A Rule-Based Tracking System for Video Surveillance Applications

Luigi Di Stefano, Martino Mola, Giovanni Neri, Enrico Varani
{ldistefano, mmola, gneri, eviarani}@deis.unibo.it, http://vision.deis.unibo.it
DEIS - Department of Electronics Computer Science and Systems, University of Bologna
ARCES – Advanced Research Centre on Electronic Systems "Ercole De Castro"
Via Risorgimento, 2 - 40136 Bologna, ITALY

Abstract. We present a general-purpose tracker structured as a rule-based expert system. The system is aimed at tracking rigid and non-rigid objects in video sequences without any assumption on the working scenario, on the nature of objects and on their possible interactions. This tracking core can easily be integrated with application-specific modules to carry out specific video surveillance task requiring a significant degree of scene understanding.

1. Introduction

A remarkable deal of work have been aimed at developing tracking systems dedicated to specific video surveillance applications, such as for example traffic monitoring [2], gesture recognition [3], detection of people exchanging objects [4]. Typically, "special-purpose" trackers form the basic core of surveillance systems capable of providing a significant degree of scene understanding but are conceived to work within a given, well-defined scenario. In [5] tracking is performed by means of a statistical approach in two predefined environments. In [6] an indoor tracker relies on a specific 3D scene model. In [7] the behaviour of the objects to track is modelled on the basis of a-priori knowledge concerning the scene under surveillance (i.e. a car park).

The goal of our work is to develop a video surveillance system capable of combining to a significant extent the requirement of generality with the ability of providing a considerable degree of scene understanding. The idea is to build a system based on a truly general-purpose tracking core which could be easily integrated with relatively small application-specific modules aimed at exploiting specific knowledge and carrying out specific tasks.

In this framework, we have designed a general-purpose tracking core, described in this paper, which is structured as *a rule-based expert system*. These *AI*-systems process the data stored into a *working memory* by means of a set of *rules*. The *working memory* contains the input data as well as the current assessments reached by the expert system. Rules, stored into a memory area referred to as *knowledge-base*, comprise a condition part and an action part. The condition is a sequence of tests verified on the data. The action is a sequence of instructions which are executed in case the condition is satisfied and that modify the content of the *working memory*. In addition, an *inferential engine* controls the execution, selecting the rules to be fired among those satisfied by data and then executing their action parts. One fundamental advantage of rule-based expert systems is that, since the behaviour of the system is embedded into a neatly identifiable part (i.e. the knowledge base) rather than scattered throughout the program, they can easily be modified or upgraded by adding or removing rules.

The architecture of rule-based expert systems matches naturally the ideas inspiring our work. In fact, the general-purpose tracking core consists of a basic set of *rule-packages* aimed at tracking, as accurately as possible, generic objects and group of objects, either rigid or non-rigid, without any assumption concerning their nature or expected interactions. Then, to perform a given surveillance tasks within a given scenario we can simply add application specific *rule-packages* embodying the knowledge associated with the actual application domain. For example, should the task be to detect luggage left unattended, we could add to the tracker a few rules aimed at triggering an alarm if a tracked object splits in two, one of the two is significantly smaller and remains motionless for a long time. Nevertheless, if the task is to detect suspicious persons in a car park, we may add to the tracker some rules aimed at detecting a relatively small object (which, in the addressed scenario, we can assume to be a person) staying for a long time close to a motionless object located within a specific image area (i.e. a parking place), which we can assume to be a car.

A tracking system structured as a rule-base expert system is described in [8]. However, unlike our general-purpose tracker, this system is explicitly conceived to track vehicles in traffic scenes.

2. Tracking Approach

Our tracking approach distinguishes between *blobs*, which are regions of connected foreground pixels, and *objects*, which are the entities of interest for the tracker and may be made out of multiple blobs. As for the input data, a robust change-detection algorithm based on background subtraction and adaptation [1] classifies the pixels of the input grey-level images into foreground (represented in white) and background (represented in black). Then, blobs are extracted by groping together the foreground pixels belonging to the a single connected component. Each blob is parameterised by the co-ordinates of two opposite corners of the *bounding box* surrounding the blob. At this stage each blob can represent either an entire object, an object subpart or can be generated by noise.

The first step of the tracking process is performed by a function, called *move_box*, that refines the *boundig boxes* associated with the blobs by including into a single, larger box those group of blobs which are located at small mutual distances.

By comparing the bounding boxes found in two contiguous frames the tracking algorithm is able to assess for each blob of the previous frame if it undergoes a split or takes part in a merger. The tracking task consists in establishing the associations between the objects found in the previous frame and the blobs just extracted and grouped within the bounding boxes. We describe now the tracking strategy, i.e. the strategy followed by the algorithm when the different events are detected.

- Straightforward tracking: *this is the simplest tracking event, and is assumed by the algorithm whenever it is possible to track an object without ambiguities, i.e. when two blobs without neighbouring ones are detected approximately in the same position in two contiguous frames and there are no splits nor merges. In this case the blob in the frame at time t will be associated to the object found in the frame at time t-1. The concept of "approximately in the same position" is implemented trough the definition of a threshold on a distance measurement between the blobs.*
- Split: *a split is detected when a blob breaks in two distinct ones. Our algorithm policy validates every split as soon as it occurs, creating two new objects to be tracked. However, the tracker will resume the original object identity if, successively, it can reliably classify the split as a temporary fragmentation of the object into two blobs, which may be due, for example, to a poor detection carried out by the change detection*

algorithm. The objects born from a split share the same trajectory up to time t-1 and have independent paths from time t on.

- Merging: *the tracking algorithm detects a merging event when two objects having close past trajectories and detected up to frame at time t-1 merge their bounding boxes in the frame at time t. If these conditions are satisfied, the algorithm creates a new object joining the trajectories of the two previous ones.*
- Occlusion: *this event is considered as a merging sub-case. Occlusions are distinguished from merging events on the basis of past trajectories: if the directions of the past trajectories of the two merged objects differ significantly, then the algorithm classifies the merging as an occlusion, i.e. a merging of two distinct objects. When an occlusion occurs, the algorithm "freezes" the updating of the features of the two distinct objects and waits for their successive separation in two distinct blobs. At the time of the separation, the algorithm calls a matching function which measures the similarity of the frozen objects to the new blobs resulting from the separation. In this process, the algorithms tries all the possible combinations in order to discover the correct association between the objects found before and after the occlusion.*

Several features can be considered to measure the similarity and carry out the blob associations in case of occlusion. Currently, the matching function compares the velocities of the frozen objects to those of the new blobs. In the comparison, it assigns more significance to the direction than to the magnitude. The underlying assumption is that after the occlusion the objects will keep roughly their originals trajectories.

- Disappearance: *an object detected in a frame at time t-1 is classified as lost in the frame at time t if no blob is present in the nearness of the expected object position at time t. If an object is lost in proximity of an image border, the algorithm assumes that the object has left the scene, else waits for the appearance of the object in proximity of the place where it disappeared.*
- Noise: *we have implemented a noise filter in the tracking algorithm that analyses the trajectories and the persistence in a place of an objects. If an object is almost stationary and does not move away significantly from the place where it has born, it is classified as noise and not shown in the output display.*

Since the current tracking assessments are reached by the algorithm based on the most mature analysis of the input data, the algorithm assume that the current assessments are more reliable than the past ones, which had been reached on the basis of a more limited analysis. Hence, if an actual assessment turns out to be incoherent with an older one, the algorithm will withdraw the old assessment in reason of the new, more reliable one. According to this strategy, the tracker is able to revise the past scene interpretation as soon as more mature information becomes available. Clearly, this "playback" feature cannot be exploited in real-time applications, since in this case the system must deliver the best possible information at each new frame. However, it is particularly effective when the task is to analyse a previously recorded video sequence.

The above described tracking strategy has been formalized through a set of rules which constitute system's *knowledge-base*. At every new frame the *inferential engine* verifies the condition parts by examining the blob and object data stored into the *working memory* and then fires the applicable rules.

The tracking system has been implemented using a package [9] that allows to encapsulate in a *C++* class the overall rule-based expert system. This class can be accessed from any *C++* application and contains the *inferential engine*, the *working memory* and the *knowledge-base*. In addition to those contained in the *knowledge-base*, other basic tracking concepts have been defined in *C++* classes. For example we have defined classes implementing the concepts of *blob* and *object*, possessing methods such as *trajectory-similarity estimation* and *velocity-similarity estimation*, as well as attributes

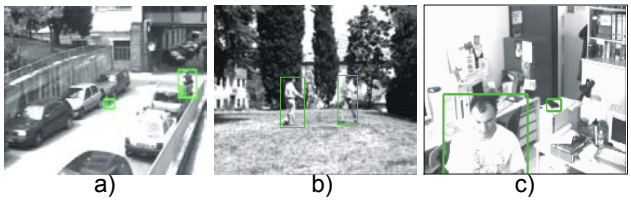

Figure 1 a): *Outdoor7*, b): *Park2*, c): *Wallet*

defining object's and blob's data structures. At runtime, when a rule needs to use a basic concept, a class instance is created and stored into the *working memory*.

3. Experimental Results

We ran the rule-based tracking algorithm on several video sequences in order to evaluate its performances. In this section we present the results obtained on three sequences (a shot taken from each sequence is shown in Figure 1) containing occlusions, splits, mergers and noise.

For each sequence, the results are presented in a graph. The graphs allow to evaluate the performance of the tracker since they show what the tracker should find (ground-truth) and what it has actually found (measurements). Graphs have a couple of bars associated with each object present in the considered sequence. The upper, solid-colour bar represents the ground truth, i.e. the interval of frames in which the object is actually present in the scene. The lower bar can be hatched in two different manners depending on the output yield by the tracker. A bar filled with vertical lines means that the object has been tracked correctly, a bar filled with oblique lines represents tracking errors. Examples of possible errors are: wrong localisation of the object's bounding-box, object not detected, object appearing in the tracking output but not present in the scene (i.e.: objects muddled with others, lost objects, non existing objects).

Each of the following sub-section contains the results on one sequence. A short description of the sequence is also given. Three videos showing the tracking output superimposed to the original sequences can be found at the web-site: http://137.204.57.5/research/ videosurveillance/demo%20tracking/default.htm

3.1. Sequence Outdoor7

This sequence (see Figure 1-a) contains an occlusion handled correctly by the system: a dog appears from left, then a bike enters from the right and around frame 160 they merge together in a single blob. After the occlusion, around frame 180, two distinct blobs appear again and hence the *inferential engine* executes the matching function. Since after the occlusion the two objects maintain their original trajectories the velocity-based matching process succeeds in recovering the correct objects associations.

This is shown in Figure 2, where for both objects the lower, measurements bar is always filled with vertical lines. It is worth pointing out that during the occlusion (i.e. roughly from frame 160 to frame 180) the two object are shown in the graph as tracked correctly. In fact, during this interval, though the system tracks a compound object (i.e. a group), it knows exactly that this compound object is made out of the two merged ones (the output shows a single bounding box with the labels of the two merged objects).

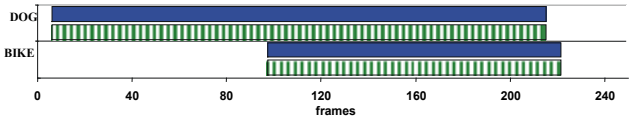

Figure 2 Tracking results on *Outdoor7*

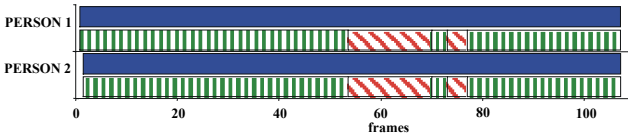

Figure 3 Tracking results on *Park 2*

3.2. Sequence Park2

In this sequence (see Figure 1-b) two persons enter the scene from opposite sides, met at the centre of the image, shakes hands and then restart walking. The trees in the background wave significantly, producing noisy intensity variations that cause the change-detection algorithm to detect spurious foreground regions. When the two persons met, their blobs merge and then split for 3 successive times. Each of the three mergers is classified by the system as an occlusion, and then handled as described in Section 2. After the split following the first occlusion, the matching function fails the objects association, swapping their labels. This happens because the merging of the two blobs is caused by the shaking of hands, with blobs splitting as soon as the two persons cease shaking hands; in such a situation the persons are almost motionless and hence velocities are scarcely meaningful to establish the associations. The second occlusion is handled correctly, and thus the objects keep the wrong labels assigned them after the first occlusion. The third occlusion is managed erroneously by the system, so that the two labels are swapped again and the final result is that, due to the compensation of two errors, the objects keep their correct labels after the three successive occlusions. The graph in Figure 3 shows the tracking errors after the first and the second occlusion. It is worth pointing out that the noise due to waving trees is completely filtered out by the system thanks to the *Noise* rule.

3.3. Sequence Wallet

This sequence is useful to illustrate the “playback” feature. A person enters a room, leaves a wallet on a PC-case and walks towards the camera (the shot in Figure 1-c is taken towards the end of the sequence).

Initially, the system cannot detect that the tracked blob contains two objects (the person holds the wallet in his hand). As soon as the person leaves the wallet on the PC-case the original object splits into two new ones. Hence, the system updates the identity of the previously tracked object classifying it as a group of two (the person and the wallet). Since we processed a previously recorded sequence, the output of the tracker shows the classification of the initial object as a group of two objects from the beginning of the sequence.

Accordingly, the graph in Figure 4 indicates that the person and the wallet are tracked correctly from the beginning of the sequence.

4. Conclusions and future work

We have presented a general-purpose tracking system structured as a rule based expert system. The system tracks the binary blobs generated by a change-detection algorithm in absence of any assumption concerning their nature and the working scenario. The tracking strategy allows for modifying the past scene interpretation as soon as more mature information becomes available. This feature is useful to analyse a previously recorded video sequence.

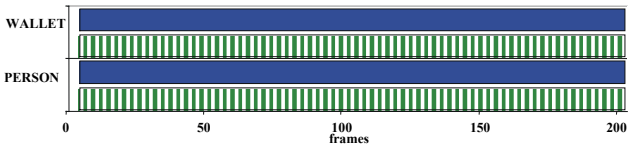

Figure 4 Tracking results on *Wallet*

Our experimental results show that the system can handle correctly occlusion events (see the results on sequence *Outdoor7*) thanks to its ability to match the objects frozen before the occlusion with the new blobs found after the separation. So far the system deploys a matching function based on velocity measurements and assumes that objects tend to keep their original trajectories. This approach is typically effective in relatively simple scenes (e.g. it is potentially effective to track vehicles in traffic scenes) but, in several circumstances, can lead to tracking errors in (see the results on sequence *Park2*).

Our future work will be aimed at developing a more general and robust matching strategy. In this context, we are currently developing a strategy, which seems very promising from our preliminary results, based on a function which matches grey-level patterns and weights the matching score depending on the degree of overlapping between the associated binary blobs.

References

[1] L. Di Stefano, G. Neri, E. Viarani, "Analysis of Pixel-Level Algorithms for Video Surveillance Applications", *Proceedings of 11th International Conference on Image Analysis and Processing ICIAP2001, 26*-28th September 2001, Palermo, Italy, Pages: 542 – 546.

[2] L. Di Stefano, E. Viarani, "Vehicle Detection and Tracking Using the Block Matching Algorithm", *Recent Advances in Signal Processing and Communications* N. Mastorakis Editor, World Scientific and Engineering Society Press, ISBN 960-8052-03-3, 1999.

[3] J. Sherrah, S. Gong, "VIGOUR: a System for Tracking and Recognition of Multiple People and their Activities", *15th Int.l conference on Pattern Recognition ICPR*, 3-8th September 2000, Barcelona, Spain, Vol. 1, Pages: 179 – 182

[4] I. Haritaoglu, R Cutler, D. Harwood, L.S. Davis, "Backpack: Detection of People Carrying Objects Using Silhouettes", *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999. Pages: 102 - 107 vol.1

[5] B. Gidas, C. Robertson, M. Pereira de Almeida, "Tracking of Moving Objects in Cluttered Environments via Monte Carlo Filter", *Proceedings of 15th International Conference on Pattern Recognition*, ICPR 2000, Pages: 175 –178, vol.1

[6] N. Rota, M. Thonnat, "Video Sequence Interpretation for Visual Surveillance", *Proc. of Third IEEE Int.l Workshop on Visual Surveillance*, Dublin - Ireland, IEEE, Pages: 59 – 68, ISBN: 0-7695-0698-4

[7] Y. Ivanov, C. Stauffer, A. Bobick, W. Grimson "Video Surveillance of Interactions" *Proc of Second International Workshop on Visual Surveillance*, June 1999 , pages 82-89, Fort Collins, Colorado

[8] R. Cucchiara, M. Piccardi, P. Mello "Image Analysis and Rule-Based Reasoning for a Traffic Monitoring System", *IEEE Trans. on Intelligent Transportation Systems*, 2000, Vol. 1, N. 2, June 2000, Pages 119-130

[9] ILOG "Rules 6.1", Users Manual, Http://www.ilog.com