A Multiprocessor-Oriented Visual Tracking System

Stefano Rovetta, *Member, IEEE,* and Rodolfo Zunino, *Member, IEEE*

***Abstract*— The design and prototypal realization of a visual tracking system is presented. The approach to target identification is nonconventional, in that it relies on an architecture composed of multiple standard neural networks (multilayer perceptrons) and exploits the information contained in simple features extracted from images, performing a small number of operations. Therefore, the tracking functions are learned by examples, rather than implemented directly. The system demonstrates that a quite complex task such as visual target tracking can be easily obtained by a suitable neural architecture. The fast tracking algorithm and the parallel structure allow a true real-time operation. The system exploits a two-level neural-network hierarchy with a number of parallel networks and an "arbiter." The training set consists of various geometrical shapes, preprocessed to yield the data vectors. The experimental hardware implementation is based on multiple processing units, implementing the neural architecture, and serves as a prototype for the analysis of the system in practice. A small-sized realization can also be obtained.**

***Index Terms*— Machine vision, multilayer perceptrons, multiprocessing, neural network applications, neurocontrollers, tracking.**

I. Introduction

VISUAL tracking of moving objects is often required in the fields of robotics, industrial automation, and automated vehicle guidance. The problem has been addressed in the past within the disciplines of pattern analysis, artificial intelligence, and neuromimetic systems. Many methods have, therefore, been developed to attain specific goals, such as biological plausibility, generality, and integration with higher level analysis systems [1]–[4]. Unfortunately, these methods are often computationally expensive or complicated, which may be an obstacle to their utilization in real-time applications.

On the other hand, in simplified situations (e.g., a small number of specific sensors instead of whole images) there are many available approaches based on control theory, either classical or using neural networks and fuzzy systems. These methods often allow an in-depth theoretical analysis and are adequate for these situations where a small input pattern, with a well-known behavior for each individual signal, is available [5]–[7].

We address the realization of an in-between solution capable of attaining good real-time performance and, at the same time, of exploiting a neural approach. We present the design and a prototypal hardware realization of a visual target tracking system, based on a neural algorithm and on a parallel processor architecture. Our approach to target identification is substantially simplified by imposing realistic constraints on the problem. This allows the use of simple and inexpensive processors to implement the control system. The resulting tracking system, although appropriate for simple tasks involving few objects, can also be successfully utilized in many situations in which real-time performance is needed.

The tracking mechanism is composed of different modules exploiting bio-inspired concepts to achieve a robust performance, such as example-based learning, parallel and distributed processing, and redundant architectures. The tracking function is not hard-wired into the algorithm, but is learned from a training set. This allows the user to easily tailor the system to the requirements for a specific application. The drawback of the neural approach is that a sound analytical treatment of stability issues is not readily available. Therefore, our approach is to experimentally verify the system's properties first with simulations, and then with actual experiments on the prototype.

The scheme is very straightforward, therefore, it can be applied to real-time operation with a limited effort, since it is based on very simple parallel procedures. This allows the designer to select from among a wide variety of implementations ranging from application-specific integrated circuits (ASIC's) to digital-signal-processor (DSP)-based software simulations. The first hardware implementation, described in this paper, is based on a Motorola workstation based on the real-time industrial standard Versa Module Europa (VME) bus architecture. Other possible realizations are considered in this paper.

The simulation of the tracking system has been introduced in [8] and [9]. In this paper, we present the actual implementation of the system. This also includes training with images acquired with a camera, rather than artificially generated. Part of this material has been presented in [10].

In the remainder of this paper, the neural system is described from the theoretical point of view (Sections II and III). The hardware implementation is then presented (Section IV). Section V contains some results from experimental verifications of the system, and Section VI compares the performance of the method presented with other approaches. Finally, Section VII draws some conclusions and presents future lines of research.

Manuscript received May 4, 1998; revised September 25, 1998. Abstract published on the Internet April 18, 1998. This work was supported in part by the Italian Ministry for the University and Research (MURST 40%).

The authors are with the Department of Biophysical and Electronic Engineering, University of Genoa, 16145 Genoa, Italy (e-mail: rovetta@dibe.unige.it).

Publisher Item Identifier S 0278-0046(99)05625-7.

II. Tracking Problem

A. Statement of the Problem

The system under consideration addresses a generic tracking problem, stated in terms of the following hypotheses. There

0278–0046/99$10.00 © 1999 IEEE

is a single object moving in a plane. The mass of the object is concentrated, but the shape is not required to be strictly compact. The body is not necessarily rigid; the possible changes in the object shape, however, are not too fast. The motion is arbitrary, again with some limitations on its speed, in such a way that the object never escapes the viewing area.

The mass of the object should be concentrated because of the mechanism used for motion estimation (differential messages, to be explained further), which can be confused by odd mass distributions. This is also the reason for requiring slow shape modifications. Finally, the object should stay within the viewing area because the system described here does not include a search mechanism to scan the scene, although it would be possible and easy to add it.

These constraints are not strict. The resulting system features a "graceful degradation" that allows it to work with a very small error when the object and the scene are not exactly as required. For instance, the motion can be three dimensional without affecting the performance at all, provided that its component in the third dimension (depth) is not too fast.

B. Existing Techniques

Apart from the simple tracking techniques based on limited sensors, the available techniques to approach this problem are usually based on *feature extraction* and on *optical flow*.

Feature extraction techniques [1], [2] belong to the class of "image understanding" algorithms, in which the main target is to provide a symbolic interpretation of an image (to recognize a particular object or event in the scene). This class of algorithms can then be adapted to estimate the current object position moving in the scene. Essentially, feature extraction methods consist of the following:

- change detection (where the difference frame background is computed);
- focus of attention, that is, inscription of changed regions into minimum bounding rectangle;
- feature extraction from each rectangle (such as points, lines, edges, etc.).

A subsequent evaluation of these features yields a match of these regions and the subsequent tracking; the segments corresponding to the shape that has to be tracked, which should be known, are identified by comparing the features extracted from different frames and used to estimate the current object position. At a higher level, there are image understanding operations such as data abstraction and pattern recognition for the global interpretation of the observed scene.

This method applies to fixed shapes (rigid-body hypothesis) and requires a large number of computations to find the solution of a set of nonlinear equations (usually obtained with iterative methods).

Optical flow methods [3], [4] are based on the observation of the trajectory of all points of the image; in particular, if $f(x, y, t)$ is a function representing the brightness of the point (x, y) at time t, the algorithm determines the *displacement vector* $(\Delta x, \Delta y)$ indicating the shift of (x, y) between t and $t + \Delta t$. Considering the displacement vector for all points of the image, we can track each object moving in the scene; to do this, we must hypothesize that $f()$ is the same in each image for all objects. Obviously, this is a very restrictive assumption, which could be removed, for instance, by adding a parameterized distortion on $f()$. However, this could be done only at the expense of further computational load.

The vector $(\Delta x, \Delta y)$ is determined by solving

$$\min_{\Delta x,\, \Delta y} |f(x, y, t) - f(x + \Delta x, y + \Delta y, t + \Delta t)|^2. \quad (1)$$

Since in an image there are many points with the same brightness, an optimization of the algorithm searches $(\Delta x, \Delta y)$ in a neighborhood of the point

$$\min_{\Delta x,\, \Delta y} \sum_{x',\, y' \in N_{x,\, y}} |f(x', y', t) - f(x' + \Delta x, y' + \Delta y, t + \Delta t)|^2 \quad (2)$$

where

$$N_{x,\, y} = \{x', y' \colon |x - x'| \leq \epsilon;\ |y - y'| \leq \epsilon\}.$$

(An alternative approach is to take into account contrast or entropy operators rather than pixel values.) This method is usually named sum of square difference (SSD) and is used because it allows the computation of optical flow, that is, the space of displacement vectors

$$\Phi_{\mathrm{O}} = \{(\Delta x, \Delta y), \forall t, \forall x, \forall y\}. \quad (3)$$

By evaluating the last formula, we can then estimate the current position of the object with another computation.

Both methods can be enhanced by addressing also the issue of scale and rotation invariance. This introduces the need for matrix computations, bringing about a further level of complexity. The methods described are fairly complicated, therefore, real-time performance is not attainable in nonsimulated situations in which the object is moving at a very high speed.

III. A Neural Approach

We observe that, to model a *perception* phenomenon, such as object localization and tracking, a faithful *restitution* system is not required. Therefore, the shape, size, mass, or other features of the tracked object should not be fully described, as long as we retain the minimum information needed to implement the required function of tracking. As opposed to the techniques briefly illustrated, and according to this philosophy, the presented method is straightforward and does not even require an accurate model of the problem.

The structure of the whole system is outlined in Fig. 1. The tracking function is learned by a multilayer perceptron that is fed with simple features extracted from the images. The system is represented by a standard feedback control scheme. Its peculiarities are the realizations of the feature extractor (here termed "message generator") and of the tracker.

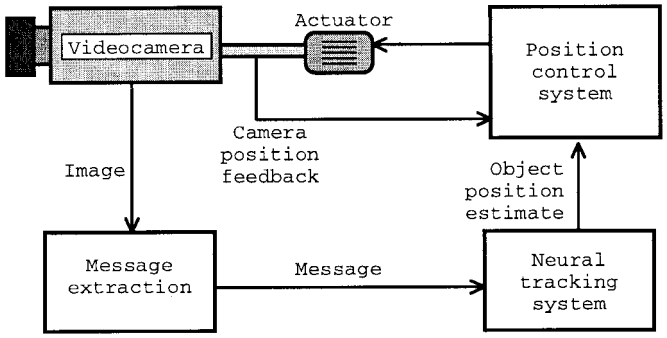

Fig. 1. Block diagram of the tracking system.

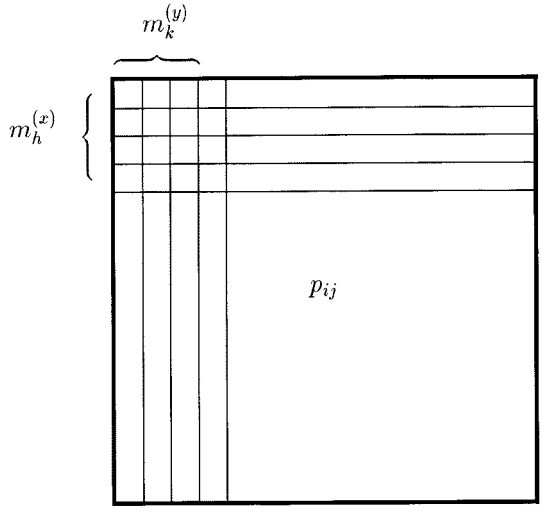

Fig. 2. Message generation.

A. Feature Extraction

Feature extraction is based on simple row-wise and column-wise averages. This solution has been chosen because the selected approach is aimed at the maximum simplicity, but other, more sensitive feature-extraction methods could easily be plugged into the system without structural modifications. The proposed solution imposes a very small computational load, while retaining the required performance level, therefore, it is a good candidate for real-time implementation.

As a remark, we note that other, well-established pattern recognition techniques adopt similar or related methods for feature extraction. Examples are found in the automated document processing context, where run-length coding [11], moments [12], and projection profiles [13] are all based on measuring the "mass" (weighted count of black pixels) along given directions. The widely used Hough transform [14] may be viewed as belonging to this class of preprocessing techniques.

The procedure is sketched in Fig. 2. The image coming from the video camera is an array $\{p_{ij}\}$ of 128×128 greyscale pixels. The resulting square matrix is summed row wise and column wise, and the resulting values are adjoined to form two 128-element vectors, one for the horizontal axis and one for the vertical axis

$$\mu_j^{(x)} = \sum_{i=1}^{128} p_{ij} \quad \mu_i^{(y)} = \sum_{j=1}^{128} p_{ij}. \tag{4}$$

These two vectors are further mapped into two reduced vectors of 32 elements each by summing neighboring components in groups of four

$$m_h^{(x)} = \sum_{i=4h-3}^{4h} \mu_i^{(x)} \quad m_k^{(y)} = \sum_{j=4k-3}^{4k} \mu_j^{(y)}. \tag{5}$$

Therefore, a message contains the spatial distribution of the volume of an object, projected on each of the two coordinate directions. Its graphical representation is a histogram, which does not uniquely determine the source image, but is, however, sufficiently characteristic.

This reduction in dimensionality is appropriate for two reasons. First, it constitutes a subsampling, producing a low-pass filtering effect. The net result of this filtering is to smooth unavoidable noise components and minor variations in the object position, which might be due to small movements of the camera.

Second, it helps keep the data dimensionality low in the subsequent neural-network training phase. Indeed, it is known [15] that the number of data samples needed to train a learning machine, with a given generalization error probability, grows with the number of parameters (in a multilayer perceptron, the number of weights). Therefore, if we increase the number of input units of a network, we should also increase the number of patterns in the training set.

The subsampling ratio of $1:4$, assessed through experiments, is a compromise between a moderate-sized training set and a sufficient sensitivity to the horizontal and vertical components of the displacement of the tracked object.

The motion estimation parameters, too, are very simple. For each coordinate, pairs of messages extracted from successive images (at time steps t and $t+1$) are subtracted, obtaining a "differential message" used to detect motion information:

$$\Delta m_h^{(\alpha)}(t+1) = m_h^{(\alpha)}(t+1) - m_h^{(\alpha)}(t) \tag{6}$$

(for $\alpha \in \{x, y\}$). As a remark, we observe that this motion estimation method is much simpler (and faster) than those found, for instance, in videocompression algorithms [16]. The reason for this is that, to identify the motion vector (a *perception* task), an accurate representation is not needed, as it would in the case of image reconstruction (decompression phase)—a *restitution* task.

B. Neural Tracking

The neural system acts as a position and motion estimator for the target. The actual positional compensation on the camera is performed by a simple proportional integral derivative (PID) algorithm, on the basis of the estimated position. Here, we describe the neural estimator design and structure.

A set of differential messages, taken from a number of sequences, is used as a training set for a multilayer perceptron. The training patterns are, therefore, composed of the messages,

as the input vector, and of the motion vector components, as the target. Since messages are computed separately for the x and y components, respectively, there are correspondingly two networks, one to learn the x component of the motion vector, and the other to learn the y component.

The training procedure is an accelerated version of the error backpropagation algorithm (using an adaptive step size). The training of a multilayer perceptron is long and requires attention to avoid the well-known problems arising from overtraining and imbalanced training sets. These problems need a sufficient number of training patterns to be selected and early-stopping, cross-validation, or regularization techniques to be adopted for training.

On the other hand, the training phase is performed once and for all off-line; in the actual system operation, the only computations needed are those of the forward pass, which is very fast. The number of computations involved in the forward propagation of activations is readily obtained; if we denote the hidden layer dimension by n_h, n_h neuron activations are required for the hidden layer, and 1 for the output layer. Each hidden neuron computes $32 + 1$ multiply-and-add operations (32 inputs and 1 bias term) plus one call to the sigmoid function, while the output neuron computes $n_h + 1$ multiply-and-add operations. If we now assume that $n_h = 20$, an average value in our case, we can estimate the number of operations to be of the order of 700 elementary operations + 20 sigmoid function calls, which is a very small computational load. Achieving real-time operation is, therefore, very simple, since no expensive hardware is needed.

Using a module trained by examples allows an extreme flexibility in the resulting system. If one has *a priori* information on the nature of the problem, its exploitation amounts only to choosing appropriate examples. For instance, if the class of typical object shapes is known, the training set can include a majority of shapes belonging to that class.

C. Combination of Multiple Neural Networks

To increase the robustness of the system behavior and the reliability of the neural stage of the tracker, a team of different networks trained on the same problem is used instead of a single network.

The rationale underlying this procedure is the following. Since the network is trained on a statistical sample, it can be viewed as an estimator of the true motion vector, subject to statistical fluctuations. Such a problem can be stated as follows: "Find the best estimate based on a finite sample of data from a signal with noise added." The error of the estimate, with respect to the sample, can be described by a bias/variance decomposition [17]

$$\epsilon^2 = b^2 + \sigma^2.$$

The bias component b is the part of the error due to an imperfect estimation mechanism, and represents a fixed offset; as such, it can be compensated for. The variance σ^2 is the part of the error due to the noise in the training set, and has to be minimized in some way.

It is known [18] that the variance of an estimator can be reduced by simply taking its average over many realizations. Therefore, the use of many independent networks in parallel helps reduce the variance of the learned mapping. In other words, for each input pattern, the output is obtained by averaging over the outputs of many networks for the same pattern. In [19], this simple technique has been studied and experimentally verified, outperforming other approaches. The problem of training such a neural structure lies often in designing a set of truly independent networks.

In our case, we decided to select different starting points and different numbers of hidden units for different networks, but not to change the training set. This results in a set of networks that are different, but not completely independent (there is a correlation between the outputs of pairs of networks, although they are not exactly equal).

Obtaining different networks from different starting points is a process which requires attention, since the final result of training is related in a complex manner to initial conditions. This is due to the many basins of attractions featured by the network cost-function profile. Therefore, a test should be performed prior to selecting the networks to be combined. Points that belong to the same basin of attraction feature similar cost bias and variance values, typical of the local profile of the cost function. The simplest test consists in verifying these values at the end of each training. If two networks feature different errors, they are almost certainly trained to different final points (provided a reasonably good optimization procedure is applied). Conversely, if they feature similar error mean and variance, they are not necessarily similar; however, more accurate tests should be performed, or another training can be run.

Following this approach, the resulting team of N networks, as expected, does not reduce the variance to σ^2/N. It turns out, however, that the error is reduced by a satisfactory amount. In the experimental section, this is demonstrated in the case of $N = 5$.

Finally, the output of the team of networks is used as an input signal to the control system of the video camera.

Real-time constraints are not included in the tracking system. However, for the time scales of many industrial and automation applications, the response is fast enough to allow real-time operation (as experimentally verified with random motion tracking). A possible stability problem, due to the lack of an explicit output delay compensation, is taken into account by the training procedure, which adopts image sequences. Due to the modularity of the system, an explicit look-ahead function could easily be added; however, our main aim is to keep the system fast and simple. The generality of the tracker would also be reduced if we took into account the specific mechanical properties of the prototype.

IV. Implementation of the System

The prototype of the system has been implemented by using VME boards hosted by a Motorola DELTA workstation, a VME machine. The VME architecture has been designed for the implementation of real-time, parallel systems. As such, it is

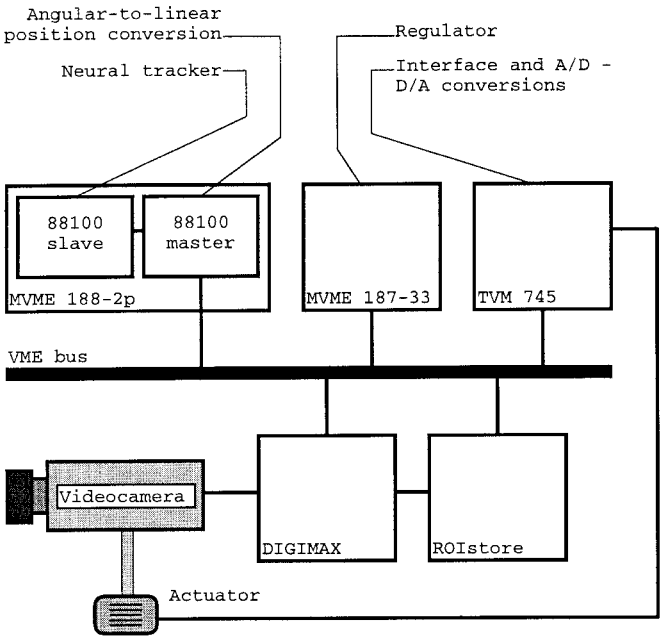

Fig. 3. Schematics of the implementation of VME bus.

typically composed of a Unix-based host and several processor and I/O boards, interfaced to the VME bus. It also includes the software environment, called VMEexec, used to allocate and control processes on the peripheral processor boards.

This realization should be considered as a prototype for the analysis of the system in practice, whereas the final goal is a small-sized realization. This can be obtained with custom VLSI integrated circuits, for a specific application with nonchangeable network functions. Another realization, currently under development, is based on standard, low-cost processor and I/O boards for the ISA or the PCI standard bus architectures. The resulting prototype will be a very flexible and powerful, yet inexpensive tracking system.

The VME host is equipped with a processor of the 68 or 88K family, in our specific case, a Motorola 68030. However, the main processor does not take part in the actual real-time operation, leaving it up to the boards. Fig. 3 shows the functional structure of the implementation.

The acquisition step is implemented with a small video camera connected to a digitizer board (DIGIMAX); the image is then stored on an image buffer board (ROISTORE).

An MVME 188-2P board equipped with 8 + 8-Mb RAM and two 88100 processors (which we denote by 188-1 and 188-2) takes care of simulation tasks. The message extractor and the neural ensemble are implemented on the 188-1 processor. A TVM 745 I/O board provides the desired voltage to the actuator motors through two servo amplifiers, imposing the position and reading the feedback signals provided by differential encoders. These feedback signals are a digital representation of the radial position of the motor with respect to the initial position. The angular position is mapped onto Cartesian coordinates by a converter implemented on the 188-2 processor. The camera position is then controlled by a discrete regulator, implemented in software on an MVME 187-33 board.

The video camera is mounted on the mechanics of a plotter (two degrees of freedom); a metallic frame supports the two axes carrying the camera. The motion takes place in a horizontal plane, about 1 m above the ground. This is a cheap and efficient method to make the camera move in a plane parallel to the motion of the object. However, this implementation could be useful, for instance, if the tracker had to be used to control the manufacturing process on an assembly line.

V. Experimental Results

The behavior of the tracking algorithm and of the complete system were experimentally assessed. Tests were performed on the preprocessing phase, on the basic motion estimation by neural networks, on the performance of the multiple network structure (team), on the stability with a long trajectory, on the ability to track an object with a background, and, finally, on the system as a whole.

There are no widely accepted parameters to assess the behavior of a tracking system. This is the subject of current research. Therefore, in this section, some quantitative parameters will be proposed and evaluated for the tracking experiments.

Let an image sequence $I(t)$ be $T+1$ frames long, from instant $t=0$ (start time) to instant $t=T$ (end time). Let $p(t)=(p^{(x)}(t), p^{(y)}(t))$ be the position of a point of the target object at time step t. We assume that the motion is the same for all such points; where this is not the case, a representative point such as the barycenter will be used. Let $S^{(x)}$ and $S^{(y)}$ be the image horizontal and vertical pixel sizes, respectively. Since in our case they are the same, they will be indicated with $S^{(x)}=S^{(y)}=S$ (and $S=128$). (Of course, if the image vertical and horizontal resolutions are not the same, the aspect ratio can be introduced in the formulas below to compensate for different vertical and horizontal dimensions of pixel.)

Let $\delta(t)$ be the normalized displacement vector of the target at time step t

$$\delta(t)=\left(\frac{p^{(x)}(t)-p^{(x)}(t-1)}{S^{(x)}},\frac{p^{(y)}(t)-p^{(y)}(t-1)}{S^{(y)}}\right)=\frac{1}{S}(p(t)-p(t-1)). \tag{7}$$

The ability to track changes in the position of the target can be measured by the average magnitude of the normalized displacement vector

$$\xi_1=\frac{1}{T}\sum_{t=1}^{T}\sqrt{\delta^{(x)}(t)^2+\delta^{(y)}(t)^2}=E\{|\delta(t)|\}. \tag{8}$$

This is an estimate of the instantaneous ability to keep the subject in the center of the viewing area. A good behavior is indicated, for instance, by $\xi_1<0.2$ (in the ideal case $\xi_1=0$).

To obtain information about the tracking stability, we can evaluate the quantity ξ_2, defined as

$$\xi_2=\frac{1}{T-1}\sum_{t=2}^{T}(|\delta(t)|-|\delta(t-1)|). \tag{9}$$

This is an estimate of the ability to reduce tracking errors over time. A good behavior is indicated by $\xi_2<0$.

A. Position Estimation

The feature extraction step has been validated by a simple set of experiments with static images. A set of geometric shapes at different scale factors has been acquired with the camera, and the resulting messages have been used to train two 32-input networks to estimate the object position relative to the viewing area. Then, a test set has been drawn from the same class of images, but shapes, positions, and scale factors have been varied to obtain a test set different from the training set. With this setup, position estimation error was constantly observed to stay within 10 pixels, but, most often, it was much less. This experiment aimed to establish the correctness of the message generation procedure.

B. Vector Estimation with Single Networks and with a Team

The networks implementing the mapping from differential messages into estimated target position are trained using different shapes at different scale factors, with varying positions. This is a simple approach to achieving a shift-invariant and scale-invariant behavior. As previously remarked, different training seeds are used for different networks, and the output arbiter is an averaging operator taking into account the responses of each network in the team. In the presented experimental results, the team was composed of five networks.

As in the previous experiment, a test set was used to assess the accuracy of the learned mapping. The test set featured geometrical shapes not present in the training set. The scale factors and motion vectors were also different.

The procedure was performed first on synthetic images and then on real images, acquired with the video camera. These images were simple geometrical shapes without background, as in the previous case. Figs. 4 and 5 show instances of the acquired images. Table I presents some test results obtained for a car silhouette. The test patterns were displacements ranging from 1.5 to 4.5 cm, with the origins at different locations in the visual field of the camera. The second and third columns are the average errors and their variances, respectively, for the five individual networks in the team, and for the ensemble.

The parameters are indicated as absolute values. It is possible to observe that the individual network obtaining the best average output error is also the one with the largest output error variance. The output of the ensemble, computed by averaging, obtains an average error that is better than that of almost each network in the team, and has the best (lowest) variance.

C. Tracking a Random Trajectory

A test for stability was performed by tracking a simulated random walk, with varying speed and direction, and with the addition of some scale variations to simulate motion in the third dimension. Motion vectors components ranged from 2 to 30 pixels. The tracked object was a square silhouette. The trajectory was developed in a 1024×1024-pixel squared area, while the viewing area was, as usual, 128×128 pixels.

Stability was satisfactorily verified on quite a long trajectory (500 steps). The evaluation parameters were $\xi_1 = 0.14$ and $\xi_2 = -0.0076$. The stability parameter is negative, as expected.

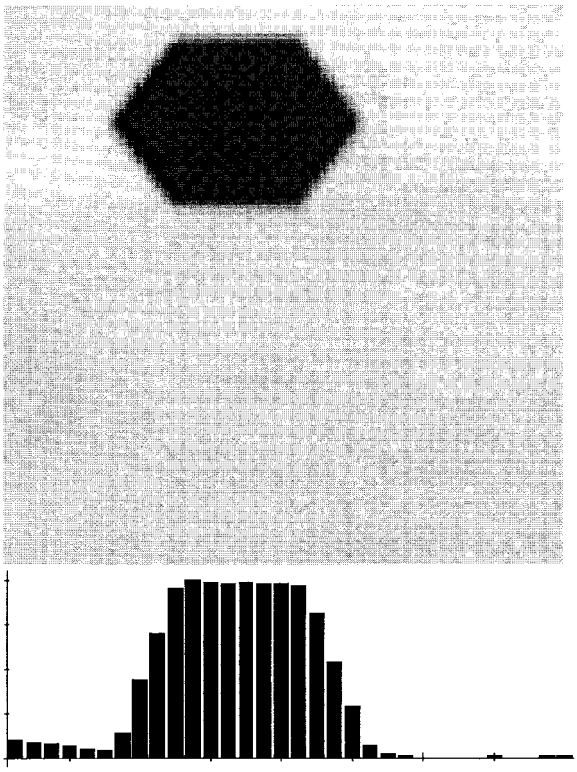

Fig. 4. A sample from the training set and the corresponding message.

D. Object Moving on a Background

Verifications have also been performed in a more complicated situation, in which the car silhouette moved in a background. The tracker was trained on the same geometrical shapes as before, hence, it was not tailored to the specific problem. The car silhouette was animated with a horizontal motion and some random vertical shifts. The background also shifted horizontally. The frame size was 256×256 pixels. The section of the viewing area falling outside the frame was filled with the background color. The tracker was initialized with its viewing area centered on the car. The sequence was 30 frames long (a sample is presented in Fig. 6).

The evaluation parameters were $\xi_1 = 0.05$ and $\xi_2 = -0.0015$. The tracking precision parameter is smaller than in the previous experiment (probably due to the more regular motion). The stability parameter is correctly negative, although smaller in magnitude than in the previous experiment, probably due to the good initial conditions imposed.

E. Verifications on the Physical System

These tests were performed on parts of the system. Similar experiments were also performed on the whole physical system. Small silhouettes were moved under the camera, in varying experimental conditions. The first test was performed using a silhouette taken from the training set; the second, with a new silhouette (not in the training set); the third, with

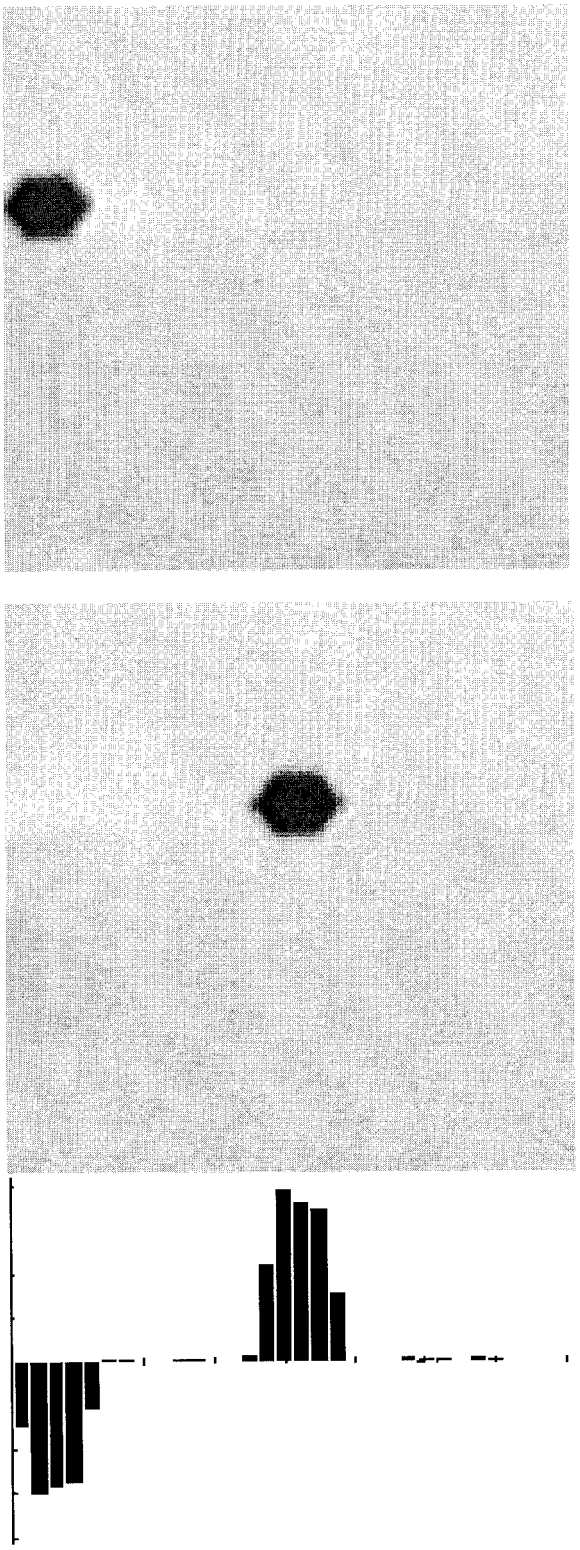

Fig. 5. A sequence of two images and the corresponding differential message.

motion in three dimensions (also along the vertical axis); and the fourth, with three-dimensional motion and a background. The evaluation parameters are displayed in Table II. The point $p(t)$ is here the barycenter of the target in the image $I(t)$

TABLE I
PERFORMANCE RESULTS ON TEST OBJECTS, SINGLE NETWORKS VERSUS TEAM

Network	Absolute average error	Variance
Network 1	0.418	0.793
Network 2	0.223	0.773
Network 3	0.235	1.220
Network 4	0.248	0.690
Network 5	0.044	1.800
Team of 5	0.200	0.630

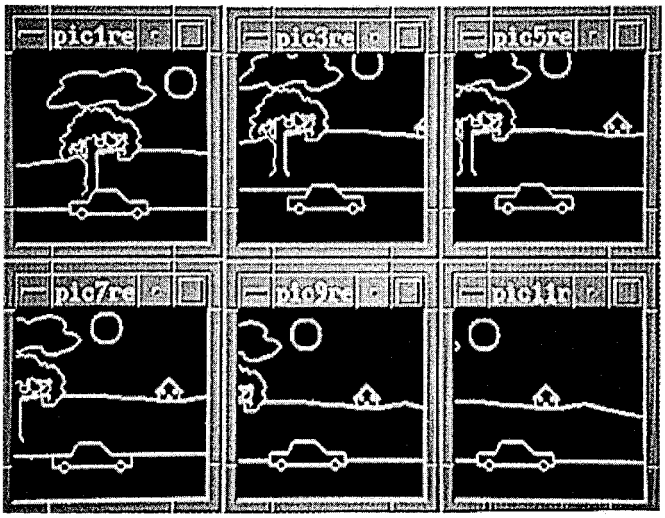

Fig. 6. An object moving on a background.

TABLE II
TESTS ON THE PHYSICAL SYSTEM, PRECISION AND STABILITY PARAMETERS

Experiment	ξ_1	ξ_2
Silhouette from training set	0.049	−0.019
Silhouette not in training set	0.070	−0.020
3D motion	0.181	−0.001
3D motion on background	0.176	−0.002

In all cases, the simulated results were confirmed. This indicates that the system is quite robust and insensitive to environmental conditions; the images acquired with the camera were corrupted by shadows present in the laboratory, the motion was very unpredictable since it was obtained (literally) by hand, and the focus of the camera was not perfect when motion was in the vertical direction. The main limitation on the system is due to the mechanics, since the maximum speed obtainable is limited.

VI. COMPARISON WITH OTHER APPROACHES

Most approaches described in the literature try to formulate a model for scene analysis and motion identification. Therefore (as already mentioned), they generally require many computation steps. In contrast, the proposed method is based on a

low-level mechanism, without explicit formation of a position estimate, but with direct generation of the control signals. This implies lower quality requirements on the image acquired by the camera.

We can note that many tracking algorithms are focused on tracking with a fixed camera position. This requires a wide viewing area, with subsequent redundancy in image data acquired. When the camera position is actively controlled (*active vision* [20]), as in our case, the frames can be much smaller and background data overhead is less. There are approaches integrating wide-area scenes with target tracking with mobile camera [21]. These require, of course, more than one camera and more computing resources.

Several interesting approaches, based on particular features of images containing moving objects, such as motion smear [22], can be computationally heavy when applied to a system with both target and camera in motion (due to the so-called problem of "egomotion estimation").

In [23], computational complexity is reduced by quantizing the possible states of the camera to a finite (and low) number. Target identification and tracking are performed on a limited set of situations, represented in a prerecorded image database. The method proposed here is more general, in that its training procedure allows the selection at the user level the class of situations to be approached. Moreover, the multineural-network structure features a graceful degradation property, enhancing the robustness with respect to unusual situations.

Reference [24] points out the problem of "temporal aliasing," consisting in the ambiguity which arises from the discrete-time image acquisition process (exemplified by the perception of wheels rotating backward in movies). The low-level, distributed structure of the presented approach, along with its inherent simplicity, allows very fast operation. This helps in alleviating the problem, since its Nyquist frequency can be higher than in other, more computation-intensive systems.

To achieve the required simplicity, we have restricted the tracking problem to the case of a target that is always visible. There are methods allowing target ambiguity (multiple hypothesis) or discontinuous trajectory, due, for instance, to target occlusion. The main problem with these more general approaches is its computational complexity, since solution methods such as statistical optimization must be used. For instance, in [25], an algorithm to find the k-best hypotheses in polynomial time is implemented and applied to visual tracking problems. In [26], a robust model including regularization is fitted to the image data to estimate discontinuous optical flow. It can be noted that ambiguities in the trajectory are, in general, greatly reduced when the frame rate is high and the nearest neighbor criterion is applicable (the target in the next frame is the nearest object to the current position). Again, in the proposed system, many of these problems are alleviated simply by increasing the computing speed.

VII. Conclusions and Future Research

In this paper, a neural system for visual target tracking has been presented. The algorithm is based on neural networks working in a team, for better estimation of the target position, and is very simple to implement and to modify for different applications. The hardware requirements for the realization are also relaxed.

The simple structure of the tracking algorithm helps keep the computation time short enough for real-time applications. However, as the experimental results show, the performance of the overall system is satisfactory. This good tradeoff between power and speed is achieved essentially by the neural structure of the tracker. Another desirable side effect of the training by examples is the remarkable flexibility obtained.

The future development of this model will be focused on integration of the control functions on a dedicated IC or on a small single board. Two applications are currently under investigation, namely, control of an atomic force microscope and tracking the plates of cars exiting highway gates. Another promising area of development is short-range on-board driving systems for autonomous vehicles. The integration with a distributed traffic control system [27] could be exploited for complete automation of traffic in a limited environment.

References

[1] D. D. Sworder, P. F. Singer, D. Doria, and R. G. Hutchins, "Image-enhanced estimation methods," *Proc. IEEE,* vol. 81, pp. 797–814, June 1993.

[2] Y. Yasumoto and G. Medioni, "Robust estimation of three-dimensional motion parameters from a sequence of image frames using regularization," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 8, pp. 464–471, July 1986.

[3] B. Ballard and O. Kimbal, "Rigid body motion from depth and optical flow," *Comput. Graph. Image Process.*, vol. 22, no. 1, pp. 95–115, Apr. 1983.

[4] B. Horn and B. Schunk, "Determining optical flow," *Artif. Intell.*, vol. 17, no. 1–3, pp. 185–203, Aug. 1981.

[5] B. B. Lithouki, A. Y. Lee, and D. B. Craig, "Estimator and controller design for LaneTrak, a vision-based automatic vehicle steering system," in *Proc. 32nd IEEE Conf. Decision Control,* San Antonio, TX, 1993, pp. 1868–1873.

[6] K. A. Ünyelioğu, C. Hatipoğlu, and Ü. Özgüner, "Design and stability analysis of a lane following controller," *IEEE Trans. Contr. Syst. Technol.*, vol. 5, pp. 127–134, Jan. 1997.

[7] Y.-C. Chang and B.-S. Chen, "A nonlinear adaptive H^{∞} tracking control design in robotic systems via neural networks," *IEEE Trans. Contr. Syst. Technol.*, vol. 5, pp. 13–29, Jan. 1997.

[8] D. Anguita, G. Parodi, and R. Zunino, "Neural structures for visual motion tracking," *Mach. Vis. Applicat.*, vol. 8, no. 5, pp. 275–288, 1995.

[9] L. D'Agnese, A. Ferro, G. Parodi, and R. Zunino, "Neural architectures for motion tracking," in *Proc. Int. Conf. Artificial Neural Networks (ICANN'93),* 1993, p. 939.

[10] A. Boni, A. Dolce, S. Rovetta, and R. Zunino, "A neural network based visual tracking system," in *Proc. 1996 Int. Workshop Neural Networks for Identification, Control, and Robotics,* Venice, Italy, Aug. 1996, pp. 128–135.

[11] W. Doyle, "Recognition of sloppy hand-printed characters," in *Proc. 1960 Western Joint Computer Conf.*, 1960, pp. 133–142.

[12] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory,* vol. IT-8, pp. 179–187, 1962.

[13] M. Krishnamoorty, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 737–747, July 1993.

[14] S. N. Srihari and V. Govindaraju, "Analysis of textual images using the Hough transform," *Mach. Vis. Applicat.*, vol. 2, no. 3, pp. 141–153, 1989.

[15] V. N. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer-Verlag, 1995.

[16] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards. Algorithms and Architectures.* London, U.K.: Kluwer, 1995.

[17] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–48, Jan. 1992.
[18] M. Perrone, "Improving regression estimates: Averaging methods for variance reduction with extension to general convex measure optimization," Ph.D. dissertation, Phys. Dep., Brown Univ., Providence, RI, 1993.
[19] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 226–239, Mar. 1998.
[20] J. Aloimonos, I. Weiss, and A. Brandyopadhyay, "Active vision," *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 333–356, 1988.
[21] J. Batista, P. Peixoto, and H. Araújo, "Real-time active visual surveillance by integrating peripheral motion detection with foveated tracking," in *Proc. 1998 IEEE Workshop Visual Surveillance*, Jan. 1998, pp. 18–25.
[22] W.-G. Chen, N. Nandhakumar, and W. N. Martin, "Image motion estimation from motion smear—A new computational model," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 412–425, Apr. 1996.
[23] Y. Ye, J. K. Trsotos, K. Benner, and E. Harley, "Tracking a person with pre-recorded image database and a pan, tilt and zoom camera," in *Proc. 1998 IEEE Workshop Visual Surveillance*, Jan. 1998, pp. 10–17.
[24] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Systems and experiment: Performance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 12, no. 1, pp. 43–47, 1994.
[25] I. J. Cox and S. L. Hingorani, "An efficient implementation of Reid's multiple hypothesis tracking algorithm and is evaluation for the purpose of visual tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 138–150, Feb. 1996.
[26] S. Ghosal and P. Vaněk, "A fast scalable algorithm for discontinuous optical flow estimation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, pp. 181–194, Feb. 1996.
[27] G. Vernazza and R. Zunino, "A distributed intelligence methodology for railway traffic control," *IEEE Trans. Veh. Technol.*, vol. 39, pp. 263–270, Aug. 1990.

Stefano Rovetta (M'98) received the Laurea degree in electrical engineering and the Ph.D. degree in models, methods, and tools for electronic and electromagnetic systems from the University of Genoa, Genoa, Italy, in 1993 and 1997, respectively.

He is currently a Postdoctoral Researcher with the Electronic Systems and Networking Group, Department of Biophysical and Electronic Engineering, University of Genoa. He is also an Invited Professor of Electronics at the University of Genoa. His research interests include electronic circuits and systems, and neural network theory, implementation, and applications.

Rodolfo Zunino (S'90–M'90) received the Laurea degree in electronic engineering from the University of Genoa, Genoa, Italy.

From 1986 to 1995, he was a Research Consultant with the Department of Biophysical and Electronic Engineering, University of Genoa, where he is currently an Assistant Professor in Electronics and Industrial Electronics. His main scientific interests include electronic systems for neural networks, distributed control, and methods for data representation and processing.