Chapter 3
Incremental Learning on Trajectory Clustering

Luis Patino, François Bremond, and Monique Thonnat

Abstract. Scene understanding corresponds to the real time process of perceiving, analysing and elaborating an interpretation of a 3D dynamic scene observed through a network of cameras. The whole challenge consists in managing this huge amount of information and in structuring all the knowledge. On-line Clustering is an efficient manner to process such huge amounts of data. On-line processing is indeed an important capability required to perform monitoring and behaviour analysis on a long-term basis. In this paper we show how a simple clustering algorithm can be tuned to perform on-line. The system works by finding the main trajectory patterns of people in the video. We present results obtained on real videos corresponding to the monitoring of the Toulouse airport in France.

3.1 Introduction

Scene understanding corresponds to the real time process of perceiving, analysing and elaborating an interpretation of a 3D dynamic scene observed through a network of sensors (including cameras and microphones). This process consists mainly in matching signal information coming from sensors observing the scene with a large variety of models which humans are using to understand the scene. This scene can contain a number of physical objects of various types (e.g. people, vehicle) interacting with each other or with their environment (e.g. equipment) more or less structured. The scene can last a few instants (e.g. the fall of a person) or a few months (e.g. the depression of a person), can be limited to a laboratory slide observed through a microscope or go beyond the size of a city. Sensors include usually cameras (e.g. omnidirectional, infrared), but also may include microphones and other sensors (e.g. optical cells, contact sensors, physiological sensors, smoke detectors, GNSS).

Luis Patino · François Bremond · Monique Thonnat

INRIA Sophia Antipolis - Méditerranée, 2004 route des Lucioles - BP 93 - 06902 Sophia Antipolis

E-mail: Jose-Luis.Patino_Vilchis@inria.fr

{Francois.Bremond,Monique.Thonnat}@inria.fr

Despite few success stories, such as traffic monitoring (e.g. Citilog), swimming pool monitoring (e.g. Poseidon) and intrusion detection (e.g. ObjectVideo, Keeneo), scene understanding systems remain erratic and can function only under restrictive conditions (e.g. during day rather than night, diffuse lighting conditions, no shadows). Having poor performance over time, they are hardly modifiable, containing little a priori knowledge on their environment. Moreover, these systems are very specific and need to be redeveloped from scratch for new applications. To answer these issues, most researchers have tried to develop original vision algorithms with focused functionalities, robust enough to handle real life conditions. Up to now no vision algorithms were able to address the large varieties of conditions characterising real world scenes, in terms of sensor conditions, hardware requirements, lighting conditions, physical object varieties, application objectives...

Here we state that the scene understanding process relies on the maintenance of the coherency of the representation of the global 3D scene throughout time. This approach which can be called 4D semantic interpretation is driven by models and invariants characterising the scene and its dynamics. The invariants (called also regularities) are general rules characterising the scene dynamics. For instance, the intensity of a pixel can change significantly mostly in two cases: change of lighting conditions (e.g. shadow) or change due to a physical object (e.g. occlusion). Another rule verifies that physical objects cannot disappear in the middle of the scene. There is still an open issue which consists in determining whether these models and invariants are given a priori or are learned. The whole challenge consists in managing this huge amount of information and in structuring all this knowledge in order to capitalise experiences, to share them with other computer vision systems and to update them along experimentations. To face this challenge several knowledge engineering tools are needed:

Tools for scene perception. A first category of tools contains vision algorithms to handle all the varieties of real world conditions. The goal of all these algorithms is to detect and classify the physical objects which are defined as interesting by the users. A first set of algorithms consists of robust segmentation algorithms for detecting the physical objects of interest. These segmentation algorithms are based on the hypothesis that the objects of interest are related to what is moving in the video, which can be inferred by detecting signal changes. Specific algorithms to separate physical objects from different categories of noise (e.g. due to light change, ghost, moving contextual object), and algorithms to extract meaningful features (e.g. 3D HOG, wavelet based descriptors, colour histograms) characterising the objects of interest belong to this category.

Tools for verification of the 3D coherency throughout time (physical world) A second category of tools are the ones combining all the features coming from the detection of the physical objects observed by different sensors and in tracking these objects throughout time. Algorithms for tracking multiple objects in 2D or 3D with one camera or a network of cameras belong here; for instance, algorithms that take advantage of contextual information and of a graph of tracked moving regions where an object trajectory can be seen as the most probable path in the graph. This

property enables to process long video sequences and to ensure the trajectory coherence. Moreover, these tracking algorithms compute the uncertainty of the tracking process by estimating the matching probability of two objects at successive instants. A second example is to fuse the information coming from several sensors at different levels depending on the environment configuration. Information fusion at the signal level can provide more precise information, but information fusion at higher levels is more reliable and easier to accomplish. In particular, we are using three types of fusion algorithms: (1) multiple cameras with overlapping field of view, (2) a video camera with pressure sensors and sensors to measure the consumption of water and electrical appliances, and (3) video cameras coupled with other sensors (contact sensors and optical cells).

Tools for Event recognition (semantic world). At the event level, the computation of relationships between physical objects constitutes a third category of tools. Here, the real challenge is to explore efficiently all the possible spatio-temporal relationships of these objects that may correspond to events (called also actions, situations, activities, behaviours, scenarios, scripts and chronicles). The varieties of these events are huge and depend on their spatial and temporal granularities, on the number of the physical objects involved in the events, and on the event complexity (number of components constituting the event and the type of temporal relationship). Different types of formalism can be used: HMM and Bayesian networks, temporal scenarios [28].

Tools for knowledge management. To be able to improve scene understanding systems, we need at one point to evaluate their performance. Therefore we have proposed a complete framework for performance evaluation which consists of a video data set associated with ground-truth, a set of metrics for all the tasks of the understanding process, an automatic evaluation software and a graphical tool to visualise the algorithm performance results (i.e. to highlight algorithm limitations and to perform comparative studies). Scene understanding systems can be optimised using machine learning techniques in order to find the best set of program parameters and to obtain an efficient and effective real-time process. It is also possible to improve system performance by adding a higher reasoning stage and a feedback process towards lower processing layers. Scene understanding is essentially a bottom-up approach consisting in abstracting information coming from signal (i.e. approach guided by data). However, in some cases, a top-down approach (i.e. approach guided by models) can improve lower process performance by providing a more global knowledge of the observed scene or by optimising available resources. In particular, the global coherency of the 4D world can help to decide whether some moving regions correspond to noise or to physical objects of interest.

Tools for Communication, Visualisation Knowledge Acquisition and learning
Even when the correct interpretation of the scene has been performed, a scene understanding system still has to communicate its understanding to the users or to adapt its processing to user needs. A specific tool can be designed for acquiring a priori knowledge and the scenarios to be recognised through end-user interactions.

3D animations can help end-users to define and to visualize these scenarios. Thus, these tools aim at learning the scenarios of interest for users. These scenarios can be often seen as the complex frequent events or as frequent combinations of primitive events called also event patterns.

The present work belongs to the latter category. We aim at designing an unsupervised system for the extraction of structured knowledge from large video recordings. By employing clustering techniques, we define the invariants (as mentioned above) characterising the scene dynamics. However, not all clustering techniques are well adapted to perform on-line. On-line learning is indeed an important capability required to perform scene analysis on long-term basis. In this work we show how meaningful scene activity characterisation can be achieved through trajectory analysis. To handle the difficulty of processing large amounts of video, we employ a clustering algorithm that has been tuned to perform on-line. The approach has been validated on video data from the Toulouse airport in France (European project COFRIEND [1]).

The reminder of the paper is organised as follows. We review some relevant work for scene interpretation from trajectory analysis in the following subsection (section 3.1.1 Related Work). We present the general structure of our approach in section 3.2 (General structure of the proposed approach). A brief description of the object detection and tracking employed in our system is given in section 3.3 (On-line processing: Real-time Object detection). The detailed description of the trajectory analysis undertaken in this work is given in section 4 (Trajectory analysis), including the algorithm to tune the clustering parameters. The evaluation of the trajectory analysis work is given in the following section. The results obtained after processing the video data from the Toulouse airport are presented in section 6. Our general remarks and conclusions are given at the end of the paper.

3.1.1 Related Work

Extraction of the activities contained in the video by applying data-mining techniques represents a field that has only started to be addressed. Recently it has been shown that the analysis of motion from mobile objects detected in videos can give meaningful activity information. Trajectory analysis has become a popular approach due to its effectiveness in detecting normal/abnormal behaviours. For example, Piciarelli et al. [22] employ a splitting algorithm applied on very structured scenes (such as roads) represented as a zone hierarchy. Foresti et al. [11] employ an adaptive neural tree to classify an event occurring on a parking lot (again a highly structured scene) as normal/suspicious/dangerous. Anjum et al. [2] employ PCA to seek for trajectory outliers. In these cases the drawback of the approach is that the analysis is only adapted to highly structured scenes. Similarly, Naftel et al. [20] first reduce the dimensionality of the trajectory data employing Discrete Fourier Transform (DFT) coefficients and then apply a self-organizing map (SOM) clustering algorithm to find normal behaviour. Antonini et al. [3] transform the trajectory data employing Independent Component Analysis (ICA), while the final clusters

are found employing an agglomerative hierarchical algorithm. In these approaches it is however delicate to select the number of coefficients that will represent the data after dimensionality reduction. Data mining of trajectories has also been applied with statistical methods. Gaffney et al. [13] have employed mixtures of regression models to cluster hand movements, although the trajectories were constrained to have the same length. Hidden Markov Models (HMM) have also been employed [5, 21, 24]. In addition to activity clustering, so as to enable dynamic adaptation to unexpected event processing or newly observed data, we need a system able to learn the activity clusters in an on-line way. On-line learning is indeed an important capability required to perform behaviour analysis on long-term basis and to anticipate the human interaction evolutions. An on-line learning algorithm gives a system the ability to incrementally learn new information from datasets that consecutively become available, even if the new data introduce additional classes that were not formerly seen. This kind of algorithm does not require access to previously used datasets, yet it is capable of largely retaining the previously acquired knowledge and has no problem of accommodating any new classes that are introduced in the new data [23]. Some various restrictions, such as whether the learner has partial or no access to previous data [26, 17, 15], or whether new classes or new features are introduced with additional data [30], have also been proposed [19]. Many popular classifiers, however, are not structurally suitable for incremental learning; either because they are stable [such as the multilayer perceptron (MLP), radial basis function (RBF) networks, or support vector machines (SVM)], or because they have high plasticity and cannot retain previously acquired knowledge, without having access to old data (such as -nearest neighbor) [19]. Specific algorithms have been developed to perform on-line incremental learning, such as Leader [14], Adaptive Resonance Theory modules (ARTMAP) [8, 7], Evolved Incremental Learning for Neural Networks [25], leaders-subleaders [29], and BIRCH [18]. Among them, the Hartigan algorithm [14], also known as Leader algorithm, is probably the most employed in the literature. The Leader algorithm, computes the distance between new data and already built clusters to decide to associate these new data with the clusters or to generate new ones better characterising the data. However, all these approaches rely on a manually-selected threshold to decide whether the data is too far away from the clusters. To improve this approach we propose to control the learning rate with coefficients indicating how flexible the cluster can be updated with new data.

3.2   General Structure of the Proposed Approach

The monitoring system is mainly composed of two different processing components (shown in Figure 3.1). The first one is a video analysis subsystem for the detection and tracking of objects. This is a processing that goes on a frame-by-frame basis. The second subsystem achieves the extraction of trajectory patterns from the video. This subsystem is composed of two modules: The trajectory analysis module and the statistical analysis module. In the first module we perform the analysis of trajectories

by clustering and obtain behavioural patterns of interaction. In the second module we compute meaningful descriptive measures on the scene dynamics.

For the storage of video streams and the trajectories obtained from the video processing module, a relational database has been setup. The trajectory analysis modules read the trajectories from the database and return the identified trajectory types; the discovered activities on the video; and resulting statistics calculated from the activities. Streams of video are acquired at a speed of 10 frames per second. The video analysis subsystem takes its input directly from the data acquisition component; the video is stored into the DB parallel to the analysis process.

The whole system helps the manager or designer who wants to get global and long-term information from the monitored site. The user can specify a period of time where he/she wishes to retrieve and analyse stored information. In particular the user can access the whole database to visualize specific events, streams of video and off-line information.
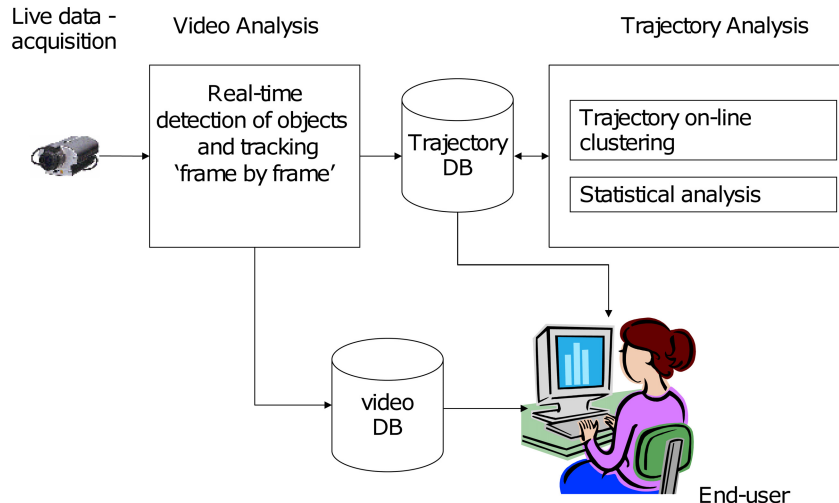

Fig. 3.1 General architecture of the system.

3.3   On-Line Processing: Real-Time Object Detection

Tracking objects in video is not the main contribution of this paper and therefore only a general description is made here. Detecting objects in an image is a difficult and challenging task. One solution widely employed consists in performing a thresholding operation between the pixel intensity of each frame with the pixel intensity of a background reference image. The latter can be a captured image of the same scene having no foreground objects, or no moving objects in front of the camera. The result of the thresholding operation is a binary mask of foreground pixels. The neighbouring foreground pixels are grouped together to form regions often referred

to as blobs which correspond to the moving regions in the image. If the moving object projection in the image plane does not overlap with each other, i.e. no dynamic occlusion, then each detected moving blob corresponds to a single moving object. The detailed description of the background subtraction algorithm, which also estimates when the background reference image needs to be updated, can be found in [12]. Having 3D information about the scene under view enables the calibration of the camera. Point correspondences between selected 3D points in the scene and their corresponding point in the 2D image plane allow us to generate the 3D location of any points belonging to moving objects. Thus, the 3D (i.e. width and height) of each detected moving blob can be measured as well as their 3D location on the ground plane in the scene with respect to a chosen coordinate system. The 3D object information is then compared against several 3D models provided by the user. From this comparison, a detected object is linked to a semantic class. Detected and classified 3D objects in a scene can be tracked within the scope of the camera using the 3D information of their location on the ground as well as their 3D dimensions. Tracking a few objects in a scene can be easy as far as they do not interact heavily in front of the camera: i.e. occlusions are rare and short. However, the complexity of tracking several mobile objects becomes a non-trivial and very difficult task to achieve when several object projected images overlap with each other on the image plane. Occluded objects have missing or wrong 3D locations, which can create incoherency in the temporal evolution of their 3D location. Our tracking algorithm [4] builds a temporal graph of connected objects over time to cope with the problems encountered during tracking. The detected objects are connected between each pair of successive frames by a frame to frame (F2F) tracker. Links between objects are associated with a weight (i.e. a matching likelihood) computed from three criteria: the comparison between their semantic class, 3D dimensions, and their 3D distance on the ground plane. The graph of linked objects provided by the F2F tracker is then analysed by the tracking algorithm, also referred to as the Long Term tracker, which builds paths of mobiles according to the link weights. The best path is then taken out as the trajectory of the related mobiles.

The proposed tracking approach has the advantage of being simple to implement and able to run at a 'high' frame rate. However, it is sensitive to noise and this could prevent tracking correctly long trajectories.

3.4 Trajectory Analysis

The second layer of analysis in our approach is related to the knowledge discovery of higher semantic information from analysis of activities recorded over a period of time that can span, for instance, from minutes to a whole day (or several days of recording). Patterns of activity are extracted from the analysis of trajectories.

3.4.1 Object Representation: Feature Analysis

For the trajectory pattern characterisation of the object, we have selected a comprehensive, compact, and flexible representation. It is suitable also for further analysis

as opposed to many video systems, which actually store the sequence of object lo-
cations for each frame of the video building thus a cumbersome representation with
little semantic information. If the dataset is made up of N objects, the trajectory for
object O_j in this dataset is defined as the set of points $[x_j(t), y_j(t)]$ corresponding to
their position points; x and y are time series vectors whose length is not equal for
all objects as the time they spend in the scene is variable. Two key points defining
these time series are the beginning and the end, $[x_j(1), y_j(1)]$ and $[x_j(end), y_j(end)]$
as they define where the object is coming from and where it is going to. We build
a feature vector from these two points. Additionally, we also include the directional
information given as $[cos(\theta), sin(\theta)]$, where θ is the angle which defines the vector
joining $[x_j(1), y_j(1)]$ and $[x_j(end), y_j(end)]$. A mobile object seen in the scene is
thus represented by the feature vector:

$$v_j = [x_j(1), y_j(1), x_j(end), y_j(end), cos(\theta), sin(\theta)] \quad\quad (3.1)$$

This feature vector constitutes a set of simple descriptors that have proven exper-
imentally to be enough to describe activities in a large variety of domains (such
as traffic monitoring, subway control, monitoring smart environments), mainly be-
cause they are the most salient, but also they are appropriate for real world videos
depicting unstructured scenes where trajectories of different types have strong over-
lap and they are usually the ones used by end-users of different domains.

3.4.2   Incremental Learning

We need a system able to learn the activity clusters in an on-line way. On-line learn-
ing is indeed an important capability required to perform behaviour analysis on a
long-term basis. A first approach proposed in the state-of-the-art for on-line clus-
tering is the Leader algorithm [14]. In this method, it is assumed that a rule for
computing the distance D between any pair of objects, and a threshold T is given.
The algorithm constructs a partition of the input space (defining a set of clusters)
and a leading representative for each cluster, so that every object in a cluster is
within a distance T of the leading object. The threshold T is thus a measure of the
diameter of each cluster. The clusters CL_i, are numbered CL_1, CL_2, CL_3, ..., CL_k.
The leading object representative associated with cluster CL_j is denoted by L_j. The
algorithm makes one pass through the dataset, assigning each object to the cluster
whose leader is the closest and making a new cluster, and a new leader, for objects
that are not close enough to any existing leaders. The process is repeated until all
objects are assigned to a cluster. Leader-subleader [29], ARTMAP [8] and BIRCH
[18] algorithms are of this type. The strongest advantages of the Leader algorithm
are that it requires a single scan of the database, and only cluster representatives
need to be accessed during processing. However, the algorithm is extremely sensi-
tive to threshold parameter defining the minimum activation of a cluster CL. A new
input object defined by its feature vector v will be allocated to cluster CL_j if v falls
into its input receptive field (hyper-sphere whose radio is given by $r_j = T$). Defining
T is application dependent. It can be supplied by an expert with a deep knowledge

of the data or employing heuristics. In this work we propose to learn this parameter employing a training set and a machine learning process.

Let each cluster CL_i be defined by a radial basis function (RBF) centered at the position given by its leader L_i:

$$CL_i(v) = \phi(L_i, v, T) = exp(-\|v - L_i\|^2 T^2) \tag{3.2}$$

The RBF function has a maximum of 1 when its input is $v = L_i$ and thus acts as a similarity detector with decreasing values outputted whenever v strides away from L_i. We can make the choice that an object element will be included into a cluster if $CL_i(v) \geq 0.5$, which is a natural choice. The cluster receptive field (hyper-sphere) is controlled by the parameter T.

Now, consider $C = \{CL_1, \ldots, CL_k\}$ is a clustering structure of a dataset $X = \{v_1, v_2, \ldots, v_N\}$; $\{L_1, L_2, \ldots, L_k\}$ are the leaders in this clustering structure and $P = \{P_1, \ldots, P_s\}$ is the true partition of the data (Ground-truth) and $\{M_1, \ldots, M_s\}$ are the main representatives (or Leaders) in the true partition. We can define an error function given by

$$E = \frac{1}{N} \sum_{j=1}^{N} E_j \tag{3.3}$$

$$E_j = \begin{cases} 0 & if \ \{L(v_j), v_j\} \in CL_i; \{L(v_j), v_j\} \in P_{i'} \\ -1 & if \ v_j \in CL_i; |CL_i| = 1 \ and \ L(v_j) \neq M(v_j) \\ 1 & otherwise \end{cases} \tag{3.4}$$

and $L(v_j)$ is the Leader associated to v_j in the clustering structure C_i and $|C_i|$ is the cluster cardinality. $M(v_j)$ is the Leader associated to v_j in the true partition P. In the above equation, the first case represents a good clustering when the cluster prototype and the cluster elements match the ground truth partition P. The error is zero and the cluster size is correct. The second case corresponds to a cluster made of a singleton element. This element prototype does not correspond to any expected cluster prototype in the 'true' partition P. In this case the cluster size has to grow in order to enclose the singleton element. The remaining case is where an element is wrongly included into a cluster; The cluster size has then to decrease to exclude unwilling border elements.

Minimising this error is equivalent to refining the clustering structure C or is equivalent to adjusting the parameter T that controls the cluster receptive field. A straightforward way to adjust T and minimise the error is employing an iterative gradient-descent method:

$$T(t+1) = T(t) - \eta \frac{\partial E(t)}{\partial T} \tag{3.5}$$

where the error gradient at time t is:

$$\frac{\partial E(t)}{\partial T} = \frac{1}{N} \sum_j E_j(t) \frac{\partial \hat{\Phi}}{\partial T} \tag{3.6}$$

and the cluster activation gradient is:

$$\frac{\partial \hat{\Phi}}{\partial T} = \frac{\partial}{\partial T} \left[\exp\left(-\left\| v_j - L(v_j) \right\|^2 T^2 \right) \right] \tag{3.7}$$

$$\frac{\partial \hat{\Phi}}{\partial T} = -\left\| v_j - L(v_j) \right\|^2 (2T) \left[\exp(-\left\| v_j - L(v_j) \right\|^2 T^2) \right] \tag{3.8}$$

The threshold update can thus be written as:

$$T(t+1) = T(t) - \eta \frac{1}{N} \sum_j E_j(t) \left[-2T \left\| v_j - L(v_j) \right\|^2 \right] \hat{\Phi} \tag{3.9}$$

The final value is typically set when the error is sufficiently small or the process reaches a given number of iterations. Convergence to an optimum value for both, T and E is only guaranteed if data in the 'true' partition P is well structured (having high intra-class homogeneity and high inter-class separation; see unsupervised/supervised evaluation below).

With the purpose of tuning parameter T, and for this application, we have defined a Training data set (with associated Ground-truth) containing sixty nine synthetic trajectories. The ground-truth trajectories were manually drawn on a top view scene image. Figure 3.2 shows the empty scene of the Toulouse airport with some drawn trajectories. Semantic descriptions such as From Taxi parking area to Tow-tractor waiting point were manually given. There are twenty three of such annotated semantic descriptions, which are called in the following trajectory types. Each trajectory type is associated with a main trajectory that best matches that description. Besides, two complementary trajectories define the confidence limits within which we can still associate that semantic description. In figure 3.2 the main trajectory of each trajectory type is represented by a red continuous line while blue broken lines represent the complementary trajectories of the trajectory type. Thus, each ground-truth trajectory is associated to a semantic descriptor or trajectory type. Each trajectory type contains a triplet of trajectories.

The proposed gradient-descent methodology was applied to the ground-truth dataset. The threshold T, in the leader algorithm, is initially set to a large value (which causes a merge of most trajectory types). Figure 3.3 shows how this threshold value evolves as the gradient algorithm iterates. The graph for the corresponding error is shown in figure 3.4. Remark that for this application we have not encountered local minima problems. However, as gradient-descent algorithms are clearly exposed to this problem, it could be envisaged to verify whether the minima found is indeed the global optima. A multiresolution analysis would be of help for this.

It is also possible to evaluate, in an unsupervised or supervised manner, the quality of the resulting clustering structure:

Unsupervised evaluation: typical clustering validity indexes evaluating the intra-cluster homogeneity and inter-cluster separation such as Silhouette [6, 16], Dunn [10] and Davies-Bouldin [9] indexes (given in Annex 1) can be employed.
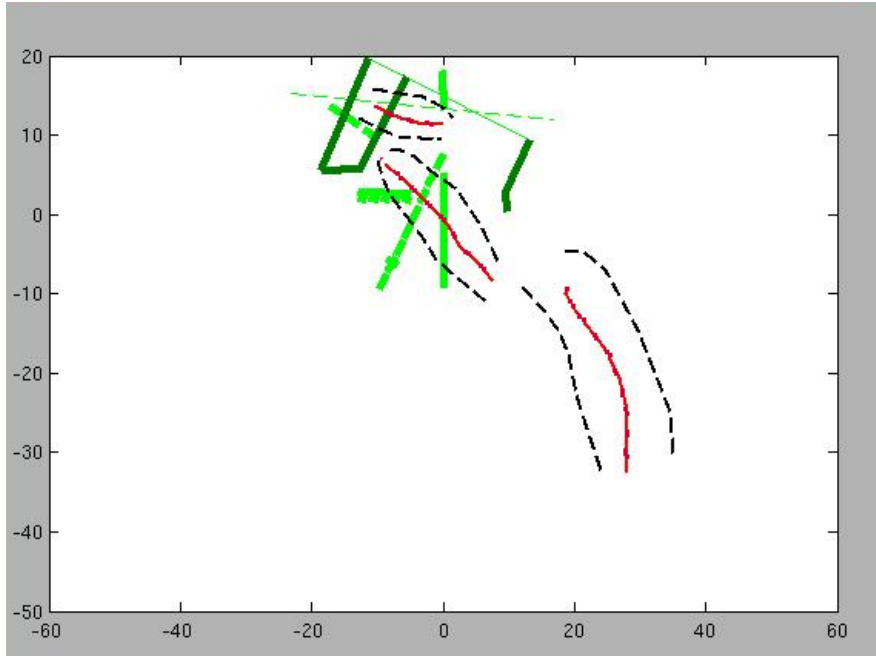
**Fig. 3.2** Ground-truth for different semantic clusters.
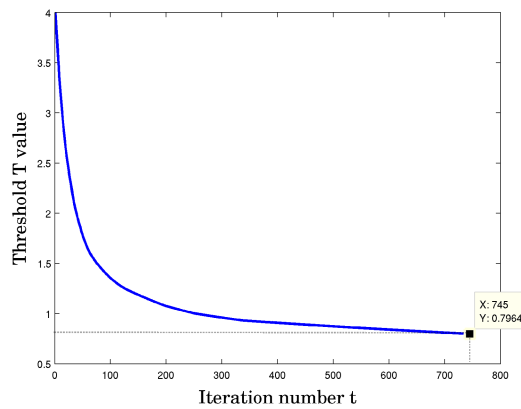
Fig. 3.3 Evolution of the threshold T controlling the cluster receptive field.

Figure 3.5 shows the evolution of these three indexes on the clustering of trajectories as the gradient-descent algorithm evolves.

Supervised evaluation: supervised validity indexes, which in this case compare the clustering results to the true data partition, such as the Jaccard index [27] (given in
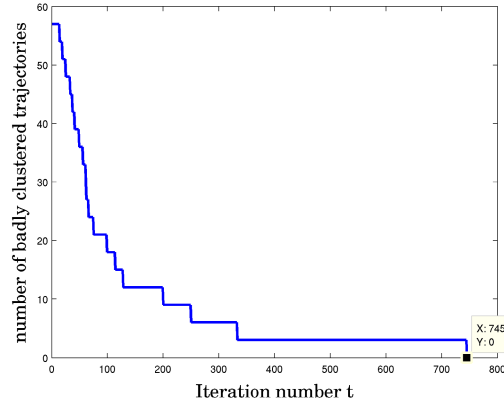
Fig. 3.4 Evolution of the gradient-descent error with the number of iterations. The error gives an indication of how many elements of different trajectory types are merged together in a single cluster.
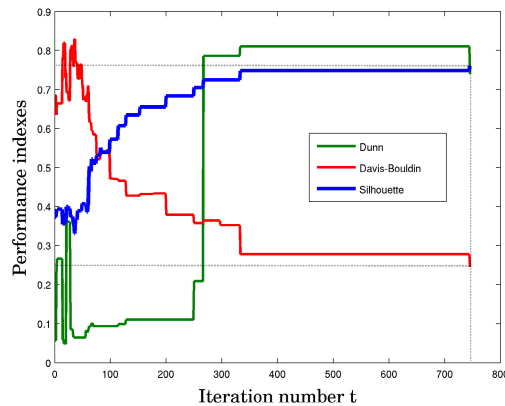
Fig. 3.5 Validity indexes such as Silhouette (higher values are better), Dunn (higher values are better) and Davies-Bouldin (lower values are better) at each iteration step of the gradient-descent algorithm.

Annex 2) can also be employed. Figure 3.6 shows the evolution of this index on the clustering of trajectories as the gradient-descent algorithm evolves.

For large values of the threshold T (above 1.5) it is possible to see that a large number of trajectories are badly clustered (about 1/3 of the dataset). The unsupervised indexes are also unstable (presenting some oscillatory changes over the different iterations) and indicative of a bad clustering structure (meaning low inter-cluster distance and high intra-cluster distance). The mapping with the true partition is also poor (indicated by low values of the Jaccard index). For values of the threshold T below 1.4 there is an almost monotonically improvement of the unsupervised and
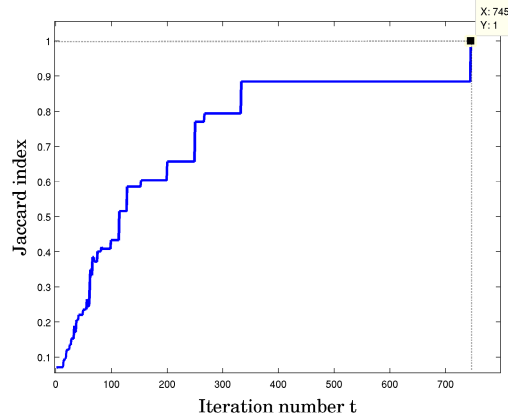
Fig. 3.6 Supervised evaluation at each iteration step of the gradient-descent algorithm. The Jaccard index compares the resulting clustering with the partition given by the trajectory ground-truth.

supervised clustering indexes. The Jaccard index reaches its maximum value (meaning a perfect matching with the true partition) for a threshold T=0.79, which is then selected for our analysis. The unsupervised indexes are also indicative of a good clustering structure. The Leaders defined from this process are selected as the initial cluster centres that will guide the partition of new incoming data.

3.5   Trajectory Analysis Evaluation

In order to test the efficiency of the trajectory clustering algorithm we have analysed a new set of synthetic trajectories, which we denote by 'experimental set'. This new set was composed of 230 trajectories, which have the same structure as the training dataset; that is, each trajectory is associated with a semantic meaning. Moreover, each trajectory in the test dataset was generated from the Ground-truth dataset in the following form. Trajectories are generated by randomly selecting among points uniformly distributed on each side between the main trajectory and the two adjacent trajectories. These points lie on segments linking the main trajectory to the two adjacent trajectories. Each segment starts on a sample point of the main trajectory and goes to the nearest sample point in the adjacent trajectory. Adjacent trajectories are up-sampled for better point distribution. Ten points lie on each segment linking the main trajectory and an adjacent trajectory. For this reason, the trajectories generated from the principal trajectory will convey the same semantic. Figure 3.7 below shows a couple of examples.

   The clustering algorithm is then run again without any knowledge of the semantic description for each trajectory on the experimental data set. After the clustering process is achieved, the resulting partition can be assessed by comparing with the one initially defined by the Ground-truth; what the Jaccard index does. In this case, the
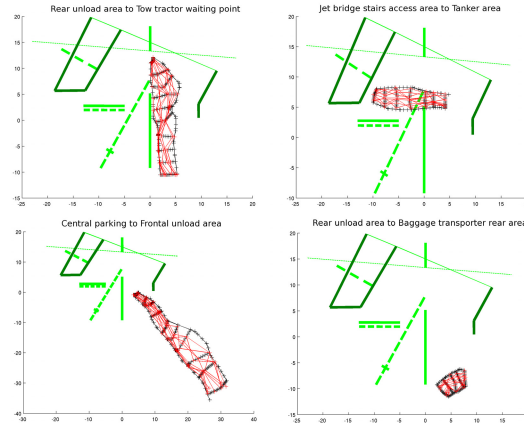
Fig. 3.7 Four different sets of synthetic trajectories. Each set contains twenty trajectories different from the main and adjacent trajectories previously defined in the Ground-truth dataset (Figure 3.2).

Jaccard index takes a value of 0.9119 (our baseline). Moreover, typical metrics related to the ROC space (Receiver operating characteristics) can be computed which evaluate both the mapping between the clustering algorithm output and the Ground-truth partition. These measures are the true positive rate (TPR) and false positive rate (FPR), which in this case take the following values: TPR=0.9565, FPR=0.002.

In order to assess the robustness of the trajectory clustering algorithm, we have evaluated our approach on more different sets of synthetic objects. Each set has the particularity of containing groups of very similar trajectories (even with an overlap in the most difficult cases), yet associated with different semantics. The evaluation consists thus in assessing how much the clustering algorithm will be affected by the different levels of complexity/noise introduced. To characterise the different datasets, we have computed the unsupervised clustering indexes Silhouette, Dunn and Davies-Bouldin. The table below (Table 3.1) summarises the results.

Each experimental set mentioned in the table above contains an increasing complexity. For instance, some groups of trajectories defined in the 'experimental set 2', which contain some overlap between them, are also present in the next experimental sets (experimental set 3, 4 , 5., . . .). For each experimental set, some new groups of trajectories are added, which in turn induce more overlapping situations and will also be present in the following experimental sets. The figure below (Figure 3.8) presents some examples of such trajectories in the different experimental sets. The structuring indexes 'Silhouette', 'Dunn' and 'Davies-Bouldin' reflect the less distinct separation induced between trajectories with different semantic meanings (Silhouette and Dunn indexes decrease, while the Davies-Bouldin index increases). The different experimental datasets cover situations presenting a very strong separation between groups of trajectories with different semantic meanings (structuring indexes Silhouette=0.85 Dunn=0.79 Davies-Bouldin=0.22), partial confusion

**Table 3.1** Clustering results on different synthetic datasets.

Input						Output						
Name	Nb. Trj.	Nb. GT's	Structure characteristics			Nb. Clusters	Structure characteristics			Jaccard Index	ROC measures	
			Sil	Dunn	DB		Sil	Dunn	DB		TPR	FPR
ExpSet	230	23	0.85	0.79	0.22	22	0.82	0.79	0.30	0.91	0.95	0.0020
ExpSet2	280	28	0.82	0.59	0.24	27	0.80	0.80	0.33	0.92	0.96	0.0013
ExpSet3	340	34	0.80	0.58	0.26	32	0.76	0.03	0.41	0.87	0.94	0.0018
ExpSet4	440	44	0.76	0.47	0.33	42	0.73	0.06	0.47	0.85	0.92	0.0016
ExpSet5	520	52	0.75	0.46	0.33	51	0.74	0.30	0.42	0.91	0.95	0.00075
ExpSet6	590	59	0.73	0.47	0.37	55	0.70	0.14	0.49	0.86	0.93	0.0012
ExpSet7	650	65	0.72	0.39	0.39	59	0.67	0.08	0.57	0.79	0.88	0.0017
ExpSet8	710	71	0.70	0.37	0.40	65	0.67	0.05	0.54	0.82	0.91	0.0012
ExpSet9	750	75	0.70	0.39	0.41	62	0.57	0.03	0.62	0.61	0.80	0.0025
ExpSet10	840	84	0.67	0.25	0.45	71	0.58	0.03	0.65	0.66	0.82	0.0020
ExpSet11	890	89	0.66	0.13	0.48	72	0.55	0.04	0.66	0.60	0.77	0.0024
ExpSet12	920	92	0.65	0.21	0.50	77	0.54	0.02	0.68	0.62	0.79	0.0022
ExpSet13	990	99	0.64	0.22	0.52	86	0.53	0.04	0.66	0.62	0.77	0.0019
ExpSet14	1070	107	0.60	0.04	0.61	77	0.45	0.02	0.74	0.45	0.66	0.0030

(Silhouette=0.7588 Dunn=0.4690 Davies-Bouldin=0.3331), high confusion (Silhouette=0.6517 Dunn=0.2131 Davies-Bouldin=0.5031) and very high confusion (Silhouette=0.6098 Dunn=0.0463 Davies-Bouldin=0.6157). The trajectory clustering algorithm performs accordingly, having more difficulty to retrieve all initial semantic groups when the confusion increases (thus the internal structure of the input data decreases); at the same time, the mapping between the trajectory clustering results and the semantic groups (GT) also worsens as exposed by the Jaccard Index. However, the overall behaviour shown by the true positive rate (TPR) and false positive rate (FPR) remains globally correct with TPR values near or above 0.77 for all studied cases except for the worst case experimental set 14 where the TPR is below 0.7.

In order to assess the generalisation capability of the trajectory clustering algorithm, we have carried out new experiments employing the CAVIAR dataset (http://www-prima.inrialpes.fr/PETS04/caviar_data.html). The dataset contains people observed at the lobby entrance of a building. The annotated ground-truth includes for each person its bounding box (id, centre coordinates, width, height) with a description of his/her movement type (inactive, active, walking, running) for a given situation (moving, inactive, browsing) and most importantly gives contextual information for the acted scenarios (browsing, immobile, left object, walking, drop down). In Figure 3.9 below, some examples of the acted scenarios in the CAVIAR dataset and the involved contextual objects are shown.

From the CAVIAR dataset we have kept only the representative trajectory of the acted scenario (which we further call principal trajectory). Other non-related trajectories like supplementary movements or non-actor trajectories, which are thus not related to the acted scenario, are filtered out. In total, forty different activities can
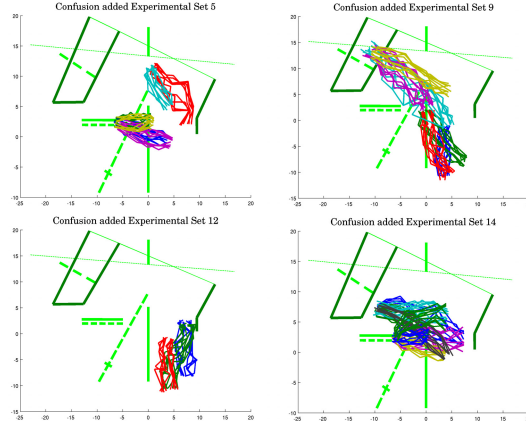
Fig. 3.8 Different examples of trajectories added to a given experimental data set under study. Different colours in an experimental data set correspond to different semantics attributed to the trajectories (each trajectory is associated with only one semantic meaning). Although the semantics between trajectories may be different, their spatial similarity can be very close.


Fig. 3.9 Two different people trajectory types while going to look for information (Browsing) at two different places.

be distinguished. They include Browsing at different places of the scene, Walking (going through the hall) from different locations, and leaving or dropping an object at different locations of the hall. We have created the new evaluation set applying the following formulae:

$$x_i' = N(\alpha r_x, x_1) + N(\beta r_x, x_i) \tag{3.10}$$

$$y_i' = N(\alpha r_y, y_1) + N(\beta r_y, y_i) \tag{3.11}$$

Where $N(\sigma_u, \overline{u})$ is a random number from a normal distribution with mean $\overline{u}$ and standard deviation σ_u; $r_x = |max(x_i) - min(x_i)|$ is the range function on x; and α, β are two different constants to control the spread of the random functions. For each principal trajectory, we have generated 30 new synthetic trajectories by adding random noise as explained above. In total, the synthetic CAVIAR dataset contained
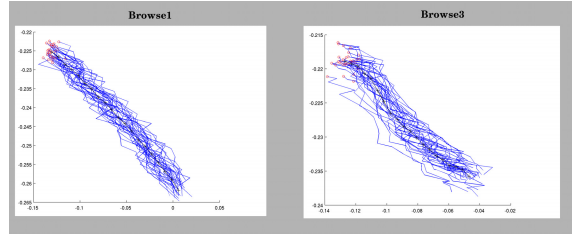
Fig. 3.10 Synthetic trajectories in the CAVIAR dataset generated from the activities shown in the previous figure. The trajectories are plotted employing their 3D coordinates on the ground.

1200 trajectories. Figure 3.10 shows the synthetic trajectories generated from the principal trajectories of the activities shown before.

The CAVIAR synthetic evaluation data set was further divided into a Learning set (containing 1/3 of the trajectories in the synthetic evaluation data set, plus all of the principal trajectories) and Test set (containing the remaining 2/3 from the trajectories in the synthetic evaluation data set). The Learning set was employed to tune the 'T' threshold, which is critical to the clustering algorithm as indicated before. In this case, the tuning algorithm has fetched a 'T' value of *T=1.05* and for which, the best clustering partition matches the 'true' data partition. When evaluating the algorithm in the Test set, the supervised Jaccard index used to compare the resulting partition with that 'true' expected partition gives a value of: Jaccard index=0.99. One supplementary evaluation set was created by adding more trajectories but which contain some spatial overlap to those already defined, yet they convey a different semantic (same procedure carried out in the first synthetic dataset). The structuring indexes 'Silhouette', 'Dunn' and 'Davies-Bouldin' reflect again the less distinct separation induced between trajectories with different semantic meanings. Table 3.2 summarises the results on both CAVIAR experimental datasets.

Again, the same trend as for the synthetic COFRIEND dataset appears. When the confusion between semantics increases (thus the internal structure of the input data decreases), retrieving all initial semantic groups is more difficult and the mapping between the trajectory clustering results and the semantic groups also worsens (Jaccard index decreases).

Table 3.2 Clustering results on CAVIAR synthetic datasets.

Input				Output				
Name	Nb. Trj.	Nb. GT's	Structure Sil	Nb. Clusters	Structure Sil	Jaccard Index	ROC measures TPR	FPR
ExpSet1	440	22	0.79	22	0.79	0.99	0.99	0.0001
ExpSet2	740	37	0.65	26	0.60	0.56	0.69	0.008

3.6 Results

We have processed in total five video datasets corresponding to different monitoring instances of an aircraft in the airport docking area (in the following, these video datasets are to be named: cof1, cof2, cof3, cof4 and cof8). These correspond to about five hours of video which corresponds to about 8000 trajectories. The system was first tuned and initialised as previously described (i.e. employing a learning dataset with 230 trajectories distributed into 23 trajectory types). Figure 3.11 shows the online system learning as the different video sequences (datasets) are processed.
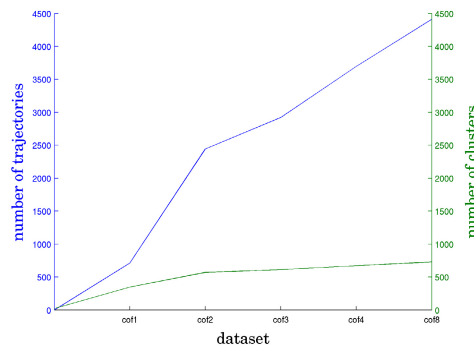
Fig. 3.11 Number of processed trajectories (blue curve) and number of trajectory clusters created by the online system as the different datasets are sequentially processed. Remark that the number of trajectory clusters does not increase much in relation to the number of trajectories analysed after the processing of the 'cof2' dataset.

The structure of the resulting clustering after the processing of a given dataset can be measured again with unsupervised evaluation indexes (i.e. the intra-cluster homogeneity and inter-cluster separation) as in section 3.4 'Trajectory analysis'. We calculated the Silhouette index for the clustering partition induced on each analysed dataset. Table 3.3 below gives such results. When comparing these Silhouette values with those obtained in section 'Trajectory analysis' for the evaluation of the clustering algorithm, we can deduce that the analysed datasets contain still high levels of complexity/noise.

Table 3.3 Clustering structure evaluated by the Silhouette index on the processed datasets.

dataset	Silhouette Index
cof1	0.45
cof2	0.24
cof3	0.43
cof4	0.39
cof8	0.39

We employed the trajectory clusters to measure the similarity between the different datasets. For this purpose a histogram was built for each dataset where each bin of the histogram represents the number of mobile objects being associated with that particular trajectory cluster. The similarity between datasets comes down to measuring the similarity between the established histograms. For this purpose we employ the Kullback-Leibler divergence measure given next for any two different histograms *h1* and *h2*:

$$KL(h1,h2) = \sum_r p_{h1}(r) \log \frac{p_{h1}(r)}{p_{h2}(r)} + \sum_r p_{h2}(r) \log \frac{p_{h2}(r)}{p_{h1}(r)} \qquad (3.12)$$

and r is a given bin on the histogram of trajectories.

Because the Kullback-Leibler divergence measure is a non-bounded measure, which equals to zero when *h1=h2*, we actually calculate the correlation (*corr*) between the different datasets by adding a normalisation factor and a unit offset:

$$corr(h1,h2) = 1 + \frac{KL(h1,h2)}{\sum_r p_{h1}(r) \log(p_{h1}(r)) + \sum_r p_{h2}(r) \log(p_{h2}(r))} \qquad (3.13)$$

The correlation between the different datasets is then given in the table below.

Table 3.4 Trajectory-based correlation between the different analysed datasets.

	cof1	cof2	cof3	cof4	cof8
cof1	1	0.77	0.75	0.79	0.76
cof2	0.77	1	0.78	0.80	0.78
cof3	0.75	0.78	1	0.76	0.71
cof4	0.79	0.80	0.76	1	0.81
cof8	0.76	0.78	0.71	0.81	1

From the trajectory-based correlation table it can then be observed that sequences cof2, cof4 and cof8 are the most similar, although in general all five sequences do contain a large number of common trajectories as their minimum correlation value is above 0.75.

3.7   Conclusions

Activity clustering is one of the new trends in video understanding. Here we have presented an on-line learning approach for trajectory and activity learning. Previous state of the art has mainly focused on the recognition of activities mostly with the aim to label them as normal/suspicious/dangerous. However, the adaptation/update of the activity model with the analysis of long term periods has only been partially adressed. Moreover, most state of the art on activity analysis has been designed for the case of structured motions such as those observed in traffic monitoring (vehicle

going straight on the road, vehicle turning on a round-about, ...) or specific iso-
lated body motions like walking, running, jumping. In this paper, we have adressed
the problem of incremental learning of unstructured spatial motion patterns. The
proposed algorithm allows monitoring and processing large periods of time (large
amounts of data), and thus perform analysis on a long-term basis. The proposed
approach employs a simple, yet advantageous incremental algorithm: The Leader
algorithm. The strongest advantage is that it requires only a single scan of the data,
and only cluster representatives need to be stored in the main memory. Generally,
incremental approaches rely on a manually-selected threshold to decide whether the
data is too far away from the clusters. To improve this approach we propose to con-
trol the learning rate with coefficients indicating when the cluster can be updated
with new data. We solve the difficulty of tuning the system by employing a training
set and machine learning. The system respects the main principles of incremental
learning: The system learns new information from datasets that consecutively be-
come available. The algorithm does not require access to previously used datasets,
yet it is capable of largely retaining the previously acquired knowledge and has no
problem of accommodating any new classes that are introduced in the new data.
In terms of the studied application, the system has thus the capacity to create new
clusters for new trajectories whose type had not been previously observed. Exhaus-
tive evaluation is made on synthetic and real datasets employing unsupervised and
supervised evaluation indexes. Results show the ability of trajectory clusters to char-
acterise the scene activities. In this work we have adressed only the recognition of
single mobiles appearing in the scene. In a future work, we will adress group-related
activity such as 'Meeting' (trajectory merging) and 'Splitting'. Our future work will
also include more exhaustive analysis with temporal information, extracted from
trajectories, to achieve more precise behaviour characterisation and to distinguish
between mobiles moving at 'walking' speed or higher speed. We will also include
normal/abnormal behaviour analysis from trajectory clustering.

Acknowledgements

References

[1] Cofriend, `http://easaier.silogic.fr/co-friend/`,
    `http://easaier.silogic.fr/co-friend/`
[2] Anjum, N., Cavallaro, A.: Single camera calibration for trajectory-based behavior
    analysi. In: IEEE Conference on Advanced Video and Signal Based Surveillance, pp.
    147–152. IEEE, Los Alamitos (2007)
[3] Antonini, G., Thiran, J.: Counting pedestrians in video sequences using trajectory clus-
    tering. IEEE Transactions on Circuits and Systems for Video Technology 16, 1008–
    1020 (2006)

[4] Avanzi, A., Bremond, F., Tornieri, C., Thonnat, M.: Design and assessment of an intelligent activity monitoring platform. EURASIP Journal on Advances in Signal Processing 2005, 2359–2374 (2005)

[5] Bashir, F., Khokhar, A., Schonfeld, D.: Object trajectory-based activity classification and recognition using hidden markov models. IEEE Transactions on Image Processing 16, 1912–1919 (2007)

[6] Campello, R., Hruschka, E.: A fuzzy extension of the silhouette width criterion for cluster analysis. Fuzzy Sets and Systems 157, 2858–2875 (2006)

[7] Carpenter, G.A., Grossberg, S.: A self-organizing neural network for supervised learning, recognition, and prediction. IEEE Communications Magazine 30, 38–49 (1992)

[8] Carpenter, G.A., Grossberg, S., Reynolds, J.: Artmap: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. Neural Networks 4, 565–588 (1991)

[9] Davies, D., Bouldin, D.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 224–227 (1979)

[10] Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Cybernetics and Systems 4, 95–104 (1974)

[11] Foresti, G., Micheloni, C., Snidaro, L.: Event classification for automatic visual-based surveillance of parking lots. In: Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol. 3, pp. 314–317. IEEE, Los Alamitos (2004)

[12] Fusier, F., Valentin, V., Bremond, F., Thonnat, M., Borg, M., Thirde, D., Ferryman, J.: Video understanding for complex activity recognition. Machine Vision and Applications 18, 167–188 (2007)

[13] Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California, United States (1999)

[14] Hartigan, J.A.: Clustering algorithms. John Wiley & Sons, Inc., New York (1975)

[15] Jantke, K.: Types of incremental learning. In: AAAI Symposium on Training Issues in Incremental Learning, pp. 23–25 (1993)

[16] Kaufman, L., Rousseeuw, P.: Finding groups in data. An introduction to cluster analysis, New York (1990)

[17] Lange, S., Grieser, G.: On the power of incremental learning. Theoretical Computer Science 288, 277–307 (2002)

[18] Livny, M., Zhang, T., Ramakrishnan, R.: Birch: an efficient data clustering method for very large databases. In: ACM SIGMOD International Conference on Management of Data, Montreal, vol. 1, pp. 103–114 (1996)

[19] Muhlbaier, M.D., Topalis, A., Polikar, R.: Learn++.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. IEEE Transactions on Neural Networks 20, 152–168 (2009)

[20] Naftel, A., Khalid, S.: Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. Multimedia Systems 12, 227–238 (2006)

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

[22] Piciarelli, C., Foresti, G., Snidaro, L.: Trajectory clustering and its applications for video surveillance. In: Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance, 2005, pp. 40–45. IEEE, Los Alamitos (2005)

[23] Polikar, R., Upda, L., Upda, S., Honavar, V.: Learn++: an incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) 31, 497–508 (2001)

[24] Porikli, F.: Learning object trajectory patterns by spectral clustering. In: 2004 IEEE International Conference on Multimedia and Expo (ICME), vol. 2, pp. 1171–1174. IEEE, Los Alamitos (2004)

[25] Seipone, T., Bullinaria, J.: Evolving neural networks that suffer minimal catastrophic forgetting. In: Modeling Language, Cognition and Action - Proceedings of the Ninth Neural Computation and Psychology Workshop, pp. 385–390. World Scientific Publishing Co. Pte. Ltd., Singapore (2005)

[26] Sharma, A.: A note on batch and incremental learnability. Journal of Computer and System Sciences 56, 272–276 (1998)

[27] Tan, P.N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Reading (2005)

[28] Vu, V.-T., Brémond, F., Thonnat, M.: Automatic video interpretation: A recognition algorithm for temporal scenarios based on pre-compiled scenario models. In: Crowley, J.L., Piater, J.H., Vincze, M., Paletta, L. (eds.) ICVS 2003. LNCS, vol. 2626, pp. 523–533. Springer, Heidelberg (2003)

[29] Vijaya, P.: Leaders subleaders: An efficient hierarchical clustering algorithm for large data sets. Pattern Recognition Letters 25, 505–513 (2004)

[30] Zhou, Z., Chen, Z.: Hybrid decision tree. Knowledge-based systems 15, 515–528 (2002)

Appendix 1

Silhouette Index

The Silhouette index is defined as follows: Consider a data object v_j, $j \in \{1, 2, \cdots, N\}$, belonging to cluster cl_i, $i \in \{1, 2, \cdots, c\}$. This means that object v_j is closer to the prototype of cluster cl_i than to any other prototype. Let the average distance of this object to all objects belonging to cluster cl_i be denoted by a_{ij}. Also, let the average distance of this object to all objects belonging to another cluster i' $i \neq i'$ be called $d_{i'j}$. Finally let b_{ij} be the minimum $d_{i'j}$ computed over $i' = 1, \cdots, c$ which represents the dissimilarity of object j to its closest neighbouring cluster. The Silhouette index is then

$$S = \frac{1}{N} \sum_{j=1}^{N} s_j \ and \ s_j = \frac{b_{ij} - a_{ij}}{max(a_{ij}, b_{ij})}$$

Larger values of S correspond to a good clustering partition.

Dunn Index

The Dunn index is defined as follows: Let cl_i and cl_i be two different clusters of the input dataset. Then, the diameter Δ of cl_i is defined as

$$\Delta(cl_i) = \max_{v_j, v_{j'} \in cl_i} \left\{ d\left(v_{j'}, v_j\right) \right\}$$

Let δ be the distance between cl_i and cl_i Then δ is defined as

$$\delta\left(cl_i, cl_{i'}\right) = \max_{v_j \in cl_i, v_{j'} \in cl_{i'}} \left\{d\left(v_{j'}, v_j\right)\right\}$$

and, $d(x, y)$ indicates the distance between points x and y. For any partition, the Dunn index is

$$D = \min_i \left\{ \min_{i'} \left\{ \frac{\delta\left(cl_i, cl_{i'}\right)}{\max_i\left(\Delta\left(cl_i\right)\right)} \right\} \right\} \text{ and } i, i' \in \{1, \cdots, N\}, i' \neq i$$

Larger values of D correspond to a good clustering partition.

Davies-Bouldin Index

The Davies-Bouldin index is defined as follows: This index is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The scatter within cluster, cl_i, is computed as

$$S_i = \frac{1}{|cl_i|} \sum_{v_j \in cl_i} \left\{ \|v_j - m_i\| \right\}$$

m_i is the prototype for cluster cl_i. The distance δ between clusters cl_i and cl_i is defined as

$$\delta\left(cl_i, cl_{i'}\right) = \|m_{i'} - m_i\|$$

The Davies-Bouldin (*DB*) index is then defined as

$$DB = \frac{1}{N} \sum_{i=1}^{N} R_i \text{ with } R_i = \max_{i,i'} R_{ii'}; \, i, i' \in \{1, \cdots, N\}, i' \neq i \text{ and } R_{ii'} = \frac{S_i + S_{i'}}{\delta\left(cl_i, cl_{i'}\right)}$$

Low values of the *DB* index are associated with a proper clustering.

Appendix 2

Jaccard Index

Consider $C = CL_1, \cdots, CL_m$ is a clustering structure of a data set $X = \{v_1, v_2, \cdots, v_n\}$; and $P = \{P_1, \cdots, P_s\}$ is a defined partition of the data .

We refer to a pair of points (v_i, v_j) from the data set using the following terms:

- *SS*: if both points belong to the same cluster of the clustering structure C and to the same group of partition P.
- *SD*: if points belong to the same cluster of C and to different groups of P.

- *DS*: if points belong to different clusters of *C* and to the same group of *P*.
- *DD*: if both points belong to different clusters of *C* and to different groups of *P*.

Assuming now that *a*, *b*, *c* and *d* are the number of *SS*, *SD*, *DS* and *DD* pairs respectively, then $a+b+c+d=M$ which is the maximum number of all pairs in the data set (meaning, $M = N(N-1)/2$ where N is the total number of points in the data set). Now we can define the Jaccard index (*J*) measuring the degree of similarity between *C* and *P*:

$$J = \frac{a}{a+b+c}$$