Tracking fast changing non-stationary distributions with a topologically adaptive neural network: Application to video tracking

Georges Drumea, Hervé Frezza-Buet

ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Tracking fast changing non-stationary distributions with a topologically adaptive neural network: Application to video tracking

Georges Adrian Drumea and Hervé Frezza-Buet

Supélec - Information, Multimodality & Signal Team
2, rue Edouard Belin, F-57070 Metz - France

Abstract. In this paper, an original method named GNG-T, extended from GNG-U algorithm by [1] is presented. The method performs continuously vector quantization over a distribution that changes over time. It deals with both sudden changes and continuous ones, and is thus suited for video tracking framework, where continuous tracking is required as well as fast adaptation to incoming and outgoing people. The central mechanism relies on the management of quantization resolution, that cope with stopping condition problems of usual Growing Neural Gas inspired methods. Application to video tracking is briefly presented.

1 Introduction

In the context of video processing, and more precisely concerning the detection of elements of interest, designers often built integrated algorithms by combining a filtering process with a clustering procedure. For example, detecting moving people can be done in such a framework by using motion detection filters on the images, labeling some pixels as belonging to moving parts of the image. This is used for traffic analysis for example [2], where patches of moving salient points are build from the image flow. Background subtraction is also used [3] to detect objects of interest, moving or not, dealing with shadows and reflections. Such preliminary stages are also motivated by content based features of ISO/MPEG-4 [4]. The difficulty is then to relate the detected pixels to the different objects in the scene. This latter stage requires clustering the pixels so that each cluster contains pixels related to each respective object.

Let us consider one image in the video stream, where object related pixels have been identified. They form several clusters, one for each object. The contours of the clusters can be extracted, and they can be reduced to polygons, as shown on fig. 1. Vertexes and edges of the polygons form a graph that is suitable for further interpretation of the scene (number of connected components could be related to the number of objects, the curvatures of the polygons could help to make a difference between a walking human and a walking dog, etc.). The motion detection process that feeds the system isn't described in the paper, and neither is the use of the graph. Concerning both of these points, let us only stress here the hypothesis that obtaining such graphs from a reliable procedure could be useful to bridge the gap between numerical analysis in the one hand, based on filtering results over the image, and symbolic scene interpretation in the other hand, based on graph analysis to extract semantics.

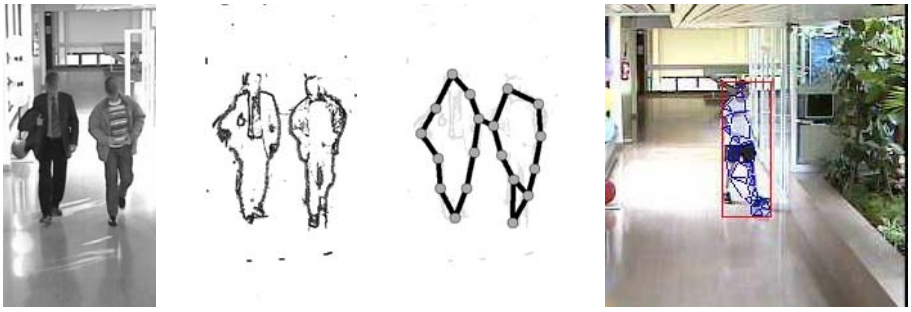

Fig. 1: Schematic description of the extraction of a graph to describe moving objects in a video scene. From left to right: input, motion detection result, expected kind of graph built from moving pixel clustering, result on real experiments with GNG-T.

What is rather presented here is focused on the clustering stage of the process. It has been previously introduced by an example involving a single image, but it has to run on the video stream, from one image to the next. As the stream feeds the algorithm with successive frames, the distribution of detected pixels in each frames changes, and it forms a non-stationary distribution. This non stationary feature has two distinct origins. First, moving people produce continuously sliding blobs of pixels where motion is detected. An update of the graph connected component related to a walking person is expected in that case, rather than a rebuild of the whole graph from scratch. This smooth update is the actual tracking. Second, when people enter or leave the scene, some blobs need to be added or removed. This requires fast adaptation to deal with changes in the number of people in the scene. These latter changes are more sudden ones, but they have to be handled while keeping on tracking smoothly moving people that are still in the scene. Dealing with these two requirements when analyzing a non stationary distribution is the central motivation for the method proposed here, and the scope of that work is more general than actual video tracking. The purpose of our method is to track distributions that have both fast changing and smooth changing components.

2 Clustering with growing networks

In order to set up the basis for the presentation of our algorithm, let us recall some aspects of vector quantization and define some notations. Let $\xi \in X$ a sample of an unknown distribution $\mathcal{P}_X$ over space X, according to a stationary density of probability $p(\xi)$. Vector quantization consists classically in finding a discrete set $\{w_i\}_{1 \leq i \leq n} \subset X$ of *prototypes* such as this set "matches" $\mathcal{P}_X$. Let $w(\xi)$ be $\text{argmin}_{w_i} \{d(\xi, w_i)\}$, where d is some proximity function (usually $d(\xi, w_i) = \|\xi - w_i\|^2$), less restrictive than pure distances, returning low values for similar arguments and higher values for less similar arguments. The

quality of the fitting depends on how well the prototypes $\{w_i\}$ are scattered over the distribution. More formally, this scattering is to minimize distortion $E = \int_X d(w(\xi), \xi)p(\xi)d\xi$. The minimization of E is performed by successive stages, until some stopping condition is met. At each stage, an input ξ is first tossed, according to $\mathcal{P}_X$. Second, a so called winner-take-all procedure (WTA) allows to determine the winning prototype $w_{i_1} = w(\xi)$. Third, w_{i_1} is modified so as to be closer to ξ.

These three stages are common to most vector quantization techniques, and we will only consider here refinements proposed by Fritzke [5]. The idea consists in accumulating error $d(w_{i_1}, \xi)$ at each w_{i_1}, and periodically add new vector w near the w_i that has accumulated most error. Moreover, the w_is are linked together to retrieve the topology of $\mathcal{P}_X$, and modification of w_{i_1} implies also a smaller modification of the w_js connected to w_{i_1}. We refer to [5] for exhaustive description of the Growing Neural Gas algorithm (GNG).

As opposed to Self-Organizing Maps (SOM) by Kohonen [6] and k-means, where decaying learning rates are used, the GNG algorithm is stationary and deals with constant learning rates. One advantage of this feature is that the algorithm can adapt to changes in the distribution $\mathcal{P}_X$. More precisely, the distribution can be non stationary, noted $\mathcal{P}_X(t)$, and the GNG can update the w_i so that they follow $\mathcal{P}_X(t)$, then forgetting ancient obsolete distributions. This can lead to unused w_is that are never updated anymore, and some supplementary mechanism has to be added to avoid this problem. This has been done in the past by Fritzke [7, 1], where the number of w_is is kept constant, and where the w_is are added an utility measure that decays for a prototype w_i if it is left out the distribution due to fast changes. GNG with such a utility-based mechanism is called GNG-U.

A problem with GNG-U and GNG is that the size of the network is limited by a parameter. For some distributions, the number of prototypes cannot be estimated beforehand. Moreover, the appropriate number may change over time. In our video tracking framework, when new moving objects are added in the scene, more prototypes are needed to keep the same accuracy of the vector quantization (i.e. to keep distortion small), and some prototypes have to be removed if some object leaves the scene (to avoid over quantization). The method that is proposed here is driven by keeping constant the accuracy of the vector quantization, leading to the adding or the removing of prototypes to do so when the distribution changes. This constant is *targeted* by the vector quantization, and we call the method GNG with targeting (GNG-T).

3 The GNG-T algorithm

Instead of dealing with decays and error accumulation, GNG-T estimates the expected quantization error from samples. This supposes the use of a time window for sampling, which is the drawback of the methods regards to GNG-U that avoids sampling windows by the actual use of decaying values and accumulators. Moreover, the estimations are made considering the prototypes constant, which

is an approximation since they update continuously when they win the WTA stage. When a prototype w_i wins for an example ξ (i.e $\xi \in V_i$, the Voronoï cell around w_i) the quantization error $d(w_i, \xi)$ is sampled for updating its mean. This estimates quantity $\bar{E}_i$ given by equation 1, sampled only for winning prototypes.

$$\bar{E}_i = \int_{V_i} d(\xi, w_i) p(\xi / \xi \in V_i) d\xi = \left(\int_{V_i} d(\xi, w_i) p(\xi) d\xi \right) / \left(\int_{V_i} p(\xi) d\xi \right) \quad (1)$$

This leads immediately to equation 2, where expression on the right is the actual contribution of the Voronoï cell V_i to the overall distortion.

$$\bar{E}_i \int_{V_i} p(\xi) d\xi = \bar{T}_i = \int_{V_i} d(\xi, w_i) p(\xi) d\xi \quad (2)$$

So, during a sampling window of N successive inputs to the algorithm, if we note $\left\{\xi_j^i\right\}_{1 \leq j \leq n_i}$ the n_i examples for which w_i has won, we can estimate the right hand of equation 2, noted $\bar{T}_i$, has in equation 3.

$$\bar{E}_i \approx \frac{1}{n_i} \sum_{j=1}^{n_i} d(\xi_j^i, w_i) \text{ and } \int_{V_i} p(\xi) d\xi \approx n_i / N \Rightarrow \bar{T}_i \approx \frac{1}{N} \sum_{j=1}^{n_i} d(\xi_j^i, w_i) \quad (3)$$

After stabilization (i.e. distortion is minimal), if number of prototypes is kept constant, a vector quantization process leads to values $\bar{T}_i$ for each w_i that are similar. If values are too small, when compared with the desired T target, some prototype have to be removed, and the one with minimal accumulated error is chosen. On the contrary, if values are bigger than T, more prototypes are needed, and we add them exactly as GNG does.

Last point with our algorithm is the learning rule. As the method is dedicated to non stationary distributions, where noise stands and where new clusters may suddenly appear, it is crucial that these effects do not to destroy current clustering, since we want smoothly changing clusters to be tracked. The usual learning rule updates a prototype w (and its neighbors) from example ξ with $\Delta w = \alpha(\xi - w)$. The problem with this learning rule is that if some new cluster (or noise) appears far from the closest available prototype, Δw is a big change, and current cluster is strongly altered. For this reason, we use $\Delta w = \alpha(\xi - w)/\|\xi - w\|$. The GNG-T algorithm is then the following, where $\star$ denotes steps that are different from original GNG.

`Initialization` Use 2 prototypes w_1 and w_2 initialized according to p. Set sample window size N, target T, learning rates $\alpha_1 > \alpha_2$, and maximum edge age A. $n \leftarrow 0$.

`Step`$\star$ `1` $n \leftarrow n + 1$, reset n_i and E_i to 0 for all prototypes w_i.

`Step 2` Get input ξ according to p, and determine the winning prototype w_{i_1} and the second closest prototype w_{i_2}, among the w_is.

`Step`$\star$ `3` $n_{i_1} \leftarrow n_{i_1} + 1$.

`Step 4` Create (or refresh) an edge between w_{i_1} and w_{i_2} with 0 age. Increment the age of all the edges from w_{i_1}, and remove the ones older than A.

`Step 5` $E_{i_1} \leftarrow E_{i_1} + d(\xi, w_{i_1})$.

`Step⋆ 6` Update weights as follows, index j denoting the neighbors of w_{i_1}.

$$w_{i_1} \leftarrow \alpha_1(\xi - w_{i_1})/\|\xi - w_{i_1}\|,\ \forall j\ w_j \leftarrow \alpha_2(\xi - w_j)/\|\xi - w_j\|$$

`Step⋆ 7` If $n < N$, then go to step 1. Else, sampling is done, and reset n to 0 for next one.

`Step⋆ 8` Compute $E_{\min} = \min_{i:n_i>0}\{E_i/N\}$ and $E_{\max} = \max_{i:n_i>0}\{E_i/N\}$

`Step⋆ 9` If $T \in [E_{\min}, E_{\max}]$, go to step 12. If $T < E_{\min}$, go to step 10. If $T > E_{\max}$, go to step 11.

`Step 10` Determine w_a the prototype w_i with strongest E_i, and find among its neighbors w_b, the prototype w_j with strongest E_j. Remove edge between them, add prototype $w_+ = 0.5 \times (w_a + w_b)$, add a new 0 aged edge between w_a and w_+, and between w_b and w_+. Go to step 12.

`Step⋆ 11` Remove the prototype i for which $E_i = E_{\min}$. Go to step 12.

`Step⋆ 12` Remove nodes that have not won (sampling windows have to be large enough). If some global stopping condition is not fulfilled, go to step 1.

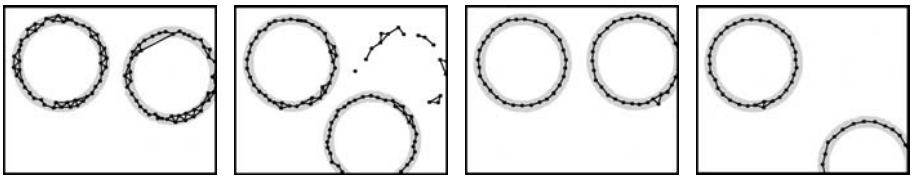

Fig. 2: Fast changing distribution. The two leftmost figures show GNG-U behavior, and the two rightmost ones shows GNG-T.

4 Experimental results and discussion

We have conducted experiments with real and synthetic data. Tests with real data have involved processing a video stream from a video platform, performing basic motion detection and then clustering on the moving pixels. The GNG-T method has been developed out of the necessity to successfully cluster moving objects in video images. Applying the GNG-T method exactly as described in the previous section yields good results with respect to the quality of cluster coverage. We can see on the right of fig.1 the algorithm correctly identifying a moving person. To compare the behavior of GNG-T and GNG-U, we have

also performed a synthetic test (see fig. 2). The test is set up in the following way: we have two ring-like clusters in the scene and we also put in some noise. One of the rings is fixed while the other can be moved freely. We start with the clusters at an initial position. After a very short time (a couple of sampling windows), the algorithm covers the two rings uniformly. We then suddenly move one of the clusters to a different position. The purpose of the test is to see if the algorithm can recover quickly from this fast change. Moreover, after only a few iterations, there aren't any trails left behind with GNG-T. This shows that the algorithm can follow fast moving clusters. We have also performed some more tests using synthetic distributions in which we have tested the ability of GNG-T to follow slowly changing distributions. Since GNG-T inherits its based features from GNG, it has performed very well in these situations.

GNG-T is based on the original GNG algorithm proposed by Fritzke. It offers more adaptability at the cost of having the input data split into sample windows. The windows introduce additional problems because, in order to have accurate results, large windows are required. In a video processing application, an alternative to this problem is to process each frame multiple times to provide enough data for the network to adapt accordingly. Multiple passes over data can be used with other algorithms, but as GNG-T offers the advantage of not having to specify a maximum network size, it may be easier to adjust the target parameter in some applications than fixing the number of prototypes. Last, let us stress here that the size of the network is managed according to statistical measures, which is more adapted to changing distributions that keeping the number of neurons constant. This criterion could also be used is traditional GNG. Ongoing work consists in providing more tests and comparative evaluations, improving motion detection on the video frames, and designing algorithms that exploit the generated graphs for a semantic interpretation of the scene.

References

[1] B. Fritzke. A self-organizing network that can follow non-stationary distributions. In *ICANN'97: International Conference on Artificial Neural Networks*, pages 613–618. Springer, 1997.

[2] C. Kamath, A. Gezahegne, S. Newsam, and G. M. Roberts. Salient points for tracking moving objects in video. In *Proceedings of Image and Video Communications and Processing*, volume 5685, pages 442–453, 2005.

[3] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1337–1342, 2003.

[4] R. Mech and M. Wollborn. A noise robust method for 2d shape estimation of moving objects in video sequences considering a moving camera. *EURASIP, Signal Processing*, 66(2):203–218, 1998.

[5] B. Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[6] T. Kohonen. *Self-Organizing Maps*. Springer, 2001.

[7] B. Fritzke. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.