Computer Vision Techniques for PDA Accessibility of In-House Video Surveillance

Rita Cucchiara, Costantino Grana, Andrea Prati, Roberto Vezzani
D.I.I. - University of Modena and Reggio Emilia
Via Vignolese, 905/b
Modena, Italy
{cucchiara.rita, grana.costantino, prati.andrea, vezzani.roberto}@unimore.it

ABSTRACT

In this paper we propose an approach to indoor environment surveillance and, in particular, to people behaviour control in home automation context. The reference application is a silent and automatic control of the behaviour of people living alone in the house and specially conceived for people with limited autonomy (e.g., elders or disabled people). The aim is to detect dangerous events (such as a person falling down) and to react to these events by establishing a remote connection with low-performance clients, such as PDA (Personal Digital Assistant). To this aim, we propose an integrated server architecture, typically connected in intranet with network cameras, able to segment and track objects of interest; in the case of objects classified as people, the system must also evaluate the people posture and infer possible dangerous situations. Finally, the system is equipped with a specifically designed transcoding server to adapt the video content to PDA requirements (display area and bandwidth) and to the user's requests. The main issues of the proposal are a reliable real-time object detector and tracking module, a simple but effective posture classifier improved by a supervised learning phase, and an high performance transcoding inspired on MPEG-4 object-level standard, tailored to PDA. Results on different video sequences and performance analysis are discussed.

Categories and Subject Descriptors

I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*motion, tracking*; I.4.2 [**Image Processing and Computer Vision**]: Compression (Coding)

General Terms

Security, Performance, Experimentation

Keywords

Indoor surveillance, people posture classifier, transcoding, PDA

1. INTRODUCTION

IWVS'03, November 7, 2003, Berkeley, California, USA.

Nowadays, video surveillance is a mature discipline aiming to define techniques and systems for processing (possibly in real-time) videos from cameras placed in a specific environment; the main goal is to extract knowledge about the environment and the actors living there, to detect objects and events of interest, and, consequently, to react by storing multimedia data, generating alarms or providing a remote access to live videos. The appeal of research activity in video-surveillance is the interesting multi-disciplinarity involved. Video surveillance needs computer vision and pattern recognition techniques for real-time video analysis, computer architecture and system integration for smart video servers, models of knowledge extraction and knowledge representation for the scene comprehension and techniques to handle communication standards used to connect networks of cameras, servers and remote clients. Moreover, in addition to the well known difficulties of computer vision tasks applied to real world non-idealities, that call for sophisticated methodologies and innovative video processing techniques, many HW/SW engineering problems arise in video-surveillance systems: in particular, problems are associated with the real-time constraint of the applications, juxtaposed to the frequent need of using off-the-shelf computing technologies only. In this case, the concept of *real-time* is adapted according to the need of system reactivity to detected events, the storage requirements of video servers and the *QoS* requests from remote users, if they exist. Thus, the video-surveillance system should find a good tradeoff between the need of processing the higher number possible of available video frames (for an accurate motion detection, a correct identification, a good video quality, ...) and, at the same time, to limit the amount of data (pixels, objects, a priori knowledge models, ...) that have to be handled.

In this work, we stress on these issues, in order to provide fast answers to events and model reactivity and, simultaneously, to cope with the limited computational power of standard network cameras and workstation servers. Moreover, we focus on the needs of remote users with very limited display and computational resources, such as the growing diffused Personal Digital Assistants (PDAs). Our proposal is specially devoted to indoor environment surveillance and to people behaviour control in home automation (or, better, *domotics*[1]) context. The reference application is a silent and automatic control of the behaviour of people living alone in the house and specially conceived for people with limited autonomy (e.g., elders or disabled people). The possibility to improve their autonomy and their quality of life can be supported by fully automatic computer vision systems that could be able to see and interpret what

[1]*Domotics* is a neologism coming from the Latin word domus (home) and informatics.

is happening, possibly interacting with the person or providing a remote connection if some dangerous events occur. Examples of dangerous situations could be people falling down and lying on the floor, people going asleep in strange locations of the home, and so on. To this aim, we have defined models and techniques for people detection and posture interpretation.

In dangerous circumstances, a remote connection should be established with someone interested to the home guest's care. In the case of hospitals or specific health care structure, the typical connection to a control center could be guaranteed without problems of bandwidth or limited resources. Instead, in private home situation, the scenario can be that of relatives, or friends that are connected with portable devices (e.g., PDAs or smart phones) in high-performance wireless technology such as IEEE 802.11a WiFi, if they stay in other parts of the home, or by more limited phone technologies (such as GPRS), if they are out of the home, in the office or everywhere around the world.

In this paper we propose the integration of several modules to build up a complete prototype: some modules have been directly inspired to the current literature, others have been originally defined to process videos, interpret people behavior and communicate events remotely to low-power devices, such as PDAs. The novelties and the meaningful points of the work, as well the system architecture itself, can be summarized as follows:

- A fast and robust module for moving objects and people detection and tracking, working on videos from fixed cameras and based on color background modeling and background suppression, and capable of dealing with background changes and shadows. We called this approach Sakbot [3, 4].
- A new approach to people posture classification that exploits a supervised learning paradigm to create probability masks.
- A smart video server, implementing new semantic transcoding techniques to connect directly with PDA clients. 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.

2. RELATED WORKS

In this work, we do not consider neither multiple-camera surveillance systems, nor moving or PTZ cameras. Focusing on static cameras, most of the works of video-surveillance are based on background modeling and background suppression. They generally differs in the model adopted to generate and update the background and in the way the moving pixels are connected in moving blobs. Most of non-trivial background models use *statistical* function over the history set of frames: Mode [18], Median [10], Mixture of Gaussians [23, 20], PCA reduction [15], and so on, have been proposed. Moreover, the background model is often *selectively* updated according with the knowledge extracted from previous frames: for instance, pixels or objects that have been detected in motion in previous frames are not used to update the background [7, 2].

In indoor applications, moving objects are mostly people, furniture and other objects that can be easily discriminated from persons with very simple geometrical considerations. Instead, due to the non-rigid human motion, the presence of occluding surfaces and possibly of more actors intersecting each other, video surveillance systems have deeply considered the problem of *tracking*. Traditionally, Kalman-based tracking systems [11] are adopted to predict the position of the moving objects in successive frames. They assume a rigid-motion model and possibly with a predictable direction. It can be applied with success also in indoor people's scenes whenever the scene is very simple, possibly with a single moving person, with few additional assumptions. Moreover, new approaches based on probabilistic tracking allow to cope with more complex situations [21]. In our system, we adopted an approach of appearance and probabilistic tracking that is inspired to the work of Senior [17]. They proposed a method that computes the probabilistic mask and the color appearance model of tracks in a very simple, and consequently quick, way.

Another enormous amount of literature works are related to the topic of *human motion capture* (HMC), including body modeling and posture detection. In its survey, Moeslund [13] classifies HMC approach in three categories: model-free, active and passive model based. Model-based approaches exploit a specific knowledge about the human body parts and their relationships, and often work in 3D space [12], or exploit very complex graph description (see, for instance, the decomposable triangulated graphs of [19]). Instead, model-free methods adopt a general pattern recognition paradigm: appearance features are extracted in 2D images and a classifier in the feature space is constructed. In this case, the efficiency of the method is affected by the significance of the selected features and the power of the classifier. In [8], the used feature is called *Star Skeletonization* and is a distance of the extremities of the silhouette from the blob's center of gravity. In [9] the vertical and horizontal projections of the blob are used to recognize people posture. For the classification of four posture classes (lying, crawling, sitting, standing), a simple nearest neighbor classifier, that compares the current projections with four manually tuned models, is adopted. Then, the system in [9] is refined with a body modeling in order to recognize salient parts of the body. The same horizontal projections have been used also in [6] to recognize parts of the body. Three projections are used: the first for head, the second for the torso and the third for the legs. We use a similar approach of [9, 6], with some suitable modifications by means of camera calibration, and a phase of supervised learning in order to construct probability masks to recognize the posture. Then, the posture is used in a Finite State Machine to recognize a dangerous situation.

Another key aspect of this work is the implementation of a remote connection with low-cost devices by means of semantic transcoding. *Semantic transcoding* is often employed in streaming servers to adapt the multimedia content and, specifically, reformat video code, in order to cope with user needs and user constraints [14]. Typically, variable compression, size, and color downscaling are useful to save bandwidth and to 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 [5], and in this paper 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. This is quite a different approach to VBR (Variable Bit Rate) implemented in MPEG-1 and MPEG-2 standards, because the semantics used to modify the bit rate is at an higher level and selected di-

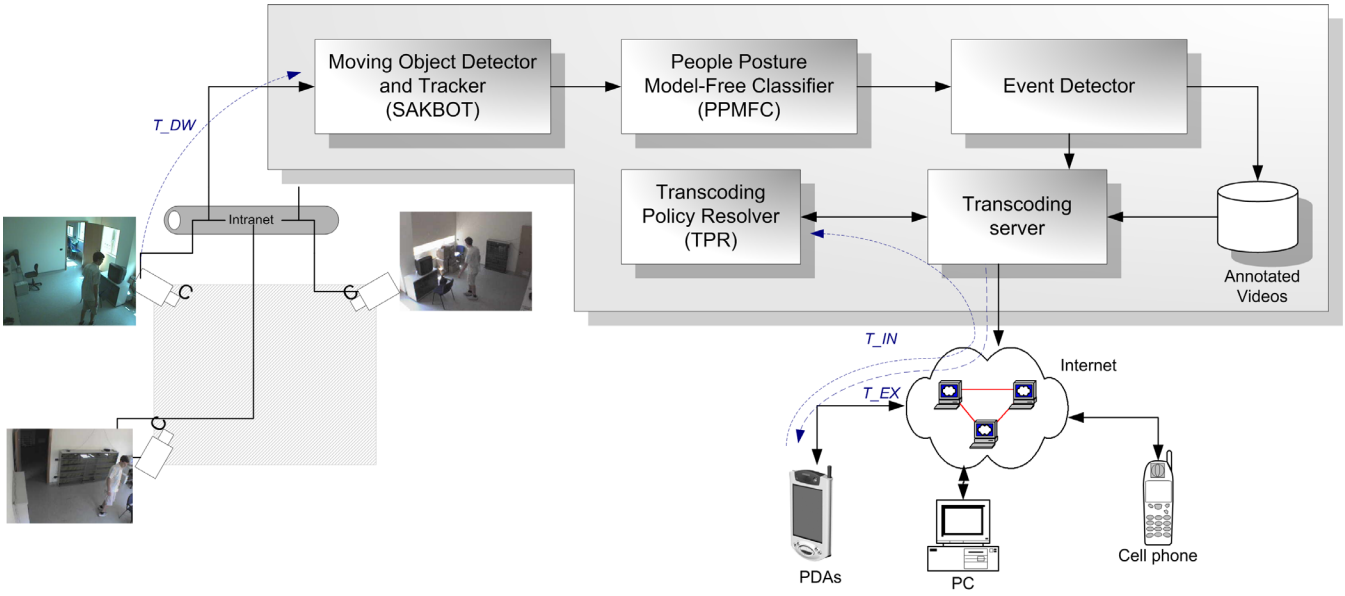

Figure 1: Scheme of the overall architecture.

rectly by the user. In such a manner the knowledge of segmented people and objects guides the adaptive compression. In [22], 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 [1] 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 domotic context and tailored for PDA.

3. SYSTEM ARCHITECTURE

Tele-presence and tele-viewing are essential for the future systems of people health care and in-house video surveillance. Our system is structured as a client-server architecture as in Fig. 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 (MVO, hereinafter), along with their features (shape, area, color distribution, average motion vector, and so on). These objects are tracked along time and processed to first classify them; the objects classified as people are further processed to detect their posture in the second module (*PPMFC*) 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. 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.

4. PEOPLE AND EVENT DETECTION

The detailed version of Fig. 1 is reported in Fig. 2. Each of the three modules of the server side will be described in the following.

4.1 Moving Object Detection and Tracking with SAKBOT

As above mentioned, the aim of the first module is to extract objects, to track them along time and to classify them into people tracks. According with the current literature, our system is based on background subtraction and models the background using statistics and knowledge based assumption: therefore, we called our approach *SAKBOT* (Statistical And Knowledge Based Object deTector) [3, 4]. In fact, the background model is computed frame by frame by using a statistical function (temporal median) to use, for each pixel, the most probable RGB value, but also by taking into account the knowledge acquired on the scene in the previous frames. In practice, the background model is updated differently if the considered pixel belongs to a previously detected MVO: in this case, the background model is kept unchanged because the current value is surely not describing the background. Moreover, if an object is detected as "stopped" (i.e., the tracking system detects that it was moving and then it stopped) for more than a "timeout" number of frames, its pixels are directly inserted into the background, without using the statistics. This updating process is summarized in the following equation:

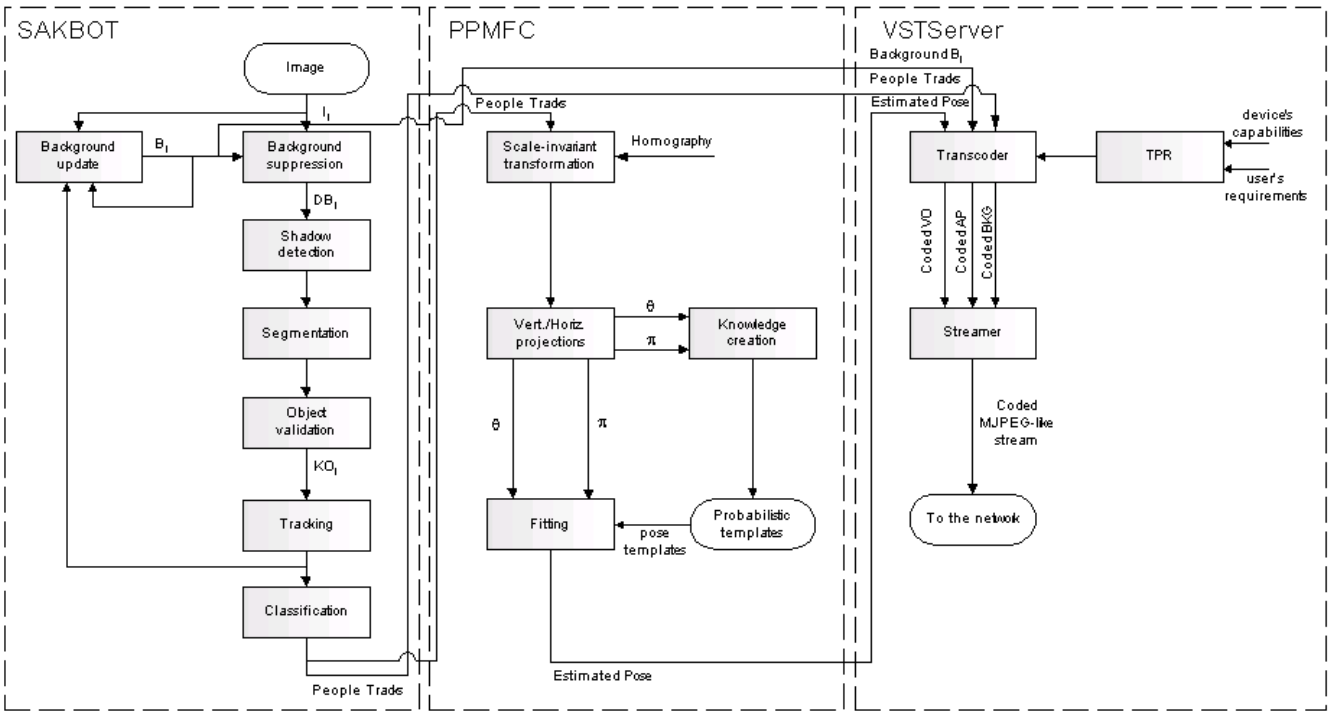

Figure 2: Detailed scheme of the proposed system.

$$\mathbf{B^{t+\Delta t}}(p) = \begin{cases} \mathbf{B^t}(p) & if \;\; p \in O, O\, in\, \{MVO^t\} \cup \{MVO\;\; shadow^t\} \vee \\ & O\, in\, \{Stopped\;\; MVO^t\} \wedge \\ & \neg(Timeout(O)) \\ \mathbf{I^{t+\Delta t}}(p) & if \;\; p \in O, \\ & O\, in\, \{Stopped\;\; MVO^t\} \wedge \\ & (Timeout(O)) \\ \mathbf{B_s^t}(p) & if \;\; p \in O, O\, in\, \{Ghost^t\} \cup \\ & \{Ghost\; Shadow^t\} \cup \{Unclas^t\} \end{cases} \quad (1)$$

For the *SAKBOT* taxonomy, each image point (p) belongs to an object (a labeled connected blob obtained after segmentation) of one of six possible classes at each frame t: the class of actually moving visual objects ($MVOs$); the shadow attached to the MVOs ($MVO\;\; shadow$); the false MVO, called "ghosts" [4], caused by a wrong background model[2] ($Ghost$ in Eq. 1); the shadow attached to ghosts ($Ghost\; Shadow$); among the MVOs, the stopped MVOs ($Stopped\;\; MVO$); and none of them ($Unclas$). $Timeout(O)$ is a boolean function that evaluate if the MVO is stopped for more than the specified timeout. In our specific application the timeout in the case of objects classified as a person is set to infinite, to avoid that a person sitting on a chair or lying down for long time is inserted into the background.

Eq. 1 says that the new background $\mathbf{B^{t+\Delta t}}(p)$ can be in some pixels equal to the previous $\mathbf{B^t}(p)$ if we have knowledge of that pixel as a foreground one, equal to the current frame $\mathbf{I^{t+\Delta t}}(p)$ if some events occur, or computed statistically as $\mathbf{B_s^t}(p)$. B_s is calculated as a *median* over the history set S of elements:

$$S = \{\mathbf{I^t}(p), \mathbf{I^{t-\Delta t}}(p), ..., \mathbf{I^{t-n\Delta t}}(p)\} \cup w_b\{\mathbf{B^t}(p)\} \quad (2)$$

As it is possible to note from Eq. 2, 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. Then, the statistical background model is computed by using the median function as follows:

$$\mathbf{B_s^{t+\Delta t}}(p) = \underset{i=1,...,k}{arg\,min} \sum_{j=1}^{k} Distance(\mathbf{x_i}, \mathbf{x_j}) \quad \mathbf{x_i}, \mathbf{x_j} \in S \quad (3)$$

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

$$Distance(\mathbf{x_i}, \mathbf{x_j}) = max(|x_i.c - x_j.c|) \; with\, c = R, G, B. \quad (4)$$

Shadows are detected by means of an appearance model that relies on the fact that cast shadows darken the background that they cover, but change slightly the color. After detected, the shadows are used to get the above-mentioned classification and to separate them from objects. An object validation task is performed to remove small objects and to distinguish between real and apparent (ghost) moving objects. More details can be found in [4]. In our experiments, the median function has proven effective while, at the same time, of less computational cost than the Gaussian or other complex statistics. Typical values are n=7, w_b=2 and Δt ranging between 5 and 30.

The tracking algorithm used in the system has been firstly presented in [17] and is based on two information stored for each pixel: its color and the probability of the correctness of that value. We perform an object (O) to tracks (T) matching: objects extracted by the background suppression module are considered and associated to the estimated position of the tracks, that are logical objects

[2]Examples are objects in the background like a door that starts its motion and leaving a "ghost" blob in its initial position. Ghosts and MVOs are discriminated with a validation step that uses the average difference of the object's pixels with previous frame.

present in the scene, by checking if their bounding box distance is sufficiently low (with respect to a parameter experimentally set):

$$BBd\left(O_i, T_j\right) = \min\left(d_B\left(c\left(O_i\right), BB\left(T_j\right)\right), d_B\left(c\left(T_j\right), BB\left(O_i\right)\right)\right) \quad (5)$$

where d_B is the minimum distance between a point and a rectangle, and c identifies the centroid of the object or the track. This produces a track-to-object correspondence matrix and many cases arise: to cope for these, if many objects correspond to the same track, the objects are merged into a single macro-object that comprises many connected components. At this point we can have three cases:

1. no track corresponds to the macro-object: a new track is generated;
2. only one track corresponds to the macro-object: the track is fitted to the image and updated using the current object.
3. many tracks correspond to the macro-object: if an ordering is available, it is exploited to fit object images to the current image, using only pixels not assigned to front objects, else the model is fitted to the whole image.

After fitting, contended points are assigned to the most similar objects, that is the ones with minimum $p_i\left(\mathbf{x}\right)$:

$$p_i\left(\mathbf{x}\right) = p_{RGB_i}\left(\mathbf{x}\right) P_{C_i}\left(\mathbf{x}\right) P_{NO}\left(i\right) \quad (6)$$

where

$$p_{RGB_i}\left(\mathbf{x}\right) = \left(2\pi\sigma^2\right)^{-\frac{3}{2}} e^{-\frac{\|I(\mathbf{x})-M(\mathbf{x})\|^2}{2\sigma^2}}. \quad (7)$$

$I\left(\mathbf{x}\right)$ is the image pixel, $M\left(\mathbf{x}\right)$ is the RGB model, $P_{C_i}\left(\mathbf{x}\right)$ is the probability for the point to belong to the object and $P_{NO}\left(i\right)$ is the probability that the object is not occluded by others. In Fig. 3 we can see the appearance RGB model $M\left(\mathbf{x}\right)$ and the probability mask. See for instance different probabilities between parts (as the torso) that are constant and parts (as legs and arms) that are continuously moving.

Figure 3: Example of appearance model and related probability mask.

After the assignment of pixels, the algorithm proceeds as for the "one object" case by updating the bounding box, the probability mask with $P_C\left(\mathbf{x}, t\right) = P_C\left(\mathbf{x}, t-1\right)\lambda + \left(1-\lambda\right)F\left(\mathbf{x}\right)$, where $F\left(\mathbf{x}\right) \in \{0,1\}$ is the foreground pixels mask and $\lambda \in [0,1]$, then updating the appearance model $M\left(\mathbf{x}, t\right) = M\left(\mathbf{x}, t-1\right)\alpha + \left(1-\alpha\right)I\left(\mathbf{x}\right)$, with $\alpha \in [0,1]$ for foreground pixels, and finally by shrinking the bounding box to contain only pixels that belong to the probability mask over a certain threshold. In indoor situation, considering the average speed of people moving in the room and the frame rate, we set $\alpha = \lambda = 0.9$. Please note that one of the more amazing properties of this probabilistic approach is to handle occlusions: the occluded parts of an object will not disappear in the appearance model, but it will only decrease their probabilities in the model. As far as the object is not occluded for too much time to bring the probability below the threshold, the object will be detected correctly.

The position's changes with respect to the previous frame are stored and used to estimate the next position according to the following equation:

$$\mathbf{v}_{EST} = \frac{3}{N\left(N+\frac{1}{2}\right)\left(N+1\right)} \sum_{t=1}^{N} t^2 \mathbf{v}_t \quad (8)$$

where $\mathbf{v}_t$ are the previous estimates and N is the number of observations available for that objects.

Tracks that have not been observed for a number of frames are deleted and additional rules are used for splitting and merging tracks, based on position and motion similarity for merging and on blob's separation inside the track for splitting.

4.2 People identification and pose estimation

After the tracking, a simple classification algorithm is used to detect people with respect to the other objects. This algorithm is based on assumptions on area (after calibration correction) and shape factors on the appearance tracks.

Once people tracks are extracted by the low-level module, they are passed to the high-level module whose aim is the determination, for each track, of the person's posture. As stated in section 2, one class of possible approaches to this problem does not exploit any information on the human body model, and, thus, is called model-free. Accordingly, our proposal, detailed in the scheme on the middle of Fig. 2 is called *PPMFC*, that stays as *People Posture Model Free Classifier*.

The posture detection is mainly based on the probabilistic classification of the vertical and horizontal projection histograms of the person's blob, similarly to the work reported in [9]. However, with respect to that work, we introduced some modifications:

- in our setup, people are used to move towards or away from the camera, and, thus, the perspective is likely to change significantly the blob's size. As a consequence, the projection histograms are not distance-invariant, causing many problems in the classification phase. In [9], the authors proposed a normalization phase after the computation of the projection histograms; unfortunately, this process makes the "standing" and "crawling" posture very similar and hard to be distinguished. We exploited camera calibration to compute the distance between the person and the camera and we use it to scale the person track.
- differently from [9], we exploited a supervised learning phase in which we acquire the probabilistic templates (see Fig. 2) that describe each posture modeled by the PPMFC. These templates are then used during the testing phase as a comparison to detect the most probable posture. Please note

that these templates must be computed only once for each camera installation, i.e. they are only dependent on the camera position and orientation. Moreover, normally to make the classifier work properly, the training set must be large enough; otherwise, the probabilistic templates are likely to be sparse. As a consequence, even if the current histogram is very similar to those used during the learning phase, the similarity between histogram and template could be very low. To solve this problem and make the classifier working also with a small training set, we used dense probabilistic templates, as reported below.

Starting from the person's blob B (defined as a cloud of points within a bounding box of size (B_x, B_y)), the projection histograms are computed as follows:

$$\theta(x) = \#\left\{(x_p, y_p) \in B | x_p = x\right\} \text{ where } x \in [0, B_x - 1] \quad (9)$$

$$\pi(y) = \#\left\{(x_p, y_p) \in B | y_p = y\right\} \text{ where } y \in [0, B_y - 1] \quad (10)$$

where θ is the horizontal projection histogram and π is the vertical one. Fig. 4 reports an example. B blobs and, consequently, projection histograms are scaled with a scale factor $sd = \frac{d}{D}$, computed on the homography. Thanks to an initial camera calibration, the homography image can be computed ([16]). On the original image the blob's support point is extracted, as the lowest point of the blob. The support point's position in the homography gives us an approximation to the real 3D distance d, that scaled with a fixed upper bound D distance produces the scaling factor sd.

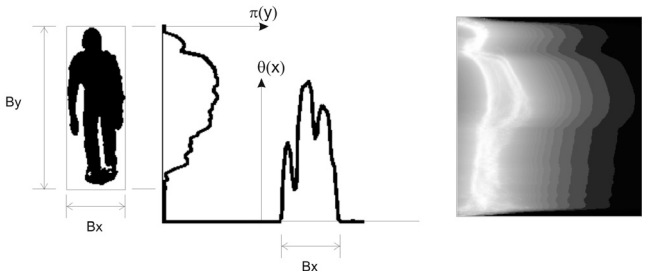

Figure 4: An example of horizontal (θ) and vertical (π) projection histograms.

During the learning phase *bi-dimensional probability density maps* are constructed. Let $B_i^t = \{(x, y)\}$, $t = 1, ..., T_i$ be a training set of T_i 2D blobs referred to the i-th posture class, and let $P_i^t = (\theta_i^t, \pi_i^t)$ be the couple of its projection histograms. All projections are not of the same size, but after the scaling they are aligned according with the blob's centroid. We construct the couple of 2D probability density maps of the state i as follow:

$$\Theta_i(x, y) = \frac{1}{T_i} \sum_{t=1}^{T_i} g\left(\theta_i^t(x), y\right) \quad (11)$$

$$\Pi_i(x, y) = \frac{1}{T_i} \sum_{t=1}^{T_i} g\left(x, \pi_i^t(y)\right) \quad (12)$$

where $g(x, y)$ is defined as:

$$g(x, y) = \frac{1}{|x - y| + 1} \quad (13)$$

Each point of a map is increased according with its distance from the histogram. The number 1 at the denominator is inserted to avoid dividing by zero. $\Theta_i(x, y)$ is the probability that an horizontal histogram θ having $\theta(x) = y$ belong to the class i. To better understand these equations refer to Fig. 4, showing the vertical probability map obtained for the standing posture after 20 training tracks.

These maps can be interpreted as an a priori conditional probability distributions. With reference to Fig. 4, for instance, a point $\Pi(x_0, y_0)$ indicates the probability (conditioned to belong to the class standing) to have x_0 points at the y_0 coordinate of the blob.

At run-time, the computed projection histograms are compared with the probability maps stored during the training phase, in the Fitting Module of Fig. 2. For each pose i, a measure of similarity S_i is extracted. In [9] also the templates are histograms, and the similarity is computed with a logarithmic likelihood formula on manually tuned histograms templates. By doing that, all the points of the template histograms have the same weight. Conversely, by using the probability maps, the most reliable parts of the histograms are highlighted.

Let $\tau_i = (\Theta_i, \Pi_i)$ be a pair of probability maps for the class i and $P = (\theta, \pi)$ the projection silhouette of the track to be classify. We compute the two similarity values S_i^θ and S_i^π as follows:

$$S_i^\theta = \frac{1}{B_x - 1} \sum_{x=0}^{B_x - 1} \Theta_i(x, \theta(x)) \quad (14)$$

$$S_i^\pi = \frac{1}{B_y - 1} \sum_{y=0}^{B_y - 1} \Pi_i(\pi(y), y) \quad (15)$$

The final score S_i is computed as the correlation between the two scores $S_i^\theta \cdot S_i^\pi$. The estimated posture is that with the maximum value of S_i.

As in [9], we first considered only four possible posture, namely "standing", "lying", "sitting", and "crawling". Examples are reported in Fig. 5.

Moreover, since the silhouettes of people sitting with a frontal, left-headed or right-headed view are different, internally the system splits each state in further three view-based subclasses, for a total of 12 classes. Once the main state is detected, the Finite State Machine reported in Fig. 6 is used to detect state transitions and, thus, events.

Transitions between "static" and "moving" states are guided by the track motion condition. When a track is in moving status its posture is classified and a changing posture causes a state transition. The permanence in the lying status for a defined period generates an alarm.

5. TRANSCODING SERVER AND PDA CLIENT APPLICATION

As depicted in Fig. 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 mo-

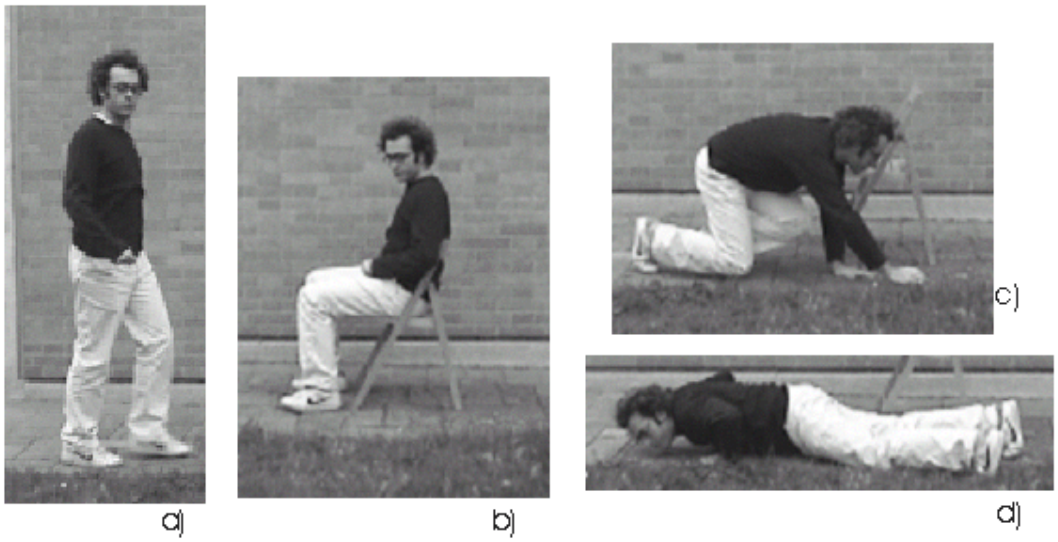

Figure 5: Examples of the four different postures considered: (a) standing, (b) sitting, (c) crawling, and (d) lying.

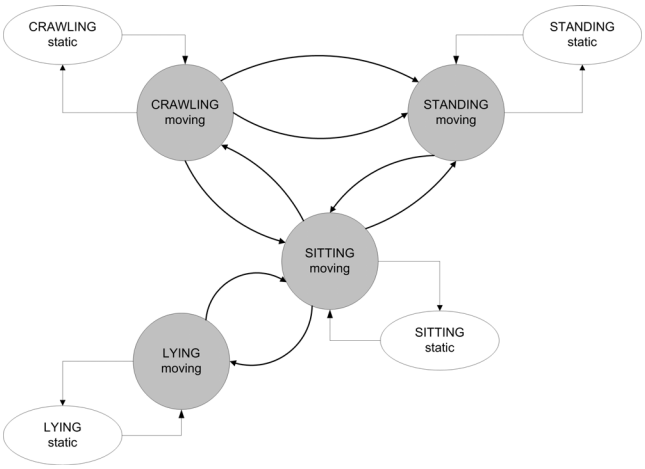

Figure 6: The Finite State Machine used for detecting events.

ment, 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 semaphore-based protocol to obtain the synchronization between the two threads.

As we saw in the previous Section, images coming from the camera are processed to detect dangerous situations or events. The corresponding alarms (in our case the person's falls) can be managed in several ways. For example, a control center can be advised and connected through a video-audio link with the assisted person. Obviously, all the events can be saved on a database for further processing. Besides, a vocal message or a SMS can be sent to a relative or a neighbor on their cell phone or PDA, and, in this latter case, a link for a low-bandwidth video connection to assert person's conditions can be provided. In this context, due to the limited computational, storage, and display capabilities of the PDA device and to the probable low-bandwidth connection, a video adaptation is mandatory. Typically, a syntactic adaptation of the video (by means of frame size reduction, frame skipping or quality deterioration) is used. Our claim is that in extreme applications like this, in which at one side the connection bandwidth can be very limited (like for GPRS), and, at the other side, a good image quality can save lives, normal only-syntactic downloading or transcoding methods are not effective. In previous works [1, 5], we proposed *semantic-based* techniques for video content adaptation (or video *transcoding*). The rationale is that 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 $C =< o_i, e_j >$, where o_i represents an object class and e_j is an event class, selected between the set of object classes O and event classes E detectable by the system:

$$O = \{o_1, o_2, ..., o_n\} \cup \{\widetilde{o}\} \quad ; \quad E = \{e_1, e_2, ..., e_m\} \cup \{\widetilde{e}\}$$

The special class $\widetilde{o}$ includes all the areas of the image that do not belong to user-defined object classes (for example, the background is considered as $\widetilde{o}$). 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 $O = \{people, background\}$ 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. 7. The stream is embodied in an HTTP connection with a multi part MIME header (to be compatible with network camera standards) 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 7: The data stream.

Thus, the background and the VOIs (Visual Object Images) 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 [1].

At the client side, this stream is decoded, background and VOI 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.

6. EXPERIMENTAL RESULTS

The architecture we have 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* updates 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, MVO feature computation, and people classification at each frame and the speed is proportional to the number and size of MVOs 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 does 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 depends 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 to not introduce delays (in average, depending on the number of objects), thanks to the multi-threaded application. In fact, besides the obvious limitation of running multiple threads on the same processor, using two de-coupled threads, one to handle the image acquisition and one to perform computation, allows us to avoid slowdowns or even deadlocks due to slow clients. 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. 10) with semantic transcoding only, we achieve an average bandwidth of 88 kbps. With GPRS (about 55 kbps) we must add a temporal downscaling 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 MVOs, shadows, ghosts [4]. 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.

A more precise analysis at cognitive level can be done at the level of posture classification. We tested the system in different environments, with five network cameras, installed in various rooms and laboratories. The performance comparisons have been conducted in two phases, learning and testing, comparing situations obtained in the same environment or in different ones. In a domotic application, the idea of learning the motion models on a person and then using those models on himself is logical and could be common practice. We also tested the performance of using models of a different person to verify the robustness of the scheme. Classification

fr. 66

fr. 232

fr. 340

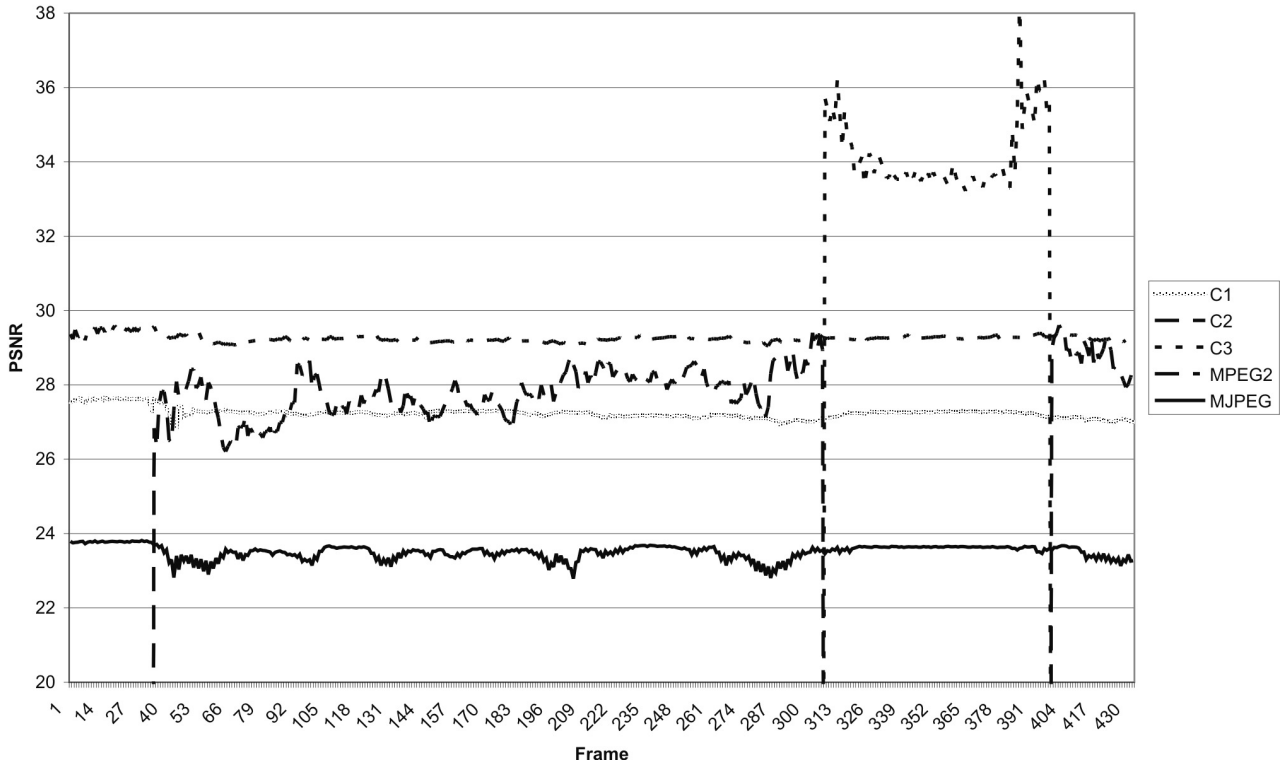

Figure 10: PSNR results for each frame, divided into classes and comparison with the other algorithms

Video	Model	Frames	Efficacy Uncalibrated	Efficacy Calibrated
Luca1	Luca2	1318	93,10%	99,01%
Luca2	Luca2	1561	96,83%	98,37%
Roberto1	Roberto1	742	67,79%	98,30%
Roberto1	Luca2	449	89,31%	93,03%

Figure 8: Posture classification results

results are reported in Fig. 8. The last row shows results obtained surveilling an actor (Roberto) different from the one (Luca) used to learn the postures. The last column reports the improvement achieved by introducing calibration and blob scaling.

To better understand the main sources of errors, we report in Fig. 9 the confusion matrices of the posture estimator. Actually, the confusion matrices should be discussed on the 12 view based postures, but for brevity we have aggregated the results showing only the four main postures. Principally, the two mistaken postures are standing and crawling, because the transitions between them are very difficult to classify also for a human observer.

For the transcoding results, we set the following classes:

$$O = \{person, chair, door\} \cup \{\widetilde{o}\}$$
$$E = \{no\_motion, O\_moving, P\_lying\} \cup \{\widetilde{e}\}$$

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. 10, it is possible to see the performance of our system on the three classes

Video: Luca1

Classifier \ Ground Truth	Standing	Sitting	Laying down	Crawling
Standing	**1048**	0	0	0
Sitting	0	**66**	0	1
Laying down	0	0	**0**	0
Crawling	39	2	0	**53**

Video: Luca2

Classifier \ Ground Truth	Standing	Sitting	Laying down	Crawling
Standing	**349**	0	0	4
Sitting	0	**55**	0	0
Laying down	0	0	**100**	2
Crawling	0	2	0	**219**

Video: Roberto1

Classifier \ Ground Truth	Standing	Sitting	Laying down	Crawling
Standing	**113**	0	0	1
Sitting	0	**43**	0	0
Laying down	0	0	**93**	0
Crawling	0	0	0	**40**

Video: Roberto2

Classifier \ Ground Truth	Standing	Sitting	Laying down	Crawling
Standing	**122**	0	0	0
Sitting	0	**0**	0	0
Laying down	1	0	**219**	0
Crawling	1	0	2	**71**

Figure 9: Confusion matrices

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. 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.

7. CONCLUSIONS

As above mentioned, our paper reports the description of a prototype system that has three main characteristics:

1. it allows **video accessibility to low-resource devices**, such as PDAs. This is obtained by applying effective transcoding policies able to keep the ratio quality/bandwidth as high as possible. Moreover, transcoding process adapts the video to the device's capabilities;

2. it provides **best quality for the defined scopes of the user**. As reference application, we used the in-house video surveillance for elders and disabled people's health care. This goal of the system requires to detect events and situations of particular relevance;

3. it exploits **computer vision techniques to extract semantics from the video**. This is achieved using a fast and robust moving object detection and a supervised learning based posture classifier.

Acknowledgments

We would like to thanks Luca Panini and Rudy Melli for their help. The project is founded by Fondazione Cassa di Risparmio, Modena, Italy and the FIRB Italian Project on Performance Analysis.

8. REFERENCES

[1] M. Bertini, R. Cucchiara, A. D. Bimbo, and A. Prati. Object and event detection for semantic annotation and transcoding. In *Proceedings of IEEE Conference on Multimedia & Expo*, volume 2, pages 421–424, 2003.

[2] A. Cavallaro, O. Steiger, and T. Ebrahimi. Semantic segmentation and description for video transcoding. In *Proceedings of IEEE Conference on Multimedia & Expo*, volume 3, pages 597–600, 2003.

[3] R. Cucchiara, C. Grana, G. Neri, M. Piccardi, and A. Prati. The sakbot system for moving object detection and tracking. In *Video-based Surveillance Systems - Computer Vision and Distributed Processing*, chapter 12. Kluwer Academic, 2001.

[4] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003.

[5] R. Cucchiara, C. Grana, and A. Prati. Semantic video transcoding using classes of relevance. *International Journal of Image and Graphics*, 3(1):145–169, Jan. 2003.

[6] A. Datta, M. Shah, and N. D. V. Lobo. Person-on-person violence detection in video data. In *Proceedings of Int'l Conference on Pattern Recognition*, volume 1, pages 433–438, 2002.

[7] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of IEEE ICCV'99 FRAME-RATE Workshop*, 1999.

[8] H. Fujiyoshi and A. Lipton. Realtime human motion analysis lly by image skeletonization. In *Proceedings of IEEE Workshop on Applications of Computer Vision (WACV)*, 1998.

[9] I. Haritaoglu, D. Harwood, and L. Davis. Ghost: a human body part labeling system using silhouettes. In *Proceedings of Int'l Conference on Pattern Recognition*, volume 1, pages 77–82, 1998.

[10] I. Haritaoglu, D. Harwood, and L. Davis. W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, Aug. 2000.

[11] Y.-K. Jung and Y.-S. Ho. Traffic parameter extraction using video-based vehicle tracking. In *Proceedings of IEEE Int'l Conference on Intelligent Transportation Systems*, pages 764–769, 1999.

[12] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal of Computer Vision*, 53(3):199–223, July-August 2003.

[13] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, Mar. 2001.

[14] K. Nagao, Y. Shirai, and K. Squire. Semantic annotation and transcoding: Making web content more accessible. *IEEE Multimedia*, 8(2):69–81, April-June 2001.

[15] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, Aug. 2000.

[16] F. Olivier. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, Mass., 1993.

[17] A. Senior, A. Hampapur, Y.-L. Tian, L. Brown, S. Pankanti, and R. Bolle. Tracking people with probabilistic appearance

models. In *Proceedings of International Workshop on Performance Evaluation of Tracking and Surveillance (PETS) systems*, 2002.

[18] A. Shio and J. Sklansky. Segmentation of people in motion. In *Proceedings of IEEE Workshop on Visual Motion*, pages 325–332, 1991.

[19] Y. Song, L. Goncalves, and P. Perona. Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):814–828, July 2003.

[20] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, Aug. 2000.

[21] H. Tao, H. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, Jan. 2002.

[22] A. Vetro, T. Haga, K. Sumi, and S. H. Object-based coding for long-term archive of surveillance video. In *IEEE Conference on Multimedia & Expo*, volume 2, pages 417–420, 2003.

[23] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, July 1997.