A Multilayer Massively Parallel Architecture for Optical Flow Computation

C. Colombo A. Del Bimbo S. Santini
Dipartimento di Sistemi e Informatica
Via S. Marta, 3
50139 Firenze, ITALIA

Abstract

A two-layers architecture for optical flow computation is here presented, which uses neural nets in the lower layer and a special relaxation system in the upper layer. The high degree of parallelism of the architecture makes it particularly suitable for real-time applications.

1 Introduction

The retrieval of 3-D motion and structure information from an image sequence is typically a two-steps process [1]. The first step is to extract, from the sequence, an estimate of the so-called *motion field*, which is the 2-D velocity field arising from the projection of the real (3-D) motion of the objects onto the image plane. The second step is to infer, from the evaluated two-dimensional field, the three-dimensional structural and/or kinematical properties of the imaged objects. The 2-D velocity field (v_x, v_y) obtained under the assumption of constant image brightness ($dE/dt = 0$) is called *optical flow* [2]; it represents a good estimate of the motion field, under particular conditions of surface reflectance, motion and illumination [3]. The basic equation that relates the variations of brightness - the spatio-temporal gradient (E_x, E_y, E_t) - to the optical flow is:

$$(v_x, v_y) \cdot (E_x, E_y) + E_t = 0. \qquad (1)$$

Unfortunately eq. 1, often referred to as the "optical flow constraint", is not sufficient to determine univocally the optical flow field. Actually, it determines a one-dimensional affine subspace where the solution lies, which is usually referred to as *constraint line*. The uncertainty underlying optical flow computation has a biological counterpart - the so-called "aperture problem" for human vision. That is, as the movement of a straight, long moving edge is perceivable only along the direction perpendicular to the edge itself [4], as the "local" information involved in eq. 1 is sufficient only for the determination of the optical flow component *parallel* to the spatial gradient (see Fig. 1). To recover the optical flow component *perpendicular* to the spatial gradient "multicontraints" strategies can be used, which are based on the solution of an overconstrained problem [5]. The idea is to combine (or "to cluster") the local informations from neighbor points in the image plane, and then to minimize the error on the cluster according to a certain functional. For example, Campani and Verri [6] used a least-squares optimization, while Schunck proposed in [7] a functional based on statistical considerations.

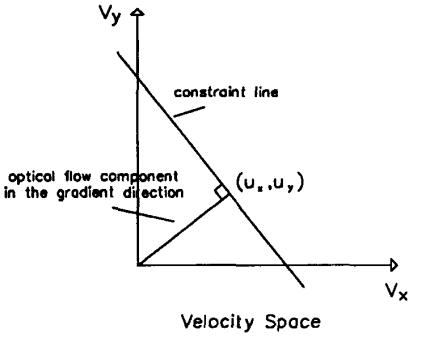

Figure 1: The constraint line and its associated gradient component of optical flow.

A key issue in optical flow computing is the development of algorithms suitable for real-time hardware implementation. Few experiences exist of parallel systems which compute optical flow using the analog CMOS technology [8] [9].

In this work, a two-layers architecture for optical

0-8186-2925-8/92 $3.00 © 1992 IEEE

flow calculation is described, which uses neural nets and a new clustering technique based on relaxation. Both the high degree of parallelism and the asynchronous nature of the architecture make it particularly suitable for analog VLSI implementation.

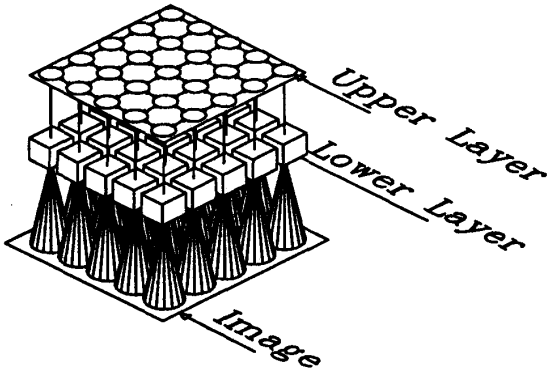

Figure 2: The two-layers architecture of the optical flow estimation system.

2 A Two-Layers Architecture

In Fig. 2 the two layers of the architecture are shown:

- The lower layer, which is completely neural, is made up of a grid of identical and *independent* multilayer perceptrons [10], one for each point where optical flow has to be determined. Each perceptron computes the constraint line which the local solution belongs to. In other words, the lower layer carries out optical flow computations in a "local" fashion, thus yelding, as mentioned above, informations on the optical flow component parallel to the spatial gradient.

- The (missing) optical flow component perpendicular to the spatial gradient is then retrieved, in the upper layer, on the basis of "global" computations, i.e. the diffusion of the local informations throughout the image. For this purpose, the upper layer is made up of a grid of *interconnected* units, identical in number and spatial arrangement to the lower level neural networks. Each upper unit iteratively updates the local optical flow components, always keeping every point in the velocity space on the relative constraint line, as given by the lower layer network. The optical flow is available at the output of this layer after a short relaxation process.

In the following, some details are reported for each layer.

2.1 The Lower Layer

Each neural network in the lower layer computes the optical flow component along the spatial gradient of brightness, that is:

$$\left[(v_x, v_y) \cdot \frac{(E_x, E_y)}{|(E_x, E_y)|}\right] \frac{(E_x, E_y)}{|(E_x, E_y)|}, \qquad (2)$$

which determines univocally the constraint line which the solution belongs to. Each network has a quite small receptive field (a square of 3×3 pixel), according to the observation that the computation of the constraint line is entirely a matter of "local" computation.

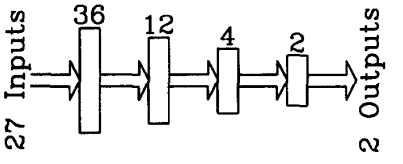

Figure 3: Architecture of one of the networks in the lower layer.

A 5-layers neural network was used, whose architecture is shown in Fig. 3. The input layer is fed by 3 receptive fields taken at consecutive time steps, in such a way that the perceptrons (which perform static, i.e. memoryless, mappings between input and output) exploit the required dynamical properties. The output is an estimate of the optical flow along the gradient direction for the middle time step. Therefore, there are $3 \times 3 \times 3 = 27$ input neurons, and 2 output neurons. The 3 hidden layers are of decreasing size, respectively of 36, 12 and 4 neurons. This sizing was suggested by the natural encoding properties of neural networks [11].

In a separate learning stage, a single prototype network has been trained by showing it moving synthetically-generated edges of different speed, inclination (with respect to the sides of the receptive field) and sharpness. An "edge" is here defined as a surface

whose transversal profile is a sigmoidal function σ of the kind:

$$\sigma(x) = E_{max} \cdot \frac{1 + \tanh(\mu x/2)}{2}, \qquad (3)$$

where the parameter μ accounts for the sharpness of the edge. After training, the prototype network has been "cloned" in several exemplars, to form the lower layer grid.

The advantage of having a small receptive field is that, in such a small area, the network is not likely to "see" complex features. A learning stage with simple features only (namely segments) has proved to be effective to make the networks to operate well under a wide span of conditions. The problem of such an approach is that a small receptive field is affected by uncertainty, as discussed referring to eq. 1. The solution of this problem is demanded to the upper layer.

2.2 The Upper Layer

The upper layer is organized so as to perform a relaxation process on the data from the lower layer. This yields to the formation of clusters in the velocity space, each one with a high probability of belonging to the same moving object. For each neural network in the lower layer, a unit in the upper layer is defined. The interconnections relative to a single upper layer unit are shown in (Fig. 4). Each unit receives as inputs the outputs of the corresponding lower network, (u_x, u_y), which represent the initial conditions for the relaxation process. Each unit delivers two outputs, (v_x, v_y), corresponding to the optical flow vector currently computed, which is available for the "outside world" only at the end of the relaxation process. Moreover, each unit has several other inputs, (v_x^i, v_y^i), which are the outputs of other upper units in a specified neighborhood. For instance, in Figg. 2 and 4 there are four of such inputs, connected so as to form a Von Neumann neighborhood; note from Fig. 2 that the neighborhoods intersect one another, thus allowing in principle an output speed relative to a certain image point to propagate its influence over all the other image points. In such sense, the relaxation process acts also as an "information diffusion process".

Consider now the details of the diffusion process. First of all, remember that the inputs from the lower layer define the constraint line at a given image point (x, y) . During the diffusion process, the corresponding unit is allowed to update its outputs, with the condition that the point they represent in the velocity space always remain onto the constraint line assigned

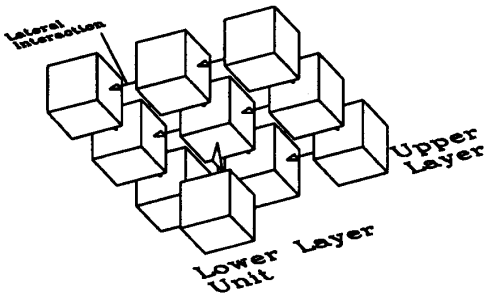

Figure 4: Interconnections between one of the upper layer units, the units in its neighborhood and the corresponding network in the lower layer (top view).

to that unit. This can be easily accomplished by defining the constraint line in a parametric fashion, namely:

$$(v_x, v_y) = (u_x, u_y) + p(\sin\theta, -\cos\theta) \qquad (4)$$

with $\mathrm{tg}\theta = (u_y/u_x)$. In so doing, the outputs (v_x, v_y) depend on the p parameter only, and the relative point in the velocity space is always ensured to belong to the proper constraint line. The system evolves by progressively adjusting the p parameter at each image point - that is $p := p + \Delta p$, starting with $p = 0$ - until a stable value, which corresponds to the formation of clusters in the velocity space, is reached.

In Schunck [7], the cluster formation was reached by finding the set $\{(w_x^i, w_y^i)\}$ of interceptions of the constraint lines of all the points in the neighborhood with the constraint line relative to the image point of optical flow evaluation, and then performing a cluster analysis on that set. In our approach, computing is performed in a highly distributed fashion. In fact, although not explicitly computed, every intersection point (w_x^i, w_y^i) in the velocity space acts as an "attraction point" for the current upper unit outputs; this is done by simply updating the p parameter of each unit based on the value of the inputs from its neighborhood. The updating rule is:

$$\Delta p = \sum_i \Phi(d^i) \qquad (5)$$

where $\Phi()$ is a bell-shaped function of the Euclidean distance

$$d^i = |(v_x^i, v_y^i) - (v_x, v_y)|$$

between the current upper unit output vector and that of any of its neighbors [12]. The global change of the

output is given by the superposition of the "elementary interactions" over the neighborhood. The special non linear shape of $\Phi()$ provides a form of *control* over the diffusion process, in that it assigns a strength to each interaction between points in the velocity space. As an effect, the reduction of the influence of neighbors whose output points in the velocity space are "too near" or "too far" from the current unit output points is achieved, thus avoiding uncorrect flow estimates which could occur, in these cases, in the presence of strong interactions between neighbors [7].

3 Experimental results

The system was tested both with synthetical and real-world 128×128 images, for which a 32×32 optical flow field was computed.

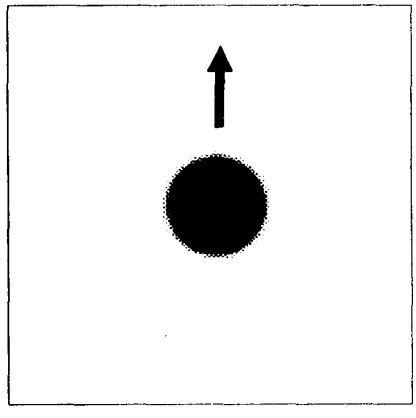

Figure 5: One of the images of the "moving circle" sequence.

Fig. 5 shows a synthetic dark circle traslating upwards on a light background. Note that the brightness changes in the image are localized in a small area between the figure and the background; therefore, according to equation 1, the optical flow is expected to be significant only in this area, and zero elsewhere. Note also that in this case the "true" optical flow is a vector field in which all vectors are identical and aligned in the same direction, since the square is translating by rigid motion. Fig. 6 shows the optical flow as computed by the lower layer. In Fig. 7 the final flow as produced after the relaxation process in the upper layer is shown (16 relaxation steps have been performed): note that the computed optical flow is quite correct, despite the low number of iterations in the diffusion process.

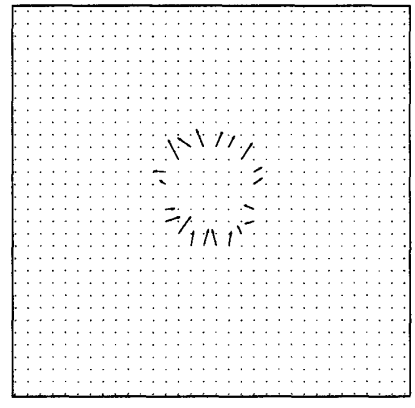

Figure 6: The optical flow in the gradient direction, as computed by the lower layer.

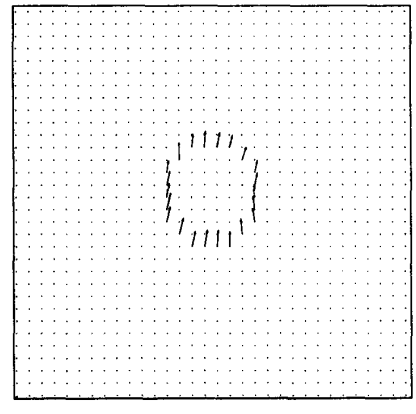

Figure 7: Optical flow, as computed by the upper layer of the system.

The clustering properties of the relaxation process are evident from Figg. 8 and 9 which show the optical flow in the velocity space - where each speed is represented by a small circle - at the input and output of the upper layer, respectively.

Experiments with 5×5 pixel square neighborhoods in the upper layer, in the place of Von Neumann neighborhoods, demonstrated that an enhancement in the quality of the computed optical flow - due to the increase of information available at each image point - is

achievable at the expense of an increment in computation time. Further experiments with real sequences have also proven that a high acquisition noise immunity is obtainable with this system, mainly due to the intrinsic selection of reliable velocity vectors which takes place at the upper layer level. Even better results in the formation of clusters in the velocity space have been achieved by performing a smoothing of the evaluated optical flow with a regularization procedure which can preserve optical flow discontinuities. Further details can be found in [12].

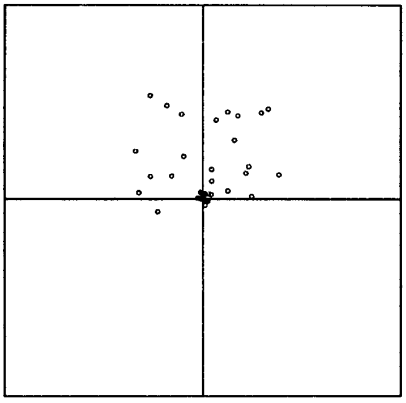

Figure 8: The optical flow computed by the lower layer (velocity space).

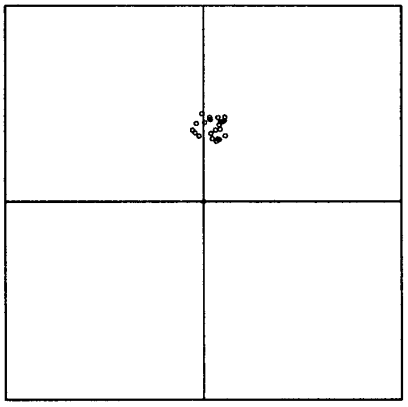

Figure 9: The optical flow computed by the upper layer (velocity space).

References

[1] G. Adiv, "Determining Three-Dimensional Motion and Structure from Optical Flow generated by Several Moving Objects". *IEEE trans. PAMI*, **7**, pp. 384-401, 1985.

[2] B.K.P. Horn, B. G. Schunck, "Determining Optical Flow". *Artificial Intelligence*, **17**, pp. 185-203, 1981.

[3] A. Verri, T. Poggio, "Motion Field and Optical Flow: Qualitative Properties". *IEEE trans. PAMI*, **11**, pp. 490-498, 1989.

[4] D. Marr, *Vision*, Freeman and Co., 1982.

[5] J.K. Aggarwal, N. Nandhakumar, "On the Computation of Motion from Sequences of Images - A review". *IEEE Proc.*, **76**, pp. 917-935, 1988.

[6] M. Campani, A. Verri, "Computing optical flow from an overconstrained system of linear algebraic equations". In *Proc. of 3rd IEEE ICCV '90*, Osaka pp. 22-26, 1990.

[7] B. G. Schunck, "Image flow Segmentation and Estimation by Constraint Line Clustering". *IEEE trans. PAMI*, **11**, pp. 1010-1027, 1989.

[8] J. Hutchinson, C. Koch, J. Luo, C. Mead, "Computing Motion Using Analog and Binary Resistive Networks". *Computer*, **21**, pp. 52-63, 1988.

[9] J. Tanner, C. Mead, "Optical Motion Sensor". In *Analog VLSI and Neural Systems*, pp. 229-255, 1989.

[10] R.P. Lippmann, "An Introduction to Computing with Neural Nets". *IEEE ASSP Magazine*, pp. 4-22, 1987.

[11] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing*, vol. 1, The MIT Press, 1986.

[12] C. Colombo, "Un'architettura ad alto parallelismo per la stima del flusso ottico in una sequenza di immagini". Tesi di Laurea, Fac. di Ingegneria, Univ. di Firenze, 1992.