Visual Dynamic Event Recognition with Support Vector Machines

M. Pittore* A. Granili† R. Montobbio‡ **A. Verri**§

Unità operativa di Genova

Computer vision systems that learn the tasks they are supposed to perform are more likely to adapt effectively to the changing environment. In this note we describe a system able to learn to represent and recognize visual dynamic events from examples. The system detects dynamic events through a simple low-level vision module and then represent and recognize a detected event by means of Support Vector Machines (SVMs). SVMs, a new method for solving function estimation problems recently introduced by Vapnik [2], are attracting increasing attention in both the machine learning [1, 5] and computer vision community [3, 4] for their theoretical and practical properties. SVMs require the solution of a QP problem with linear constraints. In the dual formulation the dimensionality of the problem is equal to the number of examples.

The system *detects* and *classifies* dynamic events from a sequence of images acquired by a camera looking down a corridor from a fixed position. Occasionally, one or more people are expected to walk up or down the corridor, possibly entering or leaving one of the adjacent rooms. The system task is to

- detect the presence of moving people in the observed scene, and
- recognize their trajectory.

The ability of learning from examples endows the system with the flexibility sufficient to effectively work in different environments and perform different tasks. In its current version the system is able to

(1) detect the beginning and end of a dynamic event, and

(2) compute a stereotyped representation of the observed event.

While in the training stage the event label is known *a priori*, in the test stage the system is able to

(3) classify the detected event on the basis of the computed representation.

The system consists of three modules, a *detection*, a *representation*, and a *recognition* module. The detection module is the only module based on low-level visual processing. Since the camera is fixed and the background stationary, we employ a simple method which, while periodically updating the background image (kept fixed), starts off by locating all pixels of the current image at which the gray level difference with respect to the background image is larger than a fixed threshold m. If a *significative* change is detected we *(a)* form a minimal rectangular box of center $\mathbf{c}$ enclosing all nonisolated pixels where the change has been detected, and *(b)* track the box over the image sequence by means of a linear Kalman filter until the change remains significant. A change is considered *significant* if the number N of connected pixels over which a change is detected exceeds a given threshold t_1, and *remains* significant until N remains higher than a second (lower) threshold t_2. Events that last for less than a minimum number of frames, f, are discarded as noise.

The choice of the representation to adopt is a crucial aspect of the *representation* module and of the entire system. The fact that the event to be represented neither lasts the same amount of time, nor has the same spatio-temporal evolution (not even within events corresponding to the same class) makes this choice particularly difficult. For the time being, we adopted a very simple representation that satisfies the basic requirement of providing a stereotyped representation (in practice, a feature vector with a fixed number of components for each event and with each component carrying comparable information across all events).

The representation is built in three steps. In the first step, the detected event is described by three T-dimensional vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ (where T is the event duration in number of frames). The vectors $\mathbf{u} =$

*INFM - DISI, Università di Genova, pittore@disi.unige.it.

†INFM - DISI, Università di Genova, granili@infm.ge.it.

‡INFM - DISI, Università di Genova, montobbi@infm.ge.it.

§INFM - DISI, Università di Genova, verri@disi.unige.it.

$(u_1, u_2, \ldots, u_T)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_T)$ consist of the horizontal and vertical positions on the image plane of the box center $\mathbf{c}$, while the vector $\mathbf{a} = (a_1, a_2, \ldots, a_T)$ of the number of pixels N within the box at each frame. In the second step, the vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ are normalized between $[0,1]$ in both range and time and regarded as the sampling of three functions $U, V, A : [0,1] \to [0,1]$. In the third and final step an SVM for regression is run on the data $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{a}$ to build a representation consisting of an array of numbers of fixed size and homogeneous meaning across the events. This is obtained through the use of a kernel K corresponding to a piecewise linear spline with m equally spaced fixed knots in the unit interval (Vapnik 1998) (with $m = 10$ in the present implementation). The detected event is finally described by means of a vector $\mathbf{x}$ the components of which are obtained as follows. The first $m-2$ components are obtained from the regression estimation on the vector $\mathbf{u}$, the second $m-2$ from $\mathbf{v}$, and the third $m-2$ from $\mathbf{a}$. In order to make use of the information contained in the duration of the observed action (effectively lost in the previous normalization stages), the last component is proportional to the elapsed time T of the event.

The last module of the system, the *recognition* module, is only used in the test stage. During the training stage each event $\mathbf{x}$ in the training set is simply given by the supervisor one of the n possible labels $y_1, ..., y_n$ fixed beforehand. The classification of a new event $\mathbf{x}$ is obtained by means of a linear SVM for pattern recognition trained to distinguish between all possible pairs of different event classes. As proposed in Pontil and Verri (1998) the multiclass classification follows the rules of a *tennis tournament*. Each class (or label) is a *player* and at each match the winner is determined by the label temporarily assigned to the unknown event $\mathbf{x}$. The final determines the label assigned to the event.

In its current version, the system is able to detect the visual dynamic events in real time, but both the representation and the recognition stages are still computed offline. Therefore, in order to evaluate the performance of the system we acquired a number of image sequences, computed the representation for all the events detected in real time, and divided the obtained events to form the training and test sets. The system has been trained to recognize different actions taking place down a corridor leading to our Lab. Examples of these actions are *somebody is walking up the corridor and entering the last door on the left*, and *somebody is exiting from room A, walking up the corridor and entering room B*. Since the major feature of the system is its ability to learn the tasks to be performed, we considered two different cases. In both cases the camera was looking down the corridor and the system was trained to recognize eight different actions. The camera orientation and position and the set of actions differ in the two cases and several peoples were involved as actors.

We checked the system performance by means of the *leave-one-out* technique. This means that after acquiring M events, we trained the SVM for classification with $M-1$ examples and use the left out for testing the system for all the M possible combinations. Overall the system classified correctly 99% of the events. In both cases no event was misclassified in training. More offline experiments in which some of the system parameters have been changed (like the dimension of the feature vectors or the number of fixed knots in the spline kernel for regression) gave very similar overall results. This suggest that none of the parameters is critical.

Concluding, we have proposed a system which can be trained to recognize people movements against a stationary background from a sequence of images taken by a fixed camera. The system effectively learns the tasks to be performed from a number of examples by means of a Support Vector Machine for regression (used to produce a stereotyped representation of the detected events) and a Support Vector Machine for classification (used to recognize the event as one of a number of possible actions defined in the training stage). The results obtained so far are very good and indicate that the system can be trained efficiently on different environments.

References

[1] V. Blanz, B. Schölkopf, H. Bulthoff, C. Burges, V.N. Vapnik, T. Vetter, *Comparison of view–based object recognition algorithms using realistic 3D models*, in LNCS **1112**, pp 251–256 (1996).

[2] C. Cortes, V.N. Vapnik, *Support Vector Network*, Machine Learning **20**, pp. 1–25 (1995).

[3] E. Osuna, R. Freund, F. Girosi, *Training Support Vector Machines: an Applications to Face Detection*, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Puertorico, 1997.

[4] M. Pontil, A. Verri, *Support Vector Machines for 3-D Object Recognition*, IEEE Trans. Pattern Anal. Mach. Intell. **20**, pp. 637–646 (1998).

[5] B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, V.N. Vapnik, *Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers*, AI Memo **1599**, CBCL Paper No. 142, Massachusetts Institute of Technology, Cambridge, 1996.

[6] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.