Clustering of Trajectories in Video Surveillance Using Growing Neural Gas

Javier Acevedo-Rodríguez[1], Saturnino Maldonado-Bascón[1], Roberto López-Sastre[1], Pedro Gil-Jiménez[1], and Antonio Fernández-Caballero[2,3]

[1] University of Alcala, Teoría de la señal y Comunicaciones, Alcalá de Henares, Spain
http://agamenon.tsc.uah.es/Investigacion/gram/index.html
[2] Instituto de Investigación en Informática de Albacete (I3A), Universidad de Castilla-La Mancha, 02071 Albacete, Spain
[3] Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, 02071 Albacete, Spain

Abstract. One of the more important issues in intelligent video surveillance systems is the ability to handle events from the motion of objects. Thus, the classification of the trajectory of an object of interest in a scene can give important information to higher levels of recognition. In this context, it is crucial to know what trajectories are commonly given in a model in order to detect suspect ones. This implies the study of a set of trajectories and grouping them into different categories. In this paper, we propose to adapt a bioinspired clustering algorithm, growing neural gas, that has been tested in other fields with high level of success due to its nice properties of being unnecessary to know a priori the number of clusters, robustness and that it can be adapted to different distributions. Due to the fact that human perception is based on atomic events, a segmentation of the trajectories is proposed. Finally, the obtained prototype sub-trajectories are grouped according to the sequence of the observed data to feed the model.

Keywords: Trajectory clustering, growing neural gas, high-level video surveillance.

1 Introduction

The research on intelligent video surveillance systems has been intensive for the last years with emphasis on obtaining high-level information for interpreting a scene in order to give some kind of alarms when an event detection has been triggered. One of the issues that has gained attention is to automatically recognize behaviors, where the movement followed by the objects of interest can give rich information about specific or suspicious behaviors. Trajectory classification is then placed between low level stages such as segmentation, object recognition or tracking and high level interpretation of the scene. The trajectories followed by objects of interest in the image is a valuable source of information to automatically detect these suspicious behaviors or generate some kind of alarms [13].

J.M. Ferrández et al. (Eds.): IWINAC 2011, Part I, LNCS 6686, pp. 461–470, 2011.

In order to define the model of the surveillance system it is of high interest to associate objects with common observed trajectories. As the number of trajectories grows it becomes impractical to group and identify them manually, so it is necessary to use a clustering algorithm that can also provide prototypes of trajectories and detect outliers.

Trajectory matching involves the measure of the similarity respect to some kind of distance metric between two or more trajectories. The most trivial distance measure is the Euclidean one, but it should be noted that two different trajectories can have different number of points. In [9] trajectories are re-sampled so that all of them have the same number of points, allowing the use of the Euclidean distance as a metric of similarity. Nevertheless, this measure is suboptimal when comparing sub-trajectories obtained as a result of some kind of occlusion in the tracking stage. This fact makes necessary the research on other measures that do not require the two trajectories to have the same number of points. One of the measures in this second group, widely used in trajectory comparison, is Dynamic Time Warping (DTW) [19], [18]. Another measure that do not need the same number of points is the Longest Common SubSequence (LCSS) used in [5] or [7]. The modified Hausdorff distance introduced in [2] adapts the Hausdorff distance to compare trajectories keeping the order of the points in the sequences. In [14] the distance used was called Trajectory Directional Histogram and it is based on the histogram of the angles obtained from the sequence.

Previous works have explored clustering methods in order to group trajectories. The most extended is K-means and its soft version fuzzy K-means, used in [12] or [9]. Agglomerative techniques have also been used as described in [1] or [5]. A common problem with these approaches is that it is necessary to know a priori the number of centroids or classes of the problem and make some assumptions about the distribution of the data. Spectral clustering is another method found in the literature, that do not make any assumption on the distribution of data points and approximates an optimal graph partition [16]. Its use in trajectory clustering is described in [3]. However, this last approach does not provide prototype trajectories, which can be one of the targets if we are interested in a model of the video scene. A deeper review of the state of the art in trajectory classification and clustering can be found in [15]. More recently, several works indicate that the problem of trajectory classification and clustering can be easily solved by dividing it into sub-trajectories. This solution is based on the way humans recognize trajectories by dividing them into atomic units of actions that are of substantial value for perception [4], [17].

Growing neural gas [8] (GNG) is a bioinspired algorithm for finding optimal representation of feature vectors. Its name comes from the behaviour of the vectors during the adaptation process that distributes them like a gas in space. It is based on growing self organizing maps and has been used for clustering analysis [6]. The advantage as a clustering algorithm is that, with enough training time, it can adapt to clusters with very different nature, keeping more neurons in those clusters that are more complex to be represented. This paper proposes to adapt this algorithm to the problem of trajectory clustering using segments

of the trajectories observed in a video scene. The obtained weights indicate prototype sub-trajectories that are useful to generate prototype trajectories to feed the model. To this end, an algorithm that studies sequences of prototype trajectories was designed. This algorithm studies the observed trajectories and associates them to one of the clusters provided by the GNG algorithm.

2 Proposed Algorithm

2.1 Trajectory Segmentation

Let's suppose that a trajectory T_i is defined by a set of n_i 2D-points, corresponding to consecutive positions of a tracked object of interest, observed at equally spaced intervals of time, $T_i = \{(x_1, y_1), (x_2, y_2), ..., (x_{n_i}, y_{n_i})\}$. As mentioned, the number of points n_i of each trajectory can change from one trajectory to other, and depends on several factors such as the path followed by the object, the velocity or a truncation in the trajectory due to some occlusions. As it was described in the preivous section, it is more complex to define a distance measure when the whole trajectory is considered. In this paper we propose to divide each trajectory in linear segments by using a fast realization of the Douglas-Peucker (DP) algorithm [11]. This algorithm was already used in trajectory compression [10] but in this case, we propose to use these segments as the inputs of the GNG network. As it is shown in Figure (1) after the DP algorithm has been applied to trajectories, these are transformed into $T_i \rightarrow S_i = \boldsymbol{s}_1^i, \boldsymbol{s}_2^i, .., \boldsymbol{s}_d^i$, being $d_i << n_i$ the number of segments extracted, and each of the elements $s_j^i \in \mathbb{R}^4$ is composed by the coordinates of the starting point (x_s, y_s) and the end point (x_f, y_f) concatenated. Segments are then treated as vectors, all of them with the same dimension, and then the clustering algorithm is expected to find prototype segments.

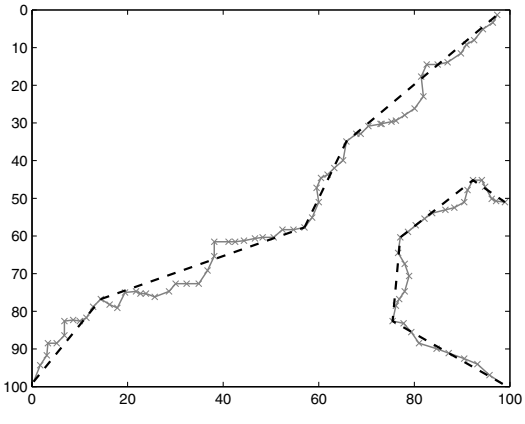

Fig. 1. Douglas-Peucker algorithm applied to extract linear segments from trajectories

The only parameter to be adjusted on this stage is τ defined as:

$$\tau = \max_i \left(\min_j d\left((x_i, y_i), \boldsymbol{s}_j \right) \right) . \tag{1}$$

where $d\left((x_i, y_i), \boldsymbol{s}_j \right)$ is the distance between point (x_i, y_i) of the original trajectory and the segment $\boldsymbol{s}_j$ of the linearized trajectory.

2.2 Growing Neural Gas

Once we have extracted all the segments from the observed set of trajectories, the GNG algorithm is used to find clusters of sub-trajectories. The data used in the training process is the whole set of segments, from all the trajectories, found in the previous stage. This training set is defined as $\boldsymbol{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, ...\boldsymbol{s}_m\}$, where m is the total number of segments. It is important to highlight that, in order to find the clusters, the information about what trajectory is producing the segment $\boldsymbol{s}_i$ is not present. The procedure can be summarized in the following steps:

1. Define a maximum number of iterations, *mxiter*, a maximum number of nodes *mxnodes*, the actual number of nodes $n = 0$, a matrix of weights $\boldsymbol{\Omega} = \emptyset$, a vector of errors, E = 0, a matrix of connections, C = 0, and set the actual iteration $iter = 1$.
2. Randomly select two segments from the training set, $(\boldsymbol{s}_a, \boldsymbol{s}_b)$ and assign them as the weights of the two first nodes:

$$\begin{array}{l} \boldsymbol{\omega_1} \leftarrow \boldsymbol{s}_a \\ \boldsymbol{\omega_2} \leftarrow \boldsymbol{s}_b \\ \boldsymbol{\Omega} = \{\boldsymbol{\omega_1}, \boldsymbol{\omega_2}\} \end{array} . \tag{2}$$

   Set a connection $C(1,2) = C(2,1) = 1$ and make $n = 2$.
3. Randomly select a segment,$\boldsymbol{s}_i$ from the training set and search the nearest weight, $\boldsymbol{\omega_f}$ and the second one $\boldsymbol{\omega_s}$. Connect nodes f and s, $C(f,s) = C(s,f) = 1$. Increment the error associated to node f by:

$$E_f = E_f + \|\boldsymbol{s}_i - \omega_f\|^2 \tag{3}$$

   Update the reference vectors of the winner and its direct topological neighbors by fractions ε_a and ε_n, respectively, of the total distance to the input vector:

$$\Delta\boldsymbol{\omega}_f = \varepsilon_a \left(\boldsymbol{s}_i - \omega_f \right); \;\; \Delta\boldsymbol{\omega}_j = \varepsilon_b \left(\boldsymbol{s}_i - \omega_j \right) \;\; j \in \boldsymbol{N}_f \tag{4}$$

   where $\boldsymbol{N}_f$ is the set of direct topological neighbors of node f.
4. Increment the connection value of all neighbors of f, $C(f,j) = C(f,j) + 1, C(j,f) = C(j,f) + 1$. Remove all the connections, $C(f,h) = C(h,f) = 0$, greater than a predefined value α. If in this step a node is found to be isolated, it is removed from all associated vectors and matrices and the number of active nodes is decreased. Increment the value of *iter*.

5. If $mod(iter/\lambda) = 0$ and $n < mxnodes$, where λ is a predefined value, a new node r is inserted. To this end, the node with maximum error is found,

$$q = \max_{i\ \boldsymbol{\omega}_i \in \Omega} E(i)\ . \tag{5}$$

and also the node with maximum error among the neighbors of q,

$$l = \max_{i\ \boldsymbol{\omega}_i \in \boldsymbol{N}_q} E(i)\ . \tag{6}$$

The new node is inserted by breaking the connection between nodes q and l, $C(q,l) = C(l,q) = 0$ and adding the new connection $C(r,q) = C(r,q) = C(l,r) = C(r,l) = 1$. The weight of the new node is:

$$\begin{array}{l} \boldsymbol{\omega}_r = (\boldsymbol{\omega}_l + \boldsymbol{\omega}_q)/2 \\ \Omega = \Omega \bigcup \boldsymbol{\omega}_r \\ E(r) = (E_l + E_q)/2 \end{array} \tag{7}$$

Increase the number n of active nodes.
6. Decrease the error in all the nodes $\Delta E(i) = -\beta E(i)$ where β is a predefined constant. If $iter < mxiter$ repeat the process from point 3. Else stop the algorithm and return the matrix $\boldsymbol{\Omega}$.
7. Find those nodes that are enough close each other, $\|\boldsymbol{\omega}_i - \boldsymbol{\omega}_j\| < \mu$, where μ is a predefined constant and group them into a unique node.

2.3 Prototype Trajectories

The GNG algorithm provides a set of n nodes, each one representing a subtrajectory prototype. In this stage the algorithm searches for common sequences of sub-trajectories in order to build a set of prototype trajectories. These common sequences are extracted from the set of observed trajectories. As it has been highlighted in the previous section, there was a lack of information in the GNG algorithm about what trajectory produce each segment. Now, we have recovered the information obtained in the first step, so instead of considering initial trajectories the sequences of linear segments are studied. Each segment is associated to its nearest node:

$$\begin{array}{l} T_i \rightarrow S_i = \boldsymbol{s}_1^i, \boldsymbol{s}_2^i, .., \boldsymbol{s}_d^i \\ \phi_j^i = \max_z \|\boldsymbol{s}_j^i - \boldsymbol{\omega}_z\|\ j = 1, 2, ..d_i\ z = 1, 2, .., n \end{array} \tag{8}$$

From this assigment, we have mapped each trajectory into a set of indexes associated to prototype sub-trajectories, $\boldsymbol{\Phi}_i = \{\phi_1, \phi_2, ...\phi_{d_i}\}$. A search is now made to obtain those $\boldsymbol{\Phi}_i$ that are repeated more than a given number of times. In our work we have considered a sequence of sub-trajectories to be a prototype if it is repeated more than four times.

3 Results and Discussion

3.1 Clustering Performance

First of all we would need to verify if the GNG method fulfills the requirements of grouping sub-trajectories. To this end we have build an artificial dataset of segments shown in Figure (2) and each one was assigned to a group. We define the accuracy of a clustering algorithm as:

$$\rho = \frac{1}{n} \sum_{i=1}^{k} \sum_{j=1}^{n} \frac{I_i(j)}{1 + O_i(j)} \ . \tag{9}$$

where n is the number of clusters, in our case nodes, k is the number of classes, $I_i(j)$ is the number of samples of class i that are associated to cluster j and $O_i(j)$ is the number of elements that do not belong to class i but are present in cluster j. This criteria is maximized in the case we have the minimal number of clusters that only contain samples of one class. One sample is said to belong to a cluster j if the euclidean distance with this cluster is the minimal among all the distances with clusters and its value is not higher than a threshold. This definition is necessary in the presence of outliers, such as those that can be observed in our example. In our case we have set the threshold value of 30.

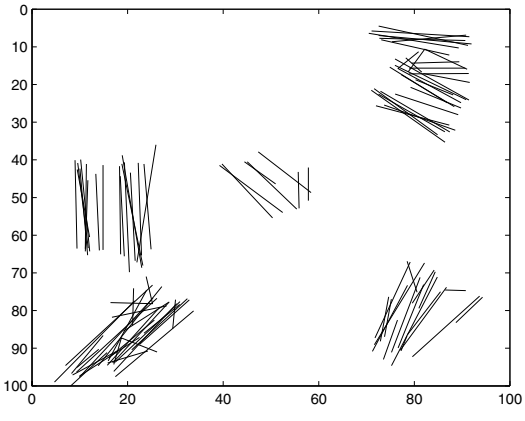

Fig. 2. Artificial problem with different segments distributed in groups and noise

We want to check the influence of two important parameters *maxiter* and λ. Results of the proposed example are shown in Table (1), where the algorithm was executed without the final cluster aggregation proposed in the last step of the algorithm.

From these results, we can extract important conclusions that must be kept in mind when the GNG algorithm is executed. When the number of iterations is very high and λ is very low the algorithm degenerates to every single sample is

Table 1. Cluster Precision for different values of *maxiter* and λ

λ	*maxiter*				
	3000	25000	50000	100000	300000
500	4.12	6.26	7.45	1	1
2000	NA	8.42	13.33	5.26	1
6000	NA	5.36	8.25	11.42	6.52
10000	NA	4.2	5.36	6.42	7.21

a cluster. When the number of iterations is very low and λ is also low there are not enough nodes and their weights are not properly adapted. The extreme case would happen when we have an only cluster. We can check that selecting these parameters produce an adequate number of clusters. In Figure (3) it is shown the prototype segments found with $maxiter = 50000$ and $\lambda = 2000$.

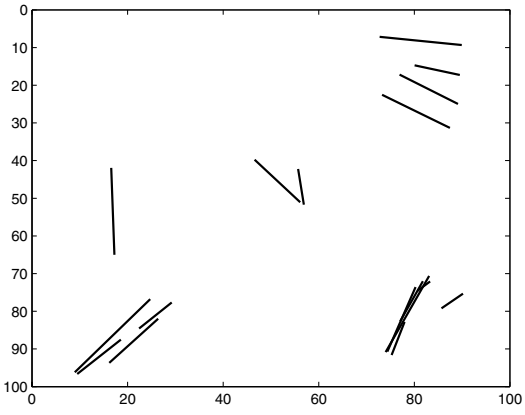

Fig. 3. Nodes found by the GNG method with $maxiter = 50000$ and $\lambda = 2000$

3.2 Trajectories in a Scene

In the previous section it was demonstrated how the GNG algorithm performs appropriately for the problem of noisy segment clustering. In this section the whole algorithm is tested within an image of the entrance lobby of the INRIA Labs at Grenoble, filmed for the CAVIAR project with a wide angle camera. It is necessary to say that, although the image come from real video scene application, the trajectories we have worked on are synthetic trajectories. This was needed to verify the behaviour of the method with complex trajectories and it is usually done in most of the works for trajectory clustering. In Figure (4) some of the synthetic trajectories are shown. It can be noticed how some of them are only parts of completed trajectories, included to mimic the effects of occlusions and problems with tracking algorithms.

Fig. 4. Synthetic observed trajectories in the INRIA scene

In Figure (5) the prototype trajectories found by the algorithm are shown. It should be noted that several trajectories share some of the segments, although in the image they has been represented only in one color. This is logical because we can see in Figure (4) that many trajectories start from narrow paths and disperse later. The parameters used to obtain these results were $\tau = 5$ for the initial DP algorithm to extract segments; the GNG algorithm was tuned according to results obtained in the previous section with $maxiter = 50000$, $\lambda = 2000$, $\varepsilon_a = 0.07$, $\varepsilon_b = 0.008$, $\beta = 0.0005$ and $\alpha = 40$. The μ constant defined to group segments that are closer enough each other was fixed to $\mu = 1000$.

Fig. 5. Prototype trajectories found with the proposed method

As it can be noticed, the algorithm finds prototype trajectories that can feed a model in order to detect abnormal behaviour when a trajectory is segmented and their sequence of segments is far away from any trajectory found by the method. It should be said that partial trajectories have also been used to build

the model, and their segments are used to train the GNG. Opposite to other trajectory clustering methods described in the literature, we do not need to select only complete trajectories and the information of these pieces of trajectories are exploited but as these incomplete trajectories are not frequently repeated they do not build a prototype.

4 Conclusions

In this paper we have introduced a new method to obtain prototype trajectories. Based on recent works that suggest that the problem of trajectory matching can be better solved if pieces of trajectories are considered, we start the problem by dividing observed trajectories into linear segments. The GNG algorithm is used then to find sub-trajectory prototypes and demonstrates that has a good behaviour due to its robustness against outliers and that a number of clusters is not needed a priori. These prototypes are used to build sequences of common subtrajectories, very useful to detect events in higher levels. Very promising results have been obtained using synthetic observed trajectories in real scenes.

Future work will be focused on working with real observed trajectories and feed back the tracking algorithm with the information provided by the trajectory identification algorithm, once that the prototypes have been learned.

Acknowledgement

This work was supported by the Spanish Ministry for Science and Innovation under Projects TIN2010-20845-C03-03, TIN2010-20845-C03-01, Plan Avanza TSI-020302-2009-59 and by the Madrid Government under project CCG10-UAH/TIC-5965.

References

1. Antonini, G., Thiran, J.: Counting pedestrians in video sequences using trajectory clustering. IEEE Transactions on Circuits and Systems for Video Technology 16(8), 1008–1020 (2006)
2. Atev, S., Masoud, O., Papanikolopoulos, N.: Learning traffic patterns at intersections by spectral clustering of motion trajectories. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4851–4856 (2006)
3. Atev, S., Miller, G., Papanikolopoulos, N.: Clustering of vehicle trajectories. IEEE Transactions on Intelligent Transportation Systems 11(3), 647–657 (2010)
4. Bashir, F.I., Khokhar, A., Schonfeld, D.: Object trajectory-based activity classification and recognition using hidden markov models. IEEE Transactions on Image Processing 16(7), 1912–1919 (2007)
5. Buzan, D., Sclaroff, S., Kollios, G.: Extraction and clustering of motion trajectories in video. In: International Conference on Pattern Recognition, pp. 521–524 (2004)
6. Canales, F., Chacón, M.: Modification of the growing neural gas algorithm for cluster analysis. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 684–693. Springer, Heidelberg (2007)

7. Cheriyadat, A., Radke, R.: Automatically determining dominant motions in crowded scenes by clustering partial feature trajectories. In: Proceedings of the First ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2007 (2007)
8. Fritzke, B.: A growing neural gas network learns topologies. In: Advances in Neural Information Processing Systems, vol. 7, pp. 625–632 (1995)
9. Fu, Z., Hu, W., Tan, T.: Similarity based vehicle trajectory clustering and anomaly detection. In: IEEE International Conference on Image Processing, ICIP 2005, vol. 2, pp. 602–605 (2005)
10. Gudmundsson, J., Katajainen, J., Merrick, D., Ong, C., Wolle, T.: Compressing spatio-temporal trajectories. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 763–775. Springer, Heidelberg (2007)
11. Hershberger, J., Snoeyink, J.: Speeding up the douglas-peucker line-simplification algorithm. In: Proc. 5th Intl. Symp. on Spatial Data Handling, pp. 134–143 (1992)
12. Hirasawa, N.S.K., Tanaka, K., Kobayashi, Y., Sato, Y., Fujino, Y.: Learning motion patterns and anomaly detection by human trajectory analysis. In: IEEE International Conference on Systems, Man and Cybernetics, ISIC 2007, pp. 498–503 (2007)
13. Jung, C., Hennemann, L., Musse, S.: Event detection using trajectory clustering and 4-d histograms. IEEE Transactions on Circuits and Systems for Video Technology 18(11), 1565–1575 (2008)
14. Li, X., Hu, W., Hu, W.: A coarse-to-fine strategy for vehicle motion trajectory clustering. In: 18th International Conference on Pattern Recognition, ICPR 2006, vol. 1, pp. 591–594 (2006)
15. Morris, B., Trivedi, M.: A survey of vision-based trajectory learning and analysis for surveillance. IEEE Transactions on Circuits and Systems for Video Technology 18(8), 1114–1127 (2008)
16. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Advances in Neural Information Processing Systems 14, pp. 849–856. MIT Press, Cambridge (2001)
17. Piciarelli, C., Foresti, G.: On-line trajectory clustering for anomalous events detection. Pattern Recognition Letters 27(15), 1835–1842 (2006)
18. Pop, I., Scuturici, M., Miguet, S.: Incremental trajectory aggregation in video sequences. In: 19th International Conference on Pattern Recognition, ICPR 2008, pp. 1–4 (2008)
19. Yanagisawa, Y., Satoh, T.: Clustering multidimensional trajectories based on shape and velocity. In: Proceedings of 22nd International Conference on Data Engineering Workshops, p. 12 (2006)