Proc. Image Understanding, L. Baumann, ed., Morgan Kaufmann Publishers, Inc. 915-920, 1987

# PARALLEL OPTICAL FLOW COMPUTATION

James Little, Heinrich Bulthoff and Tomaso Poggio

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Abstract

*We outline a new, parallel and fast algorithm for computing the optical flow. The algorithm is suggested by a regularization method that we call "constraint method". The method, based on a theorem of Tikhonov, enforces local constraints and leads to efficient, parallel algorithms. The specific constraint exploited by our algorithm can be shown to correspond, in its most general form, to 3-D rigid motion of planar surfaces. An initial segmentation of the motion field can be obtained from the optical flow field generated by the algorithm. We also suggest an iterative scheme that can provide fast, approximate solutions and refines them subsequently. We discuss the implementation of the algorithm on the Connection Machine$^{TM}$ system and its near real-time performance on synthetic and natural images.*

## 1. Introduction

The computation of motion is an important module of early vision. It is potentially useful for computing the 3-D structure of surfaces, for segmenting a scene into objects and as a control module for navigation. The 3-D motion field, however, or even its 2-D projection, cannot be directly measured from the time sequence of images. Much effort in recent years has been invested in analyzing ways to compute the optical flow – the 2-D field of motion of brightness in the image. A common assumption is that the optical flow, suitably defined, is a very close approximation to the 2-D projection of the true 3-D motion field. Verri and Poggio (these Proceedings) argue against this assumption and the related use of the optical flow to obtain a precise quantitative estimate of 3-D structure and motion under general conditions. They argue also, however, that qualitative properties of the optical flow are very useful for segmenting the scene into different objects and for providing information about the *type* of motion and the 3-D structure, and that they are more robust than quantitative estimates. Notice that a closed-loop control system does not need accurate estimates. Furthermore, Verri's analysis suggests that the precise definition of optical flow is not critical as long as it preserves the qualitative properties of the 2-D motion field.

In this paper we outline a new, robust and fast algorithm for computing a version of the optical flow that was suggested by a regularization method that we call "constraint method". The algorithm has been implemented on the Connection Machine$^{TM}$ system. Comparatively little work in the area of motion computation has made use of sequences of real images, because of technical limitations in acquiring and processing them. Our algorithm, which runs in times that are typically of the order of a few seconds, is routinely used with image sequences grabbed by the Vision machine's eye-head system (Poggio and staff, these Proceedings).

## 2. The algorithm

We can formulate a first algorithm in terms of the simplest possible assumption, that the optical flow is locally uniform. This assumption is strictly only true for translational motion of 3–D planar surface patches parallel to the image plane. It is a restrictive assumption that, however, may be a satisfactory *local* approximation in many cases. Let $E_t(x,y)$ and $E_{t+\Delta t}(x,y)$ represent transformations of two discrete images separated by time interval $\Delta t$, such as filtered images or a map of the intensity changes in the two images (more generally, they can be maps containing a feature vector at each location $x,y$ in the image). We look for a motion displacement (also discrete) $\underline{v} = (v_x, v_y)$ at each discrete location $x, y$ such that

$$||E_t(x,y) - E_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t)||_{\text{patch}_i} = \min \quad (1)$$

where the norm is over a local neighborhood centered at each location $x,y$ and $\underline{v}(x,y)$ is assumed constant in the neighborhood. Equation (1) implies that we should look at each $x, y$ for $\underline{v} = (v_x, v_y)$ such that

$$\int_{\text{Patch}_i} |\tilde{E}_t(x,y)\tilde{E}_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t)|dxdy \quad (2)$$

is maximized. Equation (2) represents the correlation between a patch in the first image centered around the location $x, y$ and a patch in the second image centered around the location $x + v_x\Delta t, y + v_y\Delta t$.

This algorithm can be translated easily into the following description. Consider a network of processors representing the result of the integrand in equation (2). Assume for simplicity that this result is either 0 or 1. (This is the case if $E_t$ and $E_{t+\Delta t}$ are binary feature maps.). The processors hold the result of multiplying (or logically "anding") the right and left image map for different values of $x, y$ and $v_x, v_y$. The next stage, corresponding exactly to the integral operation over the patch, is for each processor to count how many processors are active in an $x, y$ neighborhood at the same disparity. Each processor thus collects a vote indicating support that a patch of surface exists at that displacement. The last stage is to choose $\underline{v}(x,y)$ out of a finite set of allowed values that maximizes the integral. This is done by an operation of "non-maximum suppression" across velocities out of the finite allowed set: at the given $x, y$, the processor is found that has the maximum vote. The corresponding $\underline{v}(x,y)$ is the velocity of the surface patch found by the algorithm. This algorithm is similar to the stereo algorithm implemented by M. Drumheller and T. Poggio (1986) on the Connection Machine$^{TM}$ system.

The algorithm will not find, in general, the 2-D projection of the true 3-D velocity field. This will happen only when the features used for matching correspond to markings on the 3-D surfaces and when either the features are sparse (no ambiguity) or the disambiguation step (the voting and non-maximum suppression stage) finds the true correspondence (i.e. the underlying assumptions are satisfied). Even when the result is not the true motion field, the algorithm will usually preserve its most important qualitative properties.

## 3. The constraint method

The algorithm just described was suggested by a regularization method (Poggio and Verri, in preparation) that follows directly from some results of Tikhonov and the discrete nature of the image data. We outline here the basis of the constraint method and then discuss how it is implemented by our motion algorithm. We postpone a more formal discussion to a later paper.

### 3.1. Tikhonov Lemma

As pointed out by Poggio and Torre (1984) many problems of early vision are ill-posed. For example solutions to problems such as surface reconstruction, computation of visual motion and depth from stereo are not unique or they are not stable (for instance for edge detection and structure from motion). A lemma by Tikhonov and Arsenin (1977) suggests how to regularize these problems by exploiting the discrete nature of the image data and the boundedness of sought solutions (for instance depth of surfaces or velocity values).

Let us consider the problem of solving the equation

$$A\underline{x} = \underline{y} \quad (3)$$

for $\underline{x} \in X$, $X$ a metric space. Let $\underline{y} \in Y$, $Y$ a metric space, and let $A$ be an operator mapping $D(A) \subseteq X$ onto $R(A) \subseteq Y$. In many applications it is required that the solution $\underline{x}$ to (2.1) $i$) exists, $ii$) it is unique and $iii$) it depends continuously on $\underline{y}$. A problem whose solutions satisfies $i$), $ii$) and $iii$) is said to be *well-posed*; otherwise it is said to be *ill-posed*. It is clear that the solution does not exist if $\underline{y} \notin R(A)$ and that it is not unique if $A$ is not injective. The solution depends continuously on $\underline{y}$ when the inverse of $A$, $A^{-1}$, is continuous. Let us assume that the solution to (2.1) is given by

$$\underline{x}_0 = \inf_{x \in X} \|A\underline{x} - \underline{y}\| \quad (4)$$

Sufficient conditions for the well-posedness of (3) (or (4)) are provided by the following lemma by Tikhonov:

*Lemma* Suppose that the operator $A$ maps a compact set $F \subseteq X$ onto the set $U \subseteq Y$. If $A : F \to U$ is continuous and one-to-one, then the inverse mapping $A|_U^{-1}$ is also continuous.

The uniqueness is obviously guaranteed by the fact that $A$ is one-to-one while both the existence and the stability of the solution rely upon the compactness of the set $F$. Let us assume that $X$ (or $F$) is a closed and bounded subset of $R^n$. Therefore $X$ (or $F$) is compact. Hence, if the solution belongs to a closed and bounded set in $R^n$ and $A$ is one-to-one, the problem to solve is well-posed. Since every early vision problem is defined on a $n$-dimensional space (where $n$ is the number of pixels), sufficient conditions for the well-posedness of these problems can be given setting *a priori* bounds on the solutions. In the specific case of our motion algorithm the compactness assumption is satisfied by restricting the input and output spaces to be discrete and finite (only a small set of discrete velocities is allowed).

It is important to stress that in several early vision problems the solution is not unique. For example the same 2-D motion field can be generated by the projection of different kinds of 3-D motion fields. This corresponds to the problem of solving equation 2 when $A$ is not one-to-one. In this case the lemma is not sufficient since it does not guarantee uniqueness. Let us assume, however, that the solution is still bounded: only a finite number of solutions, then, are possible due to the quantization. Uniqueness can be achieved by giving a set of *rules* that allow the selection of at most one value (possibly none) for the solution at every point: such a strategy obviously will always succeed due to the finite number of possible values at each location. It is worth noting that the solution could actually turn out to be defined only in some locations, corresponding to the selection of no value. The set of rules corresponds in our algorithm to the voting followed by non-maximum suppression.

### 3.2. Equivalent physical constraints

The algorithm can be extended to a less restrictive constraint: that the optical flow is locally linear or even quadratic instead of simply constant. The quadratic case corresponds to a very simple constraint on 3-D motion and surfaces, *under the assumption* that the optical flow is sufficiently close to the 2-D motion field. Under this assumption we can use the following result due to Waxman (1986): *the velocity field on the image plane originated by arbitrary, rigid 3-D motion of a planar surface patch is quadratic.* In this way, the algorithm exploits the physical assumption that surfaces are—at least relative to the image resolution—locally planar (the "allowed" world is thus a world of polyhedral solids, albeit with a very high number of faces). It is worth noting that our initial experiments indicate that the quadratic and even the linear assumptions do not change significantly the results obtained with the "constant constraint" algorithm.

### 3.3. An iteration scheme

The implementation of the quadratic patch constraint is computationally expensive, even for coarse discretization of the velocity values. Though it may be unnecessary to consider in practice quadratic patches, it is of interest to develop a scheme that allows for a fast approximate solution based on the constant field assumption that is then refined in terms of higher order assumptions such as linear and quadratic patch. It is natural to consider an iteration that first finds the best "constant" solution, then refines it with the best "linear" correction and finally finds the best "quadratic" correction. In general, the best quadratic correction does not provide the best quadratic approximation. Results however about the estimation of polynomial operators (see for instance Poggio, 1975, theorem 4.2) suggest that iterating the procedure should converge to the best quadratic approximation. In this way we can find the best "constant" estimation of the optical flow and then refine it by successive iterations that cycle from the lowest to the highest order and to the lowest again.

## 4. The Connection Machine$^{TM}$ implementation

The time $\Delta t$ between images is small, on the order of one video time frame (1/30th second). During this short time, the appearance of a moving object can change due to its own motion, camera motion, light source motion, or all three, among other effects (see Verri and Poggio, this Proceedings). However, when the local intensity variation in the surface albedo is sufficiently large, the errors introduced by these effects are minimized. For this reason, we use the output of an edge detection step as the input to later stages of optical flow computation. Both the Laplacian of a Gaussian and Canny's edge detector (Canny, 1986) have been used with good results. Let $E_t$ be an image containing a description of the features at time $t$. Features can, for example, describe the sign of the $x$ and $y$ components of the image gradient at edge points.

The comparison of features in $E_t$ and $E_{t+\Delta t}$ is performed for each $\underline{v}$ and is recorded at each $x, y$ in a matching map $M(\underline{v})$, which identifies whether the feature at $E_t(x, y)$ found a match in $E_{t+\Delta t}(x, y)$ when displaced by $(v_x, v_y)\Delta t$ in the image plane. This process is spatially parallel, operating at all $x, y$ simultaneously, for each $\underline{v}$ in the discrete set that is allowed. The process iterates over this range, generating the matching map $M(\underline{v})$.

The integration stage acts only upon the matching maps $M(\underline{v})$. By assumption, we are looking for the optical flow which is locally constant. We choose the flow $\underline{v}$ which, in a neighborhood $N$ around $x, y$, maximizes the number of matches. Again this procedure iterates over all $\underline{v}$ in the bounded range. We identify a displacement only at those locations $x, y$ at which the maximum vote is unique; ties are ambiguous and are eliminated. The result is a map of the optical flow at locations in the image at which a moving feature has undergone unambiguous motion.

The comparison and integrated stages could be merged into one stage, in which case we would be directly implementing a binary correlation scheme on the feature maps.

The constraint method can also be used to exploit more sophisticated and less restrictive assumptions about surfaces. We could, for instance, assume that the motion surface is locally planar, or that the patch is locally quadratic. These more general assumptions can be implemented by changing appropriately the geometry of the neighborhoods over which the voting takes place. We have implemented the iterative scheme for determining the best linear correction (linear in $x, y$) to the constant solution. There, the separation of the two stages is crucial for a fast implementation; for corrections that are not constant, the voting neighborhood no longer corresponds to a simple patch in the matching map of just one displacement. Also, the comparison operation can be done once and used for all later integration stages.

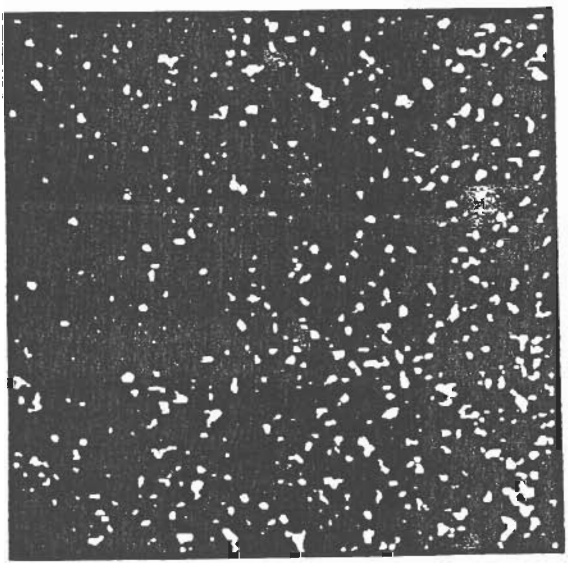


Figure 1. A disc with random texture is embedded in a background pattern with the same texture. The shape of the object becomes immediately visible when foreground and background pattern are moving with different speeds (Figure 2).

## 4.1. Examples

We used both synthetic and real images for testing the implementation of the algorithm on the Connection Machine$^{TM}$ system. The algorithm requires as much structure as possible in the images to be analyzed. In the synthetic images we produce sufficient structure by using random textures for the foreground

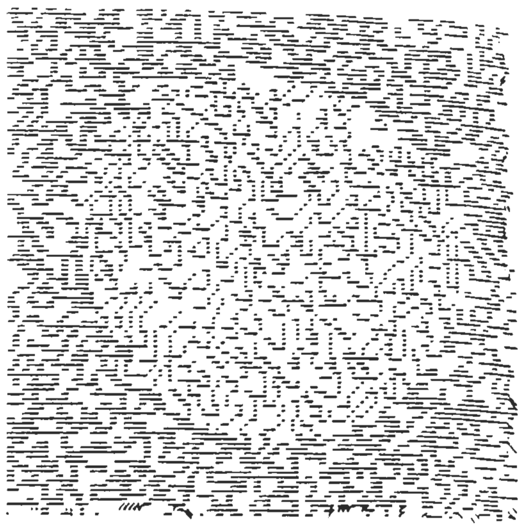


Figure 2. Needle diagram of the motion field computed by the constraint method. A disc with random texture is moving with half speed in the same direction as the background. The algorithm computes consistent velocities for foreground and background motion.

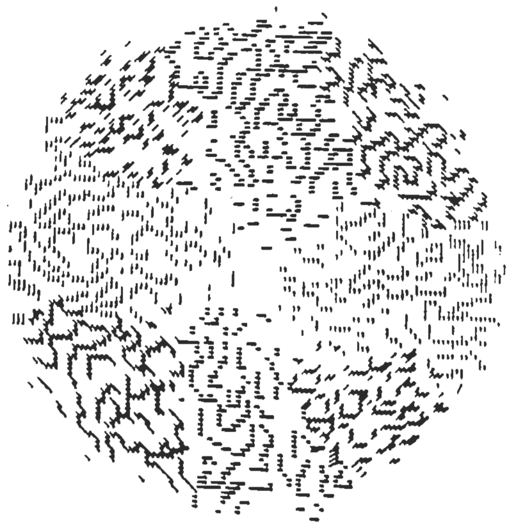


Figure 3. Needle diagram of the motion field computed by the constraint method. A disc with random texture is rotating clockwise at 2 degrees per time step in front of a stationary background.

and background patterns [1] (Figure 1). Since the foreground has the same texture as the background it remains invisible to the human observer as long as it is not moving. As soon as either the foreground or the background pattern begins to move the object becomes immediately visible. The needle diagrams of the opti-

[1] To prevent antialiasing in the motion computation output the images were appropriately bandpass filtered

cal flow field in Figure 2 show that the algorithm succesfully computes consistent motion if foreground and background patterns move with different speeds. In Figure 3 the disc-like foreground pattern is rotating clockwise in front of a stationary background. Again all motion vectors show locally consistent direction and magnitude of velocity for the foreground pattern. Figure 4 shows that the Connection Machine$^{TM}$ implementation of the constraint method is able to compute a consistent description of motion in a natural sequence of images. Two images are digitized in 30 msec intervals with the Vision machine's eye-head sytem and analyzed by the Connection Machine$^{TM}$. Almost all of the significant intensity changes on the moving robot are labelled with motion vectors pointing to the left which is consistent with the actual motion of the robot. The non-moving chair and the background is invisible to the motion detection mechanism. Only a few small patches show false motion due to apparent motion of specular reflections caused by changes in illumination between frames. On a Connection Machine$^{TM}$ having 16K processors, the optical flow of a 128 × 128 image can be computed in a few seconds, several hundreds of times faster than on a Symbolics 3640 Lisp Machine.

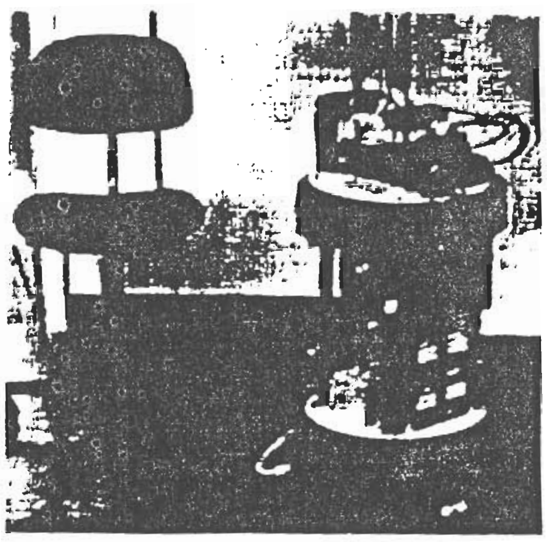
## 5. Finding discontinuities

We are presently analyzing several methods for finding efficiently initial estimates of the discontinuities of the optical flow that may be later refined (for instance at the integration stage). We give here a very brief outline of three methods:

### 5.1. Statistics of the voting step

At motion discontinuities the assumption of a constant (or linear or quadratic) motion field is obviously wrong. One would expect therefore that the "votes" at a motion discontinuity (say in the case of the "constant" motion algorithm) would fail to support clearly any single velocity. In fact, regions of close "ties", or equivalently of winners with locally minimum votes, often delinate motion discontinuities. Our implementation of this procedure scales the number of votes at a location by the total number of features in the voting neighborhood. Close ties receive values near 0.5. Figure 5 shows the result of thresholding this ratio at 0.75. This idea can be developed further by considering more complete statistics of the votes (Spoerri and Ullman, in preparation).

a



b

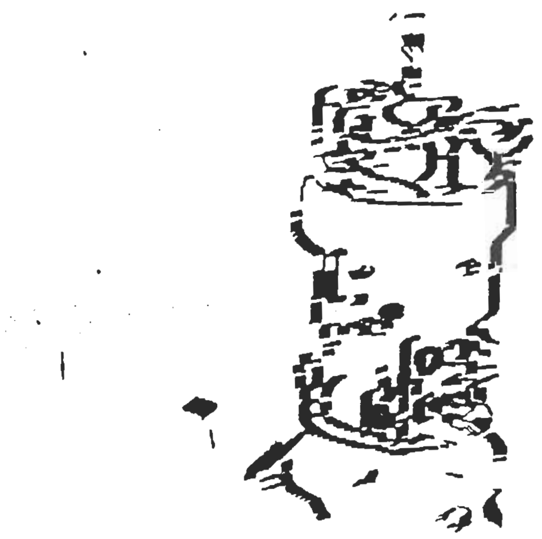


Figure 4a. Frame 1 of 2 from a motion sequence used to test parallel motion detection algorithms with real images (256 × 256). The robot is moving to the left. b. Output of the constraint method algorithm for the motion sequence shown above. Most of the needles point to the left consistently with the actual movement. The stationary background (wall, floor and chair) is invisible to the motion detectors. The small patches of indicated motion on the left are most probably caused by apparent motion of specular reflections due to changes in illumination between frames.

### 5.2. Vetoing coherent motion

Motion discontinuities can be found by using an algorithm that was suggested by data on the insect visual system (Reichardt et al., 1983). The idea is to inhibit or veto the value of the optical flow at each point by the average value of the field over a large region centered at that point *whenever* the motion is of the same type. The scheme suggested by the insect work is the following: at each $x, y$ consider separately the $x$ and the $y$ component of the optical flow, take its value and divide it by the average value, computed over a large region. Figure 6 shows the output of this operation on two examples. The average may be Gaussian weighted. It is quite intriguing to notice that the basic operation is very similar to a recent proposal by Land (1986). It is also similar to performing a center-surround operation (such as the Laplacian of a Gaussian, but with much larger surround) on the log of the optical flow.

### 5.3. Edge detection on the optical flow

Edge detection on the each component of the optical flow is the simplest way to obtain an initial estimate of discontinuities. We plan to use Canny's edge detector (Canny, 1986) on each component of the optical flow.

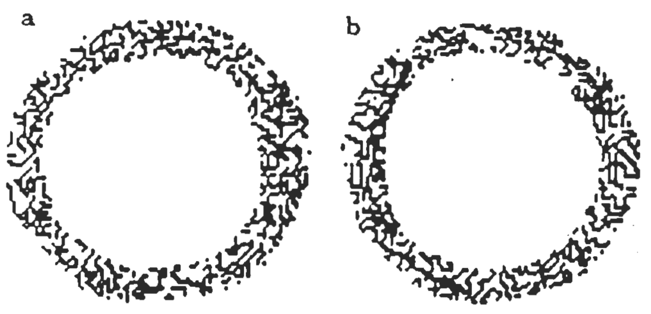


Figure 5. Edge labeling by relative motion. Motion discontinuities can be found by finding the locations that get a minimum number of votes for consistent motion. This should be the case at object boundaries for relative motion between foreground (object) and background. a. A disc moves with same speed but in opposite direction to the moving background. b. Object moves in same direction but half the speed of the background.

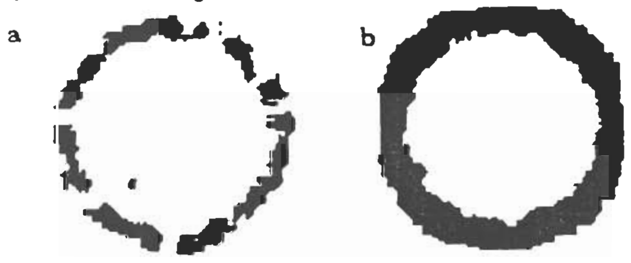


Figure 6. Edge labeling by relative motion. a. A disc with random texture is moving with opposite direction in front of a moving background with the same texture. Only those indicators of relative motion with values above a certain threshold are shown here. b. The same pattern is moving in front of the same background in the same direction but with half the velocity. The size of the labeled area depends on the size of the larger mask (see text).

### Reading list

Canny, J.F. (1986) A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Nov. 1986, Vol. 8, No. 6, 679-698.

Drumheller, M. and Poggio, T. (1986) Parallel Stereo *Proceedings of IEEE Conference on Robotics and Automation*, San Francisco.

Hassenstein, B. and Reichardt, W. (1956) Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Rüsselkäfers *Chlorophanus*. *Z. Naturforsch.* **IIb**, 513–524.

Hildreth, E. C. (1984a) *The Measurement of Visual Motion*. Cambridge: MIT Press.

Horn, B. K. P. and Schunck, B. G. (1981) Determining optical flow. *Artif. Intell.* **17**, 185–203.

Land, E.H. (1986) An alternative technique for the computation of the designator in the retinex theory of color vision. *P.N.A.S.* **83**, 3078-3080.

Marr, D. and Ullman, S. (1981) Directional selectivity and its use in early visual processing. *Proc. R. Soc. London Ser. B* **211**, 151–180.

Poggio, T. (1975) On optimal nonlinear associative recall. *Biol. Cybern.* **19**, 201–209.

Poggio, T. and Torre, V. (1984) Ill-posed problems and regularization analysis in early vision. Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo 773, C.B.I.P. Paper 001.

Poggio, T. and Reichardt, W. (1973) Considerations on models of movement detection. *Kybernetik* **13**, 223–227.

Reichardt, W., Poggio, T. and Hausen, K. (1983) Figure–ground discrimination by relative movement in the visual system of the fly. Part II: Towards the neural circuitry. *Biol. Cybern.* **46**, 1–30.

Tikhonov, A.N. and Arsenin, V.Y. (1977) *Solutions of ill-posed problems*. W.H.Winston, Washington, D.C.

Torre, V. and Poggio, T. (1978) A Synaptic mechanism possibly underlying directional selectivity to motion. *Proc. R. Soc. Lond. B* **202**, 409–416.

Ullman, S. (1981) Analysis of visual motion by biological and computer systems. *IEEE Computer*, August 1981, pp. 57–69.

Waxman, A. M. (1986) Image flow theory: A framework for 3-D inference from time–varying imagery. In *Advances in Computer Vision*, ed. C. Brown, New Jersey: Erlbaum. *In press*.