http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (4): 526-533, 2007
ISSN 1812-5638
© 2007 Asian Network for Scientific Information

Neural Network for Object Tracking

M. Bouzenada, M.C. Batouche and Z. Telli
Department of Computer Science, Lire Laboratory, Vision Group,
University of Constantine, Algeria

Abstract: Real-time object tracking is a problem which involves extraction and processing of critical information from complex and uncertain image data in a very short time. In this study, we present a global-based approach for object tracking in video images. Knowing grey level difference between target and estimated region containing the tracked object, we employ an Artificial Neural Network (ANN) to evaluate the corrective vector which is used to find the actual position of the target. Before, this ANN has been trained, during an offline stage, over a set of output and input samples to determine the relation between the intensity variations and position variations. The evaluation of the corrective vector can be obtained with small online computation and makes real-time implementation on standard workstations possible.

Key words: Object tracking, ANN, Real-time vision, video images

INTRODUCTION

Tracking objects in video images in real-time is a challenging problem. Due to the time constraint (less than 40 ms to process one frame) one has to find a robust and fast tracker. For that, many algorithms have been developed in the last two decades of vision research. These algorithms can be classified in two main groups: Appearance-based and pixel-based.

The methods belonging to the first group represent the tracked object by a model of global and high-level description. This model is defined by one or many features of object properties (Color, texture, shape, motion). The matching, between the model and the information extracted from a frame in video images, has led to two different groups of approaches. Determinist approaches (Comaniciu *et al*., 2003; Collins and Liu, 2005) treat the matching as optimization problem by trying to minimize the error between the model and the estimated information which is extracted from a frame in video images. Whereas, stochastic methods (Perez *et al*., 2002; Isard and Blake, 1998) are based on probabilistic search for the most likely configuration hypothesis with regard to the model. These methods are very effective when target objects are very far, Non-rigid and distinguishable from background.

In the pixel-based group, a spatial representation is used for the tracked object. According to this representation, this group can be divided into two subgroups: feature-based and global-based. Feature-based approaches require a weak runtime and are less sensitive to partial occlusion because they are based on local correspondences (between local features such as: points, line segments, regions, etc.). Whereas, global-based approaches take all pixels which represent the target object. The advantage of these approaches is their capability to handle complex objects which cannot be represented by local features. They are very robust and have been extensively used (Black and Jepson, 1998; Brunelli and Poggio, 1995; Cootes *et al*., 1998; Darrell *et al*., 1996; Gleicher, 1997; La Cascia *et al*., 2000).

In order to decrease runtime, the tracking in these approaches is generally done after an offline stage or learning stage. Hager and Belhumeur (1998) have proposed a framework for this kind of problem. Knowing the position of the target object in the first frame, we can estimate the position of this object in the subsequent frames by comparing the grey level values of the target object with the grey level values of the predicted region. The relation between the intensity variations and position variations is calculated during an offline stage.

This relation has been estimated by using the inverse of the jacobian image (Hager and Belhumeur, 1998) and also by an hyperplane approximation (Jurie and Dhome, 2002; Jurie and Dhome, 2001; Masson *et al*., 2004; Bencheikh El-Hocine *et al*., 2004). The aim of this article is to use an Artificial Neural Network (ANN) to determine this relation (Bouzenada and Batouche, 2005). ANN is non-linear statistical data modeling tools. So, they can be used to model complex relationships between inputs and outputs. The ANN advantage with regard to classic

Corresponding Author: M. Bouzenada, Department of Computer Science, Lire Laboratory, Vision Group,
University of Constantine, Algeria

non-linear modeling techniques lies in their capacity to realize models of equivalent precision with fewer experimental data. Also, ANN allow to change the precision or the runtime by manipulating some of its parameters (Hidden layer number, Goal, etc.).

PRINCIPLE AND NOTATIONS

Let I(p) be grey level value of pixel p at location (x,y) of an image and let the set Rref = (X_1, X_2,...., X_n) be the set of N image locations which define the target region (Fig. 1). Then the set I(Rref) = (I(X_1), I(X_2),...., I(X_n)) represent the vector of brightness values of target region (Jurie and Dhome, 2002; Hager and Belhumeur, 1998).

Let μ be the displacement vector of the region (Jurie and Dhome, 2002; Hager and Belhumeur, 1998). The perturbations (deformations) of the region Rref (Fig. 2) will be obtained by applying N displacements

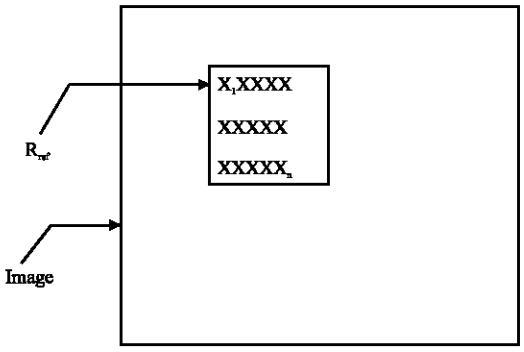


Fig. 1: Reference region including target object in the first frame of the video images

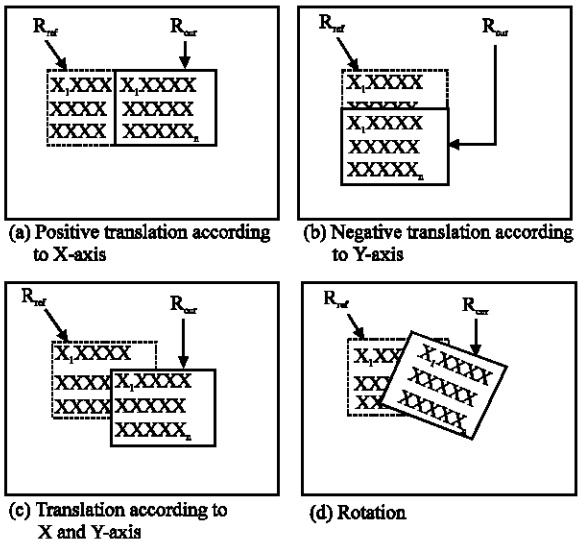


Fig. 2: The perturbations which we have done

(μ_1, μ_2,....., μ_n). For every displacement μi, a difference δIi is done between the vector I(Rref) and the vector I(Rcur) (with Rcur = (X'_1, X'_2,...., X'_n)). I(Rref) represent the grey level vector of the target region and I(Rcur) represent the grey level vector of the current region obtained after a μi displacement.

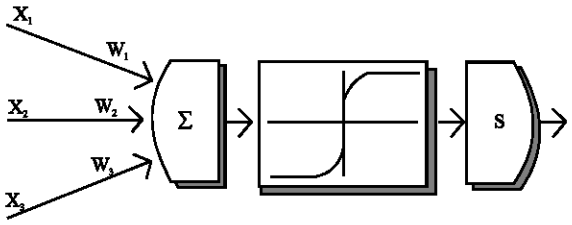
The relation which we want to estimate is between the N displacements (μ_1, μ_2,....., μ_n) and the N differences (δI_1, δI_2,....., δI_n) done after every displacement μ_i.

BRIEF DESCRIPTION OF ANN

An ANN is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on connectionist approach to computation.

Artificial neuron: Called also node, is a basic unit in an ANN. Artificial neurons are simulations of biological neurons and they are typically functions from many dimensions to one dimension. They receive one or more inputs and sum them to produce an output. Usually the sums of each node are weighted and the sum is passed through a non-linear function known as an activation or transfer function (Fig. 3). The canonical form of transfer functions is the sigmoid, but they may also take the form of other non-linear functions, piecewise linear functions or step functions. Generally, transfer functions are monotonically increasing.

Artificial Neural Network (ANN): ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs. Based on the connection pattern (architecture), ANNs can be grouped into two categories (Jain and Mao, 1996) (Fig. 4):



Where:

- (X_1, X_2, X_3) are artificial neuron imputs
- (W_1, W_2, W_3) are weights.
- $\Sigma W_i X_i$ is imputs sum.
- f is activation or transfer fonction.
- $S = f(\Sigma W_i X_i)$ is neuron output

Fig. 3: Example of artificial neuron with three inputs

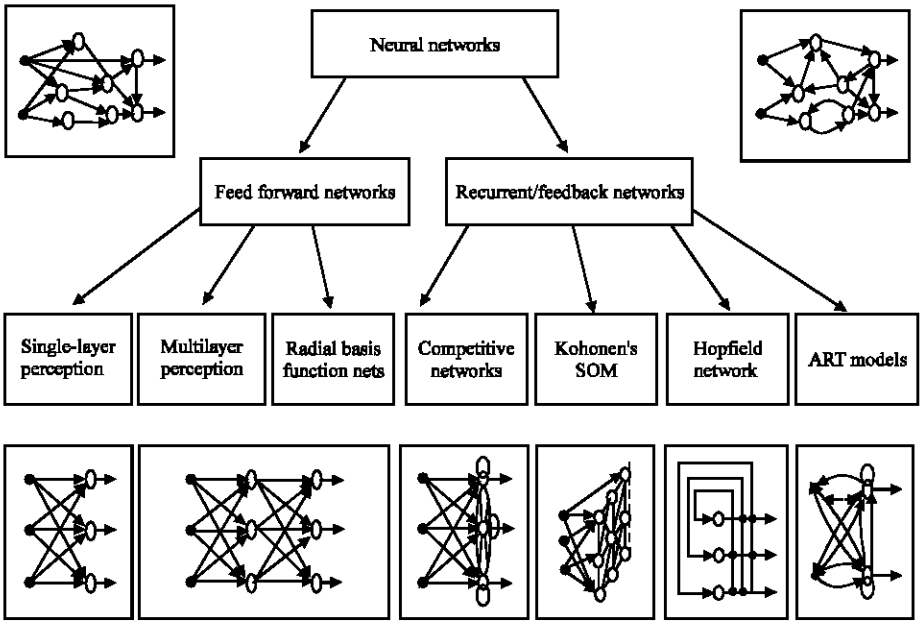


Fig. 4: Taxonomy of feedforward and feedback network architectures

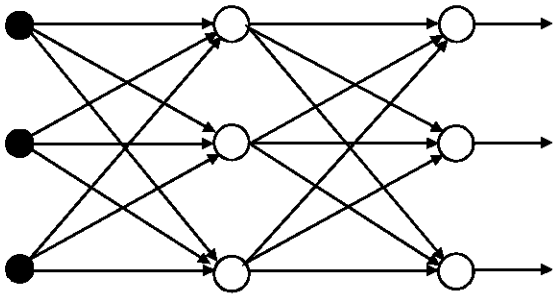


Fig. 5: Multilayer perceptron architecture

- Feed-forward networks, in which graphs have no loops
- Recurrent (or feedback) networks, in which loops occur because of feedback connections.

In the most common family of feed-forward networks, called multilayer perceptron, neurons are organized into layers that have unidirectional connections between them (Fig. 5).

The ability to learn is a fundamental trait of intelligence. Although a precise definition of learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. The network usually must learn the connection weights from available training patterns. Performance is improved over time by iteratively updating the weights in the network.

ANNs' ability to automatically learn from examples makes them attractive and exciting. To understand or design a learning process, you must first have a model of the environment in which a neural network operates, that is, you must know what information is available to the network. Second, you must understand how network weights are updated, that is, which learning rules govern the updating process. A learning algorithm refers to a procedure in which learning rules are used for adjusting the weights.

There are two main learning paradigms: supervised and unsupervised. In supervised learning, or learning with a teacher, the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. Reinforcement learning is a variant of supervised learning in which the network is provided with only a critic on the correctness of network outputs, not the correct answers themselves. In contrast, unsupervised learning, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlations between patterns in the data and organizes patterns into categories from these correlations.

Learning theory must address three fundamental and practical issues associated with learning from samples: capacity, sample complexity and computational complexity. Capacity concerns how many patterns can be

Paradigm	Learning rule	Architecture	Learning algorithm	Task
Supervised	Errot-correction	Single or Multilayer perception	Perception Learning Algorithms Back-propagation, Madaline	Pattern Classification, Functions Approximation, Predicition, Control
	Boltzmann	Recurrent	Boltzmann Learning Algorithm	Pattern Classification
	Hebbian	Multilayer Feed-Forward	Linear Diuscriminan t Analysis	Data analysis, Pattern Classification
	Competitive	Competitive	Learning Vector Quantization	Within-class Categorization Data Compression
		ART	ARTMap	Pattern Classification Within-class Categorization
Unsupervised	Error-correction	Multilayer Feed-Forward	Sammon's Projection	Data analysis
	Hebbian	Feed-Forward or competitive	Principal component analysis	Data analysis Data compression
		Hopfield Network	Associative memory learning	Associative memory
	Competitive	Competitive	Vector Quantization	Categorization Data compression
	Cohonen's sum	Kohonen's SOM	Kohonen's SOM	Categorization Data analysis

Fig. 6: Well-known learning algorithms

stored and what functions and decision boundaries a network can form. Sample complexity determines the number of training patterns needed to train the network to guarantee a valid generalization. Too few patterns may cause over-fitting (wherein the network performs well on the training data set, but poorly on independent test patterns drawn from the same distribution as the training patterns.

Computational complexity refers to the time required for a learning algorithm to estimate a solution from training patterns. Many existing learning algorithms have high computational complexity. Designing efficient algorithms for neural network learning is a very active research topic. There are four basic types of learning rules: error-correction, Boltzmann, Hebbian and competitive learning.

Each learning algorithm is designed for training a specific architecture. Therefore, when we discuss a learning algorithm, a particular network architecture association is implied. Each algorithm can perform only a few tasks well (Jain and Mao, 1996) (Fig. 6).

THE PROPOSED APPROACH

Our approach can be classed in the global-based approaches group because it considers the whole pixels of the target region. To track an object in video images, one has to select the target region (which contains the object) in the first frame. An offline stage is then automatically launched. It consists of training an ANN over a set of inputs and outputs examples. After, the approach iterates over subsequent frames of the video. In each iteration, the trained ANN receives as input the difference between target region and predicted region and gives as a result the corrections to do over predicted target to find the target object (Fig. 7).

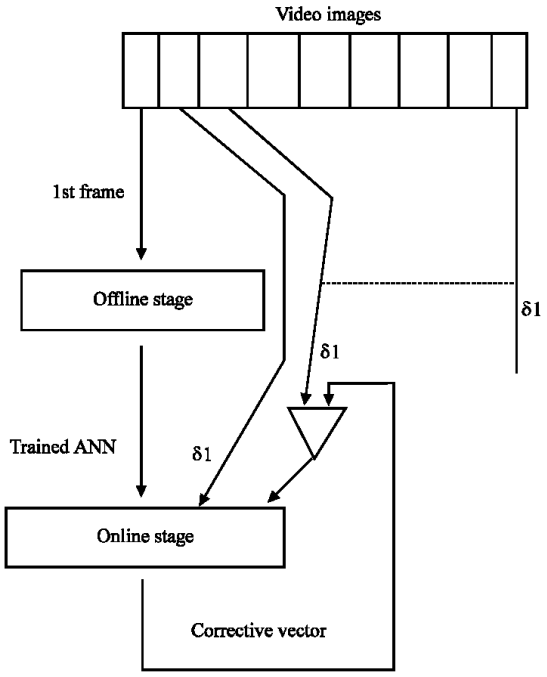


Fig. 7: Global view of proposed approach

Offline stage: The tracked object is selected in rectangular region Rref in the first frame of the video images. Thus, its position is done by [X_{LL}, Y_{LL}, W, H], where (X_{LL}, Y_{LL}) are the coordinates of the lower left corner of Rref and (W, H) its width and height, respectively. Let n be the number of Rref pixels (Rref = (X_1, X_2,...., X_n)) and I(Rref) its grey value vector. Then, to do this we have to Know:

- What information is available to the ANN.
- Which kind of ANN to choose.

Available information: In our case, we know the set of perturbations Y = (μ_1, μ_2,....., μ_n) applied to Rref which will represent the inputs of our ANN. Also, we can have the differences H = (δI_1, δI_2,....., δI_n) which will represent its outputs. Every perturbations μi, is a vector with three elements [dX, dY, θ] which represent, respectively displacement according to X axis, displacement according to Y axis and θ angle of rotational motion. So, the first parameter to deal with, is the type and the number of perturbations to apply to the target region (Rref) chosen in the first frame of a video. Three types of perturbations have been done:

- Horizontal and vertical translations but only about 15% respectively of the width and of the height. In other word, if [X_{LL}, Y_{LL}, W, H] is location of Rref, then [X_{LL}+dx, Y_{LL}, W, H] and [X_{LL}, Y_{LL}+dy, W, H] are respectively horizontal and vertical translations with max(|dx|) = 15%W and max(|dy|) = 15%H. This percentage is sufficient to establish a training set for an ANN because there is a small change between successive frames in video.
- Rotations about (+/-) 35°.
- Combinations of Horizontal translations, vertical translations and rotations.

Also, 1500 perturbations are a good tradeoff between runtime and precision. This number can be increased for more precision or decreased for faster runtime.

Choice and learning of ANN: Knowing the correct δIi difference (input) for every displacement μi (output) that has be done, we are in front of a typical problem of function approximation. Then, supervised learning is the only valid paradigm for this problem. So, we have chosen a multilayer perceptron as architecture, perceptron learning algorithm as learning algorithm and error-correction as learning rule. According to the experimentations made on different video images, we have chosen the following values for this parameters:

- The number of layers in ANN is 2 (1 hidden layer and 1 output layer).
- The number of neurons in each layer is 200 (hidden layer) and 3 (output layer).
- The activation function I (tansig function for neurons in hidden layer and purelin function for neurons in output layer).
- The training function is trainscg function.
- The goal or the allowed error is 10^{-5}.

H and Y must be normalized for more performance with fewer iterations (Fig. 8 and 9). So, the training can start now. It is done as follows:

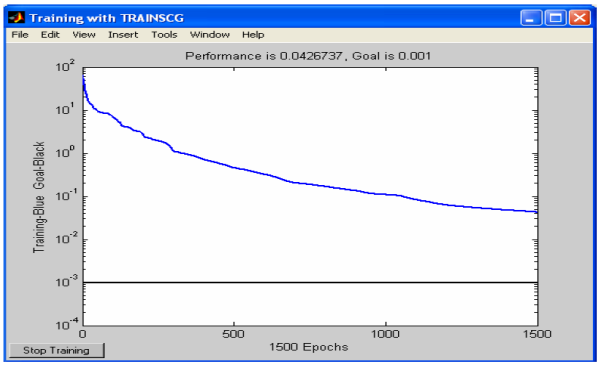


Fig. 8: Performance with unnormalized input

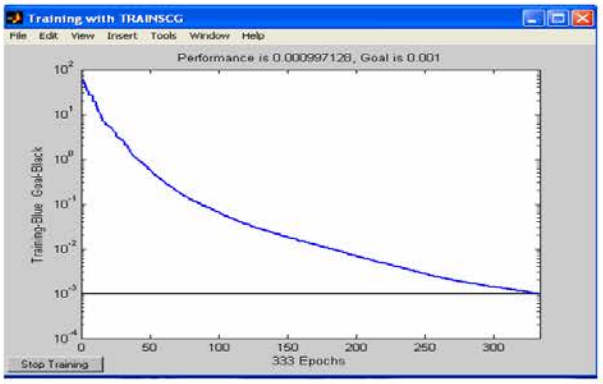


Fig. 9: Performance with normalized input

- Initialize the weights and threshold to small random numbers.
- Present a pattern vector Ht and evaluate the ANN output (D).
- Update the weights by minimizing the difference between evaluated output (D) and desired output (Y).

Online stage: In this stage, the ANN is trained and ready to be used in the tracking of the target object in the subsequent video images. This stage is an iterative process. It is done as follows:

- Given NB the number of frames in video images and I the index of the current frame.
- For I = 2,....., NB do
  - Initialize the predicted position of Rcur region in frame I at the same position of Rref region in frame I-1.
  - Subtract the grey level values of the predicted region and the target region. Thus, $\delta I = I(Rcur)-I(Rref) = (I(X_1), I(X_2),...., I(X_n))t - (I(X'_1), I(X'_2),...., I(X'_n))t$.
  - Present δI as input for the trained ANN. Then, It will give us the corrective vector [dX, dY, θ] as output,
  - Update the position of the predicted region Rcur with the corrective vector.
  - Take this updated Rcur region as the reference region Rref for the next iteration.
- Finally, the position of the target region is determinated in all frames of the video images.

EXPERIMENTAL RESULTS

Our tracker was applied to many video images and it is integrated into several applications. Here we just present some representative results. All sequences have 375 frames of 320x240 pixels. The tracked object is shown into rectangular region surrounded by a blue border.

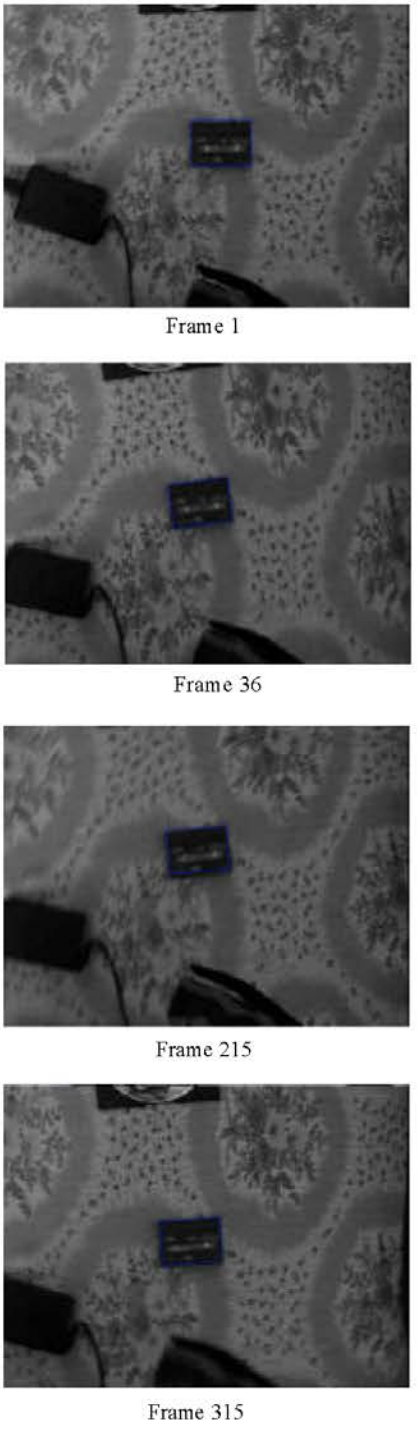


Fig. 10: A fixed object is tracked in this sequence

In the first example, we try to track a fixed object. The ANN is trained with the same values as mentioned above in offline stage at section IV. The target was initialized with a rectangular region (frame 1). The tracker proves to be effective in this case (Fig. 10).

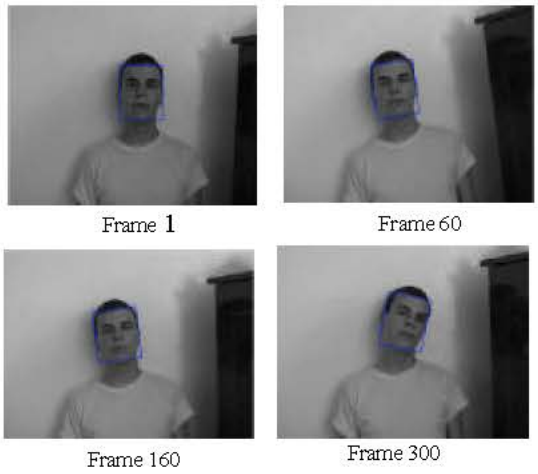


Fig. 11: A human face is tracked in this sequence

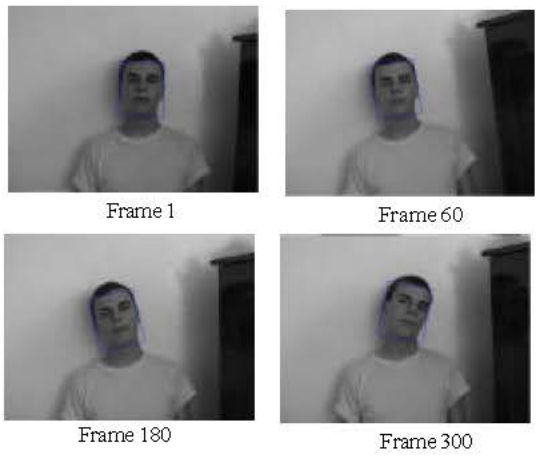


Fig. 12: The same sequence of Fig. 11 is used with modification in some parameter values

In the second example, the object to track is a dynamic human face. The ANN is trained with the same values as mentioned above in offline stage at section IV. The tracker is still effective (Fig. 11).

With the same sequence of the last example, we change some parameters values, when training ANN, as follows: 2000 perturbations, 2 hidden layers, 300 neurons in each hidden layer and 10^{-7} for the goal. The tracker gives the same results as the training with the old values (Fig. 12).

In the last sequence, we try to track the front of a lorry which is a very dynamic object. We use the same parameter values, chosen in the first example, to train ANN. The tracker results are still acceptable (Fig. 13).

Fig. 13: A Dynamic object is tracked in this sequence

CONCLUSIONS

In this research, we have developed an efficient object tracker which is belonging to the global-based approaches. We have used Artificial Neural Network in the offline stage. This ANN allow us to determine the relation between the intensity variations and position variations. Thus, in the online stage, the trained ANN receives as input the difference between target region and predicted region and gives as result the corrections to do over predicted target to find the target object. The result of this second stage can be obtained with small online computation and makes real-time implementations on standard workstations possible.

We are actively continuing to evaluate the performance of our approach and to extend its theoretical underpinnings. We are also looking at the problem of extending the method to support scale variations.

REFERENCES

Bencheikh El-Hocine, M., M. Bouzenada and M.C. Batouche, 2004. A new method of finger tracking applied to the magic board. Proc. IEEE ICIT 04, Tunisia.

Black, M.J. and A.D. Jepson, 1998. Eigen tracking: Robust matching and tracking of articulated objects using a view-based representation. Int. J. Computer Vision, 26: 63-84.

Bouzenada, M. and M.C. Batouche, 2005. A new method of tracking based on Artificial neural network. In: Proc. CIP'2005, CDTA, Algeria.

Brunelli, R. and T. Poggio, 1995. Template Matching: Matched Spatial Filter and Beyond. A.I. Memo 1549, Massachusetts Inst. of Technology.

Collins, R.T. and Y. Liu, 2005. On-line selection of discriminative tracking features. IEEE Trans. PAMI, 27: 1631-1643.

Comaniciu, D., V. Ramesh and O. Meer, 2003. Kernel-based object tracking. IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI), 25: 564-577.

Cootes, T.F., G.J. Edwards and C.J. Taylor, 1998. Active appearance models. Proc. European Conf. Computer Vision, pp: 484-498.

Darrell, T., I.A. Essa and A.P. Pentland, 1996. Task-Specific gesture analysis in real-time using interpolated views. IEEE Trans. PAMI, 18: 1236-1242.

Gleicher, M., 1997. Projective Registration with Difference Decomposition. Proc. CVPR '97, pp: 331-337.

Hager, G.D. and P.N. Belhumeur, 1998. Efficient region tracking with parametric models of geometry and illumination. IEEE Trans. PAMI, 20: 1025-1039.

Isard, M. and A. Blake, 1998. Condensation-conditional density propagation for visual tracking. Int. J. Computer Vision, 29: 5-28.

Jain, A.K. and J. Mao, 1996. Artificial neural networks: A tutorial. IEEE. Computer, 29: 31-44.

Jurie, F. and M. Dhome, 2001. Real time tracking of 3d objects with occultation. In: International Conference on Image Processing, Thessaloniki, Greece, pp: 413-416.

Jurie, F. and M. Dhome, 2002. Hyperplane approximation for template matching. IEEE Trans. PAMI, 24: 996-1000.

La Cascia, M., S. Sclaroff and V. Athitsos, 2000. Fast, reliable head tracking under varying illumination: An approach based on registration of textured-mapped 3D models. IEEE Trans. PAMI, 22: 322-336.

Masson, L., M. Dhome and F. Jurie, 2004. Robust Real time tracking of 3D objects. International Conference on Pattern Recognition, Cambridge, UK., pp: 252-255.

Perez, P., C. Hue, J. Vermaak and M. Gangnet, 2002. Color based probabilistic tracking. ECCV'02, pp: 661-675.