t\}$ of Objects (O_i), pre-classified as people.

Each person/object is associated to a blob mask:

$$B = \{b(x, y) \in \{0, 1\}, x \in [0, B_x - 1],\ y \in [0, B_y - 1]\} \tag{6.1}$$

where B_x and B_y are the bounding box sizes, and $b(x, y) = 1$ if the pixel in the bounding box of (x, y) coordinates belongs to the person's blob and 0 otherwise.

The extracted objects are processed by a tracking module that must ensure the maintenance of the tracks also in the case of occlusions due to static or moving objects (e.g., furniture or other moving people). The information extracted by the tracking module can be exploited also by the posture classifier, as we will detail in the following. In the matter of fact, there is no reliable frame-by-frame classifier based on single-perspective images able to deal with occlusions, since it can not classify something that is not visible (such as a person behind another person).

In accordance with the blob definition reported in Eq. 6.1 and similarly to [125], we can define the vertical and horizontal projection histograms (or projections), respectively θ and π, as:

$$\theta(x) = \sum_{y=0}^{B_y} b(x, y) \quad ; \quad \pi(y) = \sum_{x=0}^{B_x} b(x, y) \tag{6.2}$$

In practice, θ and π are two feature vectors associated to the blob B. A value (or bin) of θ (π) at the position $\overline{x}$ ($\overline{y}$) represents the thickness of the silhouette in the vertical (horizontal) direction. Therefore, a blob B has associated a measure $Ph_B \triangleq (\theta_B, \pi_B)$.

6.4 Frame by Frame Classification

According with the literature, we define four main postures:

$$\Gamma_M = \{STanding, CRouching, SItting, LAying\} \tag{6.3}$$

Since our approach is based on the histograms obtained by projecting the silhouette onto x and y axis, and since the silhouette of people sitting with a frontal, left or right view are very different, we have split each state into three view-based subclasses (frontal, left-headed, and right-headed), thus obtaining twelve view-based postures:

$$\begin{aligned}\Gamma_{VB} = \{&ST_F,\ ST_L,\ ST_R,\ CR_F,\ CR_L,\ CR_R,\\ &SI_F,\ SI_L,\ SI_R,\ LA_F,\ LA_L,\ LA_R\}\end{aligned} \tag{6.4}$$

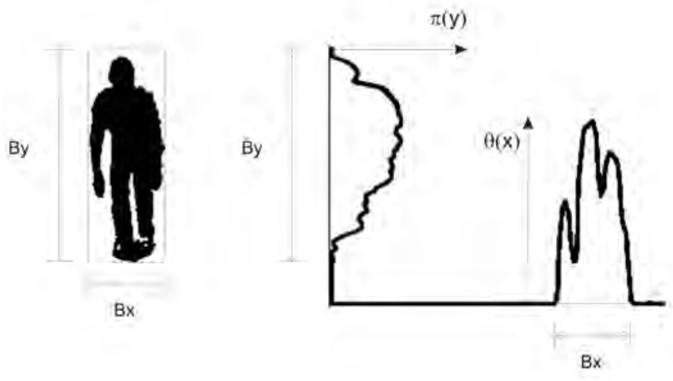


Figure 6.1: Example of projection histograms.

The approach we will describe in the following is applied by the system over the twelve Γ_{VB}. Nevertheless, we will refer in the following to the main postures only (calling them generally postures) for the sake of simplicity.

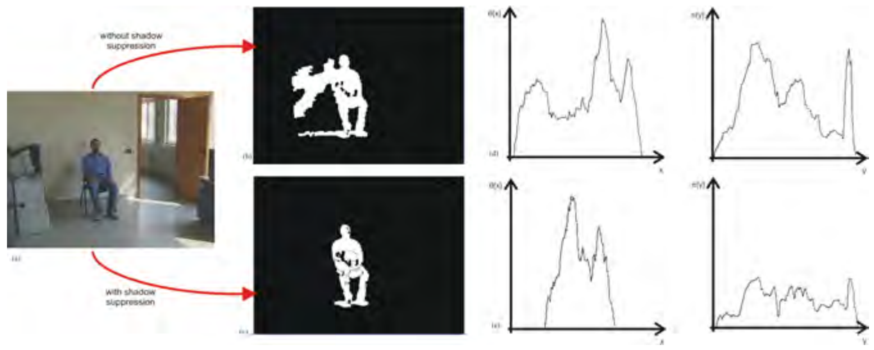


Figure 6.2: Comparison of the projection histograms obtained by preserving or removing shadows

This classifier exploits the intrinsic characteristic of the silhouette to recall the person's posture and it is based on *projection histograms* that describe how the silhouette's shape is projected on the x ad y axes. An example of projection histogram is depicted in figure 6.5(a). Moreover, in Fig. 6.2(d) and 6.2(e) the projection histograms obtained by including shadows or removing them are shown: as it has been already stated, shadows are particularly problematic for projection histograms based on blob's silhouette, and thus they must be effectively removed.

Though projection histograms are an approximation w.r.t. a complex 3D model, they have proven to be sufficiently detailed to discriminate between the postures we are interested in. However, these descriptors suffer two limitations:

- they depend on the silhouette's size;

- they are too sensitive to the unavoidable non-rigid movements of the human body.

The first drawback can be overcome with an initial scaling correction, while the second needs to compare the projection histograms to a suitable model, capable of generalizing the peculiarities of a training set of postures.

The following subsections will report the proposed solutions to these two problems.

6.5 Scaling Procedure

Due to the perspective, the size of the detected VO can differ depending on the distance from the camera. This is particularly true in the case of indoor environments in which the objects moving in the scene are relatively close to the camera. To get rid of this problem we introduced a scaling procedure.

A similar problem has been addressed in other works. For example, in [125] a normalization factor has been used to scale the histograms, but it does not take into account the actual distance of the blob from the camera and it is thus very sensitive to different postures. In [132], the normalization phase is mandatory to feed similar inputs to the neural network, but it has the drawback of considering all the persons as standing. Eventually, the authors of [133] developed a resizing procedure that scales all the images to the same size: this is because they are more interested in detecting where the person points than in classifying his/her posture. Moreover, this example does not take into account the actual 3D position of the person.

Thus, to be reliable, the scaling must be correlated with the distance of the objects from the camera in the 3D space. Assuming that the camera is fixed, we exploit camera calibration to compute the distance d between the object and the camera. Choosing a suitable normalization distance D, we define the rate s_f as a scale factor

$$s_f = d/D \tag{6.5}$$

Applying the scale factor s_f to the blobs is analogous to "moving" all the detected objects to a fixed distance D from the camera. The d measure depends on the position of the support point SP, that is the contact point of the object with the $Z = 0$ ground plane. SP can be described by the (x_{SP}, y_{SP}) coordinates in the image plane and the (X_{SP}, Y_{SP}, Z_{SP}) coordinates in the real 3D world, with $Z_{SP} = 0$. In the case of a person, normally SP corresponds with the position of the feet, but this is not necessarily true in the case of a person laying down on the floor.

Assuming that the camera's point of view is frontal, SP could be easily computed as the point with the maximum y coordinate (see Fig. 6.3). If more points present the same y-coordinate, SP could be randomly selected or computed as the middle point. Once we obtained the SP image coordinates, we can compute the

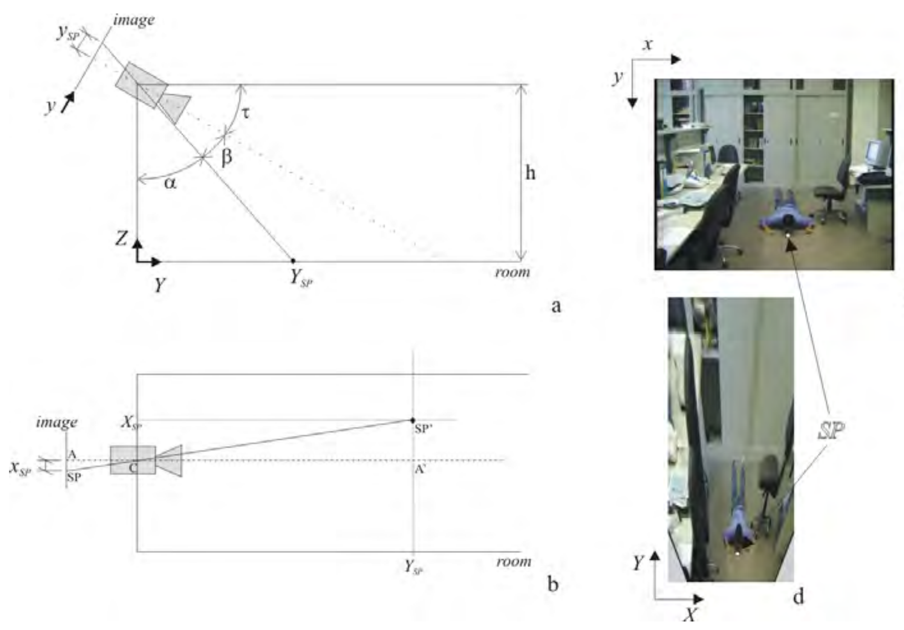


Figure 6.3: Model for perspective effect removal. (a) lateral view and (b) upper view of the pin-hole camera model. Figure (c) reports the original frame, whereas (d) shows the undistorted image obtained through an homography on plane $Z = 0$.

world coordinates of this point by using the projection equations of the pin-hole camera model.

Having the height h of the camera with respect to the floor, the focal lengths f_x and f_y, and the tilt angle τ obtained by camera calibration, we can compute the coordinates of the support point SP:

$$Y_{SP} = h \cdot tg(\alpha), \quad X_{SP} = \frac{x_{SP}}{f_x} \cdot Y_{SP} \tag{6.6}$$

with:

$$\beta = \arctan\left(\frac{y_{SP}}{f_y}\right), \quad \alpha = \frac{\pi}{2} - \tau - \beta, \quad d = \sqrt{X_{SP}^2 + Y_{SP}^2} \tag{6.7}$$

After the scale factor, we assume that the sizes B_x and B_y of the blob's bounding box have been normalized. Obviously, this does not mean that all the silhouettes should have the same sizes (as in the case of a person in the standing or crouching position).

6.6 Support Point Tracking

The correct estimation of SP (support point) position also during an occlusion is a key requirement for correct people scaling. The *SP* is normally coincident with the feet's position, but it could differ when the person is lying down on the floor. If the camera has no roll angle and the pan angle is low enough, the support point can be estimated taking the maximum y coordinate and the average of the x coordinates of the lowest part of the blob (Fig. 6.3).

This simple algorithm fails in presence of occlusions: if the occlusion includes the feet of the person, in fact, the y coordinate of the support point becomes not valid. The support point can not be computed neither on the blob nor on the probability mask: the first because the blob is incomplete and the second because the computation is imprecise and unreliable, especially in body parts with frequent motion such as the legs. To solve this problem, a dedicated tracking algorithm for the support point coordinates has been developed. Two independent constant-velocity Kalman filters are adopted, the first is a standard Kalman filter on the x coordinate, while the second Kalman filter for the y coordinate of SP is used in two modalities: when the person is visible the filter considers both the forecast and the measure (as usual), while when the person is occluded the measure of the y coordinate is ignored and the filter exploits only the forecast to estimate the current position.

The two filters have the same parameters and exploit the constant velocity assumption. Using the well-known notation of the discrete Kalman filter:

$$\begin{aligned} x(k+1) &= \mathbf{\Phi} \cdot x(k) + v \\ z(k) &= \mathbf{H} \cdot x(k) + w \end{aligned} \tag{6.8}$$

The adopted matrices are:

$$\begin{array}{lll} x(k) = \begin{pmatrix} pos_k \\ vel_k \end{pmatrix} & \mathbf{H} = \begin{pmatrix} 1 & -1 \\ 1 & 0 \end{pmatrix} & \mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & \gamma \end{pmatrix} \\ z(k) = \begin{pmatrix} pos_{k-1} \\ pos_k \end{pmatrix} & \mathbf{\Phi} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} & \mathbf{R} = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \end{array} \tag{6.9}$$

The Measurement Noise Covariance Matrix **R** of the Gaussian variable w is computed assuming that the two measured positions could be affected by noise that is frame-by-frame independent and time-constant, while the Process Noise Covariance **Q** of the variable v assumes that the process noise affects only the velocity terms. In our implementation, the parameters are set to λ = 200 and γ = 0.5.

The results obtained by the introduction of these two independent filters during a strong occlusion are visible in 6.4.

Let the *SP* position and the homography of the floor plane to be known, the distance between the person and the camera can be computed. Supposing that the image of the person entirely lies on a plane parallel to the camera, the blob and the tracking images can be projected into a metric space by a linear scaling procedure.

6.7 Probabilistic Projection Maps (PPMs)

The second problem of the classification based on projection histograms is that it is too sensitive to non-rigid movements of the human body model. This problem

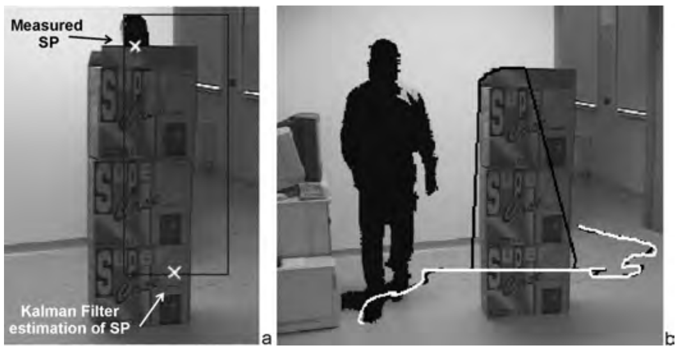


Figure 6.4: SP tracking with Kalman Filters. Trajectories of SP estimated over the blob (black color) and tracked with the Kalman Filter (white color)

has been addressed by constructing the Projection Probabilistic Maps (PPMs) with a supervised machine learning phase [139]. By means of manual annotation of the videos, the learning phase is able to build a model that represents the *memory* of the people's appearance in each posture.

The probabilistic approach included in the PPMs allows us to filter the distracting moving parts of the body (such as the arms and the legs, not important for posture classification), thus solving the above reported problem. In fact, due to the non-rigid motion of the human body, these parts are likely to distract the classifier, resulting in misclassification of the posture. It must be pointed out (and it will be clearer further on) that this probabilistic approach based on a learning phase shifts the problems to the completeness of the training set. We will demonstrate that, as soon as the training set contains a wide enough variety of samples, this approach achieves an average performance of 95% or above in correct classification.

Given a generic set of classes of postures $C = \{C_i\}, \quad i = 1, ..., K$, the probability to belong to the class C_i is:

$$P(C_B = C_i|Ph_B) = \frac{P(Ph_B|C_B = C_i)p(C_i)}{\sum_{j=1}^{K} P(Ph_B|C_B = C_j)p(C_j)} \tag{6.10}$$

The a-priori probabilities $p(C_i)$ can be estimated with respect to the habits of the observed person and the type of the supervised room (e.g., in a kitchen, people stay more often standing or sitting than laying down, whereas, in the case of a bedroom, it is more likely vice versa), or can be dynamically modified in accordance with the history of the blob B. For simplicity, we now assume $p(C_i)$ equal for all the classes C_i and independent from the blob B, that is:

$$p(C_i) = \frac{1}{K}, \quad i = 1, ..., K \tag{6.11}$$

The conditional probability of having the histograms Ph_B assumed to be in the posture class C_i can be computed with the hypothesis that the θ and π measures are independent. Thus:

$$\begin{aligned} P(Ph_B|C_B = C_i) &= P\left(\theta_B \wedge \pi_B|C_B = C_i\right) = \\ &= P(\theta_B|C_B = C_i) \cdot P(\pi_B|C_B = C_i) \end{aligned} \tag{6.12}$$

If we assume the $p(C_i)$ as a constant and neglecting the normalization factor, the Eq. 6.10 is the *similarity function* that describes the similarity between the current silhouette and the model of the silhouette in the posture C_i.

In order to calculate Eq. 6.12, the similarity between each projection histogram and the correspondent model must be computed. In a previous work [140], we tested the efficacy of PPMs by computing, in a very simple way, an average distance (arithmetic mean) between the current projection θ_B (or π_B) and the models given by the PPMs.

According to the probability theory and considering θ and π projection as vector of approximately independent measures, the two terms of Eq. 6.12 can be computed as the probability of intersection of the events:

$$\begin{aligned} P(\theta_B|C_B = C_i) &= P\left(\bigcap_{x=0}^{B_x-1} (\theta_B(x)|C_B = C_i)\right) = \\ &= \prod_{x=0}^{B_x-1} P(\theta_B(x)|C_B = C_i) \\ P(\pi_B|C_B = C_i) &= P\left(\bigcap_{y=0}^{B_y-1} (\pi_B(y)|C_B = C_i)\right) = \\ &= \prod_{y=0}^{B_y-1} P(\pi_B(y)|C_B = C_i) \end{aligned} \tag{6.13}$$

The probability distributions of the events $P(\theta_B(x)|C_B = C_i)$ and $P(\pi_B(y)|C_B = C_i)$ are estimated through a supervised learning phase, during which two 2D functions $\Theta_i(x, y)$ and $\Pi_i(x, y)$ for each class C_i are created as follows:

$$P(\theta(x) = y|C_i) = \Theta_i(x, y) \quad ; \quad P(\pi(y) = x|C_i) = \Pi_i(x, y) \tag{6.14}$$

where $\Theta_i(x, -)$ and $\Pi_i(-, y)$ are the probability distributions of $\theta(x)$ and $\pi(y)$, respectively, assuming to be in the class C_i and we will refer to them as PPMs hereinafter.

The supervised learning phase for the construction of the above mentioned maps is performed exploiting a training set TS of T_i 2D blobs referred to the i-th class $TS = \left\{B_i^t\right\}, \quad t = 1, ..., T_i$, where B_i^t are blob masks defined similarly to Eq. 6.1. For each B_i^t the couple $Ph_i^t = (\theta_i^t(x), \pi_i^t(y))$ of projection histograms is computed as in Eq. 6.2. Then, we construct $\Theta_i(x, y)$ and $\Pi_i(x, y)$ as follows:

$$\Theta_i(x, y) = \frac{1}{T_i} \cdot \sum_{t=1}^{T_i} g\left(\theta_i^t(x), y\right) \quad ; \tag{6.15}$$

$$\Pi_i(x, y) = \frac{1}{T_i} \cdot \sum_{t=1}^{T_i} g\left(x, \pi_i^t(y)\right). \tag{6.16}$$

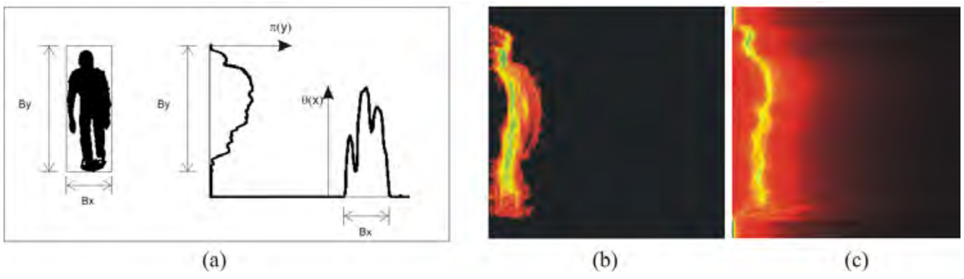


Figure 6.5: Example of PPMs compared with projection histograms (a), sparse PPMs (b), and dense PPMs (c) for $Standing_{frontal}$ posture. Brighter colors correspond to higher probabilities.

Please note that we cannot generate a PPM by either simply averaging each training set contribution, or using a Gaussian distribution, since the measures computed for each sample (i.e., for each class) are multimodal. Thus, the $g(s,t)$ function must take into account all the variations of the samples. A possible $g(s,t)$ function could be:

$$g(s,t) = \begin{cases} 1 & if\ s = t \\ 0 & otherwise \end{cases} . \tag{6.17}$$

This function simply accumulates all the training set information without generalization and it can be acceptable only if we have an almost infinite training set. In fact, also if the current histogram is very similar to those used during the learning phase, the probability (computed as the product of Eq. 6.13) could be zero if only one bin has a value which has never occurred during the training and this is due to the sparseness of the PPMs.

Consequently, the function $g(s,t)$ should be less “rough”. We adopted the following function:

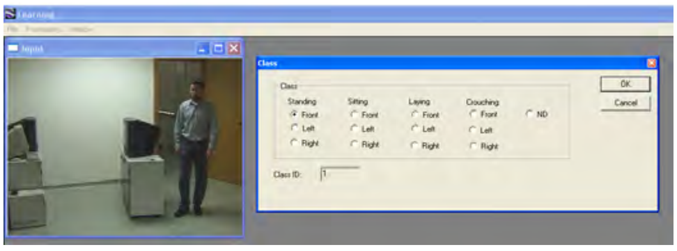
$$g(s,t) = \frac{1}{|s-t|+1} \tag{6.18}$$

The number 1 at the denominator is inserted to avoid dividing by zero. Eventually, the values of the maps obtained through Eq. 6.18 must be normalized to obtain probability distributions. A comparison between the maps created with these two $g(s,t)$ functions is reported in Fig. 6.5(b)-(c).

Once the PPMs are created during the learning phase, at the testing stage the projection histograms obtained by each blob B are compared as described with the PPM of each class and the resulting posture for the blob B is the one that maximizes the conditional probability reported in eq. 6.10, i.e.:

$$posture_B = \arg \max_{i=1,\dots,K} P(C_B = C_i | Ph_B) \tag{6.19}$$

Eventually, Fig. 6.6 shows a snapshot of the environment we used for training. Given a training video shot with a single person, we provide a simple manual annotation of the posture of the actor.



Class ID: 1

Figure 6.6: *Snapshot of the training procedure of our system.*

6.8 Track-based Classification

The previous section reported the frame by frame classifier. We apply it over a large test set of videos and the achieved results show a good robustness of the approach and a very high correct classification rate [139], at least in ideal situations when the blobs are extracted perfectly. However, by exploiting knowledge embedded in the tracking phase, also many possible classification errors due to the imprecision of the blob extraction can be corrected. These errors can arise when:

- there are frequent transitions between a posture and another: in these cases the classifier's reliability decreases;
- there are occlusions: in these cases the person's silhouette cannot be entirely viewed and the projection histograms become less reliable;
- the illumination conditions change: blob extraction based on background suppression is severely affected by this problem, but this can be partially solved by exploiting the tracking.

The former two cases are unavoidable due to person's behavior and the scene, and can be solved only at a higher level, whereas the latter is mainly due to the lower level tasks. In our system, all these cases are accounted for by defining a *state-transition graph* in which we use two states for each posture, one stable and the other unstable (corresponding the above cases).

The posture state-transition graph is a graph defined as $Posture\_STG = (E, N)$, where the set of nodes N includes both the stable states and the unstable states, respectively S_i and s_i, with $i = 1, ..., K$. E is the set of arcs representing the allowed transitions between two states. Fig. 6.7 reports the $Posture\_STG$ for the four main classes reported in Sec. 6.4. A similar graph can be derived for all the 12 subclasses.

The transitions between states are guided by two inputs:

- $posture_B$, i.e. the output of the frame by frame classifier of eq. 6.19;
- confidence (shortened with $conf$), i.e. a boolean value that gives a measure of the reliability of the frame by frame classification ($high$ means a reliable

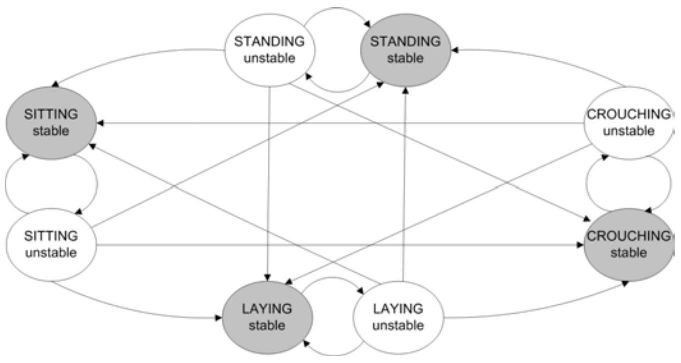


Figure 6.7: The posture state-transition graph for the four main postures.

classification, low unreliable). $conf$ is obtained by thresholding the *confidence* measure of equation 5.12

Thus, the transitions of the STG can be modelled as follows:

$$S_i \rightarrow S_i \Leftrightarrow (conf = high) \wedge (posture = i) \tag{6.20a}$$

$$S_i \rightarrow s_i \Leftrightarrow (conf = low) \vee (posture \neq i) \tag{6.20b}$$

$$s_i \rightarrow S_j \Leftrightarrow (conf = high) \wedge (posture = j) \tag{6.20c}$$

$$s_i \rightarrow s_i \Leftrightarrow (conf = low) \tag{6.20d}$$

S_i is the stable state of the posture i that is maintained whenever the classifier confirms the posture i with a good confidence (Eq. 6.20a). If the confidence decreases, the classification result is no more reliable, but we maintain the posture estimate by moving into the correspondent unstable state s_i (Eq. 6.20b). This can be due to the blob extraction errors or to blob occlusions previously mentioned. Moreover, in order to filter single-frame errors, we prevent the direct transition between two stable states (indeed, the transition $S_i \rightarrow S_j$ is not allowed). Transitions between two stable states must pass through an unstable state, at least for one frame. Obviously, this introduces a short delay in posture change detection. Please note that the loops on the states (Eqs. 6.20a and 6.20d) are not reported in Fig. 6.7.

In this general model, all posture transitions are allowed and with the same probability. Nevertheless, in real cases some transitions are quite unlikely. For instance, a direct transition between "standing" and "laying" posture is quite improbable. To include this concept we could inhibit some transitions between unstable and stable states or we could re-formulate the $p(C_i)$ of equation 6.10. In practice, we could consider $p(C_i)$ as dependent on the current state C_j. As an example, we could assume a fixed transition probability table as the one reported in Fig. 6.8.

This table can be the result of a study of people's average behaviors. It can obviously be improved by exploiting the training phase to fill in these tables.

	Current state C_j			
	ST	CR	SI	LA
$p(ST\|C_j)$	0.30	0.25	0.25	0.10
$p(CR\|C_j)$	0.30	0.25	0.25	0.30
$p(SI\|C_j)$	0.30	0.25	0.25	0.30
$p(LA\|C_j)$	0.10	0.25	0.25	0.30

(a) Example of conditional probabilities to go into a state C_i coming from a state C_j

	ST_R	ST_F	ST_L
$p(ST_R\|C_j)$	0.15	0.10	0.05
$p(ST_F\|C_j)$	0.05	0.15	0.10
$p(ST_L\|C_j)$	0.10	0.10	0.10
$p(CR_x\|C_j)$	0.10	0.10	0.10
$p(SI_x\|C_j)$	0.10	0.10	0.10
$p(LA_x\|C_j)$	0.033	0.033	0.033

(b) Zoom in on the main class ST ($x = R, F, L$)

Figure 6.8: An example of $p(C_i)$ dependent on the class.

6.9 HMM Framework for Posture Classifier

Despite the improvements given by the use of appearance mask instead of blobs, a frame-by-frame classification is not reliable enough. The adoption of a STG can slightly refine some spurious errors. However, the temporal coherence of the posture can be exploited to improve the performance: in fact, the person's posture changes slowly and through a transition phase during which its similarity with the stored templates decreases. To this aim, a Hidden Markov Models formulation has been adopted. Using the notation proposed by Rabiner in his tutorial [141], we define the followings sets:

- The state set **S**, composed by four states:

$$\mathbf{S} = \{S_i, i = 1..N\} = \{S_1, S_2, S_3, S_4\} = MainPostures \tag{6.21}$$

- The initial state probabilities $\Pi = \{\pi_i\}$: the initial probabilities are set to have the same value for each state ($\pi_i = \frac{1}{N}$. By introducing the hypothesis that a person enters a scene standing, it is possible to set the vector Π with all the elements equal to 0 except for the element corresponding to the standing state (set to the value 1). However, the choice of the values assigned to the vector Π affects the classification of the first frames only, and then it is not relevant.

- The matrix **A** of the state transition probabilities, computed as a function of a reactivity parameter α. The probabilities to remain in the same state and to pass to another state are considered equal for each posture. Then, the matrix **A** has the following structure:

$$\mathbf{A} = \mathbf{A}(\alpha) = \{A_{ij}\}\,,\ A_{ij} = \begin{cases} \alpha & i = j \\ \frac{1-\alpha}{N-1} & i \neq j \end{cases} \tag{6.22}$$

In our system we use $\alpha = 0.9$.

Then, the Observation Symbols and Observation Symbol Probability distribution **B** have to be defined. The set of possible projection histograms is used as set of observation symbols.

Even if the observation set is numerable, the matrix **B** is not computed explicitly, but the values b_j for each state (posture) j are estimated frame-by-frame using the above defined Projection Probabilistic Maps:

$$b_j = P_j = P(\widehat{PH}\Big| posture = S_j). \qquad (6.23)$$

Then, at each frame, the probability of being in each state is computed with the forward algorithm [141].

At last, the HMM input has been modified to keep into account the visibility status of the person. In fact, if the person is completely occluded, the reliability of the posture must decrease with time. In such a situation, it is preferable to set $b_j = fract1N$ as the input of the HMM. In this manner, the state probability tends to a uniform distribution (that models the increasing uncertainty) with a delay that depends on the previous probabilities: the higher the probability to be in a state S_j, the higher the time required to lose this certainty. To manage the two situations simultaneously and to cope with the intermediate cases, (i.e., partial occlusions), a generalized formulation of the HMM input is defined:

$$b_j = P(\widehat{PH}\,|S_j) \cdot \frac{1}{1+n_{fo}} + \frac{1}{N} \cdot \frac{n_{fo}}{1+n_{fo}} \qquad (6.24)$$

where n_{fo} is the number of frames for which the person is occluded. If n_{fo} is zero (i.e. the person is visible), b_j is computed as in Eq. 6.23, otherwise the higher the value of n_{fo}, the more it tends to a uniform distribution.

6.10 Experimental Results

6.10.1 Frame by Frame classifier and STG

The system has been developed to meet real-time constraints; the goal is to process a sufficient number of frames per second to be reactive and adaptive enough for possible alarms. The classification with the proposed method is not time consuming and the average performance in the tested videos on a standard PC (Pentium 4 - 3 GHz) is about 15 fps, including also the video acquisition, the segmentation, and the tracking steps.

For the benchmark suite, we use a large set of videos acquired in a room (where the camera has been calibrated) in different days and with different people. Table 6.1 reports some examples under different illumination conditions and with different people with different dresses. The benchmark has been selected in order to perform four tests:

1) the testing video is the same as that used for the training: this can be useful in the case of video surveillance applications for home automation in which we could suppose an initial training performed in the specific context and on the specific person;

Training sequence & test sequence of type 1 (384x288)	Test sequence of type 2 (360x270)
Test sequence of type 3 (384x288)	Test sequence of type 4 (360x270)

Table 6.1: Some examples from the benchmark suite.

2) the person used in the training video is the same as the testing videos', but with different clothing and in different conditions; this demonstrates the insensitivity of our system to clothing as in [134]

3) different persons (but with similar body build)[1] are used for the testing videos;

4) same as 3, but including occlusions to complicate the posture classification.

A detailed test of the accuracy of the system accounts for the number of correctly classified postures w.r.t. the total number of frames. Table 6.2 reports the average accuracy (measured as number of postures correctly classified divided by the total number of reliable - with high confidence - ground-truthed postures) for the four types of test above mentioned. Table 6.3 reports the confusion matrix (in percentage) for the four main postures obtained by averaging among the four types of experiments.

In addition, the graph in Fig. 6.9 compares the results (on the video of type 1), with and without the state-transition graph, w.r.t. the ground-truth. The graphs

[1]as is clear, a classification based on the silhouette's shape is affected by the people appearance, thus children and adult people, for instance, require different PPM models. Nevertheless, it is not sensitive to the position of the people in the room due to the calibration-based scaling.

Test Type	Number of frames	Average Accuracy W/o STG	Average Accuracy With STG	Postures St	Postures Si	Postures La	Postures Cr
Type 1	1742	98.50%	99.40%	28%	21%	40%	11%
Type 2	18222	96.90%	99.57%	6%	32%	42%	20%
Type 3	6023	90.40%	92.20%	32%	21%	22%	25%
Type 4	3141	90.80%	97.75%	35%	12%	26%	27%

Table 6.2: Accuracy for each test type.

Classified as	G.T. posture St	G.T. posture Si	G.T. posture La	G.T. posture Cr
St	**94%**	0%	0%	3%
Si	5%	**99%**	0%	3%
La	0%	0%	**100%**	3%
Cr	1%	1%	0%	**92%**

Table 6.3: Confusion matrix (in percentage) averaged over the four test types.

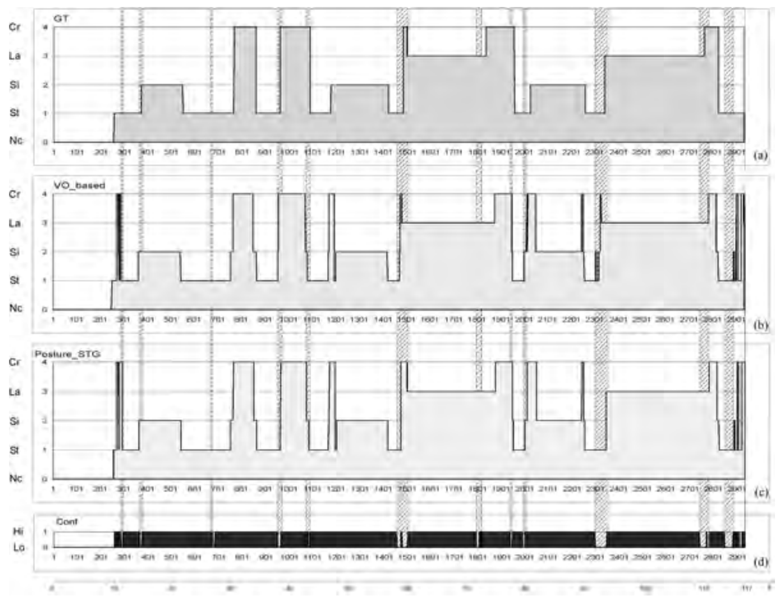


Figure 6.9: Graph reporting the comparison on the video of the test type 1.

show the postures on the ordinates (where Nc stays for “not classified”) and the frame number on the abscissas. The upper graph (a) reports the manual ground-truth classification, while the two successive graphs are the classification obtained by using only the frame by frame classifier (b) or the VO-based classifier with the track-based STG (c). The graph at the bottom (d) reports, frame-by-frame, the confidence value. This value is then used to highlight (with slashed rectangles), in

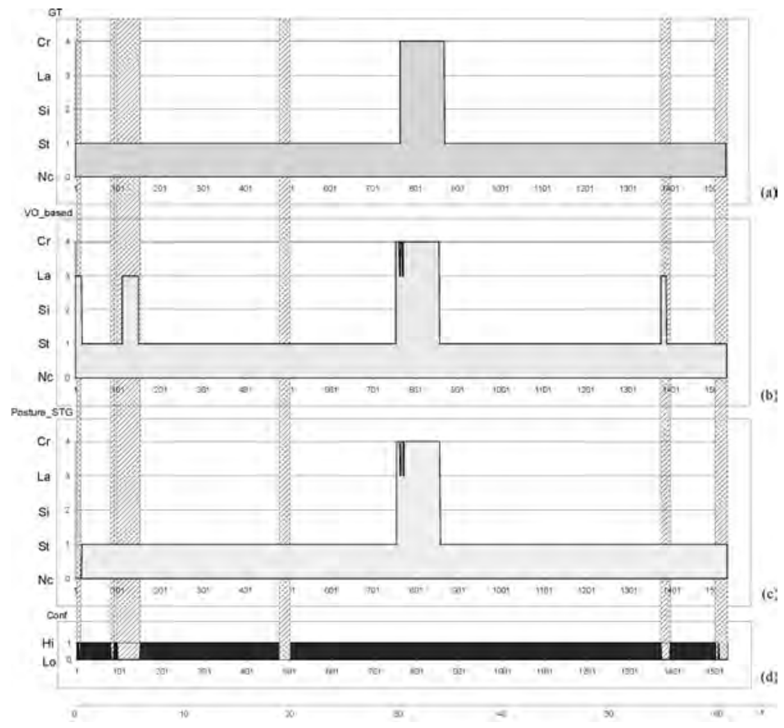


Figure 6.10: Graph reporting the comparison on the video of the test type 4.

the above graphs, the sections of the video in which confidence says that the track has a low reliability.

In our tests, we set the BFR threshold to 0.7%. The BFR threshold depends on the requirements of the system and it can be set as a trade offs between the timeliness and the reliability of the responses.

The graph in Fig. 6.10 compares the results on the video of type 4, with and without the state-transition graph, w.r.t. the ground-truth. In particular, intervals in which confidence is low correspond to object occlusions. In this graph the contribution of the STG is more evident, that, in presence of occlusions, freezes the old state and avoids misclassification.

As it is possible to see from Tables 6.2 and 6.3 and graph on Fig. 6.9, the accuracy achieved is very high and the use of the state-transition graph significantly improves the results, especially in the more challenging case in which occlusion severely affects the performance of the static, frame by frame classifier (see for instance, frame from 2299 to 2348 in Fig. 6.9). As foreseeable, the more critical posture is *crouching* that is often confused with the other postures (see the confusion matrix) and the worst result is obtained in the test of type 3 in which the crouching posture is dominant.

6.10.2 Results of the HMM approach

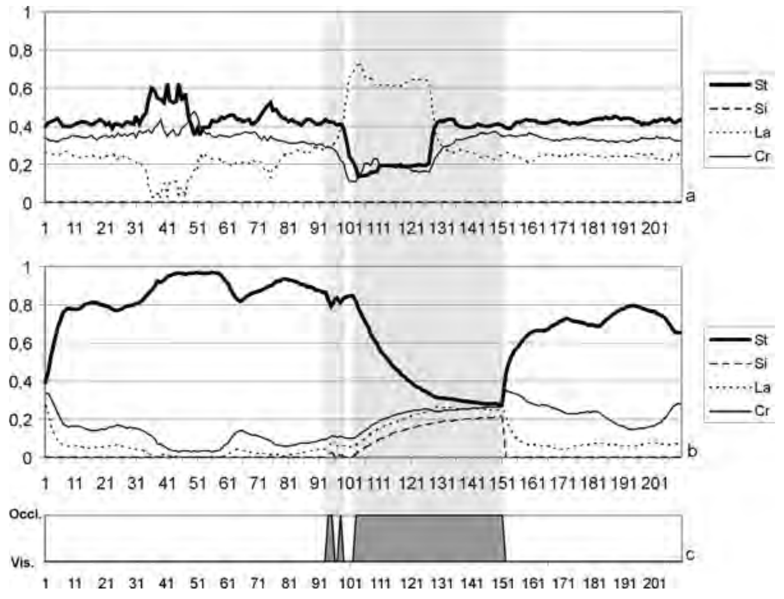


Figure 6.11: Comparison between the frame-by-frame posture classification (a) and the probabilistic classification with HMM (b) in presence of occlusions (c).

In Fig. 6.11 the benefits of the introduction of the HMM framework are evidenced. The results are related to a video sequence in which a person passes behind a stack of boxes always in a standing position. During the occlusion (highlighted by the gray strips) the frame-by-frame classifier fails (it states that the person is lying). On the other hand, through the HMM framework, the temporal coherence of the posture is preserved, even if the classification reliability decreases during the occlusion.

6.11 Discussion and Conclusions

In-house video surveillance has some peculiarities that must be taken into account to develop a reliable and efficient system. We can summarize the distinctive problems to be addressed in in-house video surveillance as in Tab.6.4.

Only by taking into account all these aspects is there very high accuracy performance of the system (in terms of correctly classified postures). We tested the system over more than 2 hours of video (provided with ground-truths), achieving an average accuracy of 97.23%. The misclassified postures were mainly due to confusion between sitting and crouching postures.

PROBLEM	CAUSE	EFFECT	SOLUTION
large shadows	diffuse and close sources of illumination	shape distortion and object merging	shadow detection in the HSV colour space (Section 2.3)
deformable object model	non-rigid human body	shape-based algorithms are misled	probabilistic tracking based on appearance models (Chapter 5)
track-based occlusions	position of the camera; interaction between humans	tracking problems; shape-based algorithms are misled	probabilistic tracking based on appearance models (Chapter 5)
object-based occlusion	presence of many objects		
scale-dependent shapes	position and distance of the camera	size of people depends on their distance from the camera	camera calibration and body model scaling (Section 6.5)
object displacement	non static scene	reference background changes	statistical and knowledge-based background model (Chapter 2)
full coverage of people's movements	non-opened, complex environments	impossibility to cover all the movements with a single view	camera handoff management (Chapter 8)

Table 6.4: Summary of in-house surveillance problems, with their cause and effect, and the solutions we propose

Chapter 7

Face detection

7.1 Video-Surveillance and Biometry

Video-surveillance and Biometry are two known application fields of computer engineering, whose products are playing an important role in the ICT markets. The market of Videosurveillance systems comprises hardware platforms, software and intranet/internet connections for video transport. Just only in this last field, in a recent relation IMS Research had predict a continuous and strong evolution of products of network video over IP. For the next five years they estimate an increasing of the 28,4% for the network cameras and of the 30,3% for the video servers. Network cameras and the video servers will reach sales for about 150 million of Euros in 2008 [142].

Nevertheless, there is still a large gap between the commercial of-the-shelf products and the research activity. The market of videosurveillance products has grown up in an impressive way after the September 11 events, even if most of the market uses the term video-surveillance to name CCTV systems composed by (several) cameras connected with a video server to a control centre, allocating all the role of intelligent control to human observers.

On the other hand, the academic research in Computer Vision-based video surveillance is increasing up to its expectation in the last decade especially due to the increased power of low cost computing systems. They allow the exploitation of complex probabilistic models to detect moving shapes and to understand their behavior, of precise 3D geometrical models to reconstruct the 3D scene using the field of views of more cameras. For this reason, interesting results in moving object segmentation from single or multiple static cameras [1, 13, 94] in detecting the presence of humans, motion trajectory and interaction [143] and their behavior computation have been proposed in indoor and outdoor environments.

Privacy laws are indeed very detailed about the possible use and adoption of video surveillance systems. In Italy, for instance, the “Decalogo della Privacy[1]” avoids the use of videosurveillance systems a part from security purposes man-

[1]Provvedimento Generale del Garante, 29/04/2004)

aged by police. Private videosurveillance systems can be installed only if they guarantee to not identify the single individuals. Therefore, most of these systems are developed for generic control, for dangerous situation detection or for starting the storing of frames when possible interesting situations arise, without the people identification. They can be adopted for security and safety purposes as a possible deterrent to criminality or to store data that will be analyzed possible in case of investigation by human experts. In this case, the research focuses on models and methodologies to prune data for useless parts, to annotate meaningful parts in order to allow a smart indexing and querying in enormous quantities of stored videos.

In another context, instead, biometric systems have been evolved in last years. Biometry is the discipline that aims at assessing the identity of an individuals by means of precise measures of physiological, anthropometric of behavior features. Some examples of these features are fingerprint, hand geometry, iris patterns (Physical Biometrics), handwriting, signature, speech, and gait (Behavioral Biometrics). Biometry is an ancient science: for example, the fingerprint analysis has been defined more than a century ago. It is exploited for dual purposes: *authentication*, that means a 1-to-1 match to verify a positive claim of identity, and *identification* (or recognition) that is instead a 1-to-n match to search in a watching list (or gallery) the presence of an individual whose features have been measured, without claim of identity (or that could be a negative claim of identity, since an individual could try not to be identified).

Biometric systems differ in the type of measured features; among them, the face recognition systems are growing their importance in the market. The total biometrics market this year will reach about $1.2 billion, with face-recognition systems accounting for $144 million, according to projections by the International Biometric Group, a research company in New York. Face-recognition technology is one of the least intrusive biometrics and potentially the most powerful because it can make use of a huge amount of existing data: there are 1.2 billion digitized photos of people in databases around the world. Furthermore, video or image analysis is not invasive; face recognition can be applied to collaborative people (e.g., for authentication or for passing through a controlled gate as in airports) or to non collaborative people (e.g., to compare photos with the archives of police). Moreover it is very natural since it corresponds to the main human approach to recognition, thus automatic systems can be used as an initial screaming that can be validated by human expert analysis.

Some of the earliest research on machine recognition of faces can be traced back to the 1960s at a company called Panoramic Research, Inc. in Palo Alto, California, one of many companies springing up in California at the time to conduct research in what was later termed artificial intelligence.

Although many face recognition systems are available in the world worked and currently used in support for investigation, their results in terms of FAR and FDD are still limited. Nevertheless their adoption is very important for crime prevention. One of the first examples was the *FaceIt* technology, installed in September 1997 as part of an Integrated Passenger and Luggage Security System of Malaysia Airport, which reconciles passengers with their luggage from check-in to boarding to prevents terrorists from checking luggage and not boarding the plane. Most of them work on frontal views only and can be easily falsified by artifacts (e.g., moustaches or glasses) or are sensitive to the age changes and so on. New algorithms are now proposed to people detection and 3D reconstruction and 3D face identification.

In conclusion, currently the state of the art of products of video-surveillance are not concerning people identification. Either because they are interested to the global scene more than to single people, either because, for ethical and privacy issues, they declare not to be able to provide identification. Conversely, biometric systems based on face recognition works normally on still images, possibly in frontal portrait fashion and can be applied automatically in limited cases. Instead, the research activity in computer vision aims at closing this gap and proposing new integrated paradigms to join the capability of most responsiveness and broadband analysis of multicamera video surveillance systems and the abilities of identification of face recognition based biometric systems. The contact point of these two fields are the growing approaches of head and face detection in videos.

Face detection algorithms have been proposed especially for still images. The recent survey of Yang at al. [144] shows a very manifold panorama of techniques, for instance based on Neural Network or statistical classifiers. Other approaches, as the one proposed here, work on videos instead of still images by exploiting the temporal redundancy of visual data. We consider known and consolidated approaches working on a well acquired frontal views of faces. In this case, approaches based on Eigenface [145] or on Fisherfaces [146] are promising. We replicated experiments of both of them finding a possible low recall but a general high precision in identification.

In applications of support of investigation this mean a frequent non-match, but with very few false positive identifications. Therefore, to improve the recall more example of the element to identify should be provided. It is possible if the source of biometry is not a still image but a video. In this case, the technique of face recognition must be juxtaposed to modules of people detection, people tracking, and face detection by exploiting the new proposal in the field of computer-vision based videosurveillance.

There is another reason to explore alternative ways to face recognition in still images (as well as the fact that face recognition in still images is not able to distinguish 2D pictures of faces as in advertisement from true 3D faces of people). The reason is that typical algorithm of face recognition can be used only with a sufficient resolution or, in other words, when the face is acquired at a reasonable size. As a proof of concept we tested one of the best assessed algorithm of face recognition, i.e. the Viola & Jones approach [147]. This method proposes the adoption of the Haar transform to create pattern of interest and AdaBoost classifier to identify pixel pattern that can be considered as "faces".

Table 7.1 shows an example of results with two different webcams (C_1 and C_2) taken at four different resolutions with frontal (F) and non-frontal(NF) (+/-15°) poses. Ten versions of the sixteen situations have been replicated. Table 7.1 shows that in our experiments using the algorithm in OpenCV library [147], when the face is larger than 54x54 pixels the face detection is always corrected.

	TN	FS $\geq$ 54px	48px $\leq$ FS $\leq$ 54px	40px $\leq$ FS $\leq$ 48px	FS $\leq$ 40px
FC_1	1005	N = 585 R = 585 R% = 100%	N = 100 R = 100 R% = 100%	N = 120 R = 72 R% = 60%	N = 200 R = 0 R% = 0%
FC_2	750	N = 416 R = 416 R% = 100%	N = 100 R = 91 R% = 91%	N = 134 R = 57 R% = 42.5%	N = 100 R = 0 R% = 0%
NFC_1	1032	N = 432 R = 432 R% = 100%	N = 99 R = 67 R% = 67.7%	N = 101 R = 29 R% = 28.7%	N = 400 R = 0 R% = 0%
NFC_2	727	N = 360 R = 360 R% = 100%	N = 115 R = 79 R% = 68.7%	N = 114 R = 23 R% = 20.2%	N = 138 R = 0 R% = 0%

Table 7.1: Experimental results using OpenCV face detector on faces of different sizes. Cn: camera n; TN: Total Number of frames of the video; FS: Face Size (the face is considered about FSxFS pixels); Nn: number of frames of the video in range n; Rn: number of correct detections in range n.

The correctness is acceptable with more than 48x48 pixels. For smaller sizes, the correctness decreases. No face detection is possible for faces smaller than 40x40 pixels.

7.2 Head detection

7.2.1 Related works

Face detection is a widely explored research area in computer vision. Two recent surveys ([144] and [148]), collect a large number of proposal about face detection. Most of them are based on a skin color detection [149] followed by a face candidate validation achieved exploiting geometrical and topological constraints. Hsu *et al.* [150], for example, propose a face detection algorithm for color images in presence of varying light conditions, based on a lighting compensation technique and a non linear color transformation. They detect skin regions over the whole image, and then they generate face candidates imposing spatial constraints.

Unfortunately, most of the color-based approaches are very expensive from the computational point of view and it is impossible to perform an accurate face detection at every frame in real time video surveillance applications. To solve this problem, the face detection can be performed only when a new person enters the scene and then adopt a face tracker as the one proposed in Birchfield [151]. A different approach, instead, is the one proposed by Maio and Maltoni in [152] that works on grey scale images. In particular, face candidates are obtained through an ellipse detection applied over the gradient. The algorithm we designed in this work is based on the elliptical approximation of the head shape and the generalized

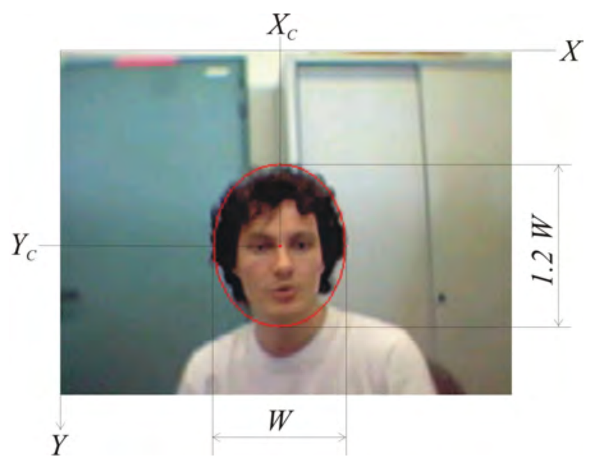


Figure 7.1: The elliptical face model adopted for the head detection. The size ratio and the orientation are fixed.

Hough transform. In particular, the method can be considered as a mixture of the two proposals reported in [151] and [152].

The proposed approach has other important characteristics, that make it particularly suitable for video surveillance applications. As first, it is featured by a low computational cost, which is a mandatory requirement to reach real time performance. Secondly, the images taken from video surveillance systems usually have a lower quality than a picture and the head sizes are smaller since the field of view is kept large to cover a wider area. In such a situation it is impossible to identify face features like eyes, mouth, lips, and so on. Thus, a feature based face detector is not employable, while algorithm based on color and shape are still valid. Finally, the proposed method is able to detect the head of the people turned back. For this reason we prefer to call the implemented algorithm “head detection” instead of “face detection”.

7.2.2 Head model

In surveillance applications the head size is too small to detect face features like eyes, lips, and so on; thus we adopt a head model based on the color histogram and the border shape only. As first, we exploit an elliptical head model with a fixed size ratio empirically sets to 1.2. Furthermore, we suppose the ellipse to be vertical; in such a manner the ellipse containing the head has three degrees of freedom that could be expressed with the coordinates of the center (X_c, Y_c) and the size W of the horizontal axis (see Figure 7.1).

In addition to the shape, the color information is used to characterize the head. In particular, we adopt as descriptor the color histogram **H** computed over the set of points internal to the above defined ellipse. To speedup the detection phase and to reduce the amount of memory required for each model, we employ the compressed space proposed in [151], which is composed by the three components (c_1, c_2, c_3) of Equation 7.1.

$$\begin{cases} c_1 &= B - G \\ c_2 &= G - R \\ c_3 &= R + G + B \end{cases} \tag{7.1}$$

We use 3 bits for representing c_1 and c_2 (chrominance components) and 2 bits for c_3 (luminance component), so that the color histograms are composed by 256 color bins. These histograms contain both face and hair colors; thus, a general and unique model can not be obtain, but a custom histogram $\tilde{\mathbf{H}}$ has to be computed and saved for each person. Summarizing, a generic head Ξ can be represented by its center coordinates, its size and a color histogram:

$$\Xi = \{(X_c, Y_c), W, \mathbf{H}\}. \tag{7.2}$$

Whenever a person enters the scene, a head model $\tilde{\Xi} = \left\{(\tilde{X}_c, \tilde{Y}_c), \tilde{W}, \tilde{\mathbf{H}}\right\}$ is computed and associated to it. In the following frames, a set $\{\Xi_i\}$ of candidate heads are generated and tested. The most likelihood head $\hat{\Xi}_i$ is chosen using the color and the gradient modules described in the following.

7.2.3 Color module

Let $\Xi = (X_c, Y_c, W, \mathbf{H})$ be a head candidate, defined by the coordinates (X_c, Y_c) of the center, the width W and the color histogram $\mathbf{H}$. Given the correspondent head model $\tilde{\Xi} = (\tilde{X}_c, \tilde{Y}_c, \tilde{W}, \tilde{\mathbf{H}})$, we compute the color-based probability P_C to be a head using the histogram intersection:

$$P_C\left(\Xi \middle| \tilde{\Xi}\right) = P_C\left(\mathbf{H} \middle| \tilde{\mathbf{H}}\right) = \frac{\sum\limits_{k=1}^{256} \min\left(\mathbf{H}(k), \tilde{\mathbf{H}}(k)\right)}{\sum\limits_{k=1}^{256} \mathbf{H}(k)} \tag{7.3}$$

From the practical point of view, the histogram intersection gives us a measure of how many colors of the head candidate are present in the reference model. In other words, the probability value is equal to 1 if the candidate is exactly the same of the model; instead it decreases if the candidate contains colors that do not appear in the model.

7.2.4 Gradient module

Goal of this module is to measure how much a head candidate Ξ has an elliptical shape. To this aim we compute the two normalized gradient maps S_X, S_Y of the image (along the horizontal and the vertical direction respectively) using the Sobel masks and we generate the set E of image coordinates belonging to the ellipse centered in (X, Y) and having a size equal to W. Formally, we can define E using the following equation (where γ and δ are used to take into account the rounding of the coordinates).

$$\begin{aligned} E(\Xi) = \ & \left\{(x,y) \middle| \left(\tfrac{(x-X_c+\gamma)^2}{1} + \tfrac{(y-Y_c+\delta)^2}{1.4}\right) = W^2\right\}; \\ & \text{with } \ \gamma, \delta \in [-0.5; 0.5] \end{aligned} \tag{7.4}$$

The gradient based probability P_G is obtained considering the gradient module in correspondence of the points of E.

$$P_G\left(\Xi \middle| \tilde{\Xi}\right) = \frac{\sum\limits_{p \in \mathrm{E}(\tilde{\Xi})} \sqrt{S_X^2(p) + S_Y^2(p)}}{\left|E(\tilde{\Xi})\right|} \tag{7.5}$$

where $|E| = \sum\limits_{p \in \mathrm{E}} 1$.

7.2.5 Head detection

Given a set $\{\Xi_i\}$ of head candidates, we select as current detection $\hat{\Xi}$ the one that maximizes a global score $\Phi(\Xi_i)$ as in Equation 7.6.

$$\begin{aligned} \Phi(\Xi) &= \alpha \cdot P_C(\Xi) + (1-\alpha) \cdot P_G(\Xi_i) \\ \hat{\Xi} &= \underset{i}{argmax}\, \Phi(\Xi_i) \end{aligned} \tag{7.6}$$

The parameter α is used to differently weight the color and the gradient module and should be adapted depending on the particular application or video characteristics. In fact, if the head size is too small, the shape term could be less significant and not so distinctive, since other almost circular objects can be present in the scene. Similarly, the video quality could be so much degraded to avoid the efficacy of the color module.

The set $\{\Xi_i(t)\}$ of head candidates is obtained starting from the head extracted on the last frame $\hat{\Xi}(t-1)$. For each head detected at the previous time step $t-1$, a prediction based on constant velocity and constant size is computed for the current frame t. To take into account scale and direction changes, the set $\{\Xi_i(t)\}$ also contains head candidates of different size and position. In particular, the position is searched in an area whose dimension is function of the person's velocity. The faster the person moves, the larger the search window is. Instead, the size $W(t)$ of the head is searched within a fixed range around the previous size $W(t-1)$.

7.3 Integration with the motion detection module

To reduce the computational cost and increase the precision of the detection, we use the foreground blobs extracted with a background subtraction module as input of the head detection subsystem. If the camera is still, a common background subtraction technique can be used in order to obtain a valid foreground region as in Fig. 7.2.

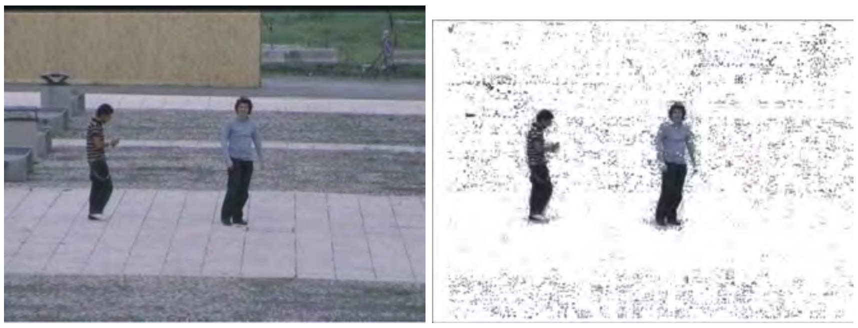

(a) Current frame (b) Extracted Foreground

Figure 7.2: Face detection over the foreground obtained with a background subtraction algorithm

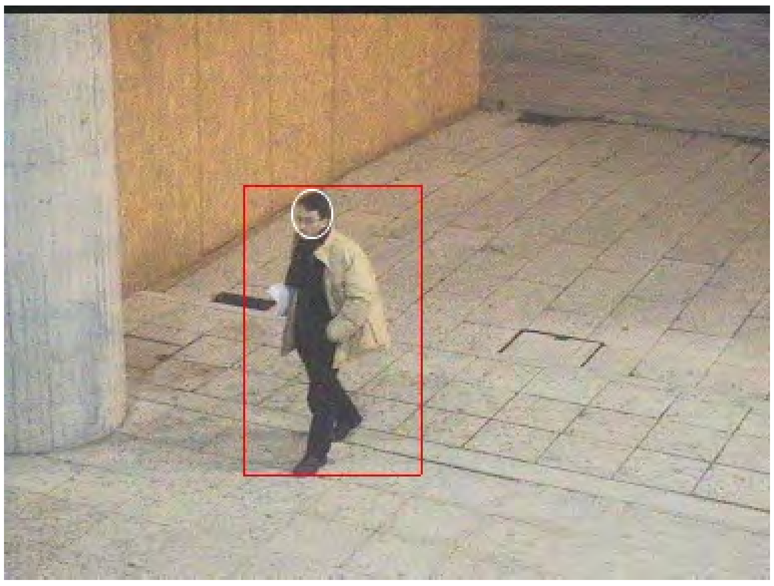


Figure 7.3: Search area obtained with the dynamic mosaicing algorithm and the detected head

If the adopted camera is moving, instead, we can apply the mosaicing algorithm described in Chapter 3. In this case, even if we cannot extract a reliable foreground region, we can estimate the bounding box of the people and reduce the head search inside these regions. In Fig. 7.3 the search area (i.e., the bounding box of the person) defined with the tracking algorithm and the detected head are highlighted. In Fig. 7.4 are reported some frames in which the head search windows have been

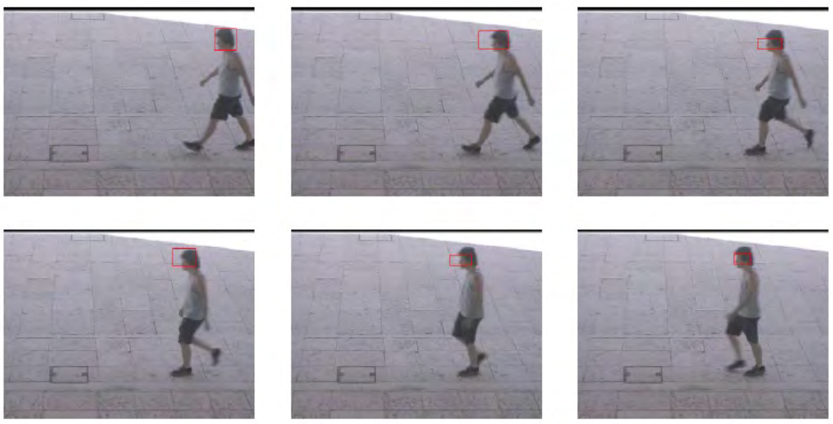


Figure 7.4: Input frames where the corresponding search areas have been superimposed. The size of the search area is dependent on the motion of the tracked person.

superimposed.

7.4 Hough Transform for Face Detection

To speedup the face detection previously described, we have developed a sort of Hough Transform using the color and the gradient measures of the previous section. For each tracked object O_j, two different Hough transforms are computed: one gradient-based T_G and one colour-based T_C. The points belonging to the edges of the track (obtained with Sobel edge detectors) vote for the first transform according to the gradient value. The selection of the voted pixels is done by moving on the image in the same direction of the gradient with a distance obtained from the estimated head size (see Figure 7.5). Calling α the angle of the gradient of the point (x, y) with respect to the horizontal axis, a and b the horizontal and the vertical half-sizes of the ellipse respectively, we can obtain the coordinates of the two candidate centers of the face (FC_1, FC_2):

$$\begin{array}{cc} \Delta x = \frac{a^2}{\sqrt{a^2+b^2 \cdot tg^2\alpha}} & \Delta y = \frac{b^2}{a^2} \cdot tg\alpha \cdot \Delta x \\ FC_1 = (x - \Delta x, y - \Delta y) & FC_2 = (x + \Delta x, y + \Delta y) \end{array} \quad (7.7)$$

Similarly, a point of the object votes for the color-based transform if its color has a non-zero value on the histogram H. In this case, it votes for all the points inside an ellipse having the same size of the head and the current pixel as the center, and the rate is proportional to the model histogram value corresponding to the color of the pixel. After that, the two transforms are normalized and multiplied pixel-by-pixel to obtain a single map that contains both color and gradient information. The point with the higher value is chosen as the head center of the object O_j. Fig 7.6 shows some transform maps together with the corresponding detection results.

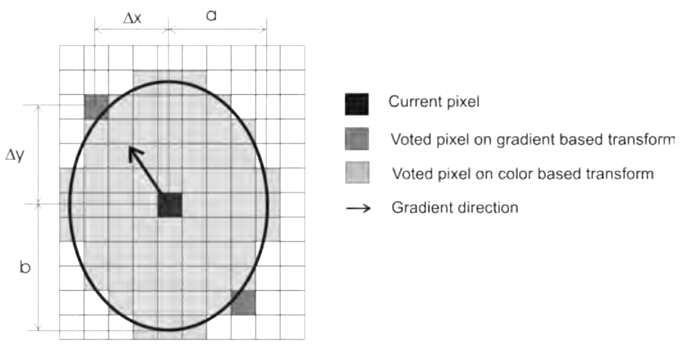

Δx
a
Δy
b
Current pixel
Voted pixel on gradient based transform
Voted pixel on color based transform
Gradient direction

Figure 7.5: Computation of the elliptical Hough transforms

7.5 Experimental Results

Figure 7.7 gives an overview of some results in indoor environment. For each frame motion and shadow detection module are exploited for the segmentation of the person. The HMM classifies the posture and the face detection module selects the head (face). The best frontal view of the face is then exploited for recognition. Table 7.5 shows some results in video with ground truth, indoor and outdoor. Differently from results of Table 7.1, face detection is not carried out at frame level but the head is detected initially and then tracked. In this case we obtain two important results: the face is detected at lower resolution also (less then 40x40 pixels) and even in different poses. These results can be useful for identification purposes in security oriented applications. For privacy issues, instead, in the case the face is too small, the person's identity is already protected by the low image resolution, as it is shown in Fig. 7.8(b).

Video	N frames	% Recogn.	Frontal View^1	Lateral view^2	Lateral Horiz. View^3	Top View^4	Mean Face Size
V3	328	100%	104	107	0	117	31x39
V4	440	99%	112	162	166	0	25x31

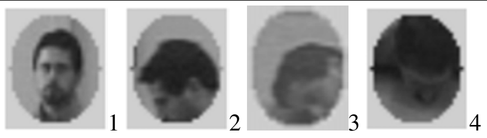


Table 7.2: Performance of the face detection and tracking module (top) split on four different head postures (bottom)

7.6 Conclusion

The world of distributed video surveillance systems is still disjoint with the world of biometry systems, for many reasons, partially technical and partially politi-

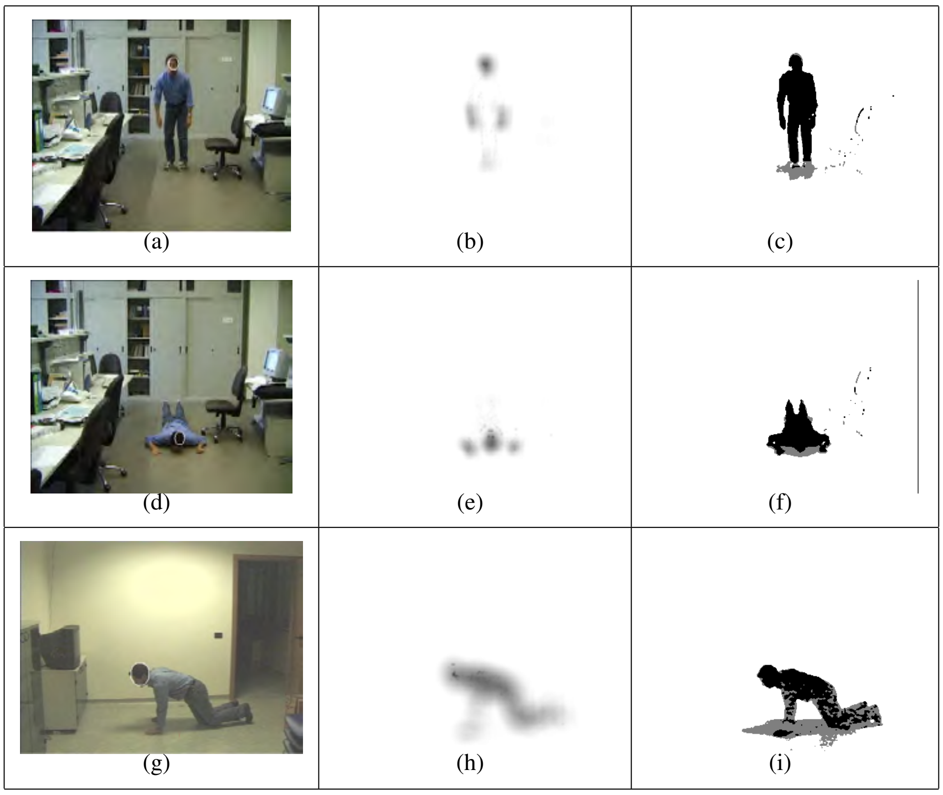


Figure 7.6: Some results obtained; left column: original frames with superimposed ellipses representing the head; middle column: results of the motion segmentation (the areas colored in gray are classified by the system as shadows); right column: Hough transform values (darker color indicates an higher probability value of the head center).

cal/ethical. The work we made aimed at addressing new paradigms of integration. We have proposed a general purpose approach to distributed video surveillance and face detection for indoor and outdoor environments that can be refined and tailored for many applications. Some examples are the surveillance of public places such as airports, stations, parks, to avoid dangerous situations (e.g. to control people in forbidden areas) or to prevent crimes (e.g. by controlling abandoned packs and luggage); the control of autonomous robots for industrial applications to improve safety of employers; the monitoring of indoor environments such as private houses, working areas, and so on to assure the safety of people living and working under the control of intelligent sensors in total respect of privacy issues.

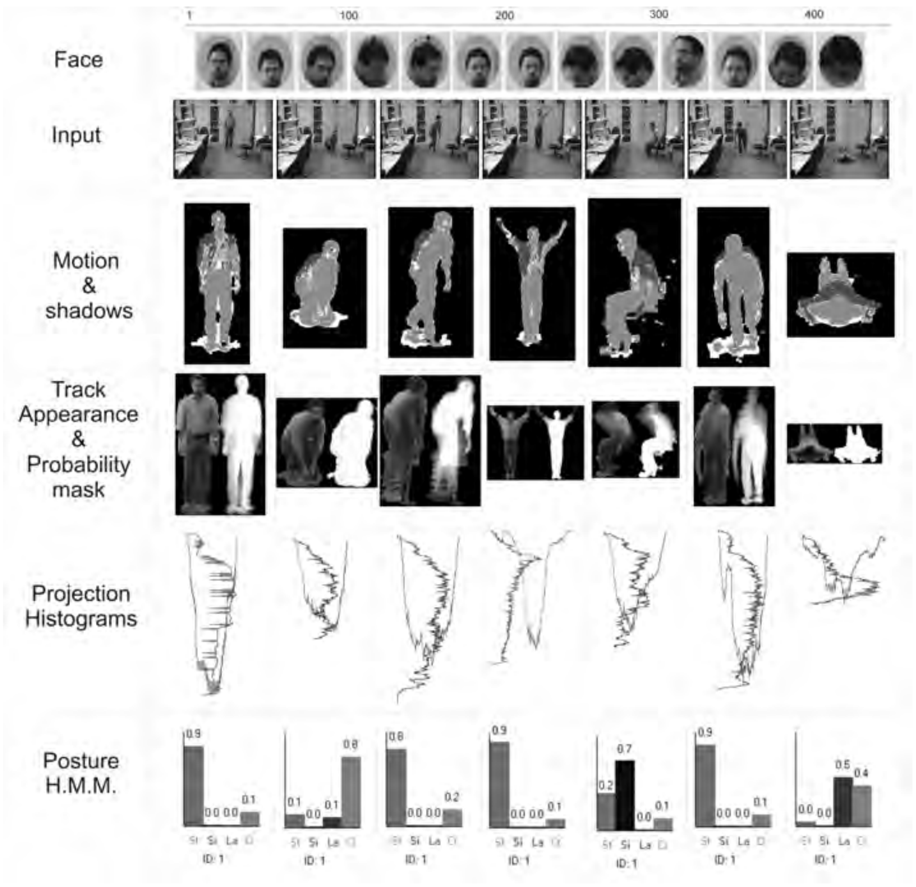


Figure 7.7: Results of the face detector algorithm in indoor surveillance.

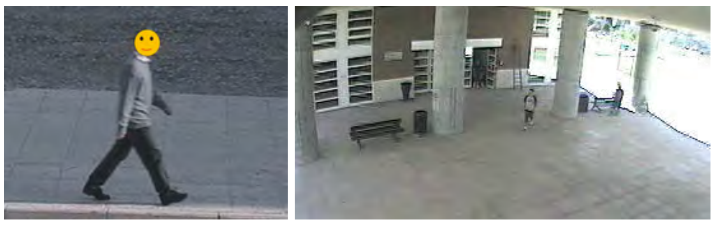


Figure 7.8: Examples of: face obscuration (a) and avoidable face obscuration (b)

Part III

Integrated VideoSurveillance Systems

Chapter 8

Multicamera Systems

8.1 Introduction and Related Work

Despite of the complexity increase, multiple camera systems exhibit the undoubt advantages of covering wide areas and enhancing the managing of occlusions (by exploiting the different viewpoints). However, the automatic merge of the knowledge extracted from single cameras is still a challenging task, especially in application of *distributed people tracking*. The goal is to track multiple people moving in an environment observed by multiple cameras tightly connected, synchronized and with partially overlapped views.

The solution to this problem must deal with two sub-problems: the reliable tracking in each camera system and the preservation of the identity of the people moving from a camera's view to the one of another camera. This second task is often known as *consistent labeling*.

Approaches to consistent labeling can be generally classified into three main categories: geometry-based, color-based, and hybrid approaches. The former exploits geometrical relations and constraints between the different views to perform the consistent labeling process. Instead, *color-based* approaches base the matching essentially on the color of the tracks [153, 154]. Finally, *hybrid* approaches mix information about the geometry and the calibration with those provided by the visual appearance, and they are based on probabilistic information fusion [155] or on Bayesian Belief Networks (BBN) [156, 157].

Geometry-based approaches can be further subdivided into calibrated and uncalibrated approaches. Among calibrated approaches, two particularly interesting papers are [158], in which homography is exploited to solve occlusions, and [159], that uses the epipolar lines. A very relevant example of the uncalibrated approaches is the work of Khan and Shah [13], based on the computation of the so-called *Edges of Field of View*, i.e. the lines delimiting the field of view of each camera. Through a learning phase in which a single track moves from one view to another, an automatic procedure computes these edges that are then exploited to keep consistent labels on the objects when they pass from one camera to the adjacent. During the training phase, the correspondences between points belonging to two overlapped cameras are extracted at the camera handoff moment. This can bring to false correspondences, as in the case of a person entering from the bottom of the image. In such a situation, the head in the first camera is put in correspondence with the feet in the other one. However, this matching is reliable enough if the goal is only the consistent labeling at the camera handoff instant (as in [13]) and if the people have the same height. Instead, if an exact correspondence is required, for example to compute an homography transformation, we must verify that the matching points belong to the same real point (e.g., the feet).

In this chapter we proposed an uncalibrated geometrical approach based on the Edge of Field of View, similar to [13]. To solve the above mentioned drawbacks, we introduce the concept of *Entry Edge of Fields of View* (E^2oFoV) to assure the consistency between the extracted lines and to compute a precise homography, used to establish the consistent labeling.

Besides being able to correctly track people, a multicamera system should also be very fast, since efficient reaction is a key point in video-surveillance. The consistent labeling can, indeed, result to be quite slow in crowded environments, where several cameras are present and many people are moving. For this reason, this chapter also reports an algorithm to reduce the computational cost of the consistent labeling establishment. In particular, a graph-based model, named *camera transition graph* (CTG), generalizable for a set of N overlapped cameras, is employed to efficiently search for the best match between objects in two overlapped cameras, similarly to the Vision graph described in [160] or the topology graph presented in [161].

The experimental result section shows very complex situations of multiple people crossing simultaneously the border of the FOV. The experiments have been provided in a real setup with partially overlapped cameras monitoring an outdoor environment (See Fig. 8.1).

8.2 Detecting Overlapping Areas

The proposed approach belongs to the class of uncalibrated geometry-based techniques. Let us suppose that the system is composed of a set $\mathbf{C} = \{C^1, C^2, ..., C^n\}$ of n cameras, with each camera C^i overlapped with at least another camera C^j. Let us call *3DFOV lines* $L^{i,s}$ the projection of the limits of the field of view (FOV) of a camera C^i on the ground plane ($z = 0$), corresponding to the intersection between the ground plane and the rectangular pyramid with its vertex at the camera optical center (the camera view frustum); s indicates the equation of the line in the image plane. In particular, four of them, $L^{i,s_h}, h = 1..4$ could be computed, with s_h corresponding to the image borders $x = 0$, $x = x_{max}$, $y = 0$, and $y = y_{max}$. They could be visible also by another camera; in such a situation we call *Edge of Field of View* $L_j^{i,s}$ the 3DFoV line corresponding to s of the Camera C^i seen by the camera C^j. The $L_j^{i,s}$ cannot be always computed, because sometimes is totally

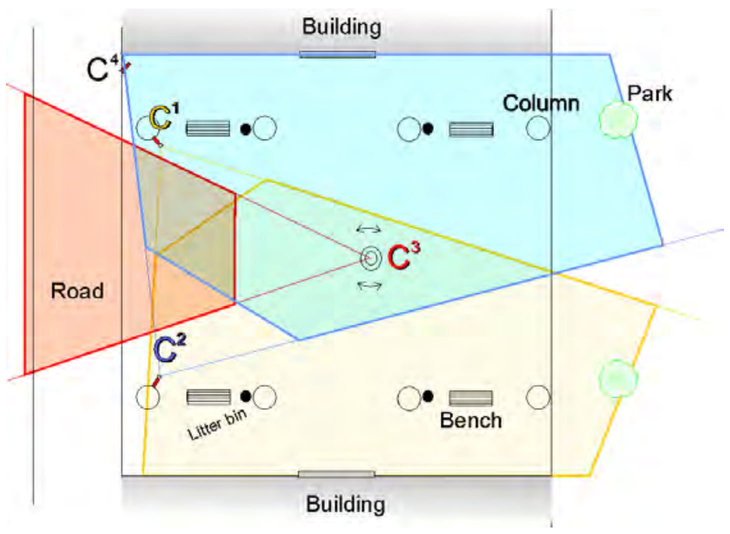


Figure 8.1: Map of our real setup.

hidden by a large object (e.g. a column). For our purposes, partial visibility is sufficient.

The EoFoV $L_j^{i,s}$ divides the image on camera C^j into two half-planes, one overlapped with camera C^i and the other one disjoint. The intersection of the overlapped semi-planes defined by the EoFoV lines from camera C^i to camera C^j generates the overlapping area $AoFoV_j^i$.

The EoFoV lines are created with a training procedure; the process is iterated for each pair (C^i, C^j) of partially overlapped cameras. To this aim, we need the correspondences of a certain number of points on the ground plane in the two considered views. Thus, as proposed in [13], during the training phase a single person moves freely in the scene, with the minimum requirement to pass through at least two points of each 3DFoV.

Therefore, even if the EoFoV is not completely visible, it can be computed if we are able to detect a sufficient number of reliable corresponding points belonging to the ground plane $z = 0$. In Fig. 8.1 a Map of the setup at the Modena's Campus is depicted where four cameras partially overlapped (one PTZ and three fixed cameras) have been installed. In Fig. 8.2 the synchronized views of C^1 and C^2 are shown.

In Fig. 8.2.a the EoFoV $L_2^{1,s}$ and $L_1^{2,s}$ are indicated. Let us suppose to have in Fig. 8.2.a only the person P (e.g. the one labelled as 3). When he enters in the FoV (camera handoff) as in Fig. 8.2.a of C^1, his support point is computed and matched with the support point of the correspondent shape K detected in C^2. The support point SP is defined as the middle point of the bottom of the bounding box of the blob, with the assumption that the training person is walking in a standing position. Therefore, collecting several matching pairs (SP_P^i, SP_K^j), the EoFoV can be computed with a Least Square Optimization. In the example in Fig. 8.2.b the

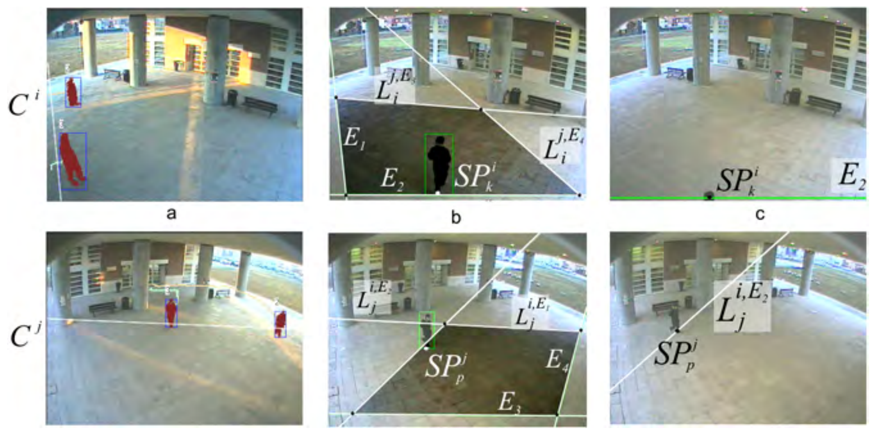


Figure 8.2: Camera handoff. a. Simultaneous camera handoff of two tracks; b. E^2oFoV creation; c. $EoFoV$ creation (using Khan-Shah approach)

$L_1^{2,s}$ corresponds approximately to the border of the image, being computed at the camera handoff moment.

However, there are cases in which, at the moment of the camera handoff, the detected parts of the person do not lie on the ground plane, as in Fig. 8.2.c, where the head is detected. Thus, matching a head's point in this camera with the SP in the other camera is incorrect and causes an erroneous EoFoV computation.

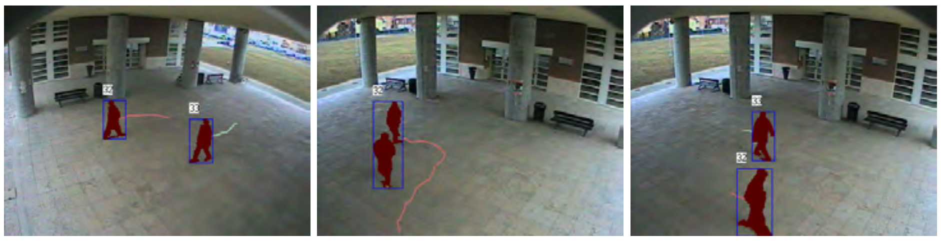
To avoid this problem, we define *Entry EoFoV (E^{2}oFoV)* as the EoFoV that is computed with the matching of (SP_P^i, SP_K^j) extracted by the bounding boxes of *totally* visible people, i.e. after the camera handoff when the blob does not "touch" the image border anymore. This approach can bring to a displacement of the line with respect to the actual limit of the image, but it assures the correct match of the feet's position in the two views. As a consequence, the actual FOV lines E_h are neither coincident nor parallel to the image borders. In Fig. 8.2.b the E^{2}oFoV lines $(L_j^{i,E_1}, L_j^{i,E_2})$, $(L_i^{j,E_3}, L_i^{j,E_4})$, correspondent to (E_1, E_2) in C^i, (E_3, E_4) in C^j respectively, are depicted. Fig. 8.2.b shows also the overlapping FoV, named Area of FoV $AoFoV$, delimited with the E^{2}oFoV and E_h lines.

8.3 Consistent Labeling with Homography

In order to propose a general approach we define the solution of consistent labeling not only at the camera handoff but whenever it is necessary to exploit homography.

For two overlapped cameras C^i and C^j, the training procedure computes the $AoFoV_j^i$ and $AoFoV_i^j$ areas. The four corners of each of these area define the two sets of four points:

$$\begin{cases} P_j^i = \{p_1^{i,j}, p_2^{i,j}, p_3^{i,j}, p_4^{i,j}\} \\ P_i^j = \{p_1^{j,i}, p_2^{j,i}, p_3^{j,i}, p_4^{j,i}\} \end{cases}, \tag{8.1}$$



(a) C^2 at frame #1250 (b) C^1 at frame #1250 (c) C^1 at frame #1260

Figure 8.3: Example of simultaneous crossing of two merged objects (frame #1250, and split after entered at frame #1260

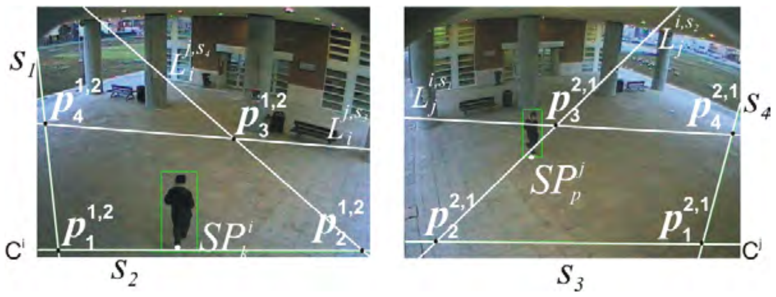


Figure 8.4: Point sets used for computation of the ground plane homography between two different cameras

where the subscripts indicate corresponding points in the two cameras (see Fig. 8.3). These four associations between points of the camera C^i and points of the camera C^j on the same plane $z = 0$ are sufficient to compute the homography matrix H^i_j from camera C^i to camera C^j. Obviously, the matrix H^j_i can be easily obtained with the equation $H^j_i = (H^i_j)^{-1}$.

Each time a new object is detected in the camera C^i in the overlapping area (not only at the moment of the camera handoff), its support point is projected in C^j by means of the homographic transformation. The coordinates of the projected point could not correspond to the support point of an actual object. For the match we select the object in C^j whose support point is at the minimum Euclidean distance in the 2D plane from these coordinates.

This approach is an efficient tradeoff between classical approaches that verify correspondences at the camera handoff only as in [13], and complex methods of 3D reconstruction that find correspondences at each frame preventing any real time implementation [159]. Instead the matching is verified whenever a new track is detected in an image and also a 1-to-n or n-to-m match could be computed for coping with the cases where more people passe through the EoFoV at the same time as in Fig. 8.2.a. Moreover, as in Fig. 8.3, some people could be initially detected as a group and thus matched with a single track. For instance, in Fig. 8.3 a group is matched to a single object (label 32). Nevertheless, whenever the people can be detected separately (after 10 frames) the correct consistent labeling is recovered.

8.4 Camera Transition Graph

When a new track is detected in camera C^i the system must check whether it is a completely new track or it is already present in other cameras. This check can be very complex and computationally expensive if many cameras with many tracks are present. To this aim, a graph model has been defined to exploit camera relationships in reducing the search space of the multi-camera matching process.

In the proposed model a graph is built using information acquired during the training phase. The graph is called *Camera Transition Graph* (CTG), because it incorporates information about camera position and it models possible tracks handoff among overlapped views.

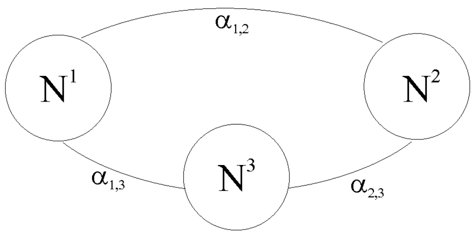


Figure 8.5: Example of Camera Transition Graph (CTG).

The problem can be viewed as a Constraint Satisfaction Problem (CSP). The CTG is a symmetric graph where each node N^i is the set of objects visible and tracked in an instant in a camera C^i, and each arc $\alpha_{i,j} = \alpha_{j,i}$ indicates the presence of a common $AoFoV$ between C^i and C^j and needs that the constraint of consistency must be verified if a track is visible inside the $AoFoV$.

In the CTG for each node N^j we denote $T^j(t) = \left\{\tau_i^j(t) | \, i = 1, \ldots, k^j(t)\right\}$ the set of variable at the time t, that are the tracks detected and with $x_i^j(t)$ their correspondent assigned labels (the instanced values for the variables).

The unary constraint at each node is that two distinct tracks must have different labels and they must be conserved during the time. This is the typical tracking problem from a single camera and the unary constraints must be checked at each frame. Instead, the binary constraint of "consistent labeling" over the $AoFoV$ is verified only when needed.

The state of the whole system at any time can be either *consistent* or *inconsistent*. We refer to a consistent state whenever the constraints are satisfied and all the projections of the same person on multiple cameras are marked with the same label, while different people have different labels.

If a new track is detected at time $t+1$, the system is switched to inconsistency, and the system must check if it appears also in other cameras or not. Exploiting graph theory, specifically solving an arc-consistency problem on the CTG, it is possible to correctly select only cameras and tracks generating inconsistencies and leaving the rest of the system unchanged.

Suppose that at time t the state of the system is *consistent*. Suppose also that on camera C^l of the node N^l, at time $t+1$ is detected a new track $\tau^l_{k^l(t)+1}$ labelled $x^l_{k^l(t)+1}(t+1)$. Instead of searching for possible matches across cameras' views, we analyze the graph node corresponding to camera C^l, N^l, and we compute the set Ψ^l of nodes linked to it by means of a consistency constraint arc:

$$\Psi^l = \left\{ N^i \,|\, \exists\, \alpha_{i,l} \right\} \tag{8.2}$$

Let us call $SP^l_{k^l(t)+1}$ the support point of the new track, computed as reported in Section 8.2. For each element N^i of the set Ψ^l we must evaluate if the support point $SP^l_{k^l(t)+1}$ lies inside the $AoFoV^i_l$ between camera C^l and camera C^i on the image plane of C^l. To this aim, we used a boolean function ϕ defined as follows:

$$\phi\left(N^l, N^i, SP\right) = \begin{cases} 1 & if\, SP \in Z^i_l \\ 0 & otherwise \end{cases} \tag{8.3}$$

In the case that ϕ returns zero for each node N^i in the set Ψ^l the new track is not visible in any other overlapped camera, thus, a new label is assigned and the system is consistent.

Otherwise, the search space $\Sigma^l_{k^l(t)+1}$ for the new track $\tau^l_{k^l(t)+1}$ is composed by the set of tracks τ^i_m such that:

$$\begin{aligned} &(i) N^i \in \Psi^l \\ &(ii) \phi\left(N^l, N^i, SP^l_{k^l(t)+1}\right) = 1 \\ &(iii) \phi\left(N^i, N^l, SP^i_m\right) = 1 \end{aligned} \tag{8.4}$$

In other words, for each camera C^i whose view is overlapped with C^l (condition (i) of equation 8.5) and in which the new track is visible (condition (ii) of equation 8.5), the set of tracks $T^i(t+1)$ at the time $t+1$ is considered. For each track of this set, the visibility on the camera C^l is checked by means of function ϕ (condition (iii)) and if it is visible, the track is considered as a candidate for the consistent labeling. It is evident that the search space $\Sigma^l_{k^l(t)+1}$ obtained with this procedure is minimal and that results in computational saving, especially if the matching procedure is complex and time consuming, as in the case that the track appearance is used.

8.5 Track warping

In this section we describe how use the consistent labeling to improve the appearance based tracking system; in particular, similarly to [162], we transfer appearance memory models among cameras applying homography transformations.

As stated above, the probabilistic tracking is able to handle occlusions and segmentation errors in a single camera system. However, to be robust to occlusions the strong hypothesis is made that the track has been seen for some frames without occlusions so that the appearance model is correctly initialized. This hypothesis is erroneous in the case the track is occluded since its creation (as in Fig. 8.6(b)).

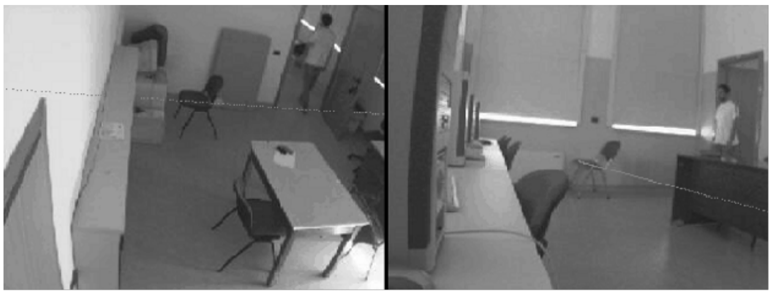


Figure 8.6: The frame extracted from the two cameras during the camera handoff. The lines are the intersections of the floor and the gate plane computed and drawn by the system

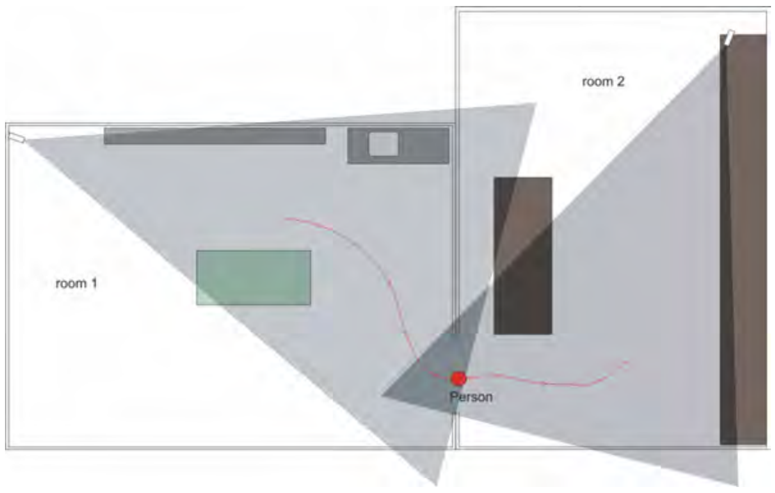


Figure 8.7: A scheme of the two rooms used for our tests

Our proposal is to exploit the appearance information from another camera (where the track is not occluded) to solve this problem. If a person passes between two monitored rooms (see for example our test bed setup reported in Fig. 8.7), it is possible to keep the temporal information stored into its track extracted from the first room (Fig. 8.6(a)) and use them to initialize the corresponding track in the second room (Fig. 8.6(b)).

To this aim the following assumptions are used:

- the two cameras are calibrated with respect to the same coordinate system

(X_W, Y_W, Z_W);

- the equation of the plane $G = f(X_W, Y_W, Z_W)$ containing the entrance is given;
- it is possible to obtain the exact instant t_{SYNC} when the person passes into the second room. To do this, the 3D position of the support point SP of the people inside the room could be used, or otherwise a physical external sensor could be adopted for a more reliable trigger;
- all the track points lie on a plane P parallel to the entrance plane G and containing the support point *SP* (hereinafter we call P Person's Plane). This assumption holds if the person passes the entrance in a posture such that the variance of its points with respect to the direction normal to P is low enough (e.g., standing posture).

The first three assumptions imply only an accurate installation and calibration of cameras and sensors, while the last one is a necessary simplification to warp the track between the two points of view. Under this condition, in fact, the 3D position of each point belonging to the appearance image of the track can be computed and then its projection on a different image plane is obtained.

In particular, the process mentioned above is applied only to the four corners of the tracks and thus the homography matrix **H** that transforms each point between the two views can be computed:

$$[x_2\ y_2\ 1]^T = \mathbf{H}_{3\times3} \cdot [x_1\ y_1\ 1]^T \tag{8.5}$$

Through **H** it is possible to re-project both the color components $(\bar{o})$ and the alpha value $\alpha(o)$ of each track point from the point of view of the leaving room to the point of view of the entering one (Fig. 8.8). The re-projected track is used as initialization for the new view that can in such a manner solve the occlusion by continuing to detect the correct posture.

As a test bed for our distributed system, a two-rooms setup has been created in our lab. An optical sensor is used to trigger the passage of a person between the two rooms. A map of the test environment is reported in Fig.8.7, while in Fig. 8.6 the frames extracted from the two cameras in correspondence of a sensor trigger are reported. In Fig. 8.6 the intersection lines between the floor plane and the entrance plane are automatically drawn by the system exploiting the calibration information. In the second room a desk that occludes the legs of a person during his/her entry has been placed in order to test the efficacy of our approach. In this situation the stand-alone people posture classifier fails, stating that the person is crouching (Fig. 8.9 top). Exploiting the described camera handoff module, instead, the legs of the person are recovered by the warped appearance mask and the posture classification produces the correct result (Fig. 8.9 bottom).

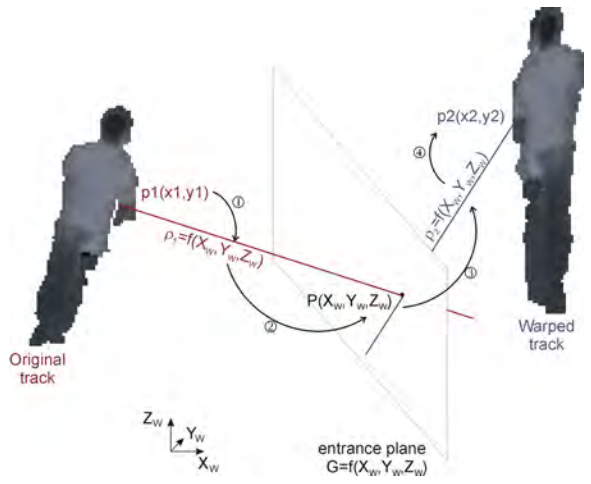


Figure 8.8: Track warping: (1) exploiting the calibration of Camera1, (2) intersection with entrance plane, (3) calibration of camera2, (4) intersection with Camera2 plane.

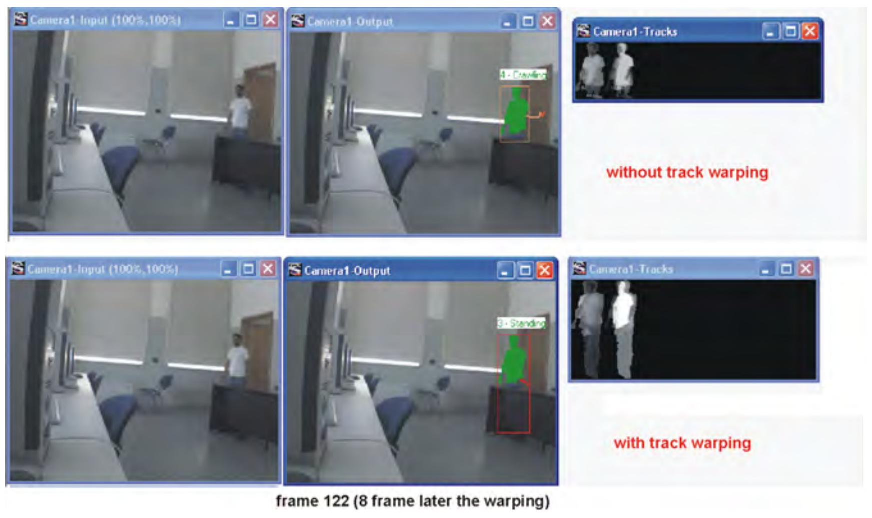


Figure 8.9: Initial occlusion resolved with track warping. a,d: input frames; b,e: output of the posture

8.6 Experimental Results

To test the system, we have installed four partially overlapped cameras in our department (see Fig. 8.1 for a map). The tests were carried out using the single camera probabilistic and appearance-based tracking module described in Chapter 5.

This approach permits to maintain the appearance of the track and thus to compute its Support Point also when it is partially occluded. E^2oFoV and $AoFoV$ of the three cameras have been computed over a training video of 8000 frames. As an evidence of the goodness of the automatically obtained homography we report in Fig. 8.10.a the mosaic image of three frames obtained merging a frame of a camera with the homographically distorted frames of the other two cameras. If the homography transformation between at least one camera and the ground plane is manually given, a sort of bird-eye view of the scene can be obtained (Fig. 8.10.b), in which the trajectory of a person is automatically drawn. In the figure some examples of the projections in the three views of the person are superimposed.

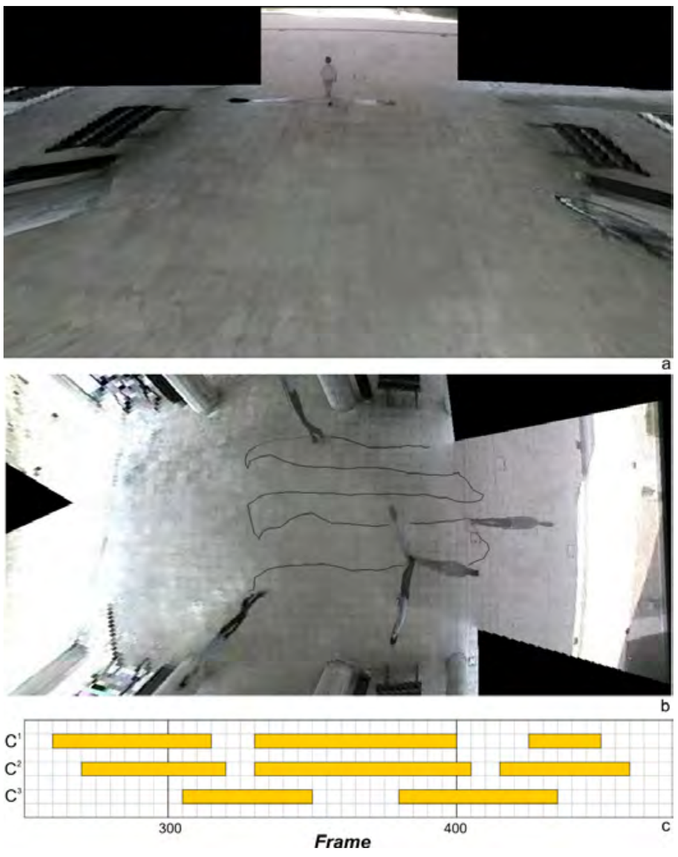
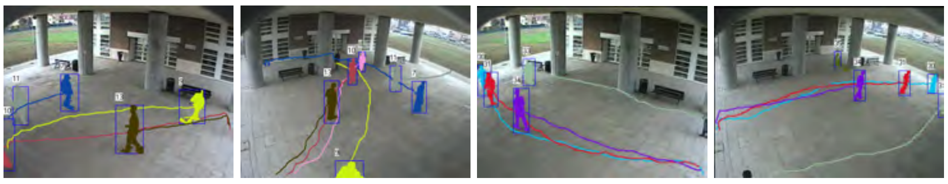


Figure 8.10: a. Automatically obtained mosaic image through homography; b. A bird-eye view of the scene with a superimposed trajectory; c. visibility over the three different cameras of the track used to generate the trajectory

Some snapshots of the output of the system (in non-trivial conditions) after the consistent labeling assignment are reported in Fig. 8.11.



(a) C^1 at frame #783 (b) C^2 at frame #783 (c) C^1 at frame #1080 (d) C^2 at frame #1080

Figure 8.11: Some snapshots of the output of the system after consistent labeling.

The track graphs in Fig. 8.10.c, and Fig. 8.12 report, for each person P_i, the slot of time (in frames) in which it is visible by the three cameras (C^1, C^2, and C^3) of our real setup. The color of the bars corresponds to the identifier assigned by the consistent labeling algorithm. In particular the graph in Fig. 8.10.c is related to the same sequence used to construct the trajectory of Fig. 8.10.b.

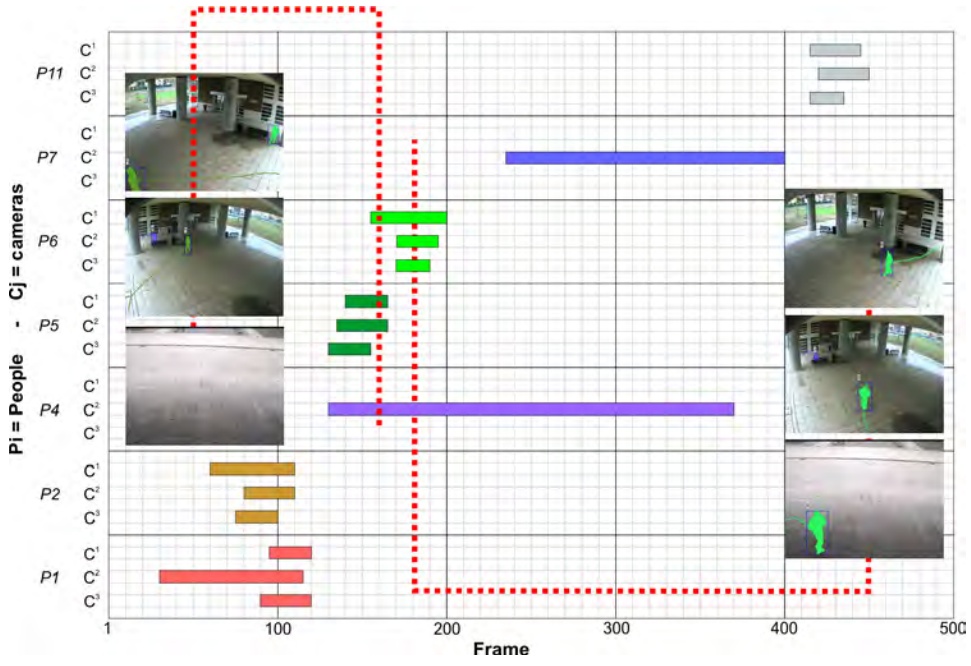


Figure 8.12: Visibility and labels (indicated with the color of the bars) of the tracks in a test sequence

8.7 Conclusions

This chapter presents a method for establishing consistent labeling in a multi-camera system. Its main contributions can be summarized as follows:

1) the computation of reliable E^2oFoV lines to obtain without calibration the correct area of overlap $AoFoV$

2) the computation of the homography matrices between two overlapped views by using the EoFoV lines;

3) the exploitation of the homographic transformation to establish consistent labeling in the whole overlapping area, in order to recover the correct labels in the case of objects that enter as merged and then split.

4) the automatic generation of a *Camera Transition Graph* that models the topology of the network of cameras reporting the field of view overlaps; this graph is used to reduce the search space during the consistent labeling phase.

The reported experiments demonstrate the accuracy of the proposed method, also in situations with many people overlapped and only partially visible.

Chapter 9

Geospatial Trajectory Estimation of a Moving Camera

9.1 Introduction

In the first part we have seen how recover the motion and thus the trajectory of a moving object by means of motion detection techniques. The video stream contains the object to be tracked and, knowing the camera position, it is possible to estimate the object trajectory.

In this chapter, instead, we suppose that the analyzed video has been captured by a moving camera, e.g. a hand held camera. A set of reference images from the same scene is available, together with information about the position where the images are taken from. Our aim is to recover the trajectory of the camera comparing the frames with the reference images.

The proposed methods have two main steps. First, scale invariant features (SIFT) are detected and matched between the reference images and the video frames, to calculate a weighted adjacency matrix (WAM) based on the number of SIFT matches. Second, using the estimated WAM the maximum matching reference image is selected for the current video frame, which is then used to estimate the relative position (rotation and translation) of the video frame using either the fundamental matrix or temporal fundamental matrix. The relative position is recovered up to a scale factor and the scale ambiguity is resolved using a triangulation between the video frame and two reference images. Results of recovering camera trajectory are reported for four real sequences, as well as on images from the ICCV Computer Vision 2005 contest dataset.

This work has been carried out at the CVLab - University of Central Florida - Orlando, under the valuable supervision of Prof. Mubarak Shah.

9.2 Description of the system

Geospatial trajectory estimation involves finding the GPS location of the captured video, using certain geometric constraints. GPS was first introduced by the US Department of Defence about 15 years ago for military personal and vehicle tracking around the world. Since then the GPS technology has been widely used in the areas of autonomous navigation and localization of vehicles and robots. It has been used by the military for terrain analysis and mapping, and targeting enemy vehicles and installations. Recently, commercial applications have employed GPS data with geo-referenced maps to recover the map of a city address or to provide directions between different city locations. In this paper we address two problems of localizing the geospatial position and estimating the trajectory of a moving camera based on the captured sequence of images. Geospatial position is localized using maximum SIFT matches between reference images (with known GPS) and video frames while camera trajectory is estimated using the geometric constraints (either spatial or spatio-temporal) between maximum matching reference images and video frames.

We start with the problem of finding the location of images with unknown GPS using images with known GPS location. We consider realistic scenarios where the method does not require 3D reconstruction and matching of the environment for localization. We recover the unknown GPS location of images using geometric constraints with the images with known GPS location and applying triangulation using two maximum matching reference images (see details in Section II). We then move on to the video domain where we recover the video trajectory using geometric constraints with images with known GPS location. In order to obtain smooth trajectory we apply b-spline smoothing using reliably localized GPS locations as control points (see details in Section III). We further extend this method by using intersection of triangulated vectors using all the matching reference images to recover the unknown GPS location of the video frames (see details in Section IV). Finally, we utilize spatio-temporal geometric constraints (instead of just spatial geometric constraints used above) using temporal fundamental matrix to recover the trajectory of the moving camera. This method does not require any trajectory smoothing as it inherently uses the neighboring video frames in the estimation process (see details in Section V). We now describe the related work in the next section and follow up with the details of the three proposed methods briefed above.

9.3 Related works

In literature, a variety of methods have been proposed for motion recovery and measurement of robot trajectory (odometry) using visual inputs. Visual odometry is a method that uses image sequences to obtain 6 degrees of freedom (dof) camera motion and dense 3D environment information. Structure from Motion (SFM) is the most common approach to solve problems such as automatic environment reconstruction, autonomous robot navigation and self-localization. In these approaches, a 3D reconstruction of the environment is performed during a learning phase or directly using the test video. Thus the actual camera position is obtained by 3D matching of the current view with the learned environment map. Methods that use such technique for recovering geospatial location include batched sequence of images [163–168], trifocal tensors [169], bundle adjustment and non-linear minimization [170, 171], supervised robot navigation [172, 173], panoramic images [174,175], stereo vision [176–178], and map correlation using visual odometry [179]. The most recent work using this approach was proposed by Royer et al. [180] for mobile robot navigation. The robot is first manually guided on a learning path. Then a map of the environment and its 3D reconstruction is performed off-line. Using this 3D reconstruction the robot is able to recover its pose with respect to the 3D environment model.

An approach for Simultaneous Localization And Mapping (SLAM) was initially proposed using active vision in [181–183]. Recently approaches for SLAM were proposed in [184–188], which are based on the probabilistic models of uncertainty. They assume a robot moving in a world with stationary landmarks (distinctive physical features) that can be observed by some sensor. The positions of the landmarks along with the robots position at a particular time are considered to be the system state. The problem consists of estimating the new state (robot and landmark positions) at the next time instance, given the last movement made by the robot and new observations provided by the sensory subsystem. The typical sensors used for measuring the distance and orientation of landmarks with respect to the robot are sonar rings [189–192], vision [193], and more frequently the laser scanners [186, 194]. Davison proposed two methods that do not require specialized hardware (laser scanners or sonar rings) for measuring distance and orientation of landmarks. His solutions are based on an active binocular head [195] and a single camera [196] that estimate the distance and orientation of visual landmarks. A disadvantage of these solutions is the need for an initial manual calibration: both for the position and orientation of the robot with respect to a predefined target of known size.

SFM and SLAM based approaches have two major disadvantages. Firstly, the task is computationally very expensive and is unnecessary to just recover the trajectory of the camera. Secondly, the 3D reconstruction of the environment may fail in certain instances where distinctive features cannot be computed e.g. images with trees only or areas with sparse buildings. Since these methods rely on 3D environment reconstruction and use a matching algorithm for pose recovery, therefore these methods may not fully recover the complete video trajectory.

For real time applications a different approach to visual odometry is followed. Commonly, a stereo rig is exploited to obtain the 3D coordinates of some feature points, and not the complete image (as in SFM). Using the corresponding feature points from the next frame, the relative camera motion is recovered. Iterating this procedure, the trajectory of the camera is recovered. This approach was originally developed by Matthies et al. [197] and was further refined by [178, 198, 199]. Nister et al. [178] estimate the camera motion using a geometric hypothesize-and-test architecture. Using the same approach, Cheng et al. [198] equipped NASAs Mars exploration Rover with a visual odometry system. Ozawa et al. [199] tested their system using a humanoid robot named H7, which is constructed for footstep planning on a 3D map, reconstructed by visual odometry. The system consists of two key components: 3D reconstruction via visual odometry from a stereo image sequence to obtain a dense local world model, and a footstep planner for biped robots, using the reconstructed 3D map.

A common problem of the real time methods is the requirement of geospatial alignment of the recovered trajectories. When applied to robot navigation, the six degrees of freedom can be reduced to only three [200], since robots normally navigate on a planar (or locally planar) ground. Another disadvantage is the requirement of a stereo rig for recovery of 3D feature points. What is novel in our approach is that we do not require a 3D reconstruction of the environment (or feature points) for recovering the camera trajectory. Rather we require a set of reference images with known GPS locations for geospatial localization of the novel video data. Also, we use sub-sampled video frames for localization of the camera trajectory. Thus our method has two advantages over existing methods. First, our method does not require all video frames to have distinctive features for geospatial localization. Second, our method is computationally less expensive since we do not require 3D reconstruction and matching of the environment for localization. We now describe our image localization scheme in detail in the next section.

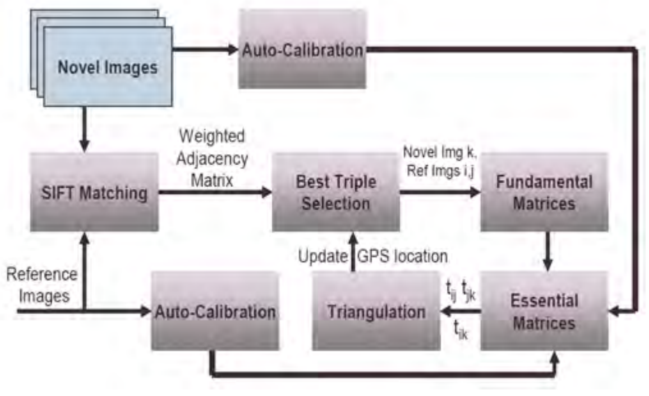


Figure 9.1: Overall geospatial image localization scheme.

9.4 Image Localization

Our first goal is to compute the geospatial localization of the novel images $\{V_t,\ t = 1..M\}$, given a set of reference images $\{I_p,\ p = 1..N\}$ with known GPS location. The assumptions made in this work are that some images have overlapping field of view with the reference images. Also, the camera does not zoom while capturing the reference images and novel images i.e. constant intrinsic parameters. These assumptions allow the auto-calibration of the capturing devices.

The overall geospatial image localization scheme is given in Fig. 9.1. Given a set of reference images we recover the camera intrinsic parameters using a method proposed by Luong and Faugeras [201]. Similarly, we recover the novel cameras intrinsic parameters using the novel images. Next, SIFT [202] features are detected and matched between the reference images and the novel images. Using the maximum matching reference images and novel images, we estimate the fundamental and essential matrices between reference image I_p and novel image V_t pairs to recover the camera pose. Finally, we apply triangulation to recover scale ambiguity in the estimated camera pose and obtain the geospatial localization of the novel image. We repeat these steps for all novel images to recover their GPS locations. The following sections detail these steps.

9.4.1 Estimating the Weighted Adjacency Matrix

In order to obtain geospatial localization using the fundamental matrix constraint, we require feature point correspondence between reference images I_p and novel images V_t. There are several methods to obtain point correspondence between images including Harris corner detector [203], Scale and affine invariant point detector [204], and Scale Invariant Feature Transform (SIFT) [202]. We empirically evaluated all three point correspondence methods and found SIFT to be the most robust matching method across a substantial range of affine distortion, change in viewpoint, addition of noise, and change in illumination. The SIFT features are highly distinctive, and each feature point is represented by a 128 dimensional feature vector.

A match is found for a feature in novel image V_t to a feature in I_p by estimating the ratio of the smallest to second smallest Euclidean distance between the feature vectors. In [202], the authors reject all matches which have a distance ratio greater than 0.8, which eliminates 90% of the false matches while discarding less than 5% of the correct matches. In our application, we set this threshold to 0.4, in order to have less number of very reliable matches. The above matching scheme may result in multiple feature points in novel image V_t matching with the same feature in image I_p. We obtain a one-to-one correspondence by maximum matching of a bipartite graph. The bipartite graph construction is obtained by treating the two feature point sets as nodes in bi-partitions (I_p and V_t) and the distance ratio as the weight on the edges between the bipartition. We obtain a weighted adjacency matrix $W(I_p, V_t)$ by finding the point correspondence between all pairs of I_p and V_t. Each entry in the matrix corresponds to the number of matching features between I_p and V_t. The corresponding set of matching point locations is stored in a matching matrix $M(I_p, V_t)$ such that:

$$M(I_p, V_t) = \{(x_1, y_1, x_2, y_2); (x_1, y_1) \in I_p \wedge (x_2, y_2) \in V_t\} \tag{9.1}$$

9.4.2 Pose Recovery of the Novel Images

Given a novel image V_t, we want to recover the position $P_{V_t} = \{X_t, Y_t, Z_t\}$ of its camera optical center with respect to a reference image I_p in the world coordinate system. Firstly, we test if the novel image is located at the same location as the reference image by applying a homography test. If it fits a homography with a reference image, then we assign the novel image the same GPS location as the reference image. Otherwise, we proceed using geometric constraints to recover the GPS location of the novel image. Since the GPS location of the reference images I_p are given in terms of longitude and latitude, we apply spherical to cartesian conversion to obtain $P_{I_p} = \{X_p, Y_p, Z_p\}$. We then find the maximum matching reference image I_p using $W(I_p, V_t)$. Furthermore, we utilize the set of corresponding points $M(I_p, V_t)$ to estimate the fundamental matrix F_t^p between images V_t and I_p using the constraint:

$$[x_1\ y_1\ 1] \cdot F_t^p \cdot [x_2\ y_2\ 1]^T = 0;\ \forall(x_1, y_1, x_2, y_2) \in M(I_p, V_t) \tag{9.2}$$

Due to noise in feature point location and incorrect point correspondence, the estimation of the fundamental matrix using the above linear constraint is erroneous. In order to obtain a robust estimate, we use RANSAC based fundamental estimation technique proposed by Torr et al. in [205]. Using the fundamental matrix and calibration matrices K_t and K_p (obtained using auto-calibration [206] of V_t and I_p respectively) we can estimate the essential matrix E_t^p by:

$$E_t^p = K_t \cdot F_t^p \cdot K_p. \tag{9.3}$$

The rotation R and translation t between V_t and I_p can be recovered from the essential matrix by using methods proposed in [206]. The translation vector t thus obtained is recovered up to a scale factor and this ambiguity is resolved using triangulation described next.

9.4.3 Resolving Scale Ambiguity

The triangulation scheme is employed to recover the scale ambiguity and thus obtaining $P_{V_t} = \{X_t, Y_t, Z_t\}$, i.e. the position of camera center for the novel image V_t. The triangulation method requires an image triplet (two reference images I_i, I_j and a novel image V_k) and the construction is depicted in Fig. 9.2. Thus, for each novel image V_t, two reference images are selected from the entire set, such that, the obtained triple has the maximum matching feature points with the novel image. The two reference images should also have different GPS locations otherwise the resulting triangulation construction is degenerate. More formally, given a novel image V_k, the best triplet $BT(V_k)$ is obtained using:

$$\begin{gathered} BT(V_k) = (V_k, I_i, I_j); \\ \mathrm{suchthat} \\ (i,j) = \operatorname*{argmax}_{i,j=1..N} \big(\min\big(\ \mathrm{W(V_k, I_i)},\quad \mathrm{W(V_k, I_j)},\quad \mathrm{W(V_k, I_i)}\ \big)\big) \end{gathered} \tag{9.4}$$

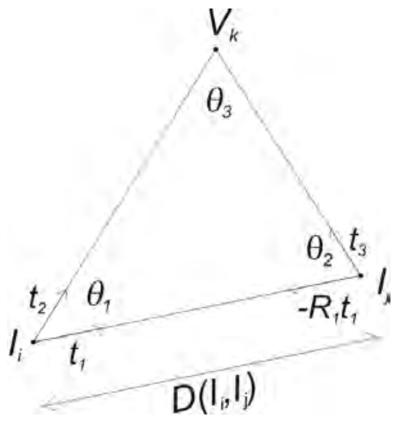


Figure 9.2: Triangulation between two reference images and novel image used to resolve scale ambiguity for GPS location estimation.

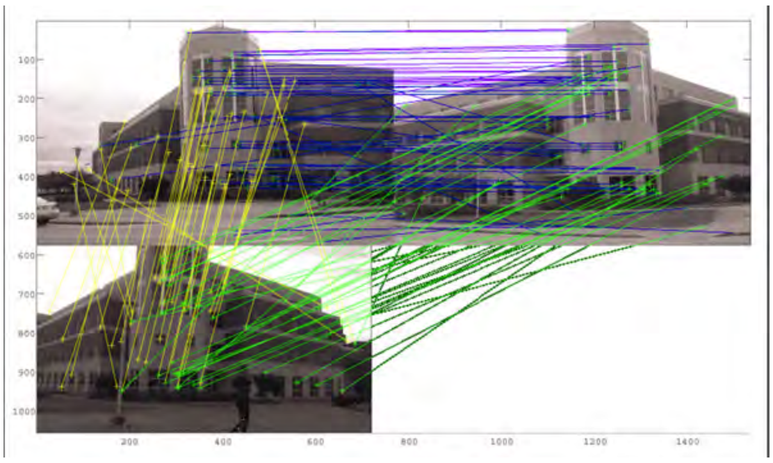


Figure 9.3: Example of best triplet selection and matching for novel image of Engineering building dataset. The two reference images are in top row while the novel image is in the bottom row.

Fig. 9.3 shows an example of best triplet selection for a novel image in the engineering building dataset. Given the rotations and translations between each pair of camera coordinate system (R_i^j, t_i^j), (R_i^k, t_i^k), and (R_j^k, t_j^k), we can compute the three internal angles (for the triangle) using:

$$\begin{cases} \theta_1 &= \cos^{-1} \frac{dot(t_1,t_2)}{norm(t_1)norm(t_2)} \\ \theta_2 &= \cos^{-1} \frac{dot(-t_1,R_1t_3)}{norm(t_1)norm(t_3)} \\ \theta_3 &= 180 - \theta_1 - \theta_2 \end{cases} \tag{9.5}$$

where $norm(x)$ is the magnitude of x. The scale factor is recovered through distance $D(I_i, I_j)$ obtained from the two GPS locations of the reference images. The location of camera center for novel image V_k is obtained using trigonometric identities (using sine law of triangles). The recovered camera center is converted from Cartesian to spherical coordinates to obtain GPS location in latitude and longitude. This process is repeated till all the GPS location for novel images are estimated. We now describe the extension of this method to video frames for recovering the trajectory of a moving camera.

9.5 Video Localization

Our second goal is to extend the geospatial localization scheme to videos. Thus, given the video frames $\{V_t,\ t = 1..M\}$ and a set of reference images $\{I_p,\ p = 1..N\}$ with known GPS location, we want to recover the trajectory of the moving video camera. The assumptions made in this work are that some video frames have overlapping field of view with the reference images. Also, the camera does not zoom while capturing the reference images and video frames i.e. constant intrinsic parameters. These assumptions allow the auto-calibration of the capturing devices.

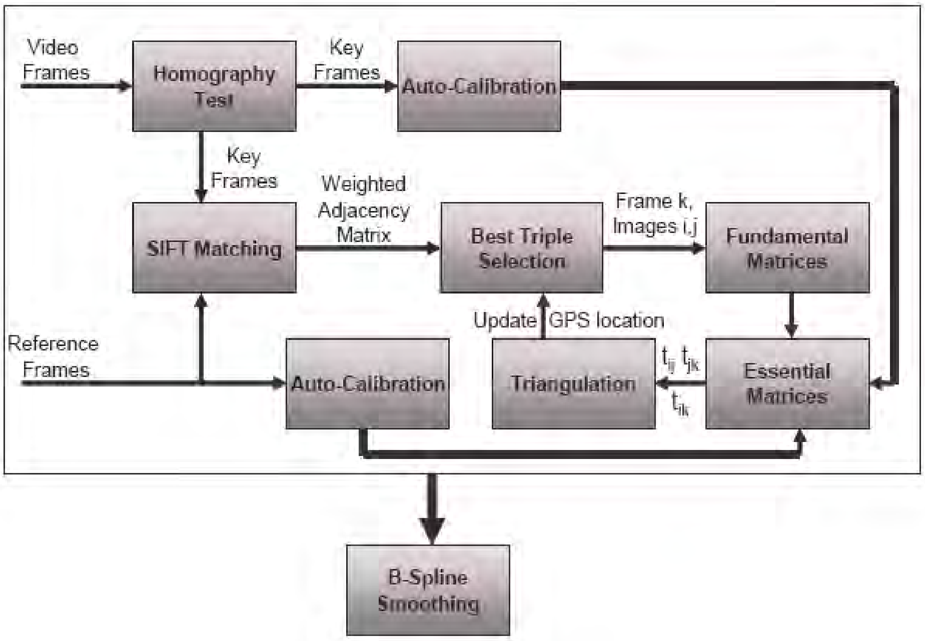


Figure 9.4: Overall geospatial video localization scheme.

The overall geospatial image localization scheme is given in Fig 9.4. Given a set of reference images we recover the camera intrinsic parameters using a method proposed by Luong and Faugeras [201]. Similarly, we recover the video cameras intrinsic parameters using the video frames. Next, we perform homography tests between the sequence of video frames to select key-frames from the video. This step reduces the computation time since the key-frames are those frames in the video that have significant camera translation between each other. Furthermore, SIFT features are detected and matched between the reference images and the video key-frames. Using the maximum matching reference images and video frames, we estimate the fundamental and essential matrices between image I_p and frame V_t pairs to recover the camera pose. Finally, we apply triangulation to recover scale ambiguity in the estimated camera pose and obtain the geospatial localization of the video key-frame. We repeat these steps for all video key-frames to recover the complete camera trajectory. The following sections detail these steps.

9.5.1 Localization of the Video Frames

Given a video frame V_t, we recover the position $PV_t = \{X_t, Y_t, Z_t\}$ of its camera optical center with respect to a reference image I_p in the world coordinate system using the similar scheme as described in the previous sections.

9.5.2 Trajectory Smoothing

The fundamental matrix estimates are highly sensitive to noise in feature correspondence. Thus, numerical errors, insufficient feature point correspondence and noise could result in incorrect estimation of GPS locations for the video frames. Performing multiple estimations of the fundamental matrix (using RANSAC), we obtain a spatial distribution of GPS locations for each video frame. If the point correspondence is reliable, the variance in the spatial distribution of GPS location will be minimal. Therefore, we discard GPS estimates of the video frames with high variance and reduce GPS estimation error for the video trajectory. Finally, we use the remaining GPS locations as control points on a b-spline and interpolate the rest of the trajectory by curve fitting.

9.6 Results and Discussion

Sequence	Frames	Keyframes	Precision
Public Affairs	405	42	6.54 meters
Engineering II	325	33	7.55 meters
Theater	645	61	6.37 meters
Health Center	1187	82	9.52 meters

Table 9.1: Mean localization error for different videos

The accuracy of our geospatial localization was tested against the ground truth GPS information obtained using a Garmin GPSMAP 76S unit that has an accuracy of 3 meters. We captured over 300 reference images (some examples shown in Fig.

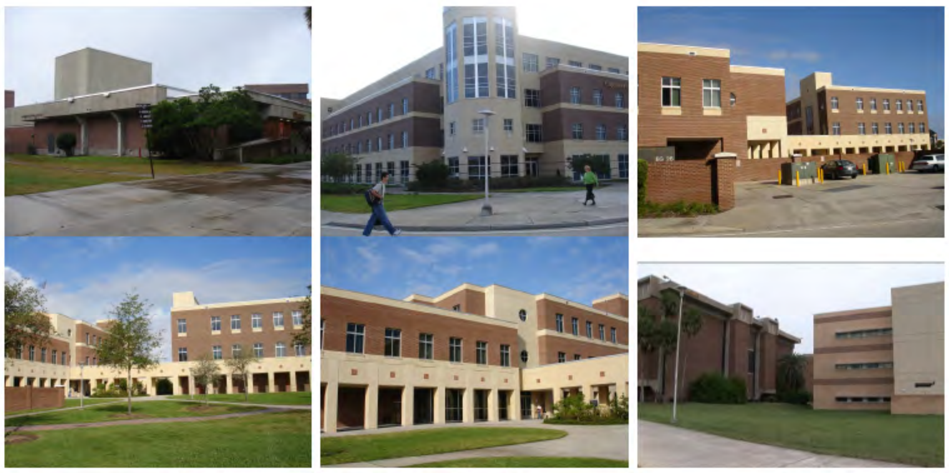


Figure 9.5: Examples of reference images with known GPS locations used in our experiments.

9.5) using a Nikon D2X camera at 4 MP (mega pixels per image) in various locations on our campus. The video sequences were captured using a Sony HDR FX1 camera at HD quality. We empirically evaluated our method to recover geospatial trajectory of a moving camera on four sequences totaling over 2500 video frames. The summary of results for the four sequences is given in Table 9.6.

Fig. 9.6 shows a video trajectory obtained and a comparison with the ground truth for the theater sequence. The average localization estimation error for this sequence is 2.77 meters with a standard deviation of 1.83 meters.

Fig. 9.7 shows a video trajectory obtained and a comparison with the ground truth for the health center sequence. The average localization estimation error for this sequence is 4.38 meters with a standard deviation of 3.91 meters. The reason for higher localization estimation error is due to the presence of trees and non-distinctive features (grass) during the middle of the sequence. Though the average estimation error for our method is about 4 meters, this sequence cannot be used for SFM based methods that rely on distinctive features throughout the sequence so that a 3D reconstruction and matching can be used for pose recovery and localization. This is the major advantage of our method and is more generally applicable to real sequences for geospatial localization. An empirical comparison of estimation errors for each key-frame (for the theater sequence) is also provided in Fig. 9.8.

Dataset	Knowns	Unknowns	Avg. Score	Avg. Est. Error
Test4	9	20	4.2	3.05 meters
Final5	16	22	3.5	6.08 meters

Table 9.2: Mean localization error for ICCV contest dataset

In order to test the robustness of our GPS estimation method, we also con-

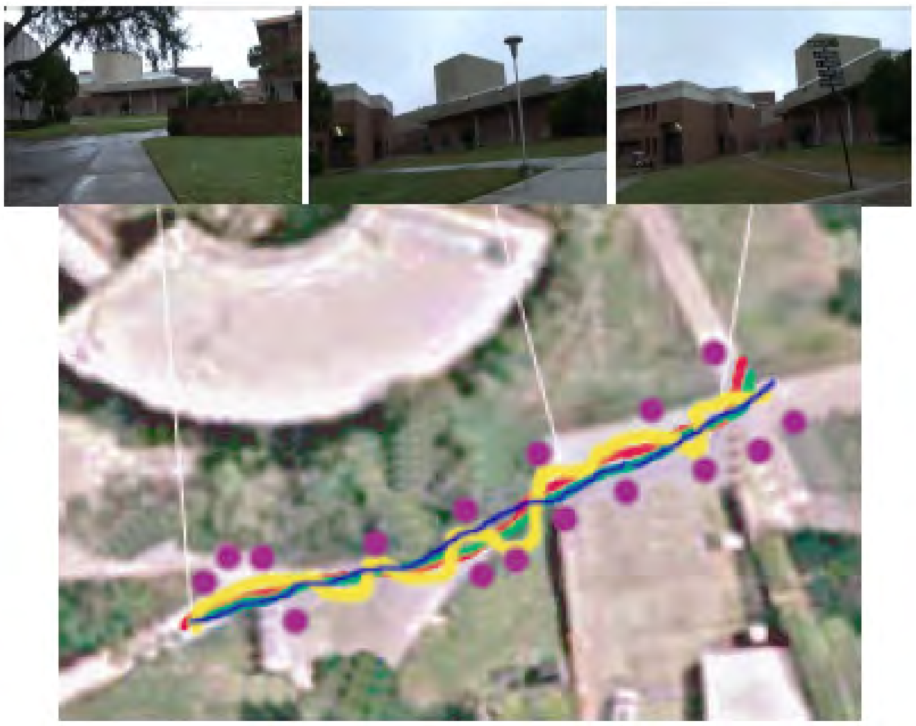


Figure 9.6: Video frames for the theater sequence. Bottom: Video trajectory (green) obtained using the temporal fundamental matrix based localization method is compared with ground truth trajectory (red). The trajectory obtained through the fundamental matrix localization is shown in blue (yellow line shows non-smooth trajectory). The location of the reference images used in localization are shown as purple dots.

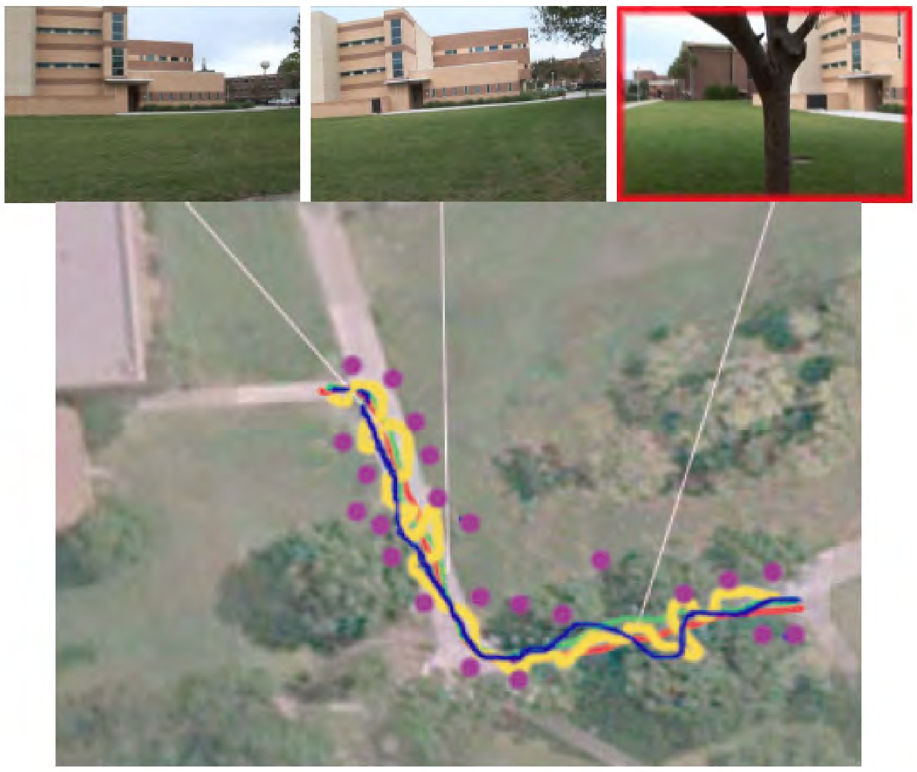


Figure 9.7: *Video frames for the health sequence. Bottom: Video trajectory (green) obtained using the temporal fundamental matrix based localization method is compared with ground truth trajectory (red). The trajectory obtained through the fundamental matrix localization is shown in blue (yellow line shows non-smooth trajectory). The location of the reference images used in localization are shown as purple dots.*

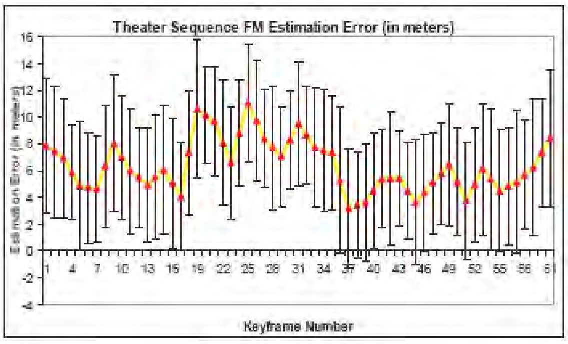


Figure 9.8: *Localization error for 40 runs of the Theater sequence; mean estimation error for each keyframe with standard deviation shown as top/bottom bars.*

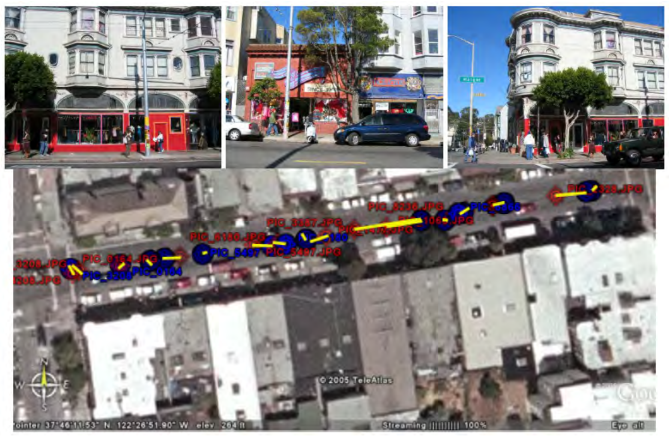


Figure 9.9: Images with unknown GPS location for Test4. Bottom: GPS locations (blue) obtained using our method is compared with ground truth GPS (red). The error in estimates are shown as yellow lines between the ground truth and estimated locations.

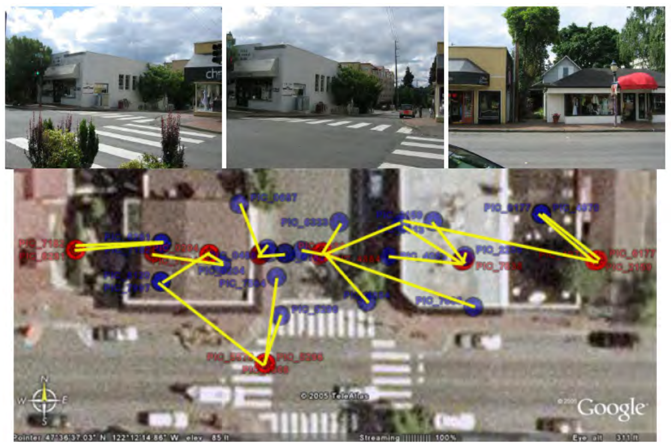


Figure 9.10: Images with unknown GPS location for Final5. Bottom: GPS locations (blue) obtained using our method is compared with ground truth GPS (red). The error in estimates are shown as yellow lines between the ground truth and estimated locations.

ducted experiments on the golden datasets (test4 and final5) of the ICCV Contest 2005. We estimate the GPS locations for each image with unknown GPS location using the best triplet and triangulation method by applying the fundamental matrix constraint (since video data is unavailable). The summary of results of our method is provided in Table 9.6. Also, a visual comparison of our results with the ground truth is given in Figures 9.9 and 9.9 for test4 and final5 respectively. As can be seen from the results, this method has an average estimation error of 3.05 meters and 6.08 meters for datasets test4 and final5 respectively. While the average score for test4 and final5 using the histogram of error method used in the ICCV Contest is 4.2 and 3.5 respectively. The best scores in the contest for these datasets were 5.0 and 3.5 respectively. This shows that our localization scheme with regular fundamental matrix performs well for estimating GPS locations for standard datasets.

9.7 Conclusions

This chapter proposed novel methods for estimating the geospatial trajectory of a moving camera. We used the video data and a set of reference images, captured from known GPS location, to recover the trajectory of the moving camera. What is novel in our approach is that we do not require a 3D reconstruction of the environment for recovering the camera trajectory. Rather we require a set of reference images with known GPS locations for geospatial localization of the novel video data. Also, we use sub-sampled video frames for localization and obtain a smooth camera trajectory. Our method has two advantages over existing methods. First, our method does not require all the video frames to have distinctive features for geospatial localization. Second, our method is computationally less expensive since we do not require 3D reconstruction and matching of the environment for localization. Results were presented on four real video sequences for geospatial localization using the proposed three methods, as well as, on the ICCV contest dataset.

Chapter 10

Sensor Integration

10.1 Introduction

Despite the efforts made by the researchers in developing a robust multi-camera vision system, computer vision algorithms have proved their limits to work in complex and cluttered environments [207]. These limits are mainly due to two classes of problems. The first is that "non-visible areas can not be processed by the system". This trivial statement is of particular importance in cluttered scenes and can be partially lessened by using multiple sensors (not only cameras). The second class of problems, instead, is due to the limited resolution of cameras. Having infinite resolution and zooming capabilities would make the job easier, but, in addition to be unfeasible, it would exponentially increase the computational load and it is typically too expensive.

An interesting solution is that of using simple but effective specialized sensors to solve the specific problems of the vision systems. In this way, vision would still provide high-level information, and low-level sensors would assure higher accuracy. In this context, the marriage between a widely distributed low-cost wireless sensor network and the coarsely distributed higher level of intelligence that can be exploited by computer vision systems may overcome many troubles in a complete tracking of large areas. For our application, we exploit passive Pyroelectric Infrared (PIR) sensors which are widely deployed in low-cost surveillance systems (e.g., for car or home alarm systems). PIR sensors are used in traditional surveillance systems to trigger video cameras [208]. A trigger just conveys a binary (yes/no) presence information, but limited signal processing effort on the output of a PIR sensor can produce much more information (e.g., target speed and direction of movement). Furthermore, integration of data from multiple networked PIR sensors can provide drastically improved spatial resolution in monitoring and tracking. Low-cost and low-power sensor nodes can now be developed with on-board processing capabilities, reconfigurability and wireless connectivity [209, 210].

This chapter reports our research in developing a multi-modal sensor network that integrates a wireless network of PIR-based sensors with a traditional vision system to provide drastically improved (in accuracy and robustness) tracking of people.

10.2 Related works

The emerging technology of multisensor data fusion has a wide range of applications, both in public and in private areas. Among them, surveillance and security have gained much interest in recent years, also due to the terroristic threats. Such applications often rely on cameras, due to the large amount of information and the high-level knowledge they provide. However, video surveillance in wide area through computer vision techniques is still a challenging problem largely faced in the last years [211].

Since cameras still have limitations due to the limited field of view or resolution, the proposal of this work is to combine data coming from cameras with information provided by a wireless sensor network based on PIR sensors, in order to improve the robustness of the system. PIR sensors are mainly known for their use in video surveillance systems, manufactured by a number of companies (e.g in [208, 212]), to detect motion and provide a trigger to cameras. For their insensitivity to light conditions, in [213] PIR sensors are used to provide a trigger event in a motion-detection application mainly based on cameras for tracking events at night, together with a floodlight. The appearance of an infrared radiating body set off the PIR sensor, which turns on the floodlight enabling the cameras to capture clearly an event such as animals passing by an outdoor detected area. Being low-cost, low-power and small form-factor devices, PIR sensors are suitable for wireless sensor networks, where energy consumption, unobtrusiveness and easy placement in the environment are critical requirements. In [214], a wireless PIR sensor network is used to detect objects and humans for security applications and provide an estimation of the direction of movements. The network is implemented using Mica2 [210] nodes and data gathered by a base station. Tracking algorithms are implemented on the nodes and speed calculation provided accurately, even if influenced by the orientation of the sensors. Sensor networks implemented with PIR devices are useful where privacy must be preserved together with security. In [215] cameras and PIR sensors are deployed respectively in public and private areas, and their information combined to correlate events such as tracking human motion and undesired access or presence in private areas, such as theft. This work demonstrate benefits of reducing camera deployment in favor of PIR sensors and reports results from a survey on 60 people, stating that people consider motion sensors less invasive for their privacy than cameras.

PIR sensors are often combined with vision systems and other kind of sensors in research focused on robot navigation and localization. In [216] a sound source localizer and a motion detector system are implemented on a human service robot called ISAC, with the purpose of redirect the attention of ISAC. The motion detector system use an infrared sensor array of five PIR sensors and it is integrated with the vision system of ISAC to perform real-time human tracking, in a most inexpensive way. Similar use of PIR sensor can be found in [217], where data from multiple PIR sensors, two color cameras and a pair of microwave sensors are collected, processed and fused to track human presence in the vicinity of a robot. A main motivation to use PIR sensors in pair with cameras is their insensitivity to lighting conditions, to avoid robot collision with humans for their safety in many different conditions.

10.3 Integrated Multi-Modal Sensor Network

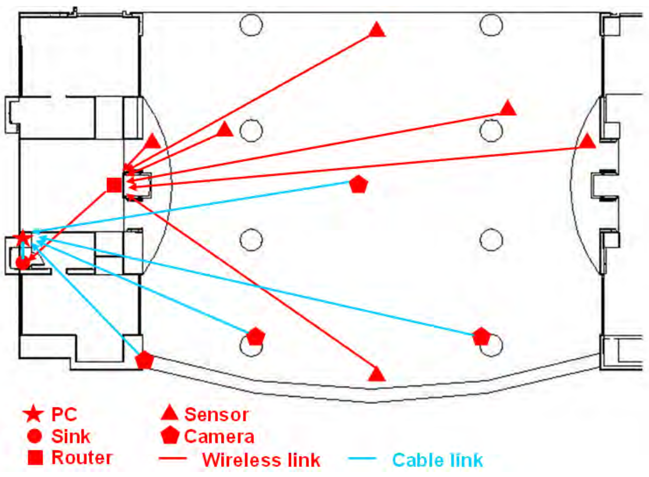


Figure 10.1: Map of our test bed system

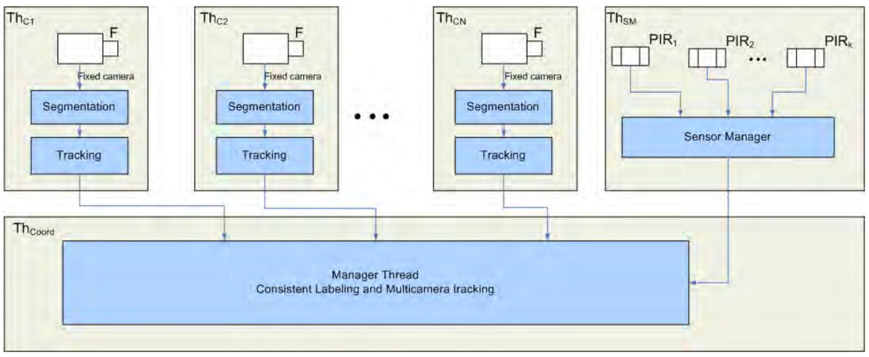


Figure 10.2: Software architecture of the system

Vision systems achieve good accuracy when working alone, but they definitely could benefit from the multi-modal integration with PIR sensors. For testing the integration, a test bed has been created at our campus. Fig. 10.1 shows the location of cameras and PIR sensors. The system we implemented is composed by several modules, working in parallel on different threads (see Fig. 10.2). In particular, a thread is generated for each camera, devoted to compute the list set of people present in the scene exploiting a two stage processing (segmentation and tracking). All the camera threads are tightly connected to a coordinator thread, that detects if the same person is visible in more than one camera, and, in such a situation, it labels the corresponding tracks with the same identifier.

At the same time, a sensor manager coordinates the network of sensors distributed over the monitored area. As explained in detail in Sec. 10.4, PIR sensors are able to detect the presence of moving objects or people within their coverage area. Observing the output of a couple of PIR sensors, the microcontroller integrated on the sensor node is able to detect both presence and direction of movement. When such a situation is detected the microcontroller creates and wirelessly sends a message to a special node which acts as a sink. The sink then forwards the message to the sensor manager via RS232 cable in order to make the information available to the tracking and labeling algorithms.

Eventually, data coming from cameras and sensors are collected and managed by a supervisor thread. The coordination between cameras and sensors is twofold. Each time the vision system requires more detailed or reliable information about the presence of people in the zones monitored by the sensors, it sends requests to the supervisor thread. Contemporaneously, whenever the sensor network detects a particular event, the manager takes care to inform the involved cameras. Detailed descriptions of the PIR sensor network, of the vision system, and of two examples of integration are reported in the following sections.

10.4 PIR sensor network

10.4.1 Sensor node architecture

Sensor nodes are more complex devices than simple sensors. A microcontroller, a transceiver and a power supply (mainly a battery) together with a variety of sensors form an entity capable of collecting events and information from the surrounding area, analysing them and sending message or data to other nodes or to the users (see Fig. 10.3).

Microprocessor and Transceiver. The hardware used to collect, process and send data from a PIR sensor is the SARD^{TM}Board provided by Freescale^{TM}to develop wireless applications over the worldwide free 2.4 GHz ISM band. This development board includes a microcontroller of the HCS08 family (MG9S08GT60), an MC13192 2.4GHz transceiver, a set of I/O pins and a RS232 port to interface the CPU with the external world and several leds and buttons. This development board is very flexible and many types of sensors can be connected to it. Furthermore, the microcontroller has the ability to operate in low power mode to save energy and to be waken up by external interrupts generated by the sensor output conditioning circuits. The transceiver is designed to be low-power, short range and fully compatible with the IEEE 802.15.4 standard. It communicates with the microcon-

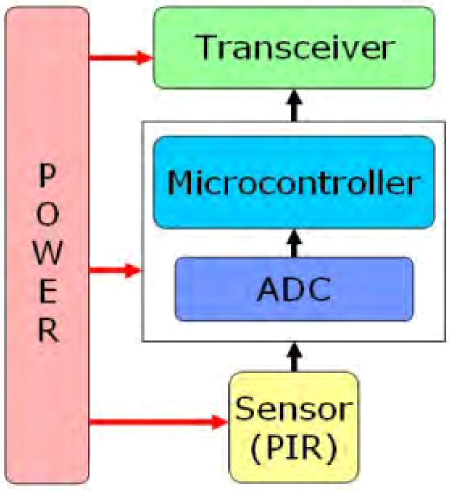


Figure 10.3: Sensor node architecture

troller via a four-wire SPI interface and an interrupt request output. The transceiver supports 250 kbps O-QPSK data in 16 5.0MHz channels and full spread-spectrum encode and decode. It enables communication with other devices up to distances of 40m outdoor.

Sensing element. The sensing element used in our application is a passive Pyroelectric InfraRed (PIR) sensor. Such devices are able to transduce incident infrared radiation into current. Commercial PIR sensors are sold in pairs with opposite polarization. Such a configuration makes the sensor able to detect variations of incident infrared radiation irradiated by bodies moving inside his coverage area which are not at thermal equilibrium with the environment. PIR sensors are used with Fresnel lens to enlarge and shape their area of sensitivity. They are passive sensors with minimal power consumption, ideal for battery powered systems. The PIR sensors used in this work exhibit high sensitivity and reliable performance, high stability to thermal change and high immunity to external noise. By suitably shading its Fresnel lenses we were able to obtain cone of coverage with a vertical angle of 60 degrees and an horizontal angle of 38 degrees.

Furthermore, a single PIR sensor can detect the direction of movement. Fig. 10.4 shows the signal detected by sensor when a person passes through the area under control from left to right (the first peak is negative) and from right to left (first peak is positive).

In our setup, the typical node includes two PIR sensors to enhance the information captured as explained in the following paragraph.

Power Supply. The system can be powered directly by main power or alternatively with a commercial battery at 9V. Internally, the voltage is stabilised at 3.3V.

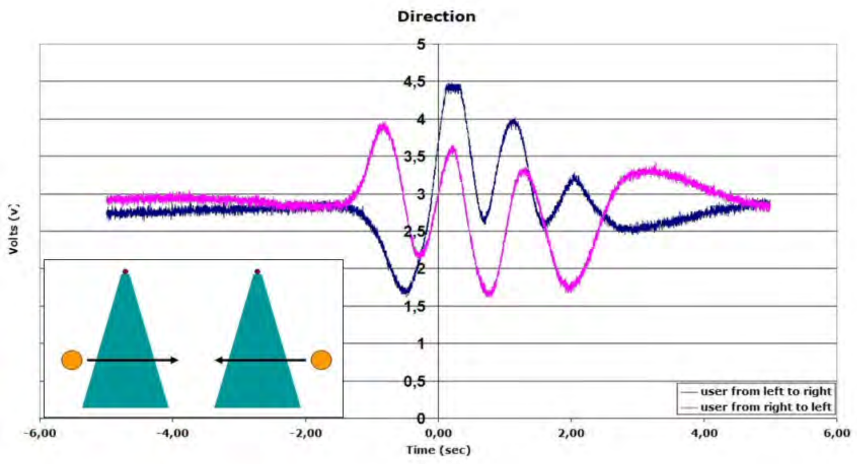


Figure 10.4: Signals detected by sensor: when a person passes through the area under control from left to right the first peak is negative and from right to left the first peak is positive

10.4.2 Sensing and Acquisition Software

Fig. 10.5 shows the processing data flow from event acquisition to generation of the packet which will be sent by the wireless node. This section describes the role of the different parts.

We are interested in precisely detecting presence and direction of movement, also in complex situations such as changes in direction within the covered area. In fact, these are information that can be exploited by the vision system for enhancing the accuracy of the video surveillance application, in which presence and direction of movement (of people) are key information.

As outlined above, we augment the information produced by a single node by using two PIR sensors (Fig. 10.6(a)) per node. The typical sensors' output when a person is walking through the sensor area is the one presented in Fig. 10.6(b). The signal collected by the sensors is digitally converted to be processed by the microcontroller. When a person crosses the monitored area each of the two sensors generates a waveform similar to the one in Fig. 10.4 depending on the direction of movement. We consider interesting events those stimulating a significant variation of the signal (Fig. 10.6(b)): when the input coming from the digital converter exceeds a lower or an upper threshold, a trigger is generated to start the processing algorithm in charge of extracting information from the signal. The analysis, as mentioned above, is aimed at understanding the direction of a person walking in the covered area. Assuming that one person is moving from left to right as in Fig. 10.6(a), he will be detected first by PIR_i then by both PIR_i and PIR_j and at last only by PIR_j as it is explained in Fig. 10.7. In general, a different activation sequence can help identifying changes in direction of movement within

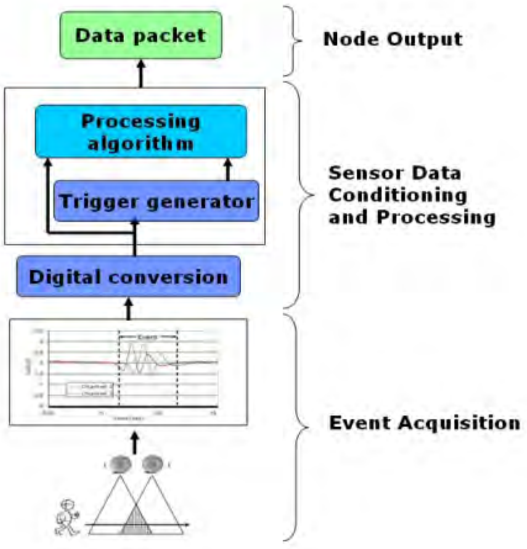
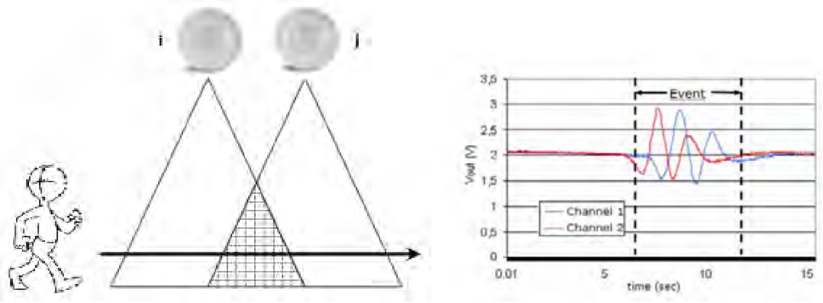


Figure 10.5: Event acquisition and Sensor Data Conditioning and Processing

the area covered by the array of sensors. Result from the processing is a message containing information about the presence and/or direction of movements in the selected area. The format of the packet is described in the following section.



(a) (b)

Figure 10.6: Sensor node composed by two PIR sensors

Note that the trigger generator is disabled for a period to be set depending on the application after the detection of an event, avoiding redundant information to be sent. In our case, the period is set to 2 seconds. This choice has been verified as not preventing a correct analysis, because it does not cause loss of events.

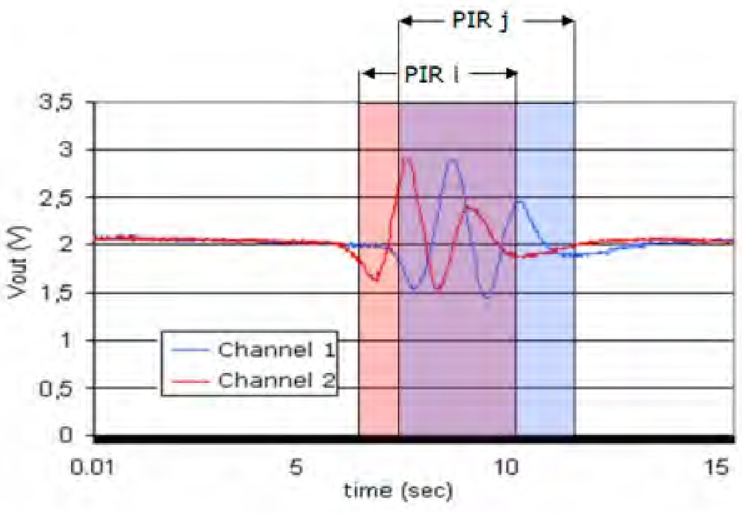


Figure 10.7: Activation sequence.

10.4.3 Network Architecture

Communication Protocol. Communication among nodes is based on IEEE 802.15.4 protocol for wireless networks. This protocol has been designed for applications with limited power budget and relaxed throughput requirements. Its main objectives are ease of installation, reliable data transfer, short range operation, extremely low cost and a reasonable battery life, while maintaining a simple and flexible protocol. This protocol defines two types of devices: full function device (FFD) and reduced function device (RFD). It also defines three possible roles for the nodes of the network: coordinator (which is unique for a network), router and end device. A coordinator and a router must be FFD while an end device may be either FFD or RFD. Usually routers and coordinators are main powered while end devices are located on the field and are battery powered. Two or more devices constitute a wireless personal area network (WPAN). However, one device must be a coordinator.

Network topology and organization. The network has a star topology, i.e., all the nodes are end devices and communicate only with a central one, the coordinator. The central node (bridge) collects data from the sensor nodes and sends them to another node (sink) which communicate to a PC through its RS232 interface (see Fig. 10.1). Hence, in our application the bridge is the network coordinator while the other nodes are end devices. The sensor nodes, which are located in the courtyard, are battery-powered while the bridge and the sink are main-powered. This topology is suitable to the characteristics of the monitored area. In fact, the sensors are located in a courtyard outside the building while the PC, due to privacy issues, is locked inside a small room, which must be kept closed within the build-

Information	Code	Values	Code
Presence	1	Present	1
		Area free	16
Direction	2	From PIR_i to PIR_j	48
		From PIR_j to PIR_i	192

Table 10.1: Examples of adopted codes

ing. Some tests shown that only the sensors close to the door of the building are able to communicate with a device inside the room, while all the courtyard can be covered by a receiver located close to the door.

Message format and set of commands. As already mentioned, the information collected by the sensors are sent to the video processing server via RS232 cable. We decided to use an asynchronous communication, that is, the sensor network send data to the server as soon as it collects them. The structure of the messages is shown in Fig. 10.8.

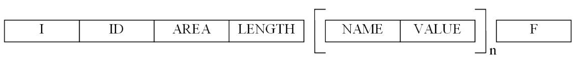


Figure 10.8: Communication protocol between nodes and sensor manager

Each message is made up of a start byte (the ASCII code 'I'), a sensor ID, an area ID, an indication of length (the number of following couples name-value), several couples name-value and a stop bit (the ASCII code 'F'). Start and stop bits are used for synchronization. Area and sensor node IDs are used to uniquely identify the node. The couple name-value encodes the information provided by the sensor. Some examples are reported in Table 10.1.

10.5 Examples of Multi-modal Integration

As stated above, the vision system achieves good accuracy when working alone, but it definitely could benefit from the multi-modal integration with PIR sensors. This section will report some results. To test the system we have equipped the atrium of our faculty with four cameras and several PIR sensors, as depicted in Fig. 10.1.

10.5.1 Sensor-guided Background Update

Algorithms of motion capture based on background subtraction rely on a very crucial task: the update of the background, especially in presence of illumination changes and moved objects inside the scene. For example, when the doors in Fig. 10.9 are opened, the background scene changes and the detection of people in that area becomes unreliable. To this aim, we use sensors to monitor the area near the

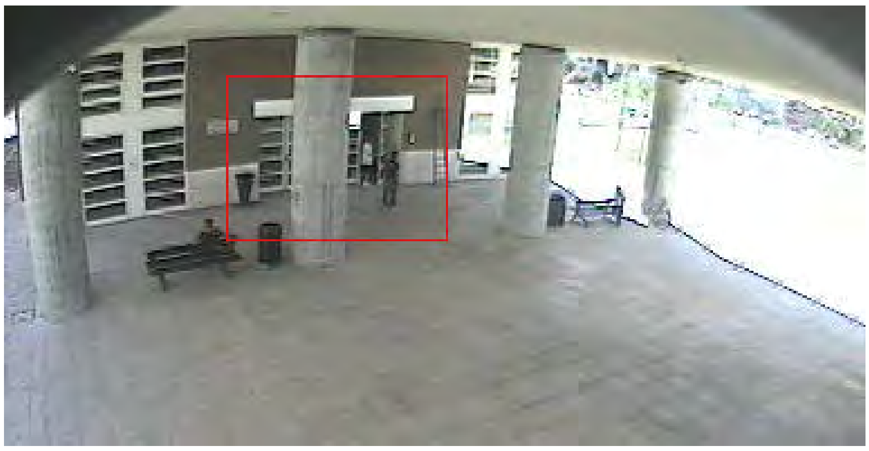


Figure 10.9: Opening and closeing doors make unreliable background suppression techniques

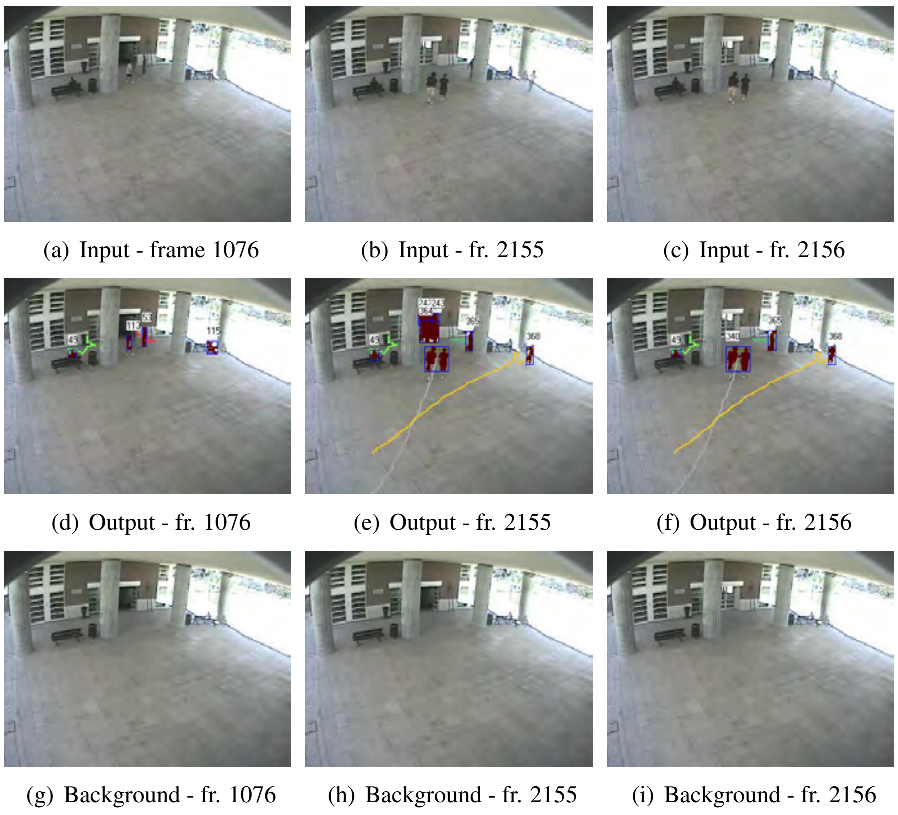
doors. If the single camera processing detects a visual object in the door area but the sensors do not capture events, then we assume that the motion is due to an incorrect background. In such a situation, the background is updated by forcing the area covered by the sensor directly with the input image.

More generally, each tracking system analyzes its list of detected objects. If an object is still for a long time, then the correspondent camera thread makes a request to the manager specifying the object location. The manager searches if the concerned zone is covered by a sensor and, in such a situation, it responds with the relative state. If the computer vision and the sensor network are discordant, then the sensor is considered more reliable, and the vision system reacts consequently, for example updating the background.

In Fig. 10.10 some frames from a single camera capturing the entrance of our faculty are shown. The rows report, from top to bottom, the input frame, the output of the tracking system, and the background model. Initially (first column) the door was open. Some frames later a person has closed the door and from this instant the background becomes inconsistent. In fact, the system erroneously detects the presence of a person in the area of the door (see Fig. 10.10(e)). When the PIR sensor placed near the door does not capture any events, the background is correctly updated (last column).

10.5.2 Detection of Direction Changes during Occlusion

Occlusions are another problem that characterize video-surveillance systems based on computer vision; for example, in the environments of Fig. 10.9, people can walk behind the columns, and, in such a situations, the system is likely to lose them. To face this problem, we have introduced some rules inside the tracking system. When a track disappears, it is not deleted immediately, but its appearance


(a) Input - frame 1076 (b) Input - fr. 2155 (c) Input - fr. 2156

(d) Output - fr. 1076 (e) Output - fr. 2155 (f) Output - fr. 2156

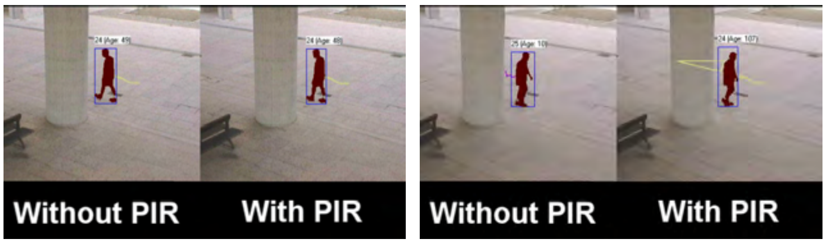
(g) Background - fr. 1076 (h) Background - fr. 2155 (i) Background - fr. 2156

Figure 10.10: Sensor-guided background update.

is kept unchanged and an estimation of the track position is computed exploiting a constant velocity assumption. If the person returns visible again with a similar appearance and a position near to the predicted one, then the system assigns the same label of the disappeared track. However, if the person changes direction during the occlusion, the system is not able to correctly assign the label anymore.

For this reason, we exploit a PIR sensor node placed behind the column. As above mentioned, these sensors detect not only the presence of a person, but also his direction. Then, we can detect a change of direction capturing couples of opposite direction events sent in a short temporal window. In such a situation, the direction of the motion applied to the track is inverted in order to estimate the position frame by frame.

In Fig. 10.11 an example of consistent labeling after an occlusion is reported. The person walks behind a column and, during the occlusion, inverts his direction. The computer vision tracking algorithm is not able to solve the consistent labeling because the person appears again too far with respect to the predicted position (computed with a constant velocity assumption). Using PIR sensors, instead, the change of direction is detected and the estimated track position can be properly updated. Then, when the person appears again, the tracker assigns the same label



(a) Before occlusion (b) After occlusion

Figure 10.11: Consistent labelling after an occlusion exploiting a PIR sensor node to detect direction changes

(24) assigned before the occlusion (see Fig. 10.11(b)).

Differently from the previous example, in this case the sensor network detects an event and the manager thread informs the involved cameras of it. Then, if a tracking system has detected an object in the corresponding position, the motion direction is changed accordingly.

10.6 Conclusions

Distributed surveillance is a challenging task for which a robust solution, working on a 7/24 basis, is still missing. This chapter is meant to propose an innovative solution that integrates cameras and PIR (Passive InfraRed) sensors. The proposed multi-modal sensor network exploits simple outputs from the PIR sensor nodes (detecting the presence and the direction of movements of people in the scene) to improve the accuracy of the vision subsystem.

Two case studies are reported. In the first, the vision system, based on background suppression, fails due to a door that is opened. Since background is not immediately updated, the door is detected as a moving object (resolution is not sufficient to enable a correct motion detection). In this case, a PIR sensor is used to discriminate between the opened door and a real moving person. In the second case study, a person changes its direction when it is occluded by a column. The vision tracking algorithm relies on the constancy of the speed during occlusions and thus fails. A pair of PIR sensors are, instead, used to detect the change in direction and alerting the vision system.

The reported results demonstrate that using the integration between PIR sensors and cameras the accuracy can significantly be increased.

Chapter 11

Semantic Video Transcoding

11.1 Introduction

In this chapter we present a framework for remote surveillance. The proposed system aims to keep under control an environment using computer vision techniques to generate a compact representation of the scene and virtually reconstructing it on mobile devices as final result.

Remote surveillance on mobile devices is becoming a wide market demand in many contexts. First, centralized control centers for visual surveillance have management costs much higher than a network of distributed and mobile control points. Centralized surveillance also has some limits in terms of efficacy, since people employed for watching tens of monitors and videos coming from hundreds of cameras cannot keep their attention on all the controlled scenes. Therefore, there is a high demand of connections between control centers and distributed mobile platforms to send in real-time surveillance data, images and videos.

In the last years the technology of robust video streaming on mobile devices has been improved but sometimes it is unfeasible or cannot be provided with an acceptable quality due to the unavailability of the connection or the lack of transmission stability. Moreover, the current standards GPRS or UMTS frequently insert a not negligible delay in streaming transmission, so that surveillance images cannot be received in real-time everywhere and every time.

A possible solution is the intra-media transcoding of visual data to textual information with a manual or automatic extraction of surveillance knowledge from the videos. Examples are the detection of people, moving objects, or suspicious abandoned packs in the scene, the counting of how many people are moving, the estimation of their position and their behavior. The textual information can be easily transmitted in real-time to mobile devices, that can show them in a textual form or with a graphical interface, e.g. virtually reconstructing the scene. The goal is the definition of a virtual environment corresponding to the real controlled one where the useful surveillance data are kept. In such a manner the textual surveillance knowledge can be transmitted in an affordable and robust way to mobile devices where the visual information can be reconstructed. Geometric 3D data on the background scene are pre-loaded in the mobile device and only the dynamic information is transmitted in real time.

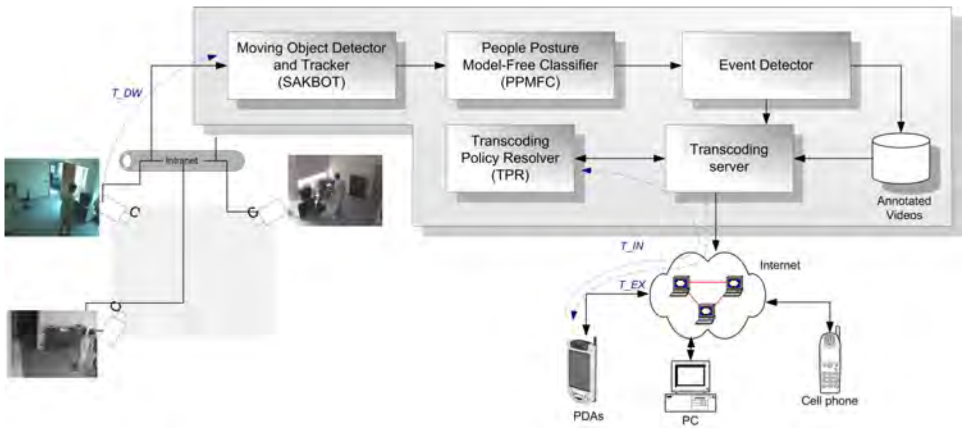

Moving Object Detector and Tracker (SAKBOT)
People Posture Model-Free Classifier (PPMFC)
Event Detector
T_DW
Intranet
Transcoding Policy Resolver (TPR)
Transcoding server
Annotated Videos
T_IN
T_EX
Internet
PDAs
PC
Cell phone

Figure 11.1: Scheme of the overall architecture.

In the proposed architecture of remote surveillance platforms (see Fig. 11.1), the video streams are processed in real-time by local servers, which extract the surveillance knowledge on the environment, provide a semantic transcoding of the visual data and make the data available for mobile connection.

The semantic transcoding can be provided in intra-media modality as well as the inter-media modality above described. In the first case videos are processed and modified in order to provide compression rates that are acceptable in remote mobile connections without loosing important data. Differently from the inter-mdeia modality the output of the transcoding system is still a video stream and not a textual information. As in [218, 219] background and moving objects can be coded at different compression rates, sent separately and reconstructed at the client-side. Instead, in the intra-media modality the scene is virtually reconstructed with computer graphic techniques, allowing the transmission of few textual data only. This second modality is less realistic but has several peculiar advantages. Firstly, the 3D scene is reconstructed so that all 3D information is available, new views can be provided, different from the field of view of the camera and a virtual interactive navigation is allowed. Secondly, there is the possibility to filter out some critical data that should not be transmitted for privacy issues. For instance, current laws of many countries do not allow the use of surveillance data in commercial sites such as offices or super-markets. With a virtual reconstruction the privacy protected data (e.g., the face) can be eliminated. Therefore we reconstruct the presence of people in an environment, their motion, their trajectory, and their possible interactions without any individual or biometric information. In this way, security employers equipped with mobile devices can manage the multimedia information in real-time interacting with the environment and in a total respect of privacy issues. In this chapter both the inter-media and the intra-media modalities are presented and discussed. We assume to have a computer vision module able to extract a list of objects of interest. As described in Chapter 5, each object O_j detected in the time t has a set of temporal attributes: position $(x(t), y(t))$ in the ground plane; the bounding box BB in the image plane, appearance image $A(t)$, that is the color aspect, and so on. The objects recognized as a person inherit some additional attributes, that are:

$$\begin{cases} Status(t) \in \{\textit{moving, still}\} \\ Posture(t) \in \{\textit{standing, sitting, crawling, laying}\}. \end{cases} \tag{11.1}$$

This extracted information are annotated in a text file, that is the basis of a standard annotation for stored video server that uses MPEG7 standard to keep information for historic and content-based search. The textual data can be downloaded, transmitted in a streaming manner over standard http-based connections to mobile clients, or used by the transcoding server to generate semantically compressed video streams.

11.2 Inter-media transcoding

11.2.1 3D virtual environment

In this section we directly present an application which consists of video surveillance of an indoor environment. First of all, the 3D virtual indoor environment has been built by using the JSR 184 software environments.

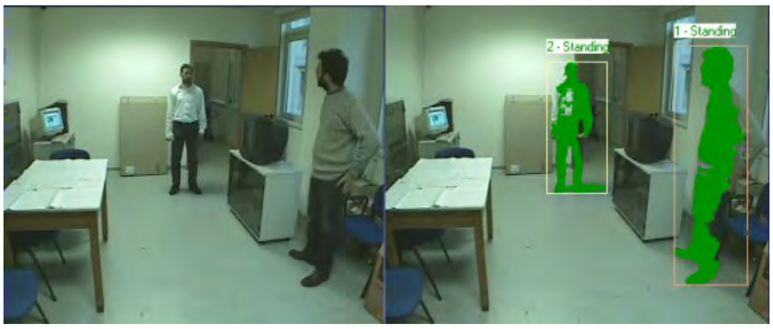


Figure 11.2: Input frame from a monitored room (left) and the output of the video surveillance module (right) corresponding to the virtual reconstruction of Fig. 11.4. Blobs, positions, and postures of the detected people are super-imposed.

A frame of the reference room used in our experiments is reported in Fig. 11.2. The background model is a 3D model made of mesh objects that refer to static elements presents in the scene: table, chairs, cabinet with TV and pavement. For representing the scene we used two ambient lights sources (instances of the ambient node class) in order to obtain a photo realistic rendering of the indoor environment. The human figures are instances of the correspondent scene graph node, and include simple human textures. In fact a photo realistic representation of the human figures is less crucial (in this application case study) than the actions taking places in the virtual environment. The application takes as input a textual data stream coming from the video surveillance extraction module. These data stream contains the extracted positions and postures of the tracked human figures. The 3D virtual environment reconstruction module takes these data and renders the human figures placing them in the environment according to their position and posture data.

The rendered postures are taken as classes from the extraction module, in the presented system four major classes are considered: standing, crawling, laying and sitting. In order to enhance the system performances the animations are executed at human figures positioning level and not on the models themselves. The system is already capable of implementing animations interpolations as already written in the past paragraph, but it is seems more reasonable to use them for other kind of domains like children surveillance (where the running speed could be a measure of the actual danger and it's crucial to visually represent that data). The 3D virtual environment also has many advantages, especially the possibility of changing the point of view. We support three different points of view:

- *Standard view*: this view is basically the same as the calibrated cameras positioned in the real environment. It is very useful in order to have a general view of the scene.

- *Bird eye view*: this view is very suitable for identifying the distances among the human figures and the objects in the scene. Spatial positioning is an important synthetic information to be used as a visual clue for detecting which actions are taking place in the environment.

- *Interactive view*: this view, not only presents the scene with a certain angle (usually the same as the standard view) but it is able to navigate the scene by using three different camera motion modalities. These modalities are: move (allow translations with respect to z and x planes), rotate (rotation among all the planes), and float (allow translations with respect to y and x planes).

The 3D virtual environment thus could represent and effective approach for easily visualize and detect people and action in a video-surveilled environment. The use of multiple synthetic points of view clearly helps users in identifying different situations and by different angles of view. Since our system is extensible, many different points of view could be implemented, for instance for certain domains, a first person view could be useful for understanding the danger or damages occurred in the environment (elderly people surveillance, and care).

11.3 Software components

The software components used in our approach are based on M3G (Mobile 3D Graphics) library. This library consists of represents a high-level API (Application Programming Interface) for the implementation of graphics rendering on mobile device. This choice leads us to develop a system in which the 3D scene rendering is carried out by on client side (mobile devices).

The mobile 3D graphics libraries used in our system are included in the JSR 184 standard specification, thus enabling the support by a wide set of devices; moreover this is a general approach because of the standard adoption of the M3G file format specification.

The M3G file format represents all the objects present inside a three-dimensional scene through the use of a tree structure called scene graph. Every node of this structure describes and defines any physical or abstract object of three-dimensional worlds (cameras, lights, meshes, animations).

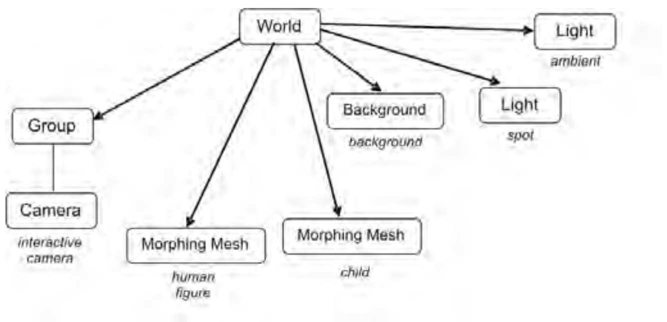


Figure 11.3: Tree structure of the 3D virtual environment

The root of this tree is always a World object. Fig 11.3 shows a reduced version of the tree structure used for building the presented 3D virtual environment.

In order to simplify the diagram, in the above scene graph we have not inserted the Mesh objects that refer to static elements presents in the scene: table, chairs, cabinet with TV and pavement. In our scene model two different classes of lights are defined: spot and ambient. The Background object is represented by a background node with a colour attribute in the format RGBA (Red, Green, Blue, and Alpha for transparencies). The interactive camera is a child of a Group object. This object has the task of managing the camera translation and rotation on the y axis. The camera object manages rotation on the x axis. With this separation all the changes on x axis don't have effect on camera movements and rotation on y axis.

The more complex objects are the two MorphinMesh that contain all necessary data for the visualization and animation of classes of human figures (e.g., men, children). The MorphingMesh class allows to define models animated through morphing techniques: it requires to define the key models of an animation and an automatic interpolation procedure will compute shapes vertexes transformations needed for a smooth animation. By using the M3G file format and supporting the JSA 184 standards we noticed that the application jar file including models images and textures, testing data and source code does not exceed the 90kb.

More in detail M3G is a new standard file format, with extension m3g, used for storing all the scene graph data and the information for loading them in a custom application. In this manner the data of a scene, comprised the animations, can be created using common existing three-dimensional modelling programs. However, there is not a standard approach for creating and exporting this kind of files. Thus we decided to use a particular technique to import models in obj (wide spread computer graphics format) data type.

Our approach for creating M3G files is inspired by Andrew Davison work [220], which is based on the Java3D API. The approach consists of converting three-dimensional models in a Shape3D object from which is possible extract the vertexes, normal vectors and coordinates of texture lists. These lists are then optimized through the use of triangle strip. Finally a class is generated containing all the necessary methods to create and manage (starting from the acquired lists) a three-dimensional object in M3G.

11.4 Experiments

We provide several experiments in both indoor and outdoor environments. The real-time performance of the served-side depends on the number of cameras, the number of people and their size in the image space. In outdoor environments with cameras mounted in a high position tens of people can be captured from 4 cameras at about 10fps. The same performance is achieved indoor with 2 or 3 people as in the previous examples.

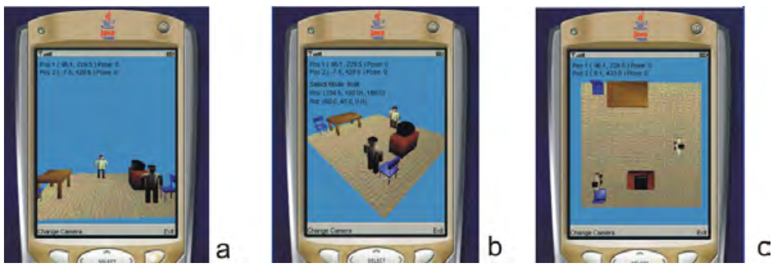


Figure 11.4: Example of different views of a3D virtual environment. a) standard view, b) interactive view, c) bird-eye view.

An example of the output of the indoor system is reported in Fig. 11.4, where the three different views (standard view, interactive view, and bird-eye view) corresponding to the input frame of Fig. 11.2 are shown.

11.5 Intra-media Transcoding

11.5.1 Introduction

In this section a smart video server, implementing new semantic transcoding techniques to connect directly with PDA clients, is described. The transcoding model, using the philosophy of MPEG-4 object-based compression, allows video streaming of part of the videos that are semantically valuable to the user: it compresses differently objects and event and operates also a temporal and size downscaling.

11.5.2 Related Works

Semantic transcoding is often employed in streaming severs to adapt the multimedia content and, specifically, reformat video code, in order to cope with user needs and user constraints [221]. Typically, variable compression, size, and color downscaling are useful to save bandwidth and deal with limited display resource of devices such as PDAs. The exploitation of semantics at this level means to use the knowledge of the video content to compress the video differently in the space and in the time. The MPEG-4 standard was the first structured proposal to handle differently background and foreground objects in the scene, to compress them and to send to the user for a specific encoding. At the same time, MPEG-4 Core Profile (needed for object functionalities) codec is computationally heavy and a real time decode and encode phase can now be achieved with hardware accelerator only. Thus, we proposed some simplified version that can be exploited also with software codec only [222], and in this section we describe a solution specially conceived for PDA clients. The valuable contribute of this proposal is to define a semantic transcoding server operating on-the-fly in cascade with the computer vision system. In such a manner the knowledge of segmented people and objects guides the adaptive compression. In [223], a similar approach for traffic surveillance is proposed, where objects are segmented and compressed differently in MPEG-4 standard for an offline annotated video server. Further, in [224] not only an object transcoding is discussed, but also the semantics is exploited at level of both objects and events. This unified framework is here presented in a domotics context and tailored for PDA.

11.5.3 System Architecture

The system is structured as a client-server architecture as in Fig. 11.1. The server side contains several pipelined modules: in domotics video surveillance, the motion is a key aspect and, thus, object detection and motion analysis is embodied in the first module. The output of this module is the set of the moving visual objects along with their features (shape, area, color distribution, average motion vector, and so on). These objects are tracked along time and processed to firstly classify them; the objects classified as people are further processed to detect their posture in the second module and, from it, to identify a given event. Events are modeled as a transition between two states of a Finite State Machine representing the posture of the person. Thus, the event “falling down” is modeled as the transition between “standing” (or “sitting”) state and the “lying down” state. The next step of semantic video transcoding is independent from the implementation of previous modules. The TPR (Transcoding Policy Resolver) input is a set of *classes of relevance* (defined as couples <object,event>, as detailed in the following) and the associated weights that define the relevance of each class (see Fig. 11.1). These information are processed by the Transcoding Server to apply selectively transcoding policies depending on the current event and on the objects in the frame.

In conclusion, what the server side sends in the network is a MJPEG-like stream of data in which the background and the moving objects are sent in a compressed, proprietary format, described in the following. At the client side, we developed a software for Pocket PC 2002 operating system working on a Compaq iPaq H3850 PDA able to interpret this stream and to re-compose the scene in an efficient way. The modularity of the proposed architecture allows quite easily to change the system to adapt to the user's requirements by adding/replacing the algorithms.

11.5.4 Transcoding Server and PDA Client Application

As depicted in Fig. 11.1, we developed a client-server architecture. To this aim, we implemented a multi-client and multi-threaded transcoding video server called *VSTServer* (Video Streaming Transcoding Server). Among the different threads present in the server, three are critical: the first downloading thread (*T_DW*) is devoted to acquire sequence of images from the network camera in streaming mode. The second inquiring thread (*T_IN*) establishes the communication between client and server and sets the transcoding policies. Whenever the initial parameters (requests of size, bandwidth, etc.) are set, the connection between client and server is passed to a third execution thread (*T_EX*). From this moment, another client can connect to the server. The threads are decoupled to allow the maximum frame rate in getting the image from the camera, despite the possible slowdowns due to slow clients. The communication between the two threads is based on shared buffers (in which the *T_DW* puts the image and from which the *T_EX* picks it up), with a semaphores based protocol to obtain the synchronization between the two threads.

If we know (from the user or automatically) which is the relevant semantics in the video context, we can exploit it to selectively transcode the video: the bandwidth saved by degrading the not relevant contents can be used to increase the quality of the relevant contents.

What we used to quantify the importance of the semantics are the *classes of relevance*. A *class of relevance* is defined as the set of meaningful elements in which the user is interested in and that the system is able to manage. Formally, a *class of relevance* C is defined as a pair $C =< oc_i, e_j >$, where oc_i represents an object class and e_j is an event class, selected between the set of object classes OC and event classes E detectable by the system:

$$OC = \{oc_1, oc_2, ..., oc_n\} \cup \{\widetilde{oc}\} \quad ; \quad E = \{e_1, e_2, ..., e_m\} \cup \{\widetilde{e}\}$$

The special class $\widetilde{oc}$ includes all the areas of the image that do not belong to user-defined object classes (for example, the background is considered as $\widetilde{oc}$). Analogously, the event $\widetilde{e}$ includes all the non interesting events or the case of no-event.

The user can then associate the set of weights w_i to each class of the set C: higher the weight, more relevant must be considered the class and the TPR will apply a less aggressive set of transcoding policies.

Thus, in the simplest case we can consider only $oc = \{people, background\}$ and, without taking into account events, we can treat object classes differently compressing them in different ways. Moreover, to allow the re-composition of the scene (background plus superimposed people tracks) at the client side, also the alpha planes (i.e., the binary mask describing the blob of the person) are sent to the client. To reduce the bandwidth occupation, the alpha planes are compressed with the lossless Run Length Encoding (RLE) coding. Summarizing, the server produces and sends to the client a stream built as in Fig. 11.5. The stream is embodied in an HTTP connection with a multi part MIME header and consists of sequences of JPEGs at different compression, preceded in the case of objects by their identities and RLE information.

```
--myboundary\r\n
Content-Type: bkg\r\n
Content-Length: <size JPEG file>\r\n
\r\n
<JPEG image data>\r\n\r\n
--myboundary\r\n
Content-Type: voi<number of VOs>\r\n
<RLE length>;<JPEG length>:<VO ID>\r\n
<RLE data><jpeg data>\r\n
\r\n
<RLE length>;<JPEG length>:<VO ID>\r\n
<RLE data><jpeg data>\r\n
\r\n
...
--myboundary\r\n
Content-Type: bkg\r\n
Content-Length: <size JPEG file>\r\n
\r\n
<JPEG image data>\r\n\r\n
...
```

Figure 11.5: Example of HTTP stream produced and sent from the server to the client

Thus, the background and the VOs are sent separately and with the syntax above reported. The VO identity is sent for another functionality of our transcoding system that keeps the people track with that identity with the best quality and crops it from the image [224].

At the client side, this stream is decoded, background and VO are superimposed, and the resulting video is visualized on the PDA by applying further scaling and general adaptation to the PDA capabilities.

More in general, not only background and VOs can be compressed differently, but both objects and events contribute in the selection of the best transcoding policy, as in the test that will be discussed in the next section.

11.6 Experimental Results

The architecture we proposed is composed by several pipelined modules and a complete performance evaluation should detail performances achieved at each stage. A first performance analysis is oriented to measure the efficiency in terms of reactivity and processing speed. *SAKBOT* upgrades the background model with n frames extracted with a Δt subsampling. If the system is directly connected to a standard camera (25 fps), the normal parameters we adopted are $\Delta t = 10$ and $n = 7$ and $w_b = 2$ in Eq. 2. Thus a changed value in background pixel due to the median function affects the background model at most after $\frac{n+w_b}{2}\Delta t$ frames and thus after about 2 seconds. Instead the *Timeout* parameter (Eq. 1) is set sufficiently high to avoid that misdetection errors cause significant objects to be included into the background model: typical values are $k\Delta t$ with $k = 10$. Therefore a little change in luminance or small background modification is captured with a delay of $2s$ while a strong change (as a moved chair) is captured normally after $4s$. If the system is connected to a network camera these times are changed since the video streams is affected by possible network bottlenecks in the *T_DW* thread; we tested that these delays are negligible in Intranets based on Ethernet or Wi-Fi connection.

Then *Sakbot* provides background suppression, segmentation, shadow removal, VO feature computation and people classification at each frame and the speed is proportional to the number and size of VOs extracted. In the common case of a single person in the center of the scene (occupying about 10% of the frame), *SAKBOT* is able to work at a speed ranging between 10 and 20 fps. The people posture classification do not introduce any sensible delay in performance. At the end the system can change the event status at least every $100ms$ and detects an alarm situation with this reaction time (actually although an alarm –person lying down– can be set after user defined delay).

Finally, the time performance depend on the network bandwidth and the client speed. The client application, called *SeeImage*, on Compaq iPAQ PDA, that has to decode the object based stream, is written in Embedded C++ for Windows CE and is sufficiently quick not to introduce delays (in average, depending on the number of objects) thanks to the multi-threaded application. Directly connecting the client to the LAN, using the USB interface of the PDA, allows to obtain between 10-18 fps at a QCIF size when interesting events are detected. This numbers decrease considerably when a low bandwidth connection is available, such as GPRS. In this case, even if semantic transcoding is performed, on average 5-8 fps are obtained for simple scenes with a single person in the scene. In the video we report as test (see Fig.11.6) with semantic transcoding only, we achieve an average bandwidth of 88 kbps. With GPRS (about 55 kbps) we must add a temporal down scaling and thus we can reach about 8 fps, that is acceptable to see the scene fluently.

Another critical point is the well known problem of evaluating performances in terms of quality and precision. In general we could consider performance at two levels:

- **perceptual** level, meaning that we should compare the output of processed videos with the original video in terms of pixels processed or transferred;
- **cognitive** level to compare the results of the video-surveillance system in terms of detected objects or events, with reference to the possible similar results of an human person controlling the scene.

The results of the initial moving visual object detector module can be evaluated at both levels. Perceptual analysis refers to the count of correctly and mistakenly segmented pixels as VOs, shadows, ghosts [1]. Cognitive level performance counts the correct visual objects that are detected and tracked frame by frame. In standard situations, testing *Sakbot* on hours of videos, all the interesting objects have been detected on the 95% of the frames, and no objects were lost due to the tracking algorithm.

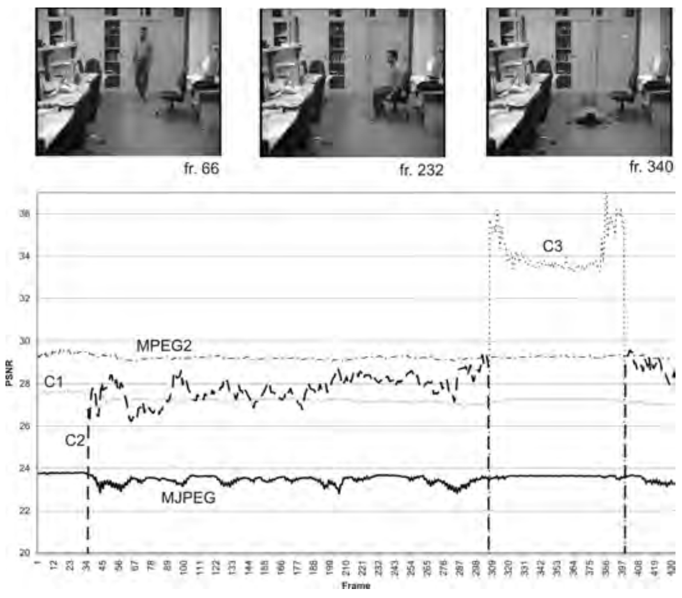


Figure 11.6: PSNR results for each frame, divided into classes and comparison with the other algorithms

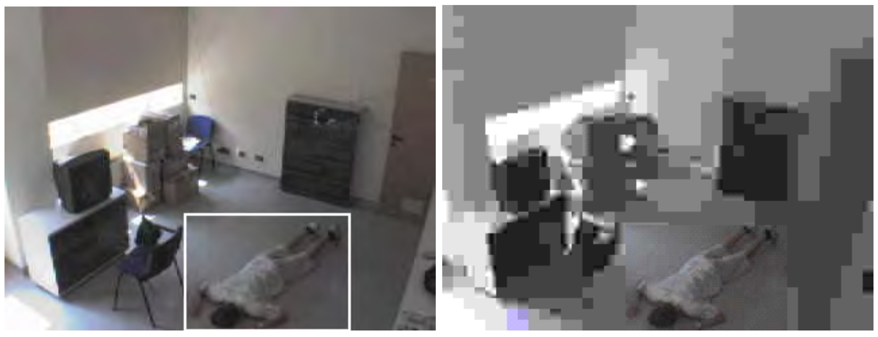
For this indoor transcoding results, we set the following classes:

$$OC = \{person, chair, door\} \cup \{\widetilde{oc}\}$$
$$E = \{no\_motion, O\_moving, P\_lying\} \cup \{\widetilde{e}\}$$

A remote client could be interested to see everything that is moving, or as in our application only in people lying on the floor. This three classes can be defined:

$$C_1 = \{chair, *\}|\{door, *\}|\{\widetilde{o}, *\}|\{*, \widetilde{e}\}$$
$$C_2 = \{person, \neg P\_lying\}$$
$$C_3 = \{person, P\_lying\}$$

(∗ is the wildcard) and a user could set a low, medium, and high interest for the three classes respectively, setting for instance the compression factors to 10, 20 and 90. In the results of Fig.11.6 it is possible to see the performance of our system on the three classes compared to MJPEG and MPEG2 (that we have available also on the PDA), on a frame by frame basis. With respect to MJPEG we can achieve better results, by the use of semantics (seldom sending of the background). On average, the results are comparable to MPEG2, but when a person falls down (frame 307 to 399), we can exploit the larger bandwidth available (because we are sending smaller images) to boost the resolution to very high quality.



(a) Bounding box detection (b) Semantic transcoding

Figure 11.7: Example of content-based video adaptation.

An example of content-based video adaptation is provided in 11.7 where the event "fall" calls for a better quality transmission of the frame region containing the fallen person.

11.7 Conclusions

Our end is the remote monitoring of the behavior of people moving in a scene exploiting a virtual reconstruction on low capabilities devices, like PDAs and cell phones. The main novelty of this system is the effective integration of the computer vision and computer graphics modules. The automatic video surveillance module extracts sufficient information to allow a virtual reconstruction of the environment on low capabilities devices, and, differently than a video stream, the bandwidth required to transmit this data is affordable and the system is working in real time.

Chapter 12

Conclusions

12.1 Achieved Objectives

This thesis is the result of three years of my Ph.D. studies. The main objective was the study, analysis, proposal and development of architectures, models and algorithms for people video surveillance. During these three years all the modules needed for a real videosurveillance system have been considered, starting from the low level foreground extraction to the high level posture detection. In particular, the main contribution of my work can be referred to the high level reasoning modules, such as the face detector and the posture classifier. The posture detector has been used in a Domotic project as a valuable help to disabled people and elders, to recognize and to send an alarm whenever the monitored person falls down. The face detection task, instead, has been used in two different situations: to extract faces for recognition purposes and to remove faces for privacy issues. Our integrated framework of head tracking and detection achieved very good performance both in term of efficacy and efficiency.

Together with the proposal of new paradigms and techniques, I have been involved in the development of real implementation of videosurveillance systems. In our campus we have installed a test setup composed by four cameras (See Fig. 8.1). I have realized a C++ Library comprising classes for image processing, motion detection, object tracking, as well as video source management, video output, graphical interface, and so on.

12.2 Publications

This research activity has produced the following publications:

- On the first part (detection and tracking), one article in international journals [71] and three publications on international conferences [17, 75, 225].
- On the second part (People Video Surveillance), three articles in international journals [87, 226, 227] and five papers in international conferences

[104, 228–231].

- On the third part (the Integrated surveillance system), one chapter in a book [232], one article in international journal [233], and seven papers on international conferences [15, 92, 140, 234–237].

Bibliography

[1] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting Moving Objects, Ghosts and Shadows in Video Streams," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1337–1342, Oct. 2003.

[2] B. Boghossian and J. Black, "The challenges of robust 24/7 video surveillance systems," in *Proceedings of the IEE International Symposium on Imaging for Crime Detection and Prevention, ICDP 2005*, 2005, pp. 33–38.

[3] B. E. Flinchbaugh and T. J.Olson, "Autonomous video surveillance," *Proceedings of the SPIE*, vol. 2962, pp. 144–151, 1997.

[4] C. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, July 1997.

[5] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[6] R. Collins, A. Lipton, and T. Kanade, "A System for Video Surveillance and Monitoring: VSAM Final Report," Tech. Rep., CMU-RI-TR-Robotics Institute, Carnegie Mellon University, May, 2000, 2000.

[7] J. Connell, A. W. Senior, A. Hampapur, Y.-L. Tian, L. Brown, and S. Pankanti, "Detection and Tracking in the IBM PeopleVision System," in *Proceedings of Int'l Conference on Multimedia and Expo*, 2004, vol. 2, pp. 1403–1406.

[8] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[9] Michael Isard and Andrew Blake, "CONDENSATION - Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[10] A. Prati, I. Mikic, M.M. Trivedi, and R. Cucchiara, "Detecting Moving Shadows: Algorithms and Evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 918–923, July 2003.

[11] R. Cucchiara, R. Melli, A. Prati, and L. De Cock, "Predictive and Probabilistic Tracking to Detect Stopped Vehicles," in *Proceedings of Workshop on Applications of Computer Vision (WACV)*, 2005, vol. 1, pp. 388–393.

[12] 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, no. 3, pp. 334–352, Aug. 2004.

[13] S. Khan and M. Shah, "Consistent labeling of tracked objects in multiple cameras with overlapping fields of view," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1355–1360, Oct. 2003.

[14] W. Hu, M. Hu, X. Zhou, T. Tan, J. Lou, and S. Maybank, "Principal Axis-based Correspondence Between Multiple Cameras for People Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 663–671, 2006.

[15] A. Calderara, A. Prati, R. Vezzani, and R. Cucchiara, "Consistent Labeling for Multi-camera Object Tracking," in *Proc. of IEEE Int'l Conference on Image Analysis and Processing*, 2005.

[16] A. D. Bagdanov, A. Del Bimbo, and F. Pernici, "Acquisition of High-resolution Images Through Online Saccade Sequence Planning," in *Proc. of ACM Workshop on Video Surveillance and Sensor Networks (VSSN)*, 2005.

[17] R. Cucchiara, A. Prati, and R. Vezzani, "Advanced Video Surveillance with Pan Tilt Zoom Cameras," in *Proc. of ACM Workshop on Video Surveillance and Sensor Networks (VSSN)*, 2006.

[18] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[19] N. Amamoto and A. Fujii, "Detecting obstructions and tracking moving objects by image processing technique," *Electronics and Communications in Japan, Part 3*, vol. 82, no. 11, pp. 28–37, 1999.

[20] C. Stauffer and W.E.L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, Aug. 2000.

[21] S.J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking groups of people," *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56, Oct. 2000.

[22] M. Seki, H. Fujiwara, and K. Sumi, "A robust background subtraction method for changing background," in *Proceedings of IEEE Workshop on Applications of Computer Vision*, 2000, pp. 207–213.

[23] N. Ohta, "A statistical approach to background suppression for surveillance systems," in *Proceedings of IEEE Int'l Conference on Computer Vision*, 2001, pp. 481–486.

[24] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel, "Towards Robust Automatic Traffic Scene Analysis in Real-Time," in *Proceedings of Int'l Conference on Pattern Recognition*, 1994, pp. 126–131.

[25] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with HSV color information," in *Proceedings of IEEE Int'l Conference on Intelligent Transportation Systems*, Aug. 2001, pp. 334–339.

[26] A. Elgammal, D. Harwood, and L.S. Davis, "Non-parametric Model for Background Subtraction," in *Proceedings of IEEE ICCV'99 FRAME-RATE Workshop*, 1999.

[27] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting objects, sahdows and ghosts in video streams by exploiting color and motion information," in *Proc. of IEEE Int'l Conference on Image Analysis and Processing*, 2001, pp. 360–365.

[28] S. Calderara, R. Cucchiara, and A. Prati, "Multimedia Surveillance: Content-based Retrieval with Multicamera People Tracking," in *Proceedings of Fourth ACM International Workshop on Video Surveillance and Sensor Networks (VSSN 2006)*, 2006.

[29] B.P.L. Lo and S.A. Velastin, "Automatic congestion detection system for underground platforms," in *Proceedings of the Int'l Symposium on Intelligent Multimedia, Video and Speech Processing*, 2000, pp. 158–161.

[30] B. Gloyer, H.K. Aghajan, K.Y. Siu, and T. Kailath, "Video-based freeway monitoring system using recursive vehicle tracking," in *Proceedings of SPIE Symposium on Electronic Imaging: Image and Video Processing*, 1995.

[31] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld, "Detection and location of people in video images using adaptive fusion of color and edge information," in *Proceedings of Int'l Conference on Pattern Recognition*, 2000, pp. 627–630.

[32] C. Stauffer and W.E.L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 1999, pp. 246–252.

[33] N.M. Oliver, B. Rosario, and A.P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, Aug. 2000.

[34] N. Rota and M. Thonnat, "Video sequence interpretation for visual surveillance," in *Proceedings of IEEE Workshop on Visual Surveillance (VS '00)*, 2000, pp. 325–332.

[35] A. Prati, R. Cucchiara, I. Mikic, and M.M. Trivedi, "Analysis and Detection of Shadows in Video Streams: A Comparative Evaluation," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 2001.

[36] C. Grana, *Segmentazione di immagini per il riconoscimento di oggetti in movimento*, Tesi di Laurea, 1999.

[37] A. Bainbridge-Smith and R.G. Lane, "Determining optical flow using a differential method," *Image and Vision Computing*, vol. 15, pp. 11–22, 1997.

[38] R. Cucchiara, C. Grana, G. Tardini, and R. Vezzani, "Probabilistic People Tracking for Occlusion Handling," in *Proceedings of Int'l Conference on Pattern Recognition*, Aug. 2004, vol. 01, pp. 132–135.

[39] M. Gelgon and P. Bouthemy, "A region-level motion-based graph representation and labeling for tracking a spatial image partition," *Pattern Recognition*, vol. 33, pp. 725–740, 2000.

[40] Y. Altunbasak, A.M. Tekalp, and G. Bozdagi, "Simultaneous motion-disparity estimation and segmentation from stereo," in *Proceedings of IEEE Int'l Conference on Image Processing*, 1994, pp. 73–77.

[41] Y. Altunbasak, P.E. Eren, and A.M. Tekalp, "Region-based affine motion segmentation using color information," in *Proceedings of Int'l Conference on Acoustic, Speech, and Signal Processing*, 1997, vol. 4, pp. 3005–3008.

[42] S. Araki, T. Matsuoka, H. Takemura, and N. Yokoya, "Real-time tracking of multiple moving objects in moving camera image sequences using robust statistics," in *Proceedings of Int'l Conference on Pattern Recognition*, 1998, vol. 2, pp. 1433–1435.

[43] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg, "A three frame algorithm for estimating two-component image motion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 12, pp. 886–896, Dec. 1992.

[44] S. Jehan-Besson, M. Barlaud, and G. Aubert, "Region-based active contours for video object segmentation with camera compensation," in *Proceedings of IEEE Int'l Conference on Image Processing*, 2001, vol. 2, pp. 61–64.

[45] K. Bhat, M. Saptharishi, and P. Khosla, “Motion detection and segmentation using image mosaics,” in *Proceedings of Int’l Conference on Multimedia and Expo*, 2000, vol. 3, pp. 1577–1580.

[46] M.M. Chang, A.M. Teklap, and M.I. Sezan, “Simultaneous motion estimation and segmentation,” *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1326–1333, Sept. 1997.

[47] G. Csurka and P. Bouthemy, “Direct identification of moving objects and background from 2D motion models,” in *Proceedings of IEEE Int’l Conference on Computer Vision*, 1999, vol. 1, pp. 566–571.

[48] R. Cutler and L.S. Davis, “Robust real-time periodic motion detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 781–796, Aug. 2000.

[49] F. Dufaux, F. Moscheni, and A. Lippman, “Spatio-temporal segmentation based on motion and static segmentation,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1995, vol. 1, pp. 306–309.

[50] R. Fablet, P. Bouthemy, and M. Gelgon, “Moving object detection in color image sequences using region-level graph labeling,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1999, vol. 2, pp. 939–943.

[51] I. Grinias and G. Tziritas, “Motion segmentation and tracking using a seeded region growing method,” in *Proceedings of European Signal Processing Conference*, 1998.

[52] C. Hennebert, V. Rebuffel, and P. Bouthemy, “A hierarchical approach for scene segmentation based on 2D motion,” in *Proceedings of Int’l Conference on Pattern Recognition*, 1996, vol. 1, pp. 218–222.

[53] K.W. Lee, S.W. Ryu, S.J. Lee, and K.T. Park, “Motion based object tracking with mobile camera,” *Electronics Letters*, vol. 34, no. 3, pp. 256–258, Mar. 1998.

[54] J.S. Lee, K.Y. Rhee, and S.D. Kim, “Moving target tracking algorithm based on the confidence measure of motion vectors,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 2001, vol. 1, pp. 369–372.

[55] Y.T. Lin, Y.K. Chen, and S.Y. Kung, “Object-based scene segmentation combining motion and image cues,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1996, vol. 1, pp. 957–960.

[56] F. Lohier, L. Lacassagne, and P. Garda, “Generic programming methods for the real time implementation of a MRF based motion detection algorithm on a multi-processor DSP with multi-dimensional DMA,” in *Proceedings of Symposium GRETSI on Signal and Image Processing*, 1999.

[57] R. Mandelbaum, G. Salgian, and H. Sawhney, “Correlation-based estimation of ego-motion and structure from motion and stereo,” in *Proceedings of IEEE Int’l Conference on Computer Vision*, 1999, vol. 1, pp. 544–550.

[58] F. Moscheni and S. Bhattacharjee, “Robust region merging for spatio-temporal segmentation,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1996, vol. 1, pp. 501–504.

[59] Y. Ninomiya, S. Matsuda, M. Ohta, and Y. Harata, “A real-time vision for intelligent vehicles,” in *Proceedings of IEEE Intelligent Vehicle Symposium*, 1996, pp. 315–320.

[60] J.M. Odobez and P. Bouthemy, “Detection of multiple moving objects using multiscale MRF with camera motion compensation,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1994, vol. 2, pp. 257–261.

[61] N. Paragios, P . Perez, G. Tziritas, C. Labit, and P. Bouthemy, “Adaptive detection of moving objects using multiscale techniques,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 1996, vol. 1, pp. 525–528.

[62] H. Sawhney and S. Ayer, “Compact representations of videos through dominant and multiple motion estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 814–830, Aug. 1996.

[63] P. Smith, T. Drummond, and R. Cipolla, “Segmentation of multiple motions by edge tracking between two frames,” in *Proceeding of British Machine Vision Conference*, 2000, pp. 342–351.

[64] S.M. Smith and J.M. Brady, “ASSET-2: Real-time motion segmentation and shape tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 814–820, Aug. 1995.

[65] V. Stephan and H.M. Gross, “A neural architecture for expectation driven detection of movign object during egomotion,” in *Proceedings of Int’l Joint Conference on Neural Networks*, 2001, pp. 2165–2170.

[66] A. Techmer, “Contour-based motion estimation and obejct tracking for real-time applications,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 2001, vol. 3, pp. 648–651.

[67] D. Tweed and A. Calway, “Motion segmentation based on integrated region layering and motion assignment,” in *Proceedings of Asian Conference on Computer Vision*, Jan. 2000, pp. 1002–1007.

[68] J.W.Y. Kam, “A real-time 3D motion tracking system,” Tech. Rep. 93-16, Laboratory for Computational Intelligence, Dept. of Computer Science, University of British Columbia, Apr. 1993.

[69] J.Y.A. Wang and E.H. Adelson, “Representing moving images with layers,” *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 625–638, May 1994.

[70] Y. Weiss and E.H. Adelson, “Perceptually organized EM: a fremework for motion segmentation that combines information about form and motion,” Tech. Rep. 315, MIT Media Lab Vismond, 1995.

[71] R. Cucchiara, A. Prati, and R. Vezzani, “Real Time Motion Segmentation From Moving Cameras,” *Real-Time Imaging*, vol. 10, no. 3, pp. 127–143, 2004.

[72] R. Megret, C. Saraceno, and W. Kropatsch, “Background mosaic from ego-motion,” in *Proceedings of Int’l Conference on Pattern Recognition*, 2000, vol. 1, pp. 571–574.

[73] T.Y. Tian, C. Tomasi, and D.J. Heeger, “Comparison of approaches to ego-motion computation,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 1996, pp. 315–320.

[74] M.B. Van Leeuwen and F.C.A. Groen, “Motion estimation with a mobile camera for traffic applications,” in *Proceedings of IEEE Intelligent Vehicle Symposium*, 2000, pp. 58–63.

[75] R. Cucchiara, A. Prati, and R. Vezzani, “Object Segmentation in Videos from Moving Camera with MRFs on Color and Motion Features,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2003, vol. 1, pp. 405–410.

[76] R. Adams and L. Bischof, “Seeded region growing,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, June 1994.

[77] H.G. Musmann, P. Pirsch, and H.J. Grallert, “Advances in picture coding,” *Proceedings of the IEEE*, vol. 73, no. 4, pp. 523–546, 1985.

[78] S. Zhu and K.K. Ma, “A new diamond search algorithm for fast block matching,” *IEEE Transactions on Circuits and System for Video Technology*, vol. 9, no. 2, pp. 287–290, Feb. 2000.

[79] D. Todorovic, “A gem from the past: Pleikart Stumpf’s (1911) anticipation of the aperture problem, Reichardt detectors, and perceived motion loss at equiluminance,” *Perception*, vol. 25, pp. 1235–1242, 1996.

[80] M. Gelgon and P. Bouthemy, “A region-level graph labeling approach to motion-based segmentation,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 1997, pp. 514–519.

[81] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance Models for Occlusion Handling," in *Proceedings of International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) systems*, 2001.

[82] Tao Zhao and Ram Nevatia, "Tracking Multiple Humans in Complex Situations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1208–1221, 2004.

[83] X. Liu, P.H. Tu, J. Rittscher, A. Perera, and N. Krahnstoever, "Detecting and counting people in surveillance applications," in *Proceedings. of IEEE Conference on Advanced Video and Signal Based Surveillance*, 2005, vol. 1, pp. 306–311.

[84] Omar Javed, Zeeshan Rasheed, Orkun Alatas, and Mubarak Shah, "M-Knight: A Real Time Surveillance System for Multiple Overlapping and Non-Overlapping Cameras," in *Proceedings of Int'l Conference on Multimedia and Expo*, 2003, vol. 1, pp. 649–652.

[85] Bangjun Lei and Li-Qun Xu, "Real-time outdoor video surveillance with robust foreground extraction and object tracking via multi-state transition management," *Pattern Recognition Letters*, vol. 27, pp. 1816–1825, 2006.

[86] Michael Isard and John MacCormick, "BraMBLe: A Bayesian Multiple-Blob Tracker.," in *Proceedings of IEEE Int'l Conference on Computer Vision*, 2001, pp. 34–41.

[87] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Probabilistic Posture Classification for Human Behaviour Analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 1, pp. 42–54, Jan. 2005.

[88] J.L. Crowley, J. Coutaz, and F. Brard, "Things That See," *Communications of the ACM*, vol. 43, no. 3, pp. 54–64, 2000.

[89] Michael J. Black and Allan D. Jepson, "EigenTracking: Robust Matching and Tracking of Articulated Objects Using a View-Based Representation," *Int. J. Comput. Vision*, vol. 26, no. 1, pp. 63–84, 1998.

[90] N. Thome and S. Miguet, "A robust appearance model for tracking human motions," in *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, 2005, vol. 1, pp. 528–533.

[91] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Kernel-Based Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, 2003.

[92] S. Calderara, R. Vezzani, A. Prati, and R. Cucchiara, "Entry Edge of Field of View for multi-camera tracking in distributed video surveillance," in *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, 2005, vol. 1, pp. 93–98.

[93] Aaron F. Bobick and James W. Davis, "The Recognition of Human Movement Using Temporal Templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[94] C. Stauffer and W.E.L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 747–757, Aug. 2000.

[95] Stewart Greenhill, Svetha Venkatesh, and Geoff A. W. West, "Adaptive Model for Foreground Extraction in Adverse Lighting Conditions.," in *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence*, 2004, pp. 805–811.

[96] L. Brown, J. Connell, A. Hampapur, M. Lu, A. Senior, C.-F. Shu, and Y. Tian, "IBM Smart Surveillance System (S3): A Open And Extensible Framework For Event Based Surveillance," in *Proceedings of the International Conference on Advanced Video- and Signal-based Surveillance 2005*, 2005, vol. 01, pp. 318–323.

[97] Deva Ramanan, David A. Forsyth, and Andrew Zisserman, "Tracking People by Learning Their Appearance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 65–81, 2007.

[98] Ying Wu, Ting Yu, and Gang Hua, "Tracking Appearances with Occlusions," *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, vol. 01, pp. 789, 2003.

[99] Oswald Lanz, "Approximate Bayesian Multibody Tracking.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1436–1449, 2006.

[100] S.J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler, "Tracking Groups of People," *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56, oct 2000.

[101] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi, "Robust online appearance models for visual tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296–1311, Oct 2003.

[102] Omar Javed and Mubarak Shah, "Tracking and Object Classification for Automated Surveillance," in *Proceedings of IEEE European Conference on Computer Vision*, 2002, pp. 343–357.

[103] Bo Wu and Ram Nevatia, “Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2006, vol. 1, pp. 951–958.

[104] R. Cucchiara, A.Prati, and R. Vezzani, “Posture Classification in a Multi-camera Indoor Environment,” in *Proceedings of IEEE Int’l Conference on Image Processing*, 2005, vol. 01, pp. 725–728.

[105] “http://pets2002.visualsurveillance.org,” .

[106] Claudette Cedras and Mubarak Shah, “Motion-based recognition a survey.,” *Image Vision Comput.*, vol. 13, no. 2, pp. 129–155, 1995.

[107] D. M. Gavrila, “The visual analysis of human movement: a survey,” *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82–98, 1999.

[108] J. K. Aggarwal and Q. Cai, “Human motion analysis: a review,” *Comput. Vis. Image Underst.*, vol. 73, no. 3, pp. 428–440, 1999.

[109] T.B. Moeslund and E. Granum, “A Survey of Computer Vision-Based Human Motion Capture,” *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231–268, Mar. 2001.

[110] Liang Wang, Weiming Hu, and Tieniu Tan, “Recent developments in human motion analysis.,” *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, 2003.

[111] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, “Human Body Model Acquisition and Tracking Using Voxel Data,” *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, July-August 2003.

[112] X. Wang, K. Tieu, and E. Grimson, “Learning Semantic Scene Models by Trajectory Analysis,” in *Proceedings of IEEE European Conference on Computer Vision*, 2006, pp. 110–123.

[113] N.T. Nguyen, H.H. Bui, S. Venkatsh, and G. West, “Recognizing and monitoring high-level behaviors in complex spatial environments,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2003.

[114] D. Makris and T. Ellis, “Path detection in video surveillance,” *Image and Vision Computing*, vol. 20, no. 12, pp. 895–903, 2002.

[115] D. Makris and T. Ellis., “Spatial and Probabilistic Modelling of Pedestrian Behavior,” *Proceeding of British Machine Vision Conference*, pp. 557–566, 2002.

[116] M. J. Swain. R. E. Kahn, “Gesture Recognition Using the Perseus Architecture,” Tech. Rep., TR-Dept. Computer Science, Univ. Chicago, 1996.

[117] C. R. Wren, B. P. Clarkson, and A. P. Pentland, "Understanding Purposeful Human Motion," in *The Fourth International Conference on Automatic Face and Gesture Recognition*, 2000.

[118] C. I. Attwood, G. D. Sullivan, and K.D. Baker, "Model-based Recognition of Human Posture Using Single Synthetic Images," in *Fifth Alvey Vision Conference*, 1989.

[119] O. Chomat and J. L. Crowley, "Recognizing Motion Using Local Appearance," in *In International Symposium on Intelligent Robotic Systems*, 1998.

[120] R. Nelson. R. Polana, "Low Level Recognition of Human Motion," in *In Workshop on Motion of Non-rigid and Articulated Objects*, 1994.

[121] T. Darell, P. Maes, B. Blumberg, and A. P. Pentland, "A Novel Environment for Situated Vision and Behavior," in *Workshop for Visual Behaviors At CVPR-94*, 1994.

[122] C. Bregler, "Learning and Recognizing Human Dynamics in Video Sequences," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 1997.

[123] T. Starner and A. Pentland, "Real-time American Sign Language Recognition from Video Using Hidden Markov Models," in *Proceedigns of International Symposium on Computer Vision*, 1995.

[124] B. Heisele and C. Wohler, "Motion-based Recognition of Pedestrians," in *Proceedings of Int'l Conference on Pattern Recognition*, 1998.

[125] I. Haritaoglu, D. Harwood, and L. S. Davis, "Ghost: A Human Body Part Labeling System Using Silhouettes," in *Proceedings of Int'l Conference on Pattern Recognition*, 1998, pp. 77–82.

[126] S.X Ju, M.J. Black, and Y. Yacob, "Cardboard People: A Parameterized Model of Articulated Image Motion," in *In 2 International Conf. on Automatic Face and Gesture Recognition*, 1996.

[127] S. Iwasawa, J. Ohya, K. Takahashi, T. Sakaguchi, K. Ebihara, and S. Morishima, "Human body postures from trinocular camera images," in *Proceedings of IEEE Int'l Conference on Automatic Face and Gesture Recognition*, 2000, pp. 326–331.

[128] N. Werghi and Y. Xiao, "Recognition of human body posture from a cloud of 3D data points using wavelet transform coefficients," in *Proceedings of IEEE Int'l Conference on Automatic Face and Gesture Recognition*, 2002, pp. 70–75.

[129] S. Iwasawa, K. Ebihara, J. Ohya, and S. Morishima, "Real-time human posture estimation using monocular thermal images," in *Proceedings of IEEE Int'l Conference on Automatic Face and Gesture Recognition*, 1998, pp. 492–497.

[130] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman, "Human Body Model Acquisition and Tracking Using Voxel Data," *International Journal of Computer Vision*, vol. 53, no. 3, pp. 199–223, July-August 2003.

[131] S. Pinzke and L. Kopp, "Marker-less systems for tracking workin postures - results from two experiments," *Applied Ergonomics*, vol. 32, no. 5, pp. 461–471, Oct. 2001.

[132] J. Freer, B. Beggs, H. Fernandez-Canque, F. Chevriet, and A. Goryashko, "Automatic recognition of suspicious activity for camera based security systems," in *Proc. of European Convention on Security and Detection*, 1995, pp. 54–58.

[133] B. Ozer and W. Wolf, "Human Detection in Compressed Domain," in *Proceedings of IEEE Int'l Conference on Image Processing*, 1998, pp. 77–82.

[134] M.M. Rahman, K. Nakamura, and S. Ishikawa, "Recognizing human behavior using universal eigenspace," in *Proceedings of Int'l Conference on Pattern Recognition*, 2002, vol. 1, pp. 295–298.

[135] H. Fujiyoshi and A.J. Lipton, "RealTime Human Motion Analysis by Image Skeletonization," in *Fourth IEEE Workshop on Applications of Computer Vision*, 1998.

[136] I.-Cheng Chang and Chung-Lin Huang, "The model-based human body motion analysis system," *Image and Vision Computing*, vol. 18, no. 14, pp. 1067–1083, Nov. 2000.

[137] Yi Li, Sondge Ma, and Hanging Lu, "Human posture recognition using multi-scale morphological method and Kalman motion estimation," in *Proceedings of Int'l Conference on Pattern Recognition*, 1998, vol. 1, pp. 175–177.

[138] Changbo Hu, Qingfeng Yu, Yi Li, and Sondge Ma, "Extraction of parametric human model for posture recognition using genetic algorithm," in *Proceedings of IEEE Int'l Conference on Automatic Face and Gesture Recognition*, 2000, pp. 518–523.

[139] L. Panini and R. Cucchiara, "A machine learning approach for human posture detection in domotics applications," in *Proc. of IEEE Int'l Conference on Image Analysis and Processing*, 2003, pp. 103–108.

[140] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "Enabling PDA Video Connection for In-House Video Surveillance," in *Proc. of First ACM Workshop on Video Surveillance (IWVS)*, 2003, pp. 87–97.

[141] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[142] http://www.imsresearch.com, ""IMS Research, CCTV & Video Surveillance Equipment 2003"," .

[143] Q. Cai and J.K. Aggarwal, "Tracking human motion in structured environments using a distributed-camera system," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 11, pp. 1241–1247, 1999.

[144] M. Yang, D.J. Kriegman, and N. Ahuja, "Detecting Faces in Images: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 1, pp. 34–58, 2002.

[145] Matthew A. Turk and Alex Pentland, "Face recognition using eigenfaces," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–591, 1991.

[146] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[147] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, 8-14 December 2001, Kauai, HI, USA*. 2001, vol. 1, pp. 511–518, IEEE Computer Society.

[148] E. Hjelm and B.K. Low, "Face Detection: A Survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[149] M.J. Jones and J.M. Rehg, "Statistical Color Models with Application to Skin Detection," *International Journal of Computer Vision*, vol. 46, pp. 81–96, 2002.

[150] R.L. Hsu, M. Abdel-Mottaleb, and A.K. Jain, "Face Detection: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002.

[151] S. Birchfield, "Elliptical Head Tracking Using Intensity Gradients and Color Histograms," *Proc. of IEEE Computer Vision and Pattern Recognition*, pp. 232–237, 1998.

[152] D. Maio and D. Maltoni, "Real-time face location on gray-scale static images," *Pattern Recognition*, vol. 33, no. 9, pp. 1525–1539, Sept. 2000.

[153] J. Li, C.S. Chua, and Y.K. Ho, "Color based multiple people tracking," in *Proc. of IEEE Intl Conf. on Control, Automation, Robotics and Vision*, 2002, vol. 1, pp. 309–314.

[154] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for EasyLiving," in *Proc. of IEEE Intl Workshop on Visual Surveillance*, 2000, pp. 3–10.

[155] J. Kang, I. Cohen, and G. Medioni, "Continuous tracking within and across camera streams," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 2003, vol. 1, pp. I–267 – I–272.

[156] S. Chang and T.H. Gong, "Tracking multiple people with a multi-camera system," in *Proc. of IEEE Workshop on Multi-Object Tracking*, 2001, pp. 19–26.

[157] S.L. Dockstader and A.M. Tekalp, "Multiple camera tracking of interacting and occluded human motion," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1441–1455, Oct. 2001.

[158] Z. Yue, S.K. Zhou, and R. Chellappa, "Robust two-camera tracking using homography," in *Proc. of IEEE Intl Conf. on Acoustics, Speech, and Signal Processing*, 2004, vol. 3, pp. 1–4.

[159] A. Mittal and L. Davis, "Unified multi-camera detection and tracking using region-matching," in *Proc. of IEEE Workshop on Multi-Object Tracking*, 2001, pp. 3–10.

[160] D. Devarajan and R. Radke, "Distributed Metric Calibration of Large Camera Networks," in *First Workshop on Broadband Advanced Sensor Networks (BASENETS)*, 2004.

[161] D. Makris, T. Ellis, and J. Black, "Bridging the Gaps between Cameras," in *Proceedings of IEEE Int'l Conference on Computer Vision and Pattern Recognition*, 2004, vol. 2, pp. 205–210.

[162] Z. Yue, S. K. Zhou, and R. Chellappa, "Robust two-camera tracking using homography," in *Proc. of IEEE Intl Conf. on Acoustics, Speech, and Signal Processing*, 2004, vol. 3, pp. 1–4.

[163] R. I. Hartley, "Euclidean Reconstruction from Uncalibrated Views," in *Proc. 2nd European-US Workshop on Invariance*, 1993, pp. 187–202.

[164] P. A. Beardsley, A. P. Zisserman, , and D. W. Murray, "Navigation Using Affine Structure and Motion," in *Proc. 3rd European Conference on Computer Vision*, 1994, pp. 85–96.

[165] P. F. McLauchlan, I. D. Reid, , and D. W. Murray, “Recursive Affine Structure and Motion from Image Sequences,” in *Proceedings of IEEE European Conference on Computer Vision*, 1994, pp. 217–224.

[166] P. A. Beardsley, I. D. Reid, A. Zisserman, , and D. W. Murray, “Active Visual Navigation Using Non-metric Structure,” in *Proceedings of IEEE Int'l Conference on Computer Vision*, 1995, pp. 58–65.

[167] P. Beardsley, P. Torr, and A. Zisserman, “3D Model Acquisition from Extended Image Sequences,” in *Proceedings of IEEE European Conference on Computer Vision*, 1996, pp. 683–695.

[168] J. Y. Bouguet, , and P. Perona, “Visual Navigation Using a Single Camera,” in *Proceedings of IEEE Int'l Conference on Computer Vision*, 1995, pp. 645–652.

[169] A. Shashua, “Trilinearity in Visual Recognition by Alignment,” in *Proceedings of IEEE European Conference on Computer Vision*, 1994, pp. 479–484.

[170] A. W. Fitzgibbon and A. Zisserman, “Automatic Camera Recovery for Closed Or Open Image Sequences,” in *Proceedings of IEEE European Conference on Computer Vision*, 1998, pp. 311–326.

[171] D. Steedly and I. Essa, “Propagation of Innovative Information in Nonlinear Least-squares Structure from Motion,” in *Proceedings of IEEE Int'l Conference on Computer Vision*, 2001, vol. 2, pp. 223–229.

[172] Y. Matsumoto, M. Inaba, and H. Inoue, “Visual Navigation Using View-Sequenced Route Representation,” in *Proceedings of IEEE Conference on Robotics and Automation*, 1996, pp. 83–88.

[173] K. Kidono, J. Miura, and Y. Shirai, “Autonomous Visual Navigation of a Mobile Robot Using a Human-Guided Experience,” *Robotics and Autonomous Systems*, vol. 40, no. 2, pp. 124–132, 2002.

[174] I. Ulrich and I. Nourbakhsh, “Appearance-based Place Recognition for Topological Localization,” in *International Conference on Robotics and Automation*, 2000, vol. 2, pp. 1023–1029.

[175] P. Blaer and P. Allen, “Topological Mobile Robot Localization Using Fast Vision Techniques,” in *International Conference on Robotics and Automation*, 2002, pp. 1031–1036.

[176] D. Demirdjian and T. Darrell, “Motion Estimation from Disparity Images,” in *Proceedings of IEEE Int'l Conference on Computer Vision*, 2001, pp. 213–218.

[177] M. A. Garcia and A. Solanas, “3D Simultaneous Localization and Modeling from Stereo Vision,” in *Proc. of IEEE Intl Conf. on Robotics and Automation*, 2004, pp. 847–853.

[178] D. Nister, O. Naroditsky, , and J. Bergen, “Visual Odometry,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2004.

[179] A. Levin and R. Szeliski, “Visual Odometry and Map Correlation,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2000, pp. 611–618.

[180] E. Royer, M. Lhuillier, M. Dhome, and T. Chateau, “Localization in Urban Environments: Monocular Vision Compared to a Differential GPS Sensor,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2005, pp. 114–121.

[181] Y. Kuniyoshi, N. Kita, S. Rougeaux, , and T. Suehiro, “Active Stereo Vision System with Foveated Wide Angle Lenses,” in *Proceedings of the Asian Conference on Computer Vision*, 1995, pp. 359–363.

[182] A. J. Davison and D. W. Murray, “Mobile Robot Localisation Using Active Vision,” in *Proceedings of IEEE European Conference on Computer Vision*, 1998, pp. 809–825.

[183] S. Jeon and B. Kim, “Monocular-based Position Determination for Indoor Navigation of Mobile Robots,” in *Proc. of the IASTED International Conference.*, 1999.

[184] J. A. Castellanos, J. M. M. Montiel, J. Neira, , and J. D. Tardos, “The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building,” *IEEE Trans. Robotics and Automation*, pp. 948–952, 1999, 15(5).

[185] J. J. Leonard and H. J. S. Feder, “A Computationally Efficient Method for Large-scale Concurrent Mapping and Localization,” in *The Ninth International Symposium on Robotics Research*, 2000, pp. 169–176.

[186] G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A Solution to the Simultaneous Localization and Map Building (SLAM) Problem,” *IEEE Transaction on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[187] P. Lamon and R. Siegwart, “3D-odometry for Rough Terrain Towards Real 3d Navigation,” in *Proceedings of IEEE International Conference on Robotics and Automation.*, 2003.

[188] K. Ohno, T. Tsubouchi, B. Shigematsu, , and S. Yuta, “Differential Gps and Odometry-based Outdoor Navigation of a Mobile Robot,” *Advanced Robotics*, vol. 18, no. 6, pp. 611–635, 2004.

[189] A. Elfes, “Sonar-based Real-world Mapping and Navigation,” *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[190] K. S. Chong and L. Kleeman, “Feature-based Mapping in Real large scale environments using an ultrasonic array.,” *In International Journal of Robotics Research*, vol. 18, no. 2, pp. 319, 1999.

[191] P. Newman, J. Leonard, J.D. Tardos, and J. Neira, “Experimental Validation of Real-Time Concurrent Mapping and Localization,” in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2002, pp. 1802–1809.

[192] J. D. Tards, J. Neira, P. M. Newman, , and J. J. Leonard, “Robust Mapping and Localization in Indoor Environments Using Sonar Data,” *International Journal of Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[193] S. Se, D. Lowe, , and J. Little, “Mobile Robot Localization and Mapping with Uncertainty Using Scale-invariant Visual Landmarks,” *International Journal of Robotic Research*, vol. 21, pp. 735–758, 2002.

[194] S. Thrun, D. Fox, , and W. Burgard, “A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots,” *Machine Learning*, p. 31, 1998.

[195] A. J. Davison and N. Kita, “3D Simultaneous Localisation and Map-Building Using Active Vision for a Robot Moving on Undulated Terrain,” in *Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition*, 2002, pp. 384–391.

[196] A. J. Davison, “Real-Time Simultaneous Localisation and Mapping with a Single Camera,” in *Proceedings of IEEE Int’l Conference on Computer Vision*, 2003, pp. 1403–1410.

[197] L. Matthies and S. Shafer, “Error Modeling in Stereo Navigation,” *IEEE Journal of Robotics and Automation*, vol. 3, no. 3, pp. 239–248, 1987.

[198] Y. Cheng, M. W. Maimone, and L. Matthies, “Visual Odometry on the Mars Exploration Rovers,” *IEEE Robotics and Automation Magazine*, vol. 2, no. 13, pp. 54–62, 2006.

[199] R. Ozawa, Y. Takaoka, Y. Kida, K. Nishiwaki, J. Chestnutt, J. Kuffner, J. Kagami, H. Mizoguch, and H. Inoue, “Using Visual Odometry to Create 3D Maps for Online Footstep Planning,” in *IEEE International Conference on Systems, Man and Cybernetics*, 2005, vol. 3, pp. 2643–2648.

[200] H. Wang, K. Yuan, W. Zou, and Q. Zhou, “Visual Odometry Based on Locally Planar Ground Assumption,” in *IEEE International Conference on Information Acquisition*, 2005, pp. 59–64.

[201] Q. T. Luong and O. D. Faugeras, “Self-Calibration of a Moving Camera from Point Correspondences and Fundamental Matrices,” *International Journal of Computer Vision*, vol. 22, no. 3, pp. 261–289, 1997.

[202] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[203] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” in *Proc. of the Alvey Vision Conference*, 1988, pp. 147–151.

[204] K. Mikolajczyk and C. Schmid, “Scale and Affine Invariant Interest Point Detectors,” *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[205] P. H. S. Torr and D. W. Murray, “Outlier Detection and Motion Segmentation,” *Sensor Fusion VI*, pp. 432–443, 1993, 2059, Boston.

[206] Q. T. Luong and O. D. Faugeras, “The Fundamental Matrix: Theory, algorithms, and stability analysis.,” *International Journal of Computer Vision*, pp. 43–76, 1996.

[207] R. Valera Espina and S.A. Velastin, “Intelligent distributed surveillance systems: A Review,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, Apr. 2005.

[208] http://www.smarthome.com/7527MC.HTML, ,” .

[209] D. Culler, D. Estrin, and M. Srivastava, “Guest Editors’ Introduction: Overview of Sensor Networks,” *IEEE Computer*, vol. 37, no. 8, pp. 41–49, Aug. 2004.

[210] http://www.xbow.com/Products /productsdetails.aspx?sid=3, ,” .

[211] G.L. Foresti, C. Micheloni, L. Snidaro, P. Remagnino, and T. Ellis, “Active Video-Based Surveillance System,” *IEEE Signal Processing Magazine*, pp. 25–37, Mar. 2005.

[212] http://www.kalatel.com, ,” .

[213] H-W. Braun P. Bryant, “Some applications of a motion detecting camera in remote environments,” *Technical Report*, Feb. 2003.

[214] S. de Vlaam, “Object Tracking in a multi sensor network,” *Master Thesis*, Aug. 2004.

[215] S. Rajgarhia, F. Stann, and J. Heidemann, “Privacy-Sensitive Monitoring With a Mix of IR Sensors and Cameras,” *Proceedings of the Second International Workshop on Sensor and Actor Network Protocols and Applications*, pp. 21–29, Aug. 2004.

[216] A.S. Sekmen, M. Wilkes, and K. Kawamura, "An Application of Passive Human-Robot Interaction: Human-Tracking Based on Attention Distraction," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 32, no. 2, pp. 248–259, Mar. 2002.

[217] Yucong Lu, Lingqi Zeng, and Gary M. Bone, "Multisensor System for Safer Human-Robot Interaction," *IEEE International Conference on Robotics and Automation*, Apr. 2005.

[218] M. Bertini, R. Cucchiara, A. Del Bimbo, and A. Prati, "An Integrated Framework for Semantic Annotation and Transcoding," *Multimedia Tools and Applications*, vol. 26, no. 3, pp. 345–363, 2005.

[219] A. Vetro, T. Haga, K. Sumi, and H.Sun, "Object-based coding for long-term archive of surveillance video," in *Proceedings of Int'l Conference on Multimedia and Expo*, 2003, vol. 2, pp. 417–420.

[220] A. Davison, *Killer Game Programming in Java*, O'Reilly Media, May 2005.

[221] K. Nagao, Y. Shirai, and K. Squire, "Semantic Annotation and Transcoding: Making Web Content More Accessible," *IEEE Multimedia*, vol. 8, no. 2, pp. 69–81, April-June 2001.

[222] R. Cucchiara, C. Grana, and A. Prati, "Semantic Video Transcoding using Classes of Relevance," *International Journal of Image and Graphics*, vol. 3, no. 1, pp. 145–169, Jan. 2003.

[223] A. Vetro, T. Haga, K. Sumi, and Sun H., "Object-based Coding for Long-term Archive of Surveillance Video," in *IEEE Conference on Multimedia & Expo*, 2003, vol. 2, pp. 417–420.

[224] M. Bertini, R. Cucchiara, A. Del Bimbo, and A. Prati, "Object and Event Detection for Semantic Annotation and Transcoding," in *Proceedings of IEEE Conference on Multimedia & Expo*, 2003, vol. 2, pp. 421–424.

[225] R. Cucchiara, C. Grana, and G. Tardini, "Track-based and Object-based Occlusion for People Tracking Refinement in Indoor Surveillance," in *Proc. of ACM Intl Workshop on Video Surveillance & Sensor Networks*, 2004, pp. 81–87.

[226] R. Cucchiara, A. Prati, and R. Vezzani, "A System for Automatic Face Obscuration for Privacy Purposes," *Pattern Recognition Letters*, vol. 27, no. 15, pp. 1809–1815, 2006.

[227] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, "A Computer Vision System for In-house Video Surveillance," *IEE Proceedings - Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 242–249, 2005.

[228] R. Cucchiara, A. Prati, and R. Vezzani, "Making the home safer and more secure through visual surveillance," in *Proceedings of Symposium on Automatic detection of abnormal human behaviour using video processing of Measuring Behaviour*, 2005.

[229] R. Vezzani R. Cucchiara, A. Prati, "Ambient Intelligence for Security in Public Parks: the LAICA Project," in *Proceedings of IEE International Symposium on Imaging for Crime Detection and Prevention*, 2005, pp. 139–144.

[230] R. Cucchiara and R. Vezzani, "Assessing Temporal Coherence for Posture Classification with Large Occlusions," in *Proceedings of IEEE Computer Society Workshop on Motion and Video Computing*, Jan. 2005, vol. 2, pp. 269–274.

[231] R. Cucchiara, C. Grana, A. Prati, G. Tardini, and R. Vezzani, "Using computer vision techniques for dangerous situation detection in domotics applications," *Proc. of IEE Intelligent Distributed Surveillance Systems (IDSS-04)*, pp. 1–5, Feb. 2004.

[232] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, *A Distributed Domotic Surveillance System*, chapter 4, pp. 91–117, IEE Press, 2006.

[233] R. Cucchiara, A. Prati, R. Vezzani, L. Benini, E. Farella, and P. Zappi, "An Integrated Multi-Modal Sensor Network for Video Surveillance," *Journal of Ubiquitous Computing and Intelligence (JUCI)*, 2006.

[234] R. Vezzani, R. Cucchiara, A. Malizia, and L. Cinque, "3-D Virtual Environments on Mobile Devices for Remote Surveillance," in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2006.

[235] A. Hakeem, R. Vezzani, M. Shah, and R. Cucchiara, "Estimating Geospatial Trajectory of a Moving Camera," in *Proceedings of Int'l Conference on Pattern Recognition*, 2006.

[236] A. Prati, R. Vezzani, L. Benini, E. Farella, and P. Zappi, "An Integrated Multi-Modal Sensor Network for Video Surveillance," in *Journal of Ubiquitous Computing and Intelligence (JUCI)*, 2006.

[237] R. Cucchiara, A. Prati, and R. Vezzani, "Domotics for disability: smart surveillance and smart video server," in *Proceedings of Workshop on Ambient Intelligence*, 2003, pp. 46–57.