Pattern Recognition Letters 30 (2009) 157–167

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Video object matching across multiple independent views using local descriptors and adaptive learning

Luis F. Teixeira *, Luis Corte-Real

INESC Porto, Faculdade de Engenharia, Universidade do Porto, Campus da FEUP, Rua Dr. Roberto Frias, No. 378, 4200-465 Porto, Portugal

ARTICLE INFO

Article history:
Available online 3 June 2008

Keywords:
Video object matching
Multiple cameras systems
Local descriptors
Vocabulary tree
Adaptive learning
Visual surveillance

ABSTRACT

Object detection and tracking is an essential preliminary task in event analysis systems (e.g. visual surveillance). Typically objects are extracted and tagged, forming representative tracks of their activity. Tagging is usually performed by probabilistic data association, however, in systems capturing disjoint areas it is often not possible to establish such associations, as data may have been collected at different times or in different locations. In this case, appearance matching is a valuable aid.

We propose using *bag-of-visterms*, i.e. an histogram of quantized local feature descriptors, to represent and match tracked objects. This method has proven to be effective for object matching and classification in image retrieval applications, where descriptors can be extracted a priori. An important difference in event analysis systems is that relevant information is typically restricted to the foreground. Descriptors can, therefore, be extracted faster, approaching real-time requirements. Also, unlike image retrieval, objects can change over time and therefore their model needs to be updated continuously. Incremental or adaptive learning is used to tackle this problem. Using independent tracks of 30 different persons, we show that the *bag-of-visterms* representation effectively discriminates visual object tracks and that it presents high resilience to incorrect object segmentation. Additionally, this methodology allows the construction of scalable object models that can be used to match tracks across independent views.

1. Introduction

In typical visual surveillance or ambient intelligence systems, event analysis is based on the tracking and identification of visual objects across multiple camera views. Objects are often captured in more than one view and it is desirable that these multiple instances of the same visual object can be automatically identified. Consider the following visual surveillance scenario: a system operator identifies a suspicious person crossing an area covered by a given camera and would like to be informed of where and when that person was previously captured by the system. To accomplish this, the surveillance system would have to track that person from the first moment it was captured by a camera and across all cameras whose field of view overlaps the person's path. The captured visual objects can be as diverse as people walking, riding a bicycle or doing other activities, cars crossing a road, etc. The tracking of multiple visual objects in one (usually static) view is a classic vision problem that has received much attention and finds application not only in surveillance systems but also in other machine vision scenarios, such as robotics. Incidentally, multiple-view tracking has only recently received much research activity. The main advantages of using many cameras for tracking in surveillance scenarios (Foresti et al., 2005; Wu et al., 2006) are an arbitrarily large coverage of any given area since, for most environments, a single camera is not able to provide adequate coverage; and, tracking performance improvement, especially in critical areas and where more robustness against occlusion is desirable (Mittal and Davis, 2003).

While a single-camera tracker searches for correspondences only between frames, the task of a multi-camera tracker is also to establish correspondences between observations of objects across cameras. The ultimate goal is to correctly tag all instances of the same visual object at any given location and at any given time instant.

Specific models can add constraints that simplify this task. For example, for human tracking, Hu et al. (2004) rely on the definition of principal axis. Moreover, for generic object tracking, a commonly used method is to find correspondences using *feature matching*. Matching colour or other features may be performed statistically using, for example, Kalman filtering (Utsumi and Ohya, 2000) or Bayesian network inference (Nillius et al., 2006; Qu et al., 2000). It is also possible to use camera calibration information to learn more about the camera geometry and derive

* Corresponding author. Fax: +35 1222094250.
E-mail addresses: luis.f.teixeira@inescporto.pt (L.F. Teixeira), lreal@inescporto.pt (L. Corte-Real).
URL: http://telecom.inescporto.pt/~lfpt (L.F. Teixeira).

0167-8655/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2008.04.001

additional constraints. Cai and Aggarwal (1999) use relative calibration between cameras and define correspondences using a set of feature points in a Bayesian probability framework. Different camera properties and illumination variations can contribute to different appearances of the same object in different cameras. To overcome this, Javed et al. (2005) proposed learning the subspace of inter-camera brightness transfer functions. Ross et al. (2007), on the other hand, presented a method that incrementally learns a subspace representation, adapting itself to appearance changes.

An alternative (or complement) to the previous method is to use a priori knowledge of the captured area's geometry, building a *3D model*. Multiple cameras are used to recover the homographic relations between each camera view. In this case, it is possible to find correspondences by projecting the location of each object in the world coordinate system, and establishing equivalences between objects that project to the same location at the same time. Likewise, equivalences between views are established by linking views that have similar projected 3D location. For example, Black et al. (2002) use this method as well as a combined 2D/3D Kalman filter for object tracking. Creating the 3D model is not always a simple task, and therefore these methods are mostly suitable for controlled environments. Alternatively, *alignment*-based approaches rely on recovering the geometric transformation between cameras automatically. This can be done using spatial image alignment methods and incorporating time information (Caspi and Irani, 2000) or by matching motion trajectories in different cameras (Lee et al., 2000). Khan and Shah (2003) propose finding the limits of the field of view of each camera that are visible by other cameras. Also, Zhao et al. (2005) propose a ground-based fusion method for camera handover using space-time constraints and stereo segmentation. However, using alignment requires overlapping fields of view which is not always feasible. To avoid using overlapping field of views, cameras are located in non-overlapping locations that nonetheless allow establishing path dependencies between them using probabilistic models (Kettnaker and Zabih, 1999; Javed et al., 2003).

In summary, most approaches of multi-camera tracking have one of the following rationales: *different angles of the same area*, to improve tracking performance in cluttered environments; *different but overlapping areas*, simplifying the camera handover of tracked objects; or *non-overlapping areas with some path correlation between them*, that can be previously defined or automatically learned.

However, in many situations it is important to keep track of persons or other video objects across independent areas, regardless of the time of capture, where it is not possible to rely on the aforementioned techniques. With this work, we propose using local feature and a *bag-of-visterms* (BOV) representation of objects which can reliably identify different visual objects in a visual surveillance scenario. Since the number of objects is unknown and existing objects can change their appearance, an incremental learning scheme is also used. The main contributions of this paper are therefore: (1) an efficient method of matching visual objects tracked across independent views using local descriptors and the visual word paradigm, (2) a scalable representation and learning of generic visual objects, and (3) a base for a surveillance system interface browsable by object tracks.

The paper is organized as follows: First, related work is discussed, as well as the options taken to accomplish our goals. Section 3 presents the proposed method to describe, match and learn new visual objects detected by a visual surveillance system. Results are presented in Section 4. Finally, in Section 5 we discuss how future extensions to this work can be used to interact with generic tracking systems and we provide a conclusion in Section 6.

2. Related work

Our work can be seen as a complement to classic tracking techniques rather than a direct alternative. It takes as inputs the results obtained by an object tracking algorithm and establishes correspondences between objects, independently of their location and the time of capture. Whereas most multi-camera tracking methods rely on geometry models, overlapping fields of view or on correlated paths between cameras, our method does not assume any of these priors.

A recent proposal by Madden et al. (2007) finds some common points with our work, namely the rationale of relying on feature matching to find track correspondences across multiple cameras. It also considers as inputs the segmentations and tags of each object track detected system-wide but a different representation is used. Scalability issues also are not approached, namely the storage of descriptors of increasing number of objects of tracks. With a very large number of tracks, the matching would require a one-to-one comparison between tracks which may become easily unfeasible – the authors only present results for a small number of objects/tracks.

Object matching using appearance features is a well studied problem, especially in the context of image retrieval. The joint use of interest point detectors and local descriptors for object detection, recognition and classification has grown significantly (Lowe, 2004). Building on top of these descriptors, Sivic and Zisserman (2003), Willamowski et al. (2004), and more recently Quelhas et al. (2007), showed the usefulness of relating image invariant local descriptors to visual words, or *visterms*. Sivic and Zisserman (2003) successfully applied this approach to retrieve shots from movies. They proposed a description scheme where descriptors are extracted from local affine-invariant regions and quantized in *visterms*, reducing noise sensitivity in matching. Objects are then matched and frames with similar content (i.e. visual objects) can be retrieved efficiently using inverted files. Nevertheless, this work still proposes solving a problem closely related to classic image retrieval and does not take into account inherent constraints of real-time video capturing systems, such as scalability – namely storage and computational restrictions.

In this article, we propose using local descriptors and the *bag-of-visterms* paradigm for object representation to match tracks detected in a visual surveillance scenario. For this specific case, we are usually interested only in the foreground objects that are captured by the system. Taking this into consideration, two differences arise when comparing this work to others like Sivic and Zisserman (2003): (1) descriptor extraction can be restricted to the foreground, which can greatly increase speed and (2) each object representation can change over time – for instance, a person can have different appearances, depending on the capture angle – which implies updating each object model continuously. These particular characteristics are explored and discussed throughout the rest of paper.

3. Scalable object matching and learning

In this section, we describe the proposed methodology to match tracked objects across independent views, that can be scalable in two distinct dimensions: an undetermined number of new objects can be added at any given instant and, existing object representations can be updated to reflect changes in time. At the same time, it needs to maintain performance at acceptable levels. As previously stated, when trying to find correspondences between objects in independent views, methods like alignment and path prediction are not suitable. Feature matching is therefore the best alternative.

Fig. 1 depicts an example of a typical scenario we would like to address: one person is detected and tracked; its description is

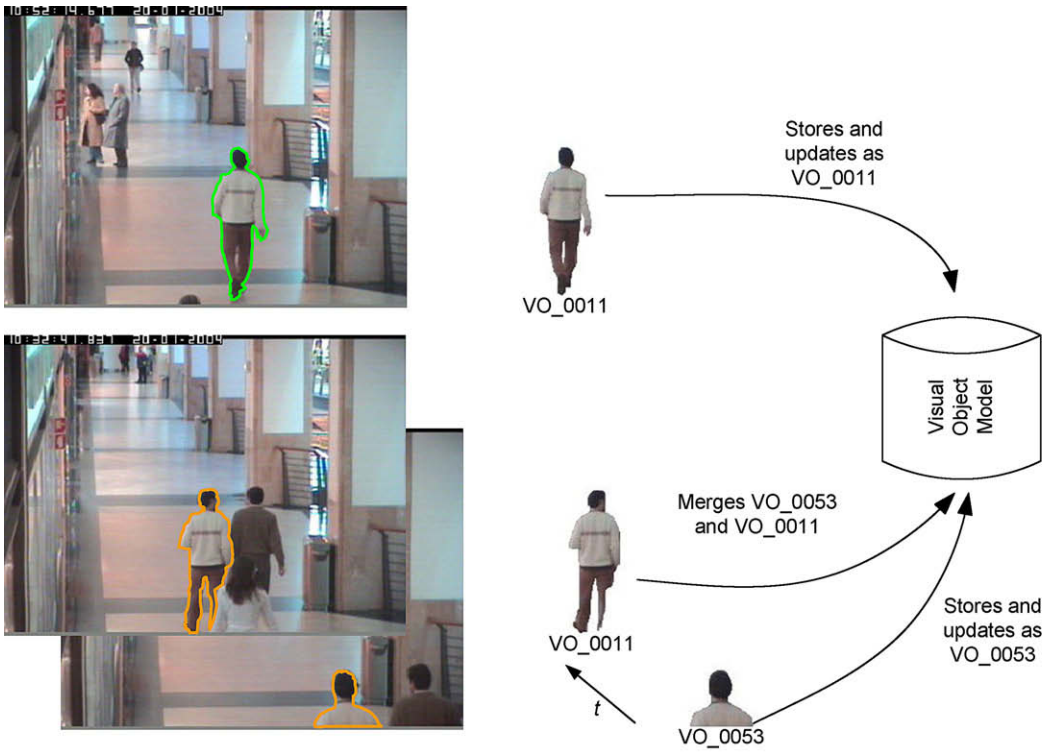

Fig. 1. The same person is tracked at different instants, although captured by the same camera; due to incomplete appearance information it is labeled incorrectly but as new information is acquired the label is corrected.

extracted and stored and an identification is assigned; in another instant, the same person is again detected and tracked; even if initially a different identity is assigned, when more information is processed, it is correctly identified. This process involves three distinct steps: (1) obtaining an object description, (2) creating and updating a model of the objects, and (3) classifying the object, given its captured track.

3.1. Describing visual objects

Choosing a representation scheme for visual objects which is compact and representative at the same time is important to assure a feasible solution to the object matching problem. The descriptors used to represent objects need to accommodate different appearances – due to different camera characteristics, capture position, illumination variations, etc. Madden et al. (2007) addressed this by a compact colour histogram representation dubbed major colour spectrum histogram representation (MCSHR) which describes the object's main colours. An incremental MCSHR (IMCSHR) is computed over a period of time to compensate for small, short-term changes in the object's pose. Finally, a transformation is applied to IMCSHR to compensate for illumination variations. The MCSHR description scheme is further discussed and evaluated in Section 4.2. As an alternative to the representation scheme used in our method, we also implemented MCSHR and evaluated it on the dataset presented in detail in Section 4.

Local descriptors are widely used for image retrieval applications, and we now propose to apply local descriptors to object track matching. For each object frame comprising the full object track, a set of descriptors is extracted in specific keypoints. These points are defined by a interest point detector (number of points depends on the image content) or by a random point selection (number of points is fixed and pre-defined). For object matching and classification, the number of words extracted from the image is the most important factor influencing performance (Nowak et al., 2006). Since objects' images are usually very small, interest point detectors tend to select an insufficient number of points to efficiently represent them. We used, therefore, random point selection from a pyramid with regular grids, as suggested by Nowak et al. (2006), since this method provides a dense representation of each object image.

A 128-dimensional vector is extracted in each keypoint using the SIFT descriptor developed by Lowe (1999). The descriptor is then quantized to form visual words using a pre-defined vocabulary. Features obtained by SIFT are invariant to image scale, rotation, and robust to changes in viewpoints and illumination. For most types of applications, it compares favourably with other local descriptor schemes (Mikolajczyk and Schmid, 2005). When affine invariance is required, SIFT is more prone to perform worse than more complex affine-invariant descriptors.

3.1.1. Building the vocabulary

Most BOV approaches rely on k-means clustering of local descriptors to create a vocabulary. The descriptor vectors of a training set are extracted and quantized into a pre-defined number of words. Nistér and Stewénius (2006) proposed an adaptation to this approach which, instead of creating a "flat" vocabulary, creates a hierarchical relation of visual words in the form of a *vocabulary tree*. This allows a more efficient search which, in turn, enables the use of large vocabularies. In our work, using a large vocabulary is a key factor since we will not be using any information of the geometric layout of visual words extracted from an image. On the other hand, given the rate at which frames are acquired and analysed, an efficient search of visual words is also required. This trade-off needs to be taken into consideration when building the vocabulary.

To build the vocabulary tree, an initial k-means clustering is first run on the training data, defining k cluster centres. The data

are then partitioned into k groups, where each group consists of the descriptor vectors closest to a particular cluster center, forming quantization cells, or *words*. The same process is then recursively applied to each group of descriptor vectors, splitting each quantization cell into k new parts. The vocabulary tree is created level by level, up to a maximum number of L levels.

When a new object's image needs to be classified, each extracted descriptor is propagated down the tree by comparing the descriptor vector at each level to the k candidate cluster centres and choosing the closest one. At each level k dot products are performed, resulting in a total of kL dot products. If k is not too large, the vocabulary tree can be very efficient, compared to an equivalent "flat" vocabulary defined by the total number nodes M in the tree, which is given by

$$M = \sum_{i=1}^{L} k^i = \frac{k^{L+1} - 1}{k - 1} - 1 \tag{1}$$

To account for different relevancies of tree nodes, a weight w_i is assigned to each node, and is defined by

$$w_i = \ln \frac{N}{N_i} \tag{2}$$

where N is the total number of images in the model and N_i is the number of images having at least one path through node i, i.e., containing the word represented by that node.

The training process of the tree uses a large set of descriptor vectors in an unsupervised fashion. Two data sources have been tested to build the vocabulary tree: (1) the segmented objects from the dataset's sequences (defined in Section 4.1) as training data forming a specific vocabulary $\mathcal{V}_s$ and (2) object frames from a different source as training data forming a generic vocabulary $\mathcal{V}_g$. By using both vocabularies, it is possible to assess the performance impact of having a generic vocabulary to represent a wide variety of objects.

3.1.2. Descriptor vector

All extracted 128-dimensional SIFT vectors are quantized in visual words using the vocabulary tree to define the final descriptor vector. For a given object c, its segmented image I_t^c at instant t is represented by

$$\upsilon(I_t^c) = \{x_1, x_2, \ldots, x_i\}, \quad i \in 1, \ldots, M \tag{3}$$

where M is the number of words in the vocabulary, given by Eq. (1). Each element x_i of (3) is the weighted histogram of the words defined by the vocabulary; x_i is, therefore, given by

$$x_i = n_i w_i \tag{4}$$

where n_i is the number of input descriptors containing the visual word i and w_i is the weight defined by Eq. (2). Unless stated otherwise, all experiments used a vocabulary with $k = 10$ and $L = 4$, resulting in a representation vector of size $M = 11{,}110$.

3.2. Adaptively updating and learning object models

In order to effectively update the previous model as new data becomes available, the model should (1) be compact, that would not imply having all previous history stored and (2) be scalable, that would imply classifying visual objects that change in time, while also having the ability to learn new ones. Recently, some methods have been proposed to solve this problem, such as Learn++ (Polikar et al., 2001). Learn++ draws its inspiration from AdaBoost, which in turn relies on an ensemble of classifiers trained using adaptive bootstrap techniques. It is iteratively updated by new sets of data, possibly containing new classes. A modification called Learn++.MT, proposed by Muhlbaier et al. (2004), improves the performance when new classes are added. Learn++.MT meets the compact and scalable requirements, and was used to create our visual object model. A brief description of it follows.

3.2.1. Learn++.MT algorithm

For each new dataset $\mathcal{D}_k$, the inputs to Learn++.MT are (1) a sequence of training data instances x_i and their correct labels y_i, (2) the classification algorithm `BaseClassifier`, and (3) T_k, the maximum number of classifiers.

As in typical boosting learning algorithms, data are drawn according to a distribution D_t. For the first set, D_t is initialized as a uniform distribution. From the second set onwards, this distribution is updated according to the performance of the ensemble on the new data. T_k classifiers are added to the ensemble when a new dataset is added. For each new classifier, a subset of $\mathcal{D}_k$ is drawn, according to D_t, and evaluated against the new classifier to obtain the hypothesis h_t. The classifier error is estimated by $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ and if $\epsilon_t > \frac{1}{2}$, a new subset is drawn, discarding the classifier. A dynamic weight voting (DWV) algorithm is then called to obtain a composite hypothesis H_t. It represents the ensemble decision of all classifiers trained until now. The distribution D_t is updated according to the performance of H_t. This process is repeated until all new T_k classifiers have been trained.

The essential difference from Learn++.MT to its parent method is the voting scheme. As in Learn++, voting is based on the weights assigned to each classifier but with DWV these weights are modified according to the classification of the specific testing instance. This is achieved by adjusting weights of classifiers that have not been trained with a given class. The adjustment is proportional to the ensemble's confidence on that class.

3.3. Classifying objects

The goal of object classification is to attribute a known label to each input object frame comprising that object's track. This can be seen as a multi-class classification problem, with a variable number of classes. As described previously, we use the Learn++.MT algorithm to handle class variability, and opted to use support vector machines (SVMs) (Burges, 1998) as its `BaseClassifier`.

SVMs are commonly used in machine learning problems, especially with large dimensional input spaces – which is the case for the object classification problem. Standard SVMs rely on margin optimization to learn a decision function $h(x)$, such that, if x belongs to the target class, $h(x)$ is large and positive; otherwise, $h(x)$ is negative. SVMs are thus binary classifiers but can be adapted to multi-class problems; we opted for the one-against–one approach, where $n(n-1)/2$ models are constructed for a n-class problem. Additionally, we used linear kernel SVMs, mainly for processing speed purposes. Specific kernels could alternatively be used, such as the pyramid match kernel proposed by Grauman and Darrell (2005) for multi-resolution histograms.

3.4. Summary

In summary, the main steps involved in the proposed solution for the surveillance scenario from Fig. 1 are (1) detect objects, including segmenting and tracking them for each view, (2) obtain a compact object description, yet sufficiently discriminative, and (3) compare the object with the current visual object model, updating it if appropriate. A block representation of these steps is shown in Fig. 2. While the first step is performed independently for each camera/view, the third step aggregates inputs from all views. The second step, on the other hand, combines object description which is performed independently for each view, and a common part to all views. This common part consists of obtaining a histogram representation of the *visterms* found in each object. A vocabulary of

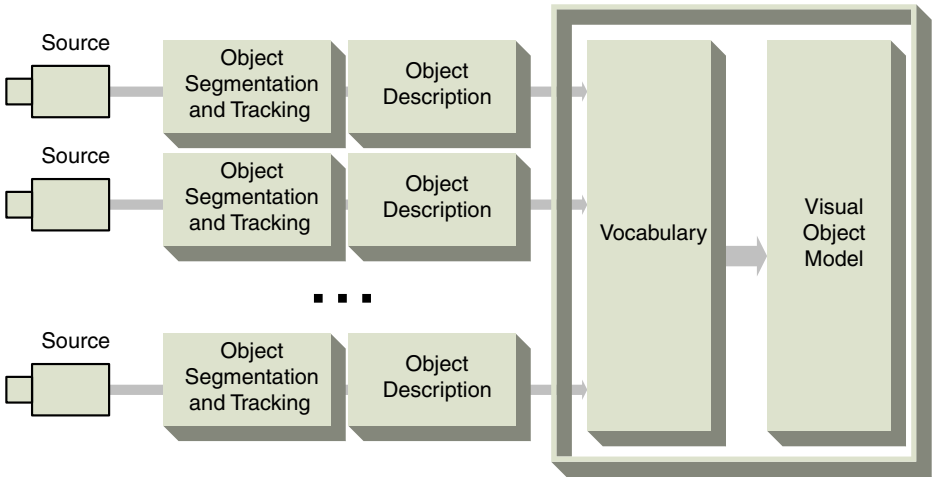

Fig. 2. Block representation of the method: objects detected in each view are first segmented and tracked; a description is extracted and quantized by a common global vocabulary; finally, the visual object model is updated accordingly.

visterms, with which object descriptors are compared, is obtained a priori and is kept unchanged.

Multiple appearances of an object are detected by matching a newly tracked object with the visual object model. The matching process takes the object tracks (i.e. sequences of segmented images) detected by any given tracking algorithm as inputs and finds multiple object occurrences. From an application point of view, it would act as a linking mechanism between sequences. An operator analysing a sequence containing an object would have the ability to know every other appearance of that object, and the possibility to jump to another sequence, pinpointed by time and location. Numerous other possibilities would than be possible, like using the previous knowledge of camera location (if any) to represent graphically the known path of an object/person.

4. Experimental setup and results

4.1. Dataset

In order to assess the performance of our methodology, a dataset was created. The dataset $\mathscr{D}_{30}$ consists of individual sequences containing 30 visual objects from the Shopping Center dataset of the EC Funded CAVIAR project IST 2001 37540.[1] We extracted each visual object from the 26 sequences that comprise the original set, using the provided CVML-based ground-truth information. The ground-truth consists of a bounding box defined in each frame, for each object.

The total number of images used to train and test the complete system is 14,506. This number includes 30 individual tracks of 30 different persons. Fig. 3 shows all visual objects composing the dataset. Individual tracks were extracted from the original video according to the bounding box defined in the ground-truth. These tracks include images with partial occlusions as well as severe occlusions by other persons (see Fig. 7 for an example). Additionally, we tested different tracks of the same person, captured at different instants and areas (see Fig. 4 for an example); they were however not used for training.

The generic vocabulary $\mathscr{V}_g$ was created with the PASCAL visual object classes (VOC) database.[2] All objects tagged as `PASperson` and all its variants (`Sitting`, `Standing` and `Walking`) were extracted from the VOC 2006 contest (Everingham et al., 2006) dataset; these images formed the generic dataset $\mathscr{D}_g$. The total number of images for this vocabulary training set is 2688.

The dataset $\mathscr{D}_{30}$ was further partitioned into subsets $\mathscr{D}^i$, $i \in \{1,2,\ldots,5\}$ to test the learning adaptability of the system. Each subset represents a time interval of 5 s (150 frames at 30 fps) of accumulated data. In Fig. 5, a representation of how the subsets are created is depicted. Note that no frames are repeated between sets and that all frames from $\mathscr{D}_{30}$ are included in the subsets, such that $\bigcup_{i=1}^{10}\mathscr{D}^i = \mathscr{D}_{30}$ and $\bigcap_{i=1}^{10}\mathscr{D}^i = \emptyset$. This incremental dataset simulates a typical visual surveillance scenario where objects are detected at different time instants. Initially only some objects are represented but, as new subsets are added new objects are added. Likewise, objects previously detected can no longer be represented in future subsets.

4.2. Discriminative capability

To test the discriminative capability of the representation scheme, $\mathscr{D}_{30}$ was divided in random training (fixed size) and validation (remaining) set partitions. We tested two representation schemes: MCSHR (Madden et al., 2007) and bag-of-visterms using a vocabulary tree defined from SIFT descriptors, which we designate hierarchical bag-of-visterms (HBOV). Both consist of a descriptor vector for each image but, while HBOV defines a fixed-size vector, MCSHR's size is variable. Due to this, evaluation consisted of a majority voting classification procedure. Each descriptor vector extracted from the test set images is compared with the training set descriptor vectors and votes are cast for the 10 closest matching, using a 1-norm distance metric. Results are obtained using a 5-fold cross validation. The SVM-based classification method described in Section 3.3 will only be used later.

The vocabulary for the HBOV was trained with dataset $\mathscr{D}_{30}$, i.e., with the same dataset to be described by the vocabulary. We tested two different vocabulary tree sizes: 125 leaf visual words ($k = 5$ and $L = 3$) and 10,000 leaf visual words ($k = 10$ and $L = 4$). The overall results are shown in Table 1.

The HBOV representation presents much better results, even with a descriptor size similar to the ones obtained with MCSHR. If we increase the vocabulary size, and hence the descriptor size, the results are further improved. Moreover, classification performance with MCSHR is less resilient to similar objects, as denoted in the confusion matrix shown in Fig. 6a in opposition to Fig. 6c. With HBOV, and especially with a larger vocabulary, all classes

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.
[2] http://www.pascal-network.org/challenges/VOC/.

Fig. 3. Images of all 30 visual objects used in the dataset. First row shows `Person[01-10]`, second row shows `Person[11-20]` and third row shows `Person[21-30]`. The dataset consists of a total of 14,506 images, with an average of 468.5 images per person.

Fig. 4. Same visual object captured at different instants, by different cameras.

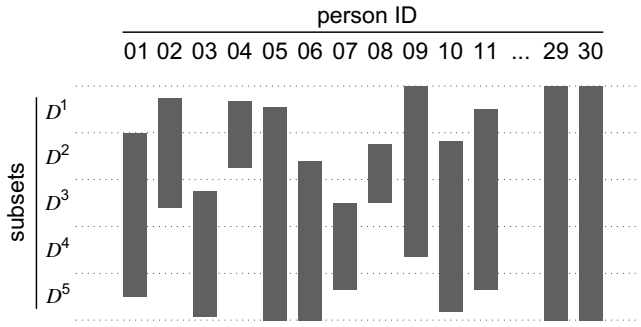

Fig. 5. Subsets $\mathscr{D}^i$, $i \in \{1,2,\ldots,5\}$ created to test incremental learning of objects. Each object starts to be captured in a given instant and leaves the field of view at another instant – the time interval when objects are is represented by the bars. Each dataset corresponds to 5 s of activity.

Table 1
Average classification rate and standard deviations using MCSHR and HBOV with different descriptor sizes

Vocabulary	Classification rate	Descriptor size
MCSHR	71.6 (17.7)	Variable, ~200
HBOV-125	82.9 (13.1)	Fixed, 155
HBOV-10000	95.0 (5.2)	Fixed, 11,110

present a very high classification performance, which clearly shows both the discriminative capability and stability of the representation scheme described in Section 3. The MCSHR scheme is used as a comparison baseline for the remaining experiments.

Since it is required that the descriptors are extracted in the shortest amount of time possible, it is also important to evaluate the time cost. For non-optimized implementations, while MCSHR descriptors can be extracted at a rate of approximately 3.7 typical track images per second,[3] HBOV-125 can be processed at 2.9 images per second and HBOV-10000 at 2.7 images per second. As expected, due to higher complexity, the HBOV extraction process is slower but nevertheless an optimized real-time implementation is equally feasible. Note that most of the time consumed in obtaining a HBOV description for an image is spent extracting the SIFT descriptor vector (≈2/3 of the total time). Due to this, the difference between HBOV-10000 and HBOV-125 is not very significant.

Despite a relatively high time cost, this method meets the needs of a real-time surveillance system since an optimized implementation could explore parallelization, redundancy of track images in short periods of time (<1 s) and inactivity.

[3] Results obtained with a C++ implementation running on a Pentium IV 3.4 GHz with 1GB of RAM.

4.3. Impact of using a generic vocabulary

The appearance of objects captured by a surveillance system is typically not known a priori. Hence, it is useful to create a vocabulary from images that are not directly related to the captured visual objects. We analyse the performance of such generic vocabulary with a similar setup as before. For each image of the generic dataset $\mathscr{D}_g$ 128-size SIFT descriptors were extracted in the points defined by a Laplacian of Gaussians (LoG) interest point detector (Lowe, 2004). The input for the vocabulary tree construction algorithm described in Section 3.1.1 consisted of the concatenation of all descriptor vectors from all images.

The dataset $\mathscr{D}_{30}$ was again divided in random training and validation partitions. In Table 2, we show the average performance of different types of vocabularies using in each case a single multi-class SVM. While the first four vocabularies were created with dataset $\mathscr{D}_{30}$, the last four were created with the generic dataset $\mathscr{D}_g$. For both groups we tested four different vocabulary sizes: 125 leaf words, with $k = 5$ and $L = 3$; 625 leaf words, with $k = 5$ and $L = 4$; 1000 leaf words, with $k = 10$ and $L = 3$; and, 10,000 leaf words, with $k = 10$ and $L = 4$. Results show that the use of a generic dataset to train the vocabulary does not affect significantly its performance.

All experiments henceforth use $\mathscr{V}_g^{10,000}$. Also, the SVM-based model trained with $\mathscr{D}_{30}$, using $\mathscr{V}_g^{10,000}$, will be designated $\mathscr{M}_{30}$.

4.4. Object representation adaptability

In the previous section, each person's data used to train the model consisted of information collected from a single track. However, it is important to know how the classification performance stands with new captured tracks. For that purpose we used $\mathscr{M}_{30}$ model to classify several different tracks. Table 3 shows the results obtained for eight new different tracks of six different persons. The first three new tracks have a similar viewing angle to the tracks used for training; the five other new tracks have a very different viewing angle and most of the time only the person's profile is visible (an example of both camera angles is shown in Fig. 4). The

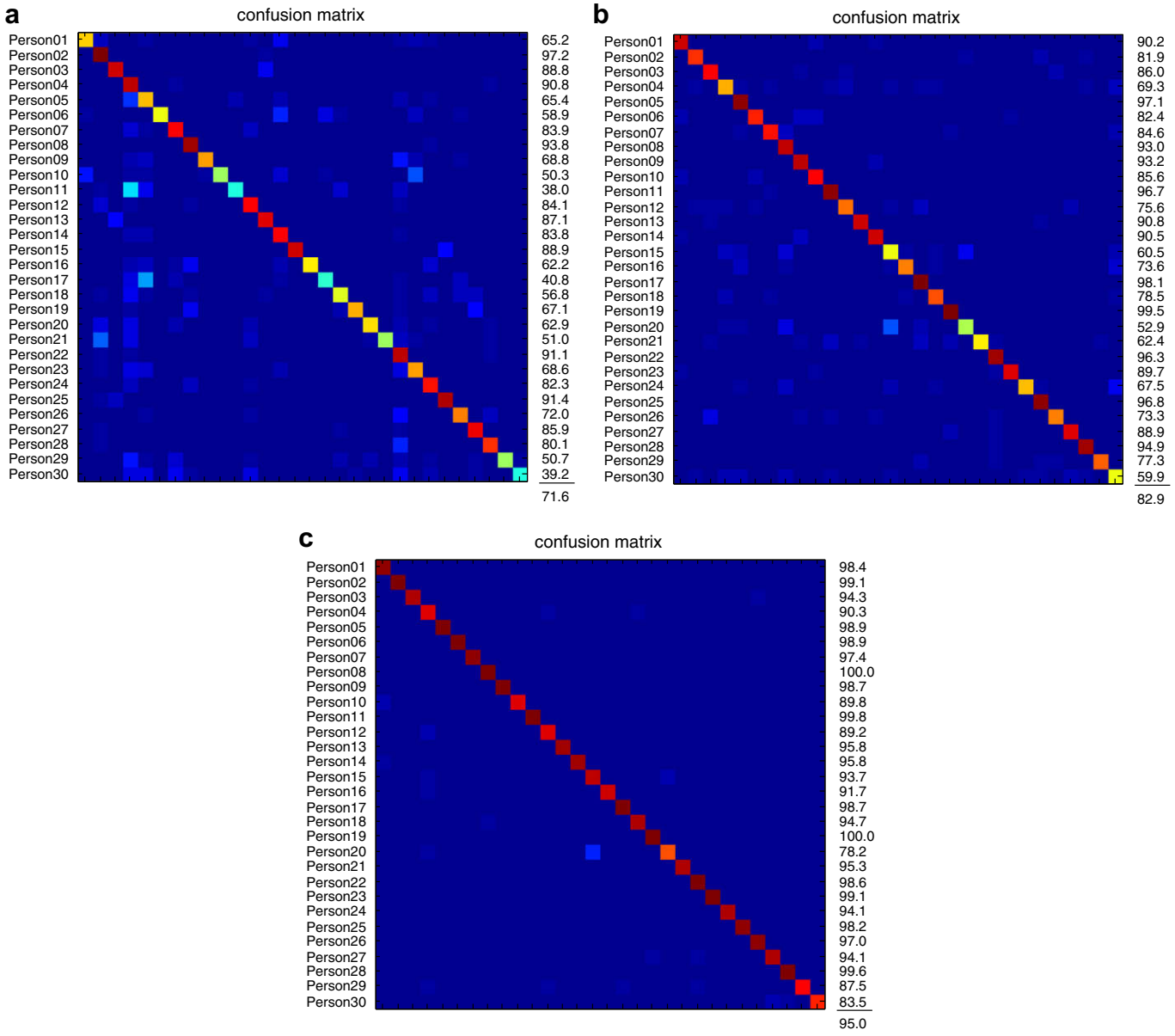

Fig. 6. Visual representation of the object classification confusion matrices with different representation schemes: (a) MCSHR, (b) HBOV-125 and (c) HBOV-10000.

Table 2
Average classification rate and standard deviations when using different vocabularies – specific $\mathcal{V}_s$ and generic $\mathcal{V}_g$ vocabularies, with tree sizes of 125, 625, 1000 and 10,000 leaf words

Vocabulary	Classification rate
Baseline	71.6 (17.7)
$\mathcal{V}_s^{125}$	82.9 (13.1)
$\mathcal{V}_s^{625}$	92.7 (7.1)
$\mathcal{V}_s^{1000}$	93.6 (6.3)
$\mathcal{V}_s^{10.000}$	95.0 (5.2)
$\mathcal{V}_g^{125}$	78.7 (14.8)
$\mathcal{V}_g^{625}$	91.4 (7.9)
$\mathcal{V}_g^{1000}$	91.6 (6.4)
$\mathcal{V}_g^{10.000}$	93.3 (6.2)

Table 3
Classification rate of different tracks using $\mathcal{M}_{30}$

Person ID	Baseline	Previous model $\mathcal{M}_{30}$	Updated model	# Of frames
Similar view angle				
01	57.5 (5.1)	76.8 (0.8)	93.4 (2.9)	331
12	84.3 (2.6)	71.6 (1.0)	95.9 (2.3)	293
20	11.9 (3.9)	43.1 (0.5)	83.4 (2.2)	422
Different view angle				
01	19.6 (8.4)	50.1 (0.6)	89.6 (3.7)	280
06	14.1 (1.5)	12.7 (1.7)	97.2 (2.5)	86
12	70.9 (6.1)	12.0 (1.3)	83.8 (4.7)	272
29	24.5 (2.5)	17.5 (1.4)	93.1 (3.6)	146
30	14.7 (1.9)	26.8 (1.4)	94.4 (2.3)	176

Results are obtained for (1) the baseline, (2) the model $\mathcal{M}_{30}$ trained previously, and (3) an updated model, including a sample of the new track.

classification performance using the baseline method is also presented. Unsurprisingly, the classification rate using the previous model $\mathcal{M}_{30}$ (second column) is much higher for the first set of new tracks. However, if we update the model with a random sample containing 50 images from each new track and retrain the model also with these samples, the results improve significantly (third column).

These results show that, given a correct model update, the visual object representation effectively adapts to the object's changes in time. Note also that the performance of the baseline method has a high variability, presenting either a high or very low classification performance. This confirms the conclusions drawn previously regarding its discriminative capability.

4.5. Resilience to incorrect segmentation

Until now, all objects were tested with the respective segmentations defined by the ground-truth's bounding box. In a real scenario, the only information about the object's location is the segmentation obtained using for example a background subtraction method. However, with such automatic segmentation methods errors occur more often. An example of an incorrect segmentation is shown in Fig. 7. The segmentation shown in green is very different from the true segmentation. It suffers from false positives errors – the person shadow to the left – as well as false negatives – part of the person is not correctly classified. Also, part of the same person is merged with the segmentation shown in red. It is clear that segmentation errors will inevitably penalise the overall performance. In order to assess the effect of imperfect segmentation on our classification performance, we used the segmentation algorithm proposed by Teixeira et al. (2007) and classified the segmented objects according to the ground-truth.

In Table 4, the confusion matrix of a model trained and tested with the images obtained with background subtraction algorithm is shown. The model consists only of the four classes of objects considered in this scenario and in this case a good discrimination is achieved.

If we test the object images extracted with the background segmentation algorithm with $\mathcal{M}_{30}$ model, results deteriorate. In Table 5, we summarise the results obtained with this model and the baseline, both using as inputs the segmentation algorithm and the ground-truth. Despite a smaller classification rate of the first three visual objects, all are successfully discriminated. The baseline representation method shows a significantly worse performance, probably due to its less resilience to occlusions and incorrect segmentation of objects. Interestingly, the results for `Person20` are better probably due to a good overall track segmentation. In theory, with a perfect segmentation, a better object description is

Table 4
Confusion matrix of the tracks segmented using a background subtraction algorithm

True class	Estimated class				# Of frames
	01	05	06	20	
01	96.8	0.0	3.0	0.1	356
05	1.7	98.0	0.1	0.2	601
06	1.1	0.8	96.5	1.6	390
20	1.1	0.7	0.2	98.0	1246

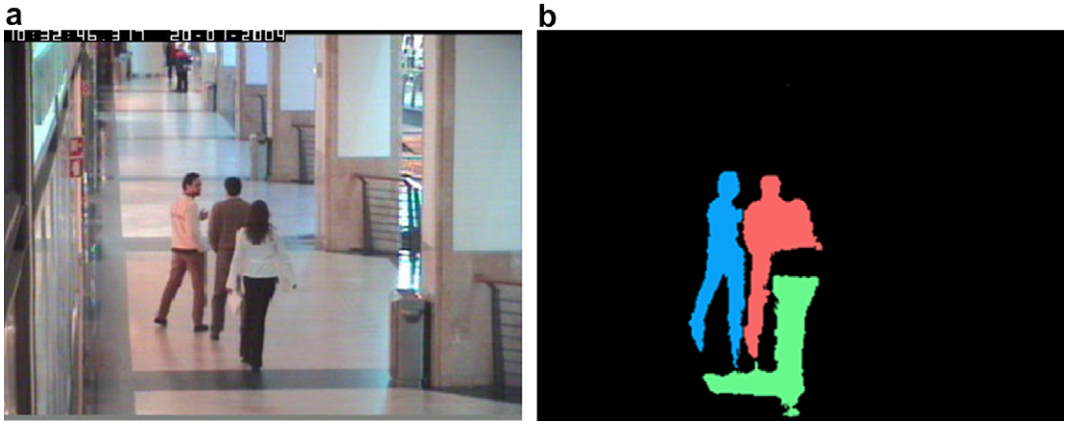

Fig. 7. Incorrect segmentation example. From left to right: `Person01` (blue segmentation), `Person20` (red segmentation), and `Person06` (green segmentation). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Average classification rates and standard deviations using the segmented tracks (second and third columns) and the ground-truth (fourth and fifth columns)

Person ID	Segmentation		Ground-truth	
	Baseline	$\mathscr{M}_{30}$	Baseline	$\mathscr{M}_{30}$
01	51.5 (2.4)	95.1 (0.2)	65.2 (2.1)	97.5 (1.5)
05	39.4 (3.5)	94.9 (0.5)	65.4 (1.7)	99.0 (0.1)
06	30.8 (1.7)	96.4 (0.5)	58.9 (1.9)	98.8 (0.8)
20	35.0 (2.1)	76.1 (0.1)	62.9 (2.3)	73.5 (2.5)

For each case, results are shown for the baseline and the $\mathscr{M}_{30}$ model.

achieved since the background is removed while the object is kept, but this is often not the case.

4.6. Incremental model learning

One of the requirements defined earlier for the object model was that it should be adaptable to changes. In a visual surveillance scenario, the objects' appearances can vary, depending on the viewing angle they are captured. We need, therefore, a way to incorporate these changes in the model. The straightforward approach to accommodate this variability of incoming data is to recreate a new model whenever a new set of data is available (similar to the approach tested in Section 4.4). When, for instance, new data become available, the previous model is forgotten and a new model is trained with that data, as well as with previous data. This approach, despite being simple, is unfeasible in real systems since it requires ever increasing storage resources, as well as computational resources. Nevertheless, this approach provides a performance reference to which our incremental learning will be compared to. The experimental setup consisted of training a model based on a multi-class SVM for all the data available until that moment. The incoming data consist of the subsets $\mathscr{D}^i$ defined in Section 4.1 and represented in Fig. 5.

The incremental learning approach described in Section 3.2 was tested in a similar fashion as the model rebuild approach. However, in this case, whenever a new set of data becomes available, only that set's data is provided to the classifier. The model is then updated, providing that new classes that appear are learnt and classification performance of other classes is kept. The number of classifiers added to Learn++.MT when a new set is added was of $T_k = 4$.

We evaluated the two update methods using a 5-fold cross-validation, and the overall results are shown in Table 6. For each subset, the total number of frames is also shown. Comparing the classification rates, it is clear that performance drops significantly if we use an incremental model update instead of the simplified approach. With a large number of new classes appearing in each new incoming data, the model is not able to adapt as promptly. Nevertheless, the overall classification rate of object images

Table 6
Classification rate for the simplified model update – at each new set a new model is built using all current available data; classification performance is kept at a very high rate

Subset	Type of model update		# Of frames
	Retrain all	Incremental	
$\mathscr{D}^1$	98.9 (2.0)	83.9 (16.8)	1864
$\mathscr{D}^2$	97.5 (5.0)	45.5 (34.2)	2813
$\mathscr{D}^3$	98.1 (2.8)	58.0 (22.4)	3158
$\mathscr{D}^4$	95.9 (6.0)	37.7 (16.9)	2770
$\mathscr{D}^5$	91.0 (8.5)	47.7 (13.1)	1951
Test set	98.0 (2.9)	52.2 (20.5)	1500

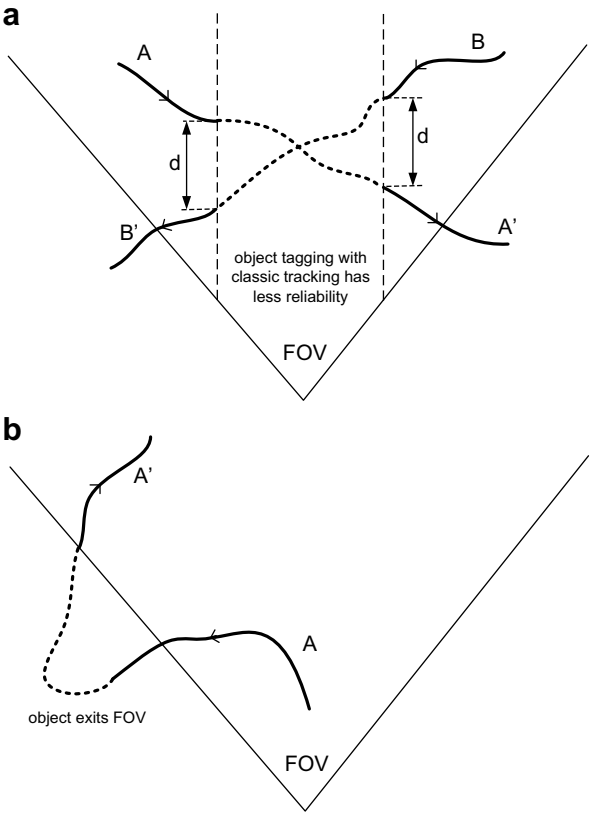

Fig. 8. Scenarios where typical tracking methods often fail to correctly track/identify an object. (a) Object occludes another object – as objects cross the tracking algorithm can incorrectly assign the label A to B′ and B to A′. (b) Object exits and re-enters FOV – the track A′, corresponding to the re-entrance of the object, would have a new assigned tag.

Table 7
Classification performance rate based on incremental learning; each subset may not contain images from a given person and, if this is the case, no classification rate is presented

Subset	Person ID															30-Class result
	01	02	04	05	08	11	15	17	19	20	23	24	27	27	30	
$\mathscr{D}^1$	–	81.3	46.2	62.7	–	86.8	61.4	100.0	95.7	74.0	–	85.1	97.7	96.2	100.0	83.9 (16.8)
$\mathscr{D}^2$	4.4	72.4	38.3	49.9	2.8	87.1	50.6	84.1	16.3	39.7	72.3	41.7	91.5	99.6	73.5	45.5 (34.2)
$\mathscr{D}^3$	69.2	27.2	–	37.1	71.4	100.0	–	69.9	–	–	41.4	83.2	65.6	69.1	72.3	58.0 (22.4)
$\mathscr{D}^4$	20.2	–	–	64.3	49.4	83.4	–	63.9	–	–	38.1	37.0	22.2	–	29.8	37.7 (16.9)
$\mathscr{D}^5$	–	–	–	37.4	–	66.7	–	38.5	–	–	21.7	24.4	–	–	23.2	47.7 (13.1)
Test set	30.0	74.0	54.0	80.0	42.0	96.0	50.0	82.0	72.0	32.0	30.0	52.0	52.0	82.0	56.0	52.2 (20.5)
Track classification	75.0	75.0	100.0	100.0	66.7	100.0	100.0	100.0	50.0	66.7	50.0	60.0	100.0	100.0	100.0	76.2 (24.3)

For brevity, results are presented only for some object classes. The last line shows the results for track classification; track segments are associated to each set and are classified by a simple rule: the most representative visual object among that segment's frames. The overall correct track classification is 76.2%.

Fig. 9. Common tracking error using a state-of-the-art particle filtering tracker (Chateau et al., 2005). The tracker changes targets because the new target has a higher classifier score then the initial one.

reaches around 50%, which is an acceptable value given the number of object classes. Also, an increase of performance in the last subset suggests that with more data the model adapts adequately. Table 7 shows the partial results for each subset. Note that only the performance results for half of the classes are shown for brevity (no particular choice).

A better measure of performance of our system is how successfully can the system classify the whole track. In our case, track classification is performed for each time window. The window, or segment, comprises a large number of images – 150 in this case – and a simple classification rule can be applied: a segment is classified as the visual object gathering the largest amount of frames labeled as that object. The last line in Table 7 shows the results for track classification, with an overall correct classification rate of 76.2%. Note that this result is for the model that resulted from all incremental updates.

5. Integration with tracking

Until now we assumed that captured tracks were the inputs for our method and no further interaction happened. However, track matching can be integrated with the tracking algorithm to improve the overall performance. The interaction between both can be twofold:

(1) Tracking helps build the database – the same person can have a different description depending on the view angle. By tracking a person with more than one camera with overlapping fields of viewing, for example, we can provide more information to the database, knowing that it's the same person.
(2) Object matching can correct incorrect tagging – matching can help disambiguate two tracks when, for example, two or more objects cross in the field of view.

While the first interaction is largely explored in our approach, the second interaction is discussed in this section.

Consider the scenario depicted in Fig. 8a. Objects A and B are tracked across a single field of view (FOV) and at some point in time both tracks cross. Without depth information or another view over the same area, disambiguating the tracks can be difficult. As objects get closer, typical segmentation outputs a single large combined object. When objects eventually are again farther apart the tracking system should to assign A and B to A′ and B′, respectively, but errors often occur. An example of this using a particle filtering tracker from Chateau et al. (2005) is depicted in Fig. 9. Using a track matching method inconsistent assignments could be detected and corrected.

Another similar problem occurs when a tracked object leaves the field of view and re-enters later, as depicted in Fig. 8b. In most tracking systems, the track corresponding to the re-entrance of the object would have a new tag assigned to it. The same object would then have two different identities. By comparing the visual representation of the new track with the visual objects model would correct the previous assignment.

The second type of interaction can also be extended to multiple independent views, where typical tracking methods cannot be used. A track produced by the tracking algorithm at a given view is matched with a global model. An existing tag is assigned to it, or a new one, if the object is not known. In multi-camera systems with overlapping FOVs, the spatial information can be a useful to aid in matching. Although this is a possibility, it is not explored in this work as we do not assume any dependence between views.

6. Conclusion

Event analysis systems based on visual information rely on an important preliminary step – object segmentation and tracking. Most current tracking techniques perform multi-target tracking in a single view. There are also many proposed algorithms that perform tracking across multiple views but usually some dependency between the views is assumed.

In this paper, we presented a combined representation scheme and incremental object model to find matches between visual object tracks. We relax the assumption of dependence between views, since: (1) no calibration information is used, (2) no spatial registration is made, and (3) no correlation is assumed between tracks detected by different cameras. The scheme was experimentally validated for a surveillance scenario using a publicly available dataset.

The description scheme relies on SIFT local descriptors and a text-like bag-of-words representation. Object images are identified by a histogram of visual words that are identified in the image. The vocabulary is constructed once using a generic dataset which, as our results show, performed closely to a vocabulary constructed with data from the images of tracked objects. Results show that this object representation scheme can be used to aid tracking of generic objects in visual surveillance systems, since it can discriminate a large quantity of different visual objects very well, and can be adapted to reflect object changes. It also presented a good resilience to incorrect segmentations when extracting the visual objects from complex scenes. The object model is updated through incremental learning, avoiding excessive data storage while maintaining performance and allowing new objects to be learnt by the system. To test the ability of our incremental learning to adapt to new data, we evaluated our model on a challenging scenario where data is processed in chunks as it becomes available. Results showed that our model has the ability to adapt to object modifications and to learn new objects.

In summary, the proposed methodology achieved an overall classification performance of 76.2% correct assignments for a classification problem of tracks from 30 different persons in a dataset that contained more than 14,000 images. This work can be a step forward to have a visual surveillance system capable of establishing connections between tracked objects across an arbitrarily complex camera system, independent of the camera location and time of capture.

Acknowledgements

This work has been partially supported by Fundação para a Ciência e a Tecnologia (Portuguese Science and Technology Foundation) and VISNET II, a Network of Excellence funded by the sixth Framework Programme of the European Commission. We also thank Kevin Smith and Pedro Quelhas for the fruitful exchange of ideas that helped improve this article.

References

Black, J., Ellis, T., Rosin, P., 2002. Multi view image surveillance and tracking. In: Proc. IEEE Workshop on Motion and Video Computing, pp. 169–174.

Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining Knowl. Disc. 2 (2), 121–167.

Cai, Q., Aggarwal, J.K., 1999. Tracking human motion in structured environments using a distributed-camera system. IEEE Trans. Pattern Anal. Machine Intell. 21 (11), 1241–1247.

Caspi, Y., Irani, M., 2000. A step towards sequence-to-sequence alignment. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 682–689.

Chateau, T., Gay-Belille, V., Chausse, F., Lapresté, J., 2005. Real-time tracking with classifiers. In: Workshop on Dynamical Vision at ECCV 2006.

Everingham, M., Zisserman, A., Williams, C.K.I., Gool, L.V., 2006. The PASCAL visual object classes challenge 2006 (VOC2006) results. <http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf>.

Foresti, G.L., Micheloni, C., Snidaro, L., Remagnino, P., Ellis, T., 2005. Active video-based surveillance system: The low-level image and video processing techniques needed for implementation. IEEE Signal Process. Magazine 22 (2), 25–37.

Grauman, K., Darrell, T., 2005. The pyramid match kernel: Discriminative classification with sets of image features. In: Proc. Internat. Conf. Computer Vision, vol. 2, pp. 1458–1465.

Hu, W., Tan, T., Wang, L., Maybank, S., 2004. A survey on visual surveillance of object motion and behaviors. IEEE Trans. Systems Man Cybernet. Part C: Appl. Rev. 34 (3), 334–352.

Javed, O., Rasheed, Z., Shafique, K., Shah, M., 2003. Tracking across multiple cameras with disjoint views. In: Proc. IEEE Internat. Conference on Computer Vision, vol. 2, pp. 952–957.

Javed, O., Shafique, K., Shah, M., 2005. Appearance modeling for tracking in multiple non-overlapping cameras. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 26–33.

Kettnaker, V., Zabih, R., 1999. Bayesian multi-camera surveillance. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 253–259.

Khan, S., Shah, M., 2003. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. IEEE Trans. Pattern Anal. Machine Intell. 25 (10), 1355–1360.

Lee, L., Romano, R., Stein, G., 2000. Monitoring activities from multiple video streams: Establishing a common coordinate frame. IEEE Trans. Pattern Anal. Machine Intell. 22 (8), 758–767.

Lowe, D.G., 1999. Object recognition from local scale-invariant features. IEEE Internat. Conf. Computer Vision 2, 1150–1157.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Internat. J. Computer Vision 60 (2), 91–110.

Madden, C., Cheng, E.D., Piccardi, M., 2007. Tracking people across disjoint camera views by an illumination-tolerant appearance representation. Machine Vision Appl. 18 (3), 233–247.

Mikolajczyk, K., Schmid, C., 2005. A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Machine Intell. 2 (10), 1615–1630.

Mittal, A., Davis, L.S., 2003. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. Internat. J. Computer Vision 51 (3), 189–203.

Muhlbaier, M., Topalis, A., Polikar, R., 2004. Learn++.MT: A new approach to incremental learning. In: Workshop on Multiple Classifier Systems. Springer Lecture Notes in Computer Science (LNCS), vol. 3077, pp. 52–61.

Nillius, P., Sullivan, J., Carlsson, S., 2006. Multi-target tracking – linking identities using bayesian network inference. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2187–2194.

Nistér, D., Stewénius, H., 2006. Scalable recognition with a vocabulary tree. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2161–2168.

Nowak, E., Jurie, F., Triggs, B., 2006. Sampling strategies for bag-of-features image classification. In: Proc. European Conference on Computer Vision.

Polikar, R., Udpa, L., Udpa, S.S., Honavar, V., 2001. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Trans. Systems Man Cybernet. – Part C: Appl. Rev. 31 (4), 497–508.

Qu, W., Schonfeld, D., Mohamed, M., 2000. Distributed bayesian multiple-target tracking in crowded environments using multiple collaborative cameras. EURASIP J. Adv. Signal Process. 22 (8), 758–767.

Quelhas, P., Monay, F., Odobez, J.-M., Gatica-Perez, D., Tuytelaars, T., 2007. A thousand words in a scene. IEEE Trans. Pattern Anal. Machine Intell. 29 (9), 1575–1589.

Ross, D.A., Lim, J., Lin, R.-S., 2007. Incremental learning for robust visual tracking. Internat. J. Computer Vision (online).

Sivic, J., Zisserman, A., 2003. Video Google: A text retrieval approach to object matching in videos. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1470–1477.

Teixeira, L.F., Cardoso, J.S., Corte-Real, L., 2007. Object segmentation using background modelling and cascaded change detection. J. Multimedia 2 (5), 55–64.

Utsumi, A., Ohya, J., 2000. Multiple-camera-based human tracking using non-synchronous observations. In: Proc. Asian Conference on Computer Vision, pp. 1034–1039.

Willamowski, J., Arregui, D., Csurka, G., Dance, C.R., Fan, L., 2004. Categorizing nine visual classes using local appearance descriptors. In: Proc. Workshop on Learning for Adaptable Visual Systems in IEEE Internat. Conference on Pattern Recognition.

Wu, G., Wu, Y., Jiao, L., Wang, Y.-F., Chang, E.Y., 2006. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In: Proc. ACM Internat. Conference on Multimedia, pp. 528–538.

Zhao, T., Aggarwal, M., Kumar, R., Sawhney, H., 2005. Real-time wide area multi-camera stereo tracking. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 976–983.