A Neural Networks Based Visual Tracking System

A. Boni, A. Dolce, S. Rovetta, and R. Zunino

DIBE – University of Genova
Via all'Opera Pia 11a 16145 Genova (Italy)
Phone +39-10-353 2268 – Fax +39-10-353 2175
E-mail Rovetta@dibe.unige.it

Abstract

A visual tracking system based on a neural architecture is presented. The approach to target identification is non-conventional in that it relies on an architecture composed of standard neural networks (multi-layer perceptrons), and exploits the information contained in simple features extracted from the images, using a small number of operations. Therefore the tracking functions are learned by examples, rather than implemented directly.
The training set is composed of various geometrical shapes, in various sizes, with and without a background, pre-processed to yield the data vectors. The system exploits a two-level neural networks hierarchy with a number of parallel networks and an "arbiter."
The selected hardware implementation is based on a cartesian arm and a Motorola VMEexec workstation, that hosts the system but does not take part in the actual computation. This allows a true real-time operation.

1: Introduction

Visual tracking of moving objects is often required in the fields of robotics, industrial automation, automated vehicule guidance. The problem has been succesfully addressed in the past, and many methods have been developed to achieve specific goals such as biological plausibility, generality, integration with higher-level analysis systems [9][11][2][6].

Unfortunately, these methods are often computationally expensive or complicated, which may be an obstacle to their utilization in real-time applications. We present the hardware realization of a visual target tracking system based on a neural architecture [1][4]. To overcome the drawbacks above outlined, our approach to target identification is substantially simplified by imposing 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 be successfully utilized in many situations where real-time performance is needed.

0-8186-7456-3/96 $5.00

The tracking mechanism is composed of different modules exploiting ensemble methods to achieve robust performance. The tracking function is not hard–wired into the algorithm, but is learned on the basis of a training set. This allows the user to easily tailor the system to the requirements of a specific application.

The schema is very straightforward, therefore it can be applied to real–time operation with a limited effort, since it is based on massive parallelism of very simple procedures. This allows the designer to select among a wide variety of implementations ranging from ASICs to DSP–based software simulations. The selected hardware implementation is based on a Motorola workstation with VME bus. 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, exploiting either VLSI integrated circuits or a board equipped with standard components.

In the remainder of the paper, the neural system is described from the theoretical point of view (Section 2). The hardware implementation is then presented (Section 3). Section 4 contains some result from experimental verifications of the system. Finally, Section 5 draws some conclusions and presents future lines of research.

2: The tracking problem and the neural approach

2.1: Statement of the problem

The system under consideration should be able to follow the position of a single object of a compact shape moving in a plane. The object is the only moving part of the scene. The motion is not very fast, but is otherwise arbitrary. However, it may not exceed the limits of the viewing area.

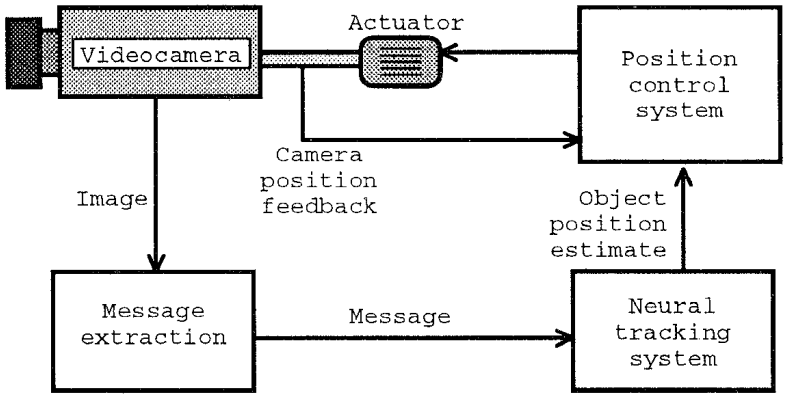
These constraints are not strict. The resulting system features a "graceful degradation" that allows it to work with very small error when the object and the scene are not exactly as required. For instance, the motion can be three–dimensional without affecting the performance at all, provided that its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

2.2: Existing techniques

The available techniques to approach this problem are usually based on *image segmentation* or on *optical flow*. In the first case, features (such as edges) are extracted from the image and used to perform a segmentation. The segment corresponding to the required shape, which should be known, is used to estimate the current object position. This method applies to fixed shapes, and requires a large number of computations to find the solution (usually with iterative methods) of a set of non-linear equations. Real–time performance is not attainable in non–simulated situations.

In the second case, the image is analyzed to extract information about the light levels that is analogous to the flow of a field or of a fluid. The flow of this field is computed by solving the corresponding model equations. This method is more powerful than the first one, but it is very complex both from a formal and from a computational point of view.

Figure 1. Block diagram of the tracking system.



2.3: A neural approach

As opposed to the techniques briefly illustrated, the presented method is straightforward and does not even require an accurate model of the problem. The tracking function is learned by a multi–layer perceptron, that is fed with simple features extracted from the images.

The structure of the whole system is outlined in Fig. 1. The system is configured as a standard feedback control. Its peculiarities are constituted by the realization of the feature extractor (termed "message generator") and of the tracker.

The feature extraction is based on simple row–wise and column–wise averages. This solution has been chosen because the selected approach is aimed at maximum simplicity, but other, more sensitive feature extraction methods could easily be plugged into the system without structural modifications. The image coming from the videocamera is an array $\{p_{ij}\}$ of 128 by 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:

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

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

$$m_h^{(x)} = \sum_{i=4h-3}^{4h} \mu_i^{(x)} \quad ; \quad m_k^{(y)} = \sum_{j=4k-3}^{4k} \mu_j^{(y)} \tag{2}$$

This massive reduction in dimensionality filters out minor variations in the object position, which could be due to small movements of the camera, while retaining sufficient sensitivity to the horizontal and vertical components of the displacement of the tracked object.

For each coordinate, pairs of messages extracted from successive images (at time steps t and $t+1$) are subtracted, obtained a "differential message" used to detect motion information:

$$\Delta m_h^{(\alpha)}(t+1) = m_h^{(\alpha)}(t+1) - m_h^{(\alpha)}(t) \tag{3}$$

(for $\alpha \in \{x, y\}$). As a remark, we may observe that this motion estimation method is much simpler (and faster) than those found for instance in videocompression algorithms [3]. The reason for this is that to track an object does not require an accurate representation as would in the case of image reconstruction (decompression phase).

The differential message is fed into a neural network (a multi–layer perceptron) that has been trained to map its input onto estimates of the object's position. The training of a multilayer perceptron is long and requires attention to avoid the well–known problems arising from overtraining and imbalanced training sets. On the other hand, it is done once off line; after that, the only computations needed are those of the forward pass, which is very fast. Using a module that is trained by examples allows an extreme flexibility in the resulting system. If one has a priori informations on the nature of the problem, its exploitation amounts only to choosing appropriate examples. For instance, if the motion features a preferential direction, the network can be trained by including in the training set more examples of motion in that direction.

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. 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. As such, it can be described by a bias/variance decomposition [5]. It is known [7] that the variance of an estimator can be reduced by simply taking its average over many realizations. Therefore, the use of many networks in parallel helps reducing the variance of the learned mapping. A description of this method can be found for instance in [8].

The output of all the networks in the team is averaged, bias–compensated, and then used as an input signal for the control system of the videocamera.

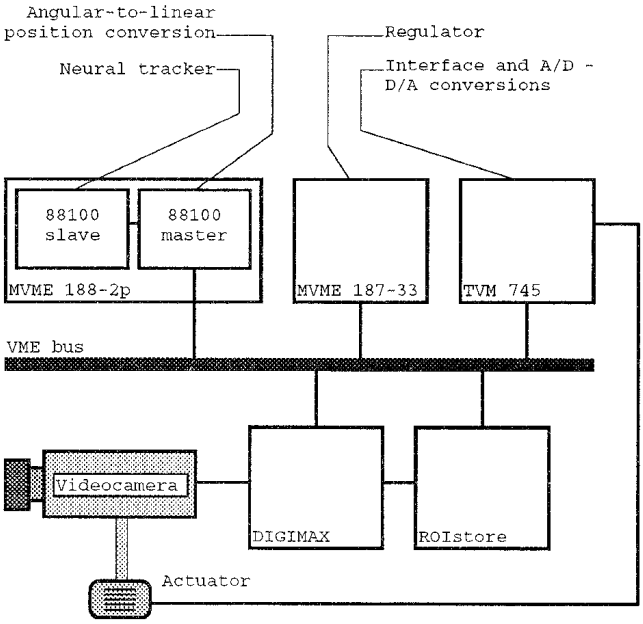
3: Implementation of the system

The prototype of the system has been implemented using VME boards hosted by a Motorola workstation. The host is equipped with a 68030 Motorola processor; however, it does not take part to the actual processing. Figure 2 shows the functional structure of the implementation.

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

A MVME 188–2P board equipped with 8+8 Mb RAM and two 88100 processors (which we indicate with 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 interacts with the actuator motors, imposing the position and reading the feedback signals. These are read as a digital representation of the radial position of the motor. The angular

Figure 2. Schematics of the implementation on VME bus.



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 a MVME 187–33 board.

The videocamera is mounted on the mechanics of a plotter. This choice has been made to allow the camera to be moved, in a cheap and efficient way, in a plane parallel to the motion of the object. This implementation could be useful, for instance, if the tracker had to be used for controlling the manufacturing process on an assembly line.

4: Experimental results

4.1: 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 have 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 disjoint from the training set. With this setup, position estimation errors were constantly observed to stay within 10 pixels, but most often they were much less. This experiment aimed to establish the correctness of the message generation procedure.

Table 1. 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

4.2: Motion vector estimation with single networks and with a team

The networks implementing the mapping from differential messages to estimated target position are trained using different shapes in different scale factors, with varying positions. This is a simple approach to achieve the shift–invariance and scale–invariance of the 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.

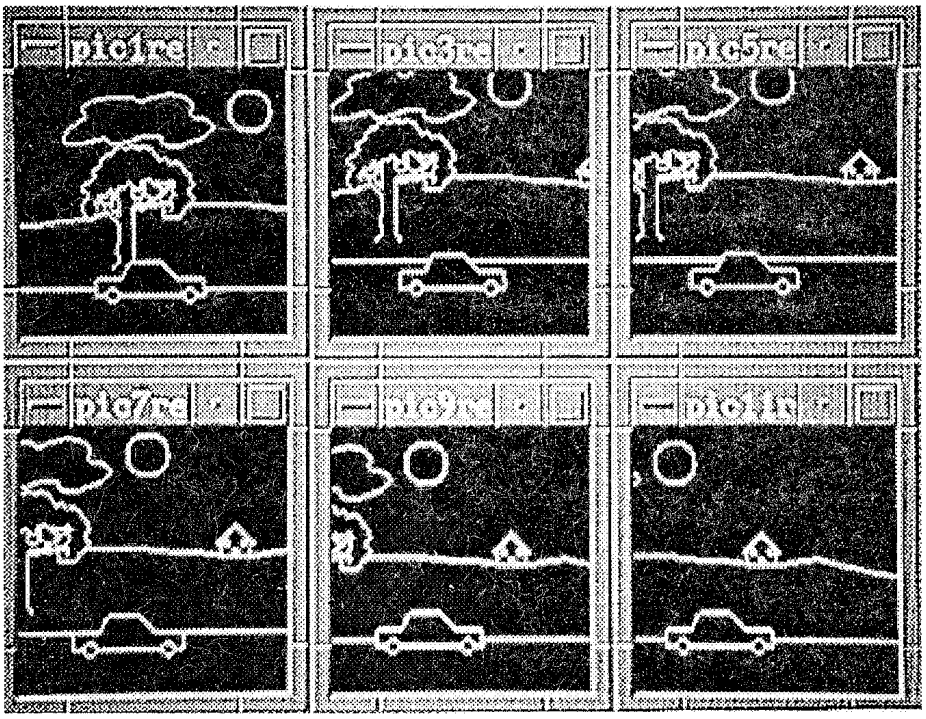
The procedure was performed first on synthetic images, and then on real images, acquired from the videocamera. These images were simple geometrical shapes without background, as in the previous case. Table 1 presents some test results obtained with a car silhouette. The test patterns were displacement ranging from 1.5 to 4.5 centimeters, with origin in different locations in the visual field of the camera. The second and third column are the average error and its variance respectively for the five individual networks in the team, and for the ensemble.

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

4.3: 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. Stability, that we empirically define as the ability to avoid error accumulation, was satisfactorily verified on a quite long trajectory (500 steps).

Figure 3. An object moving on a background.



4.4: Object moving on a background

Verifications have been made also for a more complicated situation, in which the car silhouette was moved on a background. The tracker was trained on the same geometrical shapes as before, hence it is not tailored on the specific problem. The satisfactory results are illustrated in Fig. 3, a sequence of screen shots from the test.

5: Conclusions and future research

The simple structure of the tracking algorithm helps keeping 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 trade–off 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 flexibility obtained.

The main research effort for the future development of this model will be devoted to the engineering of the system, aimed at an eventual integration of the control functions on an ASIC or on a small single board. Another promising area of development is the implementation of a short–range, on–board driving system for autonomous vehicles; the

integration with a distributed traffic control system designed by one of the present authors [10] could be exploited for complete automation of traffic in a limited environment. An appropriate application could be for instance the task of docking/undocking items in a warehouse.

References

[1] D. Anguita, G. Parodi, and R. Zunino. Neural structures for visual motion tracking. *Machine Vision and Applications*, 8:275–288, 1995.

[2] B. Ballard and O. Kimbal. Rigid body motion from depth and optical flow. *Computer Graphics and Image Processing*, 22:95–115, 1983.

[3] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards. Algorithms and Architectures.* Kluwer Academic Publishers, London, 1995.

[4] L. D'Agnese, A. Ferro, G. Parodi, and R. Zunino. Neural architectures for motion tracking. In *Proceedings of the International Conference on Artificial Neural Networks - ICANN93*, page 939, 1993.

[5] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–48, 1992.

[6] B. Horn, B. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[7] M. Perrone. *Improving regression estimates: averaging methods for variance reduction with extension to general convex measure optimization.* PhD thesis, Brown University, Physics Department, 1993.

[8] M.P. Perrone and L.N. Cooper. Learning from what's been learned: supervised learning in multineural network systems. In *Proceedings of the World Congress on Neural Networks III*, pages 354–357, Baltimore, MD, 1993.

[9] D.D. Sworder, P.F. Singer, D. Doria, and R.G. Hutchins. Image–enhanced estimation methods. *Proceedings of the IEEE*, 81:797–814, 1993.

[10] G. Vernazza and R. Zunino. A distributed intelligence methodology for railway traffic control. *IEEE Transactions on Vehicular Technology*, 39(3):263–270, August 1990.

[11] Y. Yasumoto and G. Medioni. Robust estimation of three–dimensional motion parameters from a sequence of image frames using regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:464–471, 1986.