The MIT Vision Machine

T. Poggio, J. Little, E. Gamble, W. Gillett, D. Geiger, D. Weinshall, M. Villalba, N. Larson, T. Cass, H. Bülthoff, M. Drumheller, P. Oppenheimer, W. Yang, A. Hurlbert, D. Beymer, P. O'Donnell

Introduction: The Project and Its Goals

Computer vision has developed algorithms for several early vision processes, such as edge detection, stereopsis, motion, texture, and color, which give separate cues as to the distance from the viewer of three-dimensional surfaces, their shape, and their material properties. Biological vision systems, however, greatly outperform computer vision programs. It is clear that one of the keys to the reliability, flexibility, and robustness of biological vision systems in unconstrained environments is their ability to integrate many different visual cues. For this reason, we continue the development of a *Vision Machine* system to explore the issue of integration of early vision modules. The system also serves the purpose of developing parallel vision algorithms, because its main computational engine is a parallel supercomputer, the Connection Machine.

The idea behind the Vision Machine is that the main goal of the integration stage is to compute a map of the visible discontinuities in the scene, somewhat similar to a cartoon or a line-drawing. There are several reasons for this. First, experience with existing model-based recognition algorithms suggest that the critical problem in this type of recognition is to obtain a reasonably good map of the scene in terms of features such as edges and corners. The map does not need to be perfect (human recognition works with noisy and occluded line drawings) and, of course, it cannot be. But it should be significantly cleaner than the typical map provided by an edge detector. Second, discontinuities of surface properties are the most

important locations in a scene. Third, we have argued that discontinuities are ideal for integrating information from different visual cues.

It is also clear that there are several different approaches to the problem of how to integrate visual cues. Let us list some of the obvious possibilities:

1 There is no active integration of visual processes. Their individual outputs are "integrated" at the stage at which they are used, for example by a navigation system. This is the approach advocated by Brooks [1987]. While it makes sense for automatic, insect-like, visuo-motor tasks such as tracking a target or avoiding obstacles (for example, the fly's visuo-motor system [Reichardt & Poggio 1976]), it seems quite unlikely for visual perception in the wider sense.

2 The visual modules are so tightly coupled that it is impossible to consider visual modules as separate, even to a first order approximation. This view is unattractive on epistemological, engineering, and psychophysical grounds.

3 The visual modules are coupled to each other and to the image data in a parallel fashion—each process represented as an array coupled to the arrays associated with the other processes. This point of view is in the tradition of Marr's 2 $\frac{1}{2}$-D sketch, and especially of the "intrinsic images" of Barrow and Tenenbaum [1978]. Our present scheme is of this type, and exploits the machinery of Markov Random Field (MRF) models.

4 Integration of different vision modalities is taking place in a task-dependent way at specific locations—not over the whole image—and when it is needed—therefore not at all times. This approach is suggested by psychophysical data on visual attention and by the idea of visual routines [Ullman 1984] (see also Hurlbert and Poggio [1986], Mahoney [1987], and Bülthoff and Mallot [1988]).

We are presently exploring the third of these approaches. We believe that the last two approaches are compatible with each other. In particular, visual routines may operate on maps of discontinuities such as those delivered by the present Vision Machine, and therefore be located after a parallel, automatic integration stage. In real life, of course, it may be more a matter of coexistence. We believe, in fact, that a control structure based on specific knowledge about the properties of the various modules, the specific scene and the specific task will be needed in a later version of the Vision Machine to overview and control the MRF integration stage itself and its parameters. It is possible that the integration stage should be much more goal-directed than what our present methods (MRF based) allow. The main goal of our work is to find out whether this is true.

The Vision Machine project has a number of other goals. It provides a focus for developing parallel vision algorithms and for studying how to

organize a real-time vision system on a massively parallel supercomputer. It attempts to alter the usual paradigm of computer vision research over the past years: choose a specific problem, for example stereo, find an algorithm, and test it in isolation. The Vision Machine allows us to develop and test an algorithm in the context of the other modules and the requirements of the overall visual task, above all, visual recognition. For this reason, the project is more than an experiment in integration and parallel processing: it is a laboratory for our theories and algorithms.

Finally, the ultimate goal of the Vision Machine project is no less than the ultimate goal of vision research: to build a vision system that achieves human-level performance.

The Vision Machine System

The overall organization of the system is shown in figure 1. The image(s) are processed in parallel through independent algorithms or modules corresponding to different visual cues. Edges are extracted using Canny's edge detector. The stereo module computes disparity from the left and right images. The motion module estimates an approximation of the optical flow from pairs of images in a time sequence. The texture module computes texture attributes (such as density and orientation of textons [Voorhees 1987]). The color algorithm provides an estimate of the spectral albedo of the surfaces, independently of the *effective illumination*, that is, illumination gradients and shading effects, as suggested by Hurlbert and Poggio (see Poggio and Staff [1985]).

The measurements provided by the early vision modules are typically noisy, and possibly sparse (for stereo and motion). They are smoothed and made dense by exploiting known constraints within each process (for instance, that disparity is smooth). This is the stage of *approximation* and *restoration* of data, performed using a Markov Random Field model. Simultaneously, discontinuities are found in each cue. Prior knowledge of the behavior of discontinuities is exploited, for instance, the fact that they are continuous lines, not isolated points. Detection of discontinuities is aided by the information provided by brightness edges. Thus each cue, disparity, optical flow, texture, and color, is coupled to the edges in brightness.

The full scheme involves finding the various types of physical discontinuities in the surfaces, *depth discontinuities* (extremal edges and blades), *orientation discontinuities, specular edges, albedo edges* (or marks), and *shadow edges*, and coupling them with each other and back to the discontinuities in the visual cues, as illustrated in figure 1 (and suggested by Gamble, Geiger, Poggio, and Weinshall [1989]). So far we have implemented only the coupling of brightness edges to each of the cues provided by the early algorithm. As we will discuss later, the technique we use to

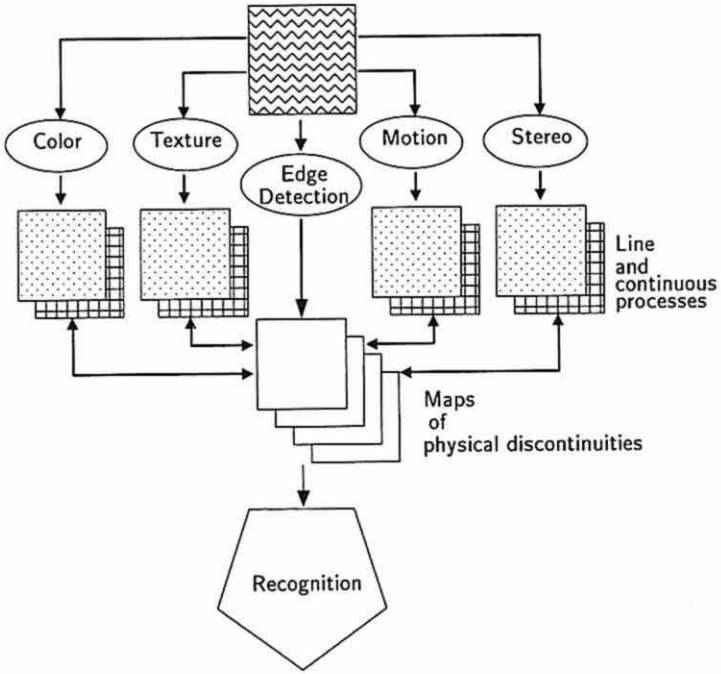

Figure 1. Overall organization of the Vision Machine.

approximate, to simultaneously detect discontinuities, and to couple the different processes, is based on MRF models. The output of the system is a set of labeled discontinuities of the surfaces around the viewer. Thus the scheme—an instance of inverse optics—computes *surface properties*, that is, attributes of the physical world and not anymore of the images. Note that we attempt to find discontinuities in surface properties and therefore qualitative surface properties: the *inverse optics* paradigm does **not** imply that physical properties of the surfaces, such as depth or reflectance, should be extracted *precisely, everywhere.* These discontinuities, taken together, represent a "cartoon" of the original scene which can be used for recognition and navigation (along with, interpolated depth, motion, texture, and color fields). As yet we did not integrate our ongoing work on grouping in the Vision Machine. We expect to use a saliency operation on the output of the edge detection process possibly before the use of intensity edges by the MRF stage. The grouping based on T-junctions [Beymer 1989] should take place on the intensity edges at the same level as the MRF stage. Initial work in recognition has been integrated in the system. The Vision Machine has been demonstrated working form images to recognition through the integration of visual cues.

The plan of this chapter is as follows. We will first review the current hardware of the Vision Machine: the eye-head system and the Connection Machine. We will then describe in some detail each of the early vision algorithms that are presently running and are part of the system. After this, the integration stage will be discussed. We will analyze some results, and illustrate the merits and the pitfalls of our present system. The last section will discuss a real-time visual system, and some ideas on how to put the system into VLSI circuits of analog and digital type.

Hardware

The Eye-Head system

Because of the scope of the Vision Machine project, a general purpose image input device is required. Such a device is the eye-head system. Here we discuss its current and future configurations.

The eye-head system consists of two CCD cameras, which act as eyes, mounted on a variable-attitude platform, which acts as the head. Inspired by biology, the apparatus is configured such that the head moves the eyes as a unit, while allowing the eyes to point independently. Each eye is equipped with a motorized zoom lens ($F1.4$, focal length from 12.5 to 75mm), allowing control of the iris, focus, and focal length by the host computer (currently a Symbolics 3600 Lisp Machine). Other hardware allows for repeatable calibration of the entire apparatus.

Because of the size and weight of the motorized lenses, it would be impractical to achieve eye movement by pointing the camera/lens assemblies directly. Instead, each assembly is mounted rigidly on the head, with eye movement achieved indirectly. In front of each lens is a pair of front surface mirrors, each of which can be pivoted by a galvanometer, providing two degrees of freedom in aiming the cameras. At the expense of a more complicated imaging geometry, we get a simple and fast pointing system for the eyes.

The head is attached to its mount via a spherical joint, allowing head rotation about two orthogonal axes (pan and tilt). Each axis is driven by a stepper motor coupled to its drive shaft through a harmonic drive. The latter provides a large gear ratio in conjunction with very little mechanical backlash. Under control of the stepper motors, the head can be panned 180 degrees from left to right, and tilted 90 degrees (from vertical-down to horizontal). Each of the stepper motors is provided with an optical shaft encoder for shaft position feedback (a closed-loop control scheme is employed for the stepper motors). The shaft encoders also provide an index pulse (one per revolution) which is used for joint calibration in conjunction

with mechanical limit switches. The latter also protect the head from damage due to excessive travel.

The overall control system for the eye-head system is distributed over a micro-processor network (UNET) developed at the MIT AI Laboratory for the control of vision/robotics hardware. The UNET is a "multi-drop" network supporting up to 32 micros, under the control of a single host. The micros normally function as network slaves, with the host acting as the master. In this mode the micros only "speak when spoken to," responding to various network operations either by receiving information (command or otherwise) or by transmitting information (such as status or results). Associated with each micro on the UNET is a local 16-bit bus (UBUS), which is totally under the control of the micro. Peripheral devices such as motor drivers, galvanometer drivers, and pulse width modulators (PWMs), to name a few, which can be interfaced at this level.

At present, three micro-processors are installed on the eye-head UNET: one each for the galvanometers, motorized lenses, and stepper motors. The processors currently employed are based on the Intel 8051. Each of these micros has an assortment of UBUS peripherals under its control. By making these peripherals sufficiently powerful, each micro's control task can remain simple and manageable. Code for the micros, written in both assembly language and C, is facilitated by a Lisp-based debugging environment.

Our computational engine: The Connection Machine

The Connection Machine is a powerful fine-grained parallel machine which has proven useful for implementation of vision algorithms. In implementing these algorithms, several different models of using the Connection Machine have emerged, because the machine provides several different communication modes. The Connection Machine implementation of algorithms can take advantage of the underlying architecture of the machine in novel ways. We describe here several common, elementary operations which recur throughout the following discussion of parallel algorithms.

◇ The Connection Machine

The CM-2 version of the Connection Machine [Hillis 1985] is a parallel computing machine with between 16K and 64K processors, operating under a single instruction stream broadcast to all processors. It is a Single Instruction Multiple Data (SIMD) machine; all processors execute the same control stream. Each processor is a simple 1-bit processor, currently with 64K bits of memory, optionally with a floating point arithmetic accelerator,

shared among 16 (or 32) processors. There are two modes of communication among the processors: the NEWS network and the *router*. The NEWS network (so-called because the connections are in the four cardinal directions) provides rapid direct communication between neighboring processors in an rectangular grid of arbitrary dimension. For example, 64K processors could be configured into a two-dimensional 256×256 grid, or into a four-dimesional $64 \times 64 \times 4 \times 4$ grid. The second mode of communication is the *router*, which allows messages to be sent from any processor to any other processor in the machine. The processors in the Connection Machine can be envisioned as being the vertices of a 16-dimensional hypercube (in fact, it is a 12-dimensional hypercube; at each vertex of the hypercube resides a chip containing 16 processors). Each processor in the Connection Machine is identified by its hypercube address in the range $0 \ldots 65535$, imposing a linear order on the processors. This address denotes the destination of messages handled by the router. Messages pass along the edges of the hypercube from source processors to destination processors. The Connection Machine also has facilities for returning to the host machine the result of various operations on a field in all processors; it can return the global maximum, minimum, sum, logical AND, and logical OR of the field.

The floating-point arithmetic accelerator, which may optionally be added to the Connection Machine, provides a significant increase in the speed of both single and double precision computations. One floating-point processor chip serves a pair Connection Machine processor chips with 32 total processors in a pipelined fashion, and can produce a speed-up of more than a factor of twenty.

To allow the machine to manipulate data structures with more than 64K elements, the Connection Machine supports *virtual processors*. A single physical processor can operate as a set of multiple virtual processors by serializing operations in time, and by partitioning the memory of each processor. This is otherwise invisible to the user. Connection Machine programs utilize Common Lisp syntax, in a language called *Lisp, and are manipulated in the same fashion as Lisp programs.

◇ Powerful primitive operations

Many vision problems must be solved by a combination of communication modes on the Connection Machine. The design of these algorithms takes advantage of the underlying architecture of the machine in novel ways. There are several common, elementary operations used in this discussion of parallel algorithms: routing operations, scanning, and distance doubling.

Routing. Memory in the Connection Machine is associated with processors. Local memory can be accessed rapidly. Memory of processors nearby

in the NEWS network can be accessed by passing it through the processors on the path between the source and the destination. At present, NEWS accesses in the machine are made in the same direction for all processors. The *router* on the Connection Machine provides parallel reads and writes among processor memory at arbitrary distances and with arbitrary patterns. It uses a packet-switched message routing scheme to direct messages along the hypercube connections to their destinations. This powerful communication mode can be used to reconfigure completely, in one parallel write operation taking one router cycle, a field of information in the machine. The Connection Machine supplies instructions so that many processors can read from the same location or write to the same location, but because these memory references can cause significant delay, we will usually only consider exclusive read, exclusive write instructions. We will usually not allow more than one processor to access the memory of another processor at one time. The Connection Machine can combine messages at a destination by various operations, such as logical AND, inclusive OR, summation, and maximum or minimum.

Scanning. The *scan* operations [Blelloch 1987] can be used to simplify and speed up many algorithms. They directly take advantage of the hypercube connections underlying the router, and can be used to distribute values among the processors and to aggregate values using associative operators. Formally, the *scan* operation takes a binary associative operator $\oplus$, with identity 0, and an ordered set $[a_0, a_1, \ldots, a_{n-1}]$, and returns the set $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1})]$. This operation is sometimes referred to as the *data independent prefix operation* [Kruskal *et al.* 1985]. Binary associative operators include *minimum, maximum,* and *plus.*

The four scan operations *plus-scan, max-scan, min-scan,* and *copy-scan* are implemented in microcode, and take about the same amount of time as a routing cycle. The *copy-scan* operation takes a value at the first processor and distributes it to the other processors. These scan operations can take *segment bits* that divide the processor ordering into segments. The beginning of each segment is marked by a processor whose segment bit is set, and the scan operations start over again at the beginning of each segment.

The *scan* operations also work using the NEWS addressing scheme, termed *grid-scans.* These compute the sum, and quickly find the maximum, copy, or number values along rows or columns of the NEWS grid.

For example, *grid-scans* can be used to find, for each pixel, the sum of a square region with width $2m+1$ centered at the pixel. This sum is computed using the following steps. First, a *plus-scan* operation accumulates partial sums for all pixels along the rows. Each pixel then gets the result of the scan from the processor m in front of it and m behind it; the difference of these two values represents the sum, for each pixel, of its neighborhood

along the row. We now execute the same calculation on the columns, resulting in the sum, for each pixel, of the elements in its square. The whole process only requires a few *scans* and routing operations, and runs in time independent of the size of m. The summation operations are generally useful to accumulate local support in many of our algorithms, such as stereo and motion.

Distance Doubling. Another important primitive operation is *distance doubling* [Wyllie 1979; Lim 1986] which can be used to compute the effect of any binary, associative operation, as in *scan,* on processors linked in a list or a ring. For example, using *max, distance doubling* can find the extremum of a field contained in the processors. Using message-passing on the router, *distance doubling* can propagate the extreme value to all processors in the ring of N processors in $O(\log N)$ steps. Each step involves two *send* operations. Typically, the value to be maximized is chosen to be the hypercube address. At termination, each processor in the ring knows the label of the maximum processor in the ring, hereafter termed the *principal processor.* This labels all connected processors uniquely, and nominates a processor as the representative for the entire set of connected processors. At the same time, the distance from the *principal* can be computed in each processor. Each processor initially, at step 0, has the address of the next processor in the ring, and a value which is to be maximized. At the termination of the i^{th} step, a processor knows the addresses of processors $2^i + 1$ away, and the maximum of all values within 2^{i-1} processors away. In the example, the maximum value has been propagated to all 8 processors in $\log 8 = 3$ steps.

Early Vision Algorithms and their Parallel Implementation

Edge detection

Edge detection is a key first step in correctly identifying physical changes. The apparently simple problem of measuring sharp brightness changes in the image has proven to be difficult. It is now clear that edge detection should be intended not simply as finding "edges" in the images, an ill-defined concept in general, but as measuring appropriate derivatives of the brightness data. This involves the task-dependent use of different two-dimensional derivatives. In many cases, it is appropriate to mark locations corresponding to appropriate critical points of the derivative such as maxima or zeroes. In some cases, later algorithms based on these binary features (presence or absence of edges) may be equivalent, or very similar, to algorithms that directly use the continuous value of the derivatives. A case

in point is provided by our stereo and motion algorithms, to be described later. As a consequence, one should not always make a sharp distinction between edge-based and intensity based algorithms; the distinction is more blurred, and in some cases it is almost a matter of implementation.

In our current implementation of the Vision Machine, we are using two different kinds of edges. The first consists of zero-crossings in the Laplacian of the image filtered through an appropriate Gaussian. The second consists of the edges found by Canny's edge detector. Zero-crossings can be used by our stereo and motion algorithms (though we have mainly used Canny's edges at fine resolution). Canny's edges (at a coarser resolution) are input to the MRF integration scheme.

◊ Zero-crossings

Because the derivative operation is ill-posed, we need to filter the resultant data through an appropriate low-pass filter [Torre & Poggio 1984]. The filter of choice (but not the only possibility!) is a Gaussian at a suitable spatial scale. An interesting and simple implementation of Gaussian convolution relies on the binomial approximation to the Gaussian distribution. This algorithm requires only integer addition, shifting, and local communication on the 2D mesh, so it can be implemented on a simple 2D mesh architecture (such as the NEWS network on the Connection Machine).

The Laplacian of a Gaussian is often approximated by the difference of Gaussians. The Laplacian of a Gaussian can also be computed by convolution with a Gaussian followed by convolution with a discrete Laplacian; we have implemented both on the Connection Machine. To detect zero-crossings, the computation at each pixel need only examine the sign bits of neighboring pixels.

◊ Canny edge detection

The Canny edge detector is often used in image understanding. It is based on directional derivatives, so it has improved localization. The Canny edge detector on the Connection Machine consists of the following steps:

- Gaussian smoothing
- Directional derivative
- Non-maximum suppression
- Thresholding with hysteresis.

Gaussian filtering, as described above, is a local operation. Computing directional derivatives is also local, using a finite difference approximation referencing only local neighbors in the image grid.

Non-maximum Suppression. Non-maximum suppression selects as edge candidates those pixels for which the gradient magnitude is maximal in the direction of the gradient. This involves interpolating the gradient magnitude between each of two pairs of adjacent pixels among the eight neighbors of a pixel, one forward in the gradient direction, and one backward. However, it may not be crucial to use interpolation, in which case magnitudes of neighboring values can be directly compared.

Thresholding with Hysteresis. Thresholding with hysteresis eliminates weak edges due to noise, using the threshold, while connecting extended curves over small gaps using hysteresis. Two thresholds are computed, *low* and *high*, based on an estimate of the noise in the image brightness. The non-maximum suppression step selects those pixels where the gradient magnitude is maximal in the direction of the gradient. In the thresholding step, all selected pixels with gradient magnitude below *low* are eliminated. All pixels with values above *high* are considered as edges. All pixels with values between *low* and *high* are edges if they can be connected to a pixel above *high* through a chain of pixels above *low*. All others are eliminated.

This is a spreading activation operation; it propagates information along a set of connected edge pixels. The algorithm iterates, in each step marking as *edge* pixels any *low* pixels adjacent to *edge* pixels. When no pixels change state, the iteration terminates, taking $O(m)$ steps, a number proportional to the length m of the longest chain of *low* pixels which eventually become *edge* pixels. The running time of this operation can be reduced to $O(\log m)$, using *distance doubling*.

Noise Estimation. Estimating noise in the image can be done by analyzing a histogram of the gradient magnitudes. Most computational implementations of this step perform a global analysis of the gradient magnitude distribution, which is essentially non-local; we have had success with a Connection Machine implementation using local histograms. The thresholds used in Canny edge detection depend on the final task for which the edges are used. A conservative strategy can use an arbitrary low threshold to eliminate the need for the costly processing required to accumulate a histogram. Where a more precise estimate of noise is needed, it may be possible to find a scheme that uses a coarse estimate of the gradient magnitude distribution, with minimal global communication.

Stereo

The Drumheller-Poggio parallel stereo algorithm [Drumheller & Poggio 1986] runs as part of the Vision Machine. Disparity data produced by the

algorithm comprise one of the inputs to the MRF-based integration stage of the Vision Machine. We are exploring various extensions of the algorithm, as well as the possible use of feedback from the integration stage. In this section, we will review the algorithm briefly, then proceed to a discussion of current research.

The stereo algorithm runs on the Connection Machine system with good results on natural scenes in times that are typically on the order of one second. The stereo algorithm is presently being extended in the context of the Vision Machine project.

◇ The Drumheller-Poggio stereo algorithm

Stereo matching is an ill-posed problem (see Bertero *et al.* [1988]) that cannot be solved without taking advantage of natural constraints. The *continuity constraint* (see, for instance, Marr and Poggio [1976]) asserts that the world consists primarily of piecewise smooth surfaces. If the scene contains no transparent objects, then the *uniqueness constraint* applies: there can be only one match along the left or right lines of sight. If there are no narrow occluding objects, the *ordering constraint* [Poggio & Yuille 1984] holds: any two points must be imaged in the same relative order in the left and right eyes.

The specific *a priori* assumption on which the algorithm is based is that the disparity, that is, the depth of the surface, is locally constant in a small region surrounding a pixel. It is a restrictive assumption which, however, may be a satisfactory *local* approximation in many cases (it can be extended to more general surface assumptions in a straightforward way, but at a high computational cost). Let $E_L(x,y)$ and $E_R(x,y)$ represent the left and the right image of a stereo pair, or some transformation of it, such as filtered images or a map of the zero-crossings 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 discrete disparity $d(x,y)$ at each location x,y in the image that minimizes

$$\|E_L(x,y) - E_R(x + d(x,y), y)\|_{\text{patch}_i} \tag{1}$$

where the norm is a summation over a local neighborhood centered at each location (x,y); $d(x)$ is assumed constant in the neighborhood. The previous equation implies that we should look at each (x,y) for $d(x,y)$ such that

$$\int_{\text{patch}_i} (E_L(x,y)E_R(x + d(x,y), y))^2 dxdy \tag{2}$$

is maximized.

The algorithm that we have implemented on the Connection Machine is actually somewhat more complicated, because it involves geometric constraints that affect the way the maximum operation is performed (see Drumheller and Poggio [1986]). The implementation currently used in the Vision Machine at the Artificial Intelligence Laboratory uses the maps of Canny edges obtained from each image for E_L and E_R.

In more detail, the algorithm is composed of the following steps:

1 Compute features for matching.

2 Compute potential matches between features.

3 Determine the degree of continuity around each potential match.

4 Choose correct matches based on the constraints of continuity, uniqueness, and ordering.

Potential matches between features are computed in the following way. Assuming that the images are registered so that the epipolar lines are horizontal, the stereo matching problem becomes one-dimensional: an edge in the left image can match any of the edges in the corresponding horizontal scan line in the right image. Sliding the right image over the left image horizontally, we compute a set of *potential match planes*, one for each horizontal disparity. Let $p(x,y,d)$ denote the value of the (x,y) entry of the potential match plane at disparity d. We set $p(x,y,d) = 1$ if there is an edge at location (x,y) in the left image and a compatible edge at location $(x-d,y)$ in the right image; otherwise, set $p(x,y,d) = 0$. In the case of the DOG edge detector, two edges are compatible if the sign of the convolution for each edge is the same.

To determine the degree of continuity around each potential match (x,y,d), we compute a local support score $s(x,y,d) = \sum_{Patch} p(x,y,d)$, where *patch* is a small neighborhood of (x,y,d) within the *dth* potential match plane. In effect, nearby points in *patch* can "vote" for the disparity d. The score $s(x,y,d)$ will be high if the continuity constraint is satisfied near (x,y,d), that is, if *patch* contains many votes. This step corresponds to the integral over the patch in the last equation.

Finally, we attempt to select the correct matches by applying the uniqueness and ordering constraints (see above). To apply the uniqueness constraint, each match suppresses all other matches along the left and right lines of sight with weaker scores. To enforce the ordering constraint, if two matches are not imaged in the same relative order in left and right views, we discard the match with the smaller support score. In effect, each match suppresses matches with lower scores in its forbidden zone [Yuille & Poggio 1984]. This step corresponds to choosing the disparity value that maximizes the integral of the last equation.

◊ Improvements

Using this algorithm as a base, we have explored several of the following topics:

Detection of Depth Discontinuities. The Marr-Poggio continuity constraint is both a strength and a weakness of the stereo algorithm. Favoring continuous disparity surfaces reduces the solution space tremendously, but also tends to smooth over depth discontinuities present in the scene. Consider what happens near a linear depth discontinuity, say a point near the edge of a table viewed from above. The square local support neighborhood for the point will be divided between points on the table and points on the floor; thus, almost half of the votes will be for the wrong disparity.

One solution to this problem is feedback from the MRF integration stage. We can take the depth discontinuities located by the integration stage (using the results from a first pass of the stereo algorithm, among other inputs) and use them to restrict the local support neighborhoods so that they do not span discontinuities. In the example mentioned above, the support neighborhood would be trimmed to avoid crossing the discontinuity between the table and the floor, and thus would not pick up spurious votes from the floor.

We can also try to locate discontinuities by examining intermediate results of the stereo algorithm. Consider a histogram of votes versus disparity for the table/floor example. For a support region centered near the edge of the table, we expect to see two strong peaks: one at the disparity of the floor, and the other at the disparity of the table. Therefore a bimodal histogram is strong evidence for the presence of a discontinuity.

These two ideas can be used in conjunction. Discontinuity detection within stereo can take advantage of the extra information provided by the vote histograms. By passing better depth data (and perhaps candidate discontinuity locations) to the integration stage, we improve the detection of discontinuities at the higher level.

Improving the Stereo Matcher. The original Drumheller-Poggio algorithm matched DOG zero-crossings, where the local support score counted the number of zero-crossings in the left image patch matching edges in the right image patch at a given disparity. We have modified the matcher in a variety of ways.

1 Canny edges. The matcher now uses edges derived by a parallel implementation of the Canny edge detector [Canny 1983; Little *et al.* 1987] rather than DOG zero-crossings, for better localization.

2 Gradient direction constraint. We allow two Canny edges to match only if the associated brightness gradient directions are aligned within a parameterized tolerance. This is analogous to the restriction in the Marr-Poggio-Grimson stereo algorithm [Grimson 1981] where two zero-crossings can match only if the directions of the DOG gradients are approximately equal. Matching gradient orientations is a tighter constraint than matching the sign of the DOG convolution. Furthermore, the DOG sign is numerically unstable for horizontally oriented edges.

3 The scores are now normalized to take into account the number of edges in the left and right image patches eligible to match, so that patches with high edge densities do not generate artificially high scores. We plan to change the matcher so that edges that fail to match would count as negative evidence by reducing the support score, but this has not yet been implemented.

In the near future, we will explore matching brightness values as well as edges, using a cross-correlation approach similar to that of Little, for motion estimation.

Identifying Areas that are Outside of the Matcher's Disparity Range. The stereo algorithm searches a limited disparity range, selected manually. Every potential match in the scene (an edge with a matching edge at some disparity) is assigned the in-range disparity with the highest score, even though the correct disparity may be out of range. How can we tell when an area of the scene is out of range? The most effective approach that we have attempted to date is to look for regions with low matching scores. Two patches that are incorrectly matched will, in general, produce a low matching score.

◊ Memory-based registration and calibration

Registration of the image pair for the stereo algorithm is done by presenting to the system a pattern of dots, roughly on a sparse grid, at the distance around which stereo has to operate. The registration is accomplished using a warping computed by matching the dots from the left and right images. The dots are sparse enough that matching is unambiguous. The matching defines a warping vector for each dot; at other points the warping is computed by bilinear interpolation of the two components of warping vectors. The warping necessary for mapping the right image onto the left image is then stored. Prior to stereo-matching, the right image is warped according to the pre-stored addresses by sending each pixel in the right image to the processor specified in the table.

The warping table corrects for deformations, including those due to vertical disparities and rotations, those due to the image geometry (errors in the alignment of the cameras, perspective projection, errors introduced

by the optics, etc.) We plan to store several warping tables for each of a few convergence angles of the two cameras (assuming symmetric convergence). We conjecture that simple interpolation can yield sufficiently accurate warping tables for fixation angles intermediate to the ones stored. Note that these tables are independent of the position of the head. Absolute depth is not the concern here (we are not using it in our present Vision Machine), but it could easily be recovered from knowledge of the convergence angle. Note also that the whole registration scheme has the flavor of a learning process. Convergence angles are inputs and warping tables are the outputs of the modules; the set of angles, together with the associated warping tables, represent the set of input-output examples. The system can "generalize" by interpolating between warping tables and providing the warping corresponding to a vergence angle that does not appear in the set of "examples." Calibration of disparity to depth could be done in a similar way.

Motion

The motion algorithm [Bülthoff *et al.* 1989] computes the optical flow field, a vector field that approximates the projected motion field. The procedure produces sparse or dense output, depending on whether it uses edge features or intensities. The algorithm assumes that image displacements are small, within a range $(\pm\delta, \pm\delta)$. It is also assumed that the optical flow is locally constant in a small region surrounding a point. This assumption is strictly only true for translational motion of 3D planar surface patches parallel to the image plane. It is a restrictive assumption which, 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 Δt, such as filtered images, or a map of the brightness changes in the two images (more generally, they can be maps containing a feature vector at each location (x, y) in the image) [Kass 1986; Nishihara 1984].

We look for a discrete motion displacement $\underline{v} = (v_x, v_y)$ at each location x, y in the image that minimizes

$$\|E_t(x, y) - E_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t)\|_{\text{patch}_i} = \min \tag{3}$$

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

$$\int_{\text{patch}_i} (E_t(x, y) - E_{t+\Delta t}(x + v_x\Delta t, y + v_y\Delta t))^2 dxdy \tag{4}$$

is minimized. Alternatively, one can maximize the negative of the integrated result. The last equation represents the sum of the pointwise squared differences 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 the previous expression. 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 differencing (taking the logical "exclusive or") 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 summate the total in an (x, y) neighborhood at the same disparity. Note that this summation operation is efficiently implemented on the Connection Machine using *scan computations*. Each processor thus collects a vote indicating support that a patch of surface exists at that displacement. The algorithm iterates over all displacements in the range $(\pm\delta, \pm\delta)$, recording the values of the integral for each displacement. The last stage is to choose $\underline{v}(x, y)$ among the displacements in the allowed range 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. The actual implementation of this scheme can be simplified so that the "non-maximum suppression" occurs during iteration over displacements, so that no actual table of summed differences over displacements need be constructed. In practice, the algorithm has been shown to be effective both for synthetic and natural images using different types of features or measurements on the brightness data, including edges (both zero-crossings of the Laplacian of Gaussian and Canny's method), which generate sparse results along brightness edges, or brightness data directly, or the Laplacian of Gaussian, or its sign, which generate dense results. Because the optical flow is computed from quantities integrated over the individual patches, the results are robust against the effects of uncorrelated noise.

The comparison stage employs patchwise cross-correlation, which exploits local constancy of the optical flow (the velocity field is guaranteed to be constant for translations parallel to the image plane of a planar surface patch); it is a cubic polynomial for arbitrary motion of a planar surface (see Waxman [1987], and Little *et al.* [1987]). Experimentally, we have used zero-crossings, the Laplacian of Gaussian filtered image, its sign, and the smoothed brightness values, with similar results. It is interesting that methods *superficially* so different (edge-based and intensity-based) give such similar results. As we mentioned earlier, this is not surprising. There are theoretical arguments that support, for instance, the equivalence of cross-correlating the sign bit of the Laplacian filtered image and the Laplacian filtered image itself. The argument is based on the following theorem,

which is a slight reformulation of a well-known theorem.

Theorem. If $f(x,y)$ and $g(x,y)$ are zero mean jointly normal processes, their cross-correlation is determined fully by the correlation of the sign of f and of the sign of g (and determines it). In particular

$$R_{\bar{f},\bar{g}} = \frac{2}{\pi} arcsin(R_{f,g})$$

where $\bar{f} = sign\ f$ and $\bar{g} = sign\ g$.

Thus, cross-correlation of the sign bit is exactly equivalent to cross-correlation of the signal itself (for Gaussian processes). Note that from the point of view of information, the sign bit of the signal is completely equivalent to the zero-crossing of the signal. Nishihara [1984] first used patchwise cross-correlation of the sign bit of DOG filtered images, and has implemented it more recently on real-time hardware [Nishihara & Crossley 1988].

The existence of discontinuities can be detected in optical flow, as in stereo, both during computation and by processing the resulting flow field. The latter field is input to the MRF integration stage. During computation, discontinuities in optical flow arising from occlusions are indicated by low normalized scores for the chosen displacement.

Color

The color algorithm that we have implemented is a very preliminary version of a module that should find the boundaries in the surface spectral reflectance function, that is, discontinuities in the surface color. The algorithm relies on the idea of *effective illumination* and on the *single source* assumption, both introduced by Hurlbert and Poggio.

The single source assumption states that the illumination may be separated into two components, one dependent only on wavelength, and one dependent only on spatial coordinates; this generally holds for illumination from a single light source. It allows us to write the image irradiance equation for a Lambertian world as

$$I^\nu = k^\nu E(x,y)\rho^\nu(x,y) \tag{5}$$

where I^ν is the image irradiance in the νth spectral channel (ν = *red, green, blue*), $\rho^\nu(x,y)$ is the surface spectral reflectance (or albedo), and the effective illumination $E(x,y)$ absorbs the spatial variations of the illumination and the shading due to the 3D shape of surfaces (k^ν is a constant for each channel, and depends only on the luminant). A simple segmentation algorithm is then obtained by considering the equation

$$H(x,y) = \frac{I^r}{I^r + I^g} = \frac{k^r\ \rho^r}{k^r\rho^r\ +\ k^g\rho^g} \tag{6}$$

which changes only when ρ^r, or ρ^g, or both change. Thus H, which is piecewise constant, has discontinuities that mark changes in the surface albedo, independently of changes in the effective illumination.

The quantity $H(x,y)$ is defined almost everywhere, but is typically noisy. To counter the effect of noise, we exploit the prior information that H should be piecewise constant with discontinuities that are themselves continuous, non-intersecting lines. As we will discuss later, this restoration step is achieved by using a MRF model. This algorithm works only under the restrictive assumption that specular reflections can be neglected. Hurlbert [1989] discusses in more detail the scheme outlined here and how it can be extended to more general conditions.

Texture

The texture algorithm is a greatly simplified parallel version of the texture algorithm developed by Voorhees and Poggio [1987]. Texture is a scalar measure computed by summation of texton densities over small regions surrounding every point. Discontinuities in this measure can correspond to occlusion boundaries, or orientation discontinuities, which cause foreshortening. Textons are computed in the image by simple approximation to the methods presented in Voorhees and Poggio [1987]. For this example, the textons are restricted to blob-like regions, without regard to orientation selection.

To compute textons, the image is first filtered by a Laplacian of Gaussian filter at several different scales. The smallest scale selects the textural elements. The Laplacian of Gaussian image is then thresholded at a non-zero value to find the regions which comprise the blobs identified by the textons. The result is a binary image with non-zero values only in the areas of the blobs. A simple summation counts the density of blobs (the portion of the summation region covered by blobs) in a small area surrounding each point. This operation effectively measures the density of blobs at the small scale, while also counting the presence of blobs caused by large occlusion edges at the boundaries of textured regions. Contrast boundaries appear as blobs in the Laplacian of Gaussian image. To remove their effect, we use the Laplacian of Gaussian image at a slightly coarser scale. Blobs caused by the texture at the fine scale do not appear at this coarser scale, while the contrast boundaries, as well as all other blobs at coarser scales, remain. This coarse blob image filters the fine blobs; blobs at the coarser scale are removed from the fine scale image. Then, summation, whether with a simple scan operation, or Gaussian filtering, can determine the blob density at the fine scale only. This is one example where multiple spatial scales are used in the present implementation of the Vision Machine.

The integration stage and MRF

Whereas it is reasonable that combining the evidence provided by multiple cues (for example, edge detection, stereo, and color) should provide a more reliable map of the surfaces than any single cue alone, it is not obvious how this integration can be accomplished. The various physical processes that contribute to image formation, *surface depth, surface orientation, albedo* (Lambertian and specular component), *illumination*, are coupled to the image data, and therefore to each other, through the imaging equation. The coupling is, however, difficult to exploit in a robust way, because it depends critically on the reflectance and imaging models. We argue that the coupling of the image data to the surface and illumination properties is of a more qualitative and robust sort at locations in which image brightness changes sharply and surface properties are discontinuous, in short, at edges. The intuitive reason for this is that at discontinuities, the coupling between different physical processes and the image data is robust and qualitative. For instance, a depth discontinuity usually originates a brightness edge in the image, and a motion boundary often corresponds to a depth discontinuity (and an brightness edge) in the image. This view suggests the following integration scheme for restoring the data provided by early modules. The results provided by stereo, motion, and other visual cues are typically noisy and sparse. We can improve them by exploiting the fact that they should be smooth, or even piecewise constant (as in the case of the albedo), between discontinuities. We can exploit *a priori* information about generic properties of the discontinuities themselves, for instance, that they usually are continuous and non-intersecting.

The idea, is then, to detect discontinuities in each cue, for instance depth, simultaneously with the approximation of the depth data. The detection of discontinuities is helped by information on the presence and type of discontinuities in the surfaces and surface properties (see figure 1), which are coupled to the brightness edges in the image.

Note that reliable detection of discontinuities is critical for a vision system, because discontinuities are often the most important locations in a scene; depth discontinuities, for example, normally correspond to the boundaries of an object or an object part. The idea is thus to couple different cues through their discontinuities and to use information from several cues simultaneously to help refine the initial estimation of discontinuities, which are typically noisy and sparse.

How can this be done? We have chosen to use the machinery of Markov Random Fields (MRFs), initially suggested for image processing by Geman and Geman [1984]. In the following section, we will give a brief, informal outline of the technique and of our integration scheme. More detailed information about MRFs can be found in Geman and Geman [1984] and Marroquin [1987]. Gamble and Poggio [1987] describe an earlier version

of our integration scheme and its implementation as outlined in the next section.

◊ MRF models

Consider the prototypical problem of approximating a surface given sparse and noisy data (depth data) on a regular 2D lattice of sites. We first define the prior probability of the class of surfaces we are interested in. The probability of a certain depth at any given site in the lattice depends only upon neighboring sites (the Markov property). Because of the Clifford-Hammersley theorem, the prior probability is guaranteed to have the Gibbs form

$$P(f) = \frac{1}{Z} e^{-\frac{U(f)}{T}} \tag{7}$$

where Z is a normalization constant, T is called temperature, and $U(f) = \sum_C U_C(f)$ is an energy function that can be computed as the sum of local contributions from each neighborhood. The sum of the *potentials*, $U_C(X)$, is over the neighborhood's *cliques*. A clique is either a single lattice site or a set of lattice sites such that any two sites belonging to it are neighbors of one another. Thus $U(f)$ can be considered as the sum over the possible configurations of each neighborhood (see Marroquin [1987]). As a simple example, when the surfaces are expected to be smooth, the prior probability can be given as sums of terms such as

$$U_c(f) = (f_i - f_j)^2 \tag{8}$$

where i and j are neighboring sites (belonging to the same clique).

If a model of the observation process is available (that is, a model of the noise), then one can write the conditional probability $P(g/f)$ of the sparse observation g for any given surface f. Bayes Theorem then allows one to write the posterior distribution

$$P(f/g) = \frac{1}{Z} e^{\frac{-U(f/g)}{T}} \tag{9}$$

In the simple earlier example, we have (for Gaussian noise)

$$U(f/g) = \sum_C \alpha\gamma_i (f_i - g_i)^2 + (f_i - f_j)^2 \tag{10}$$

where $\gamma_i = 1$ only where data are available. More complicated cases can be handled in a similar manner.

The posterior distribution cannot be solved analytically, but sample distributions can be obtained using Monte Carlo techniques such as the Metropolis algorithm. These algorithms sample the space of possible surfaces according to the probability distribution $P(f/g)$ that is determined by the prior knowledge of the allowed class of surfaces, the model of noise, and the observed data. In our implementation, a highly parallel computer

generates a sequence of surfaces from which, for instance, the surface corresponding to the maximum of $P(f/g)$ can be found. This corresponds to finding the global minimum of $U(f/g)$ (simulated annealing is one of the possible techniques). Other criteria can be used: Marroquin [1985] has shown that the average surface f under the posterior distribution is often a better estimate, and one which can be obtained more efficiently by simply finding the average value of f at each lattice site.

One of the main attractions of MRFs is that the prior probability distribution can be made to embed more sophisticated assumptions about the world. Geman and Geman [1984] introduced the idea of another process, the line process, located on the dual lattice, and representing explicitly the presence or absence of discontinuities that break the smoothness assumption. The associated prior energy then becomes

$$U_C(f) = (f_i - f_j)^2(1 - l_i^j) + \beta V_C(l_i^j) \qquad (11)$$

where l is a binary line element between site i, j. V_C is a term that reflects the fact that certain configurations of the line process are more likely than others to occur. In our world, depth discontinuities are usually themselves continuous, non-intersecting, and rarely isolated joints. These properties of physical discontinuities can be enforced locally by defining an appropriate set of energy values $V_C(l)$ for different configurations of the line process in the neighborhood of the site (note that the assignment of zero energy values to the non-central cliques mentioned in Gamble and Poggio [1987] is wrong, as pointed out to us by Tal Symchony).

◇ Organization of integration

It is possible to extend the energy function of Equation (5) to accommodate the interaction of more processes and their discontinuities. In particular, we have extended the energy function to couple several of the early vision modules (depth, motion, texture, and color) to brightness edges in the image. This is a central point in our integration scheme; brightness edges guide the computation of discontinuities in the physical properties of the surface, thereby coupling surface depth, surface orientation, motion, texture, and color, each to the image brightness data and to each other. The reason for the role of brightness edges is that changes in surface properties usually produce large brightness gradients in the image. It is exactly for this reason that edge detection is so important in both artificial and biological vision.

The coupling to brightness edges may be done by replacing the term $V_C(l_i^j)$ in the last equation with the term

$$V(l, e) = g(e_i^j, V_C(l_i^j)) \qquad (12)$$

with e_i^j representing a measure of the presence of an brightness edge between site i, j. The term g has the effect of modifying the probability

of the line process configuration depending on the brightness edge data ($V(l, e) = -log\ p(l/e)$). This term facilitates formation of discontinuities (that is, l_i^j) at the locations of brightness edges. Ideally, the brightness edges (and the neighboring image properties) activate, with different probabilities, the different surface discontinuities (see figure 1), which in turn are coupled to the output of stereo, motion, color, texture, and possibly other early algorithms.

We have been using the MRF machinery with prior energies like that shown above (see also figure 1) to integrate edge brightness data with stereo, motion, and texture information on the MIT Vision Machine System.

We should emphasize that our present implementation represents a subset of the possible interactions shown in figure 1, itself only a simplified version of the organization of the likely integration process. The system will be improved in an incremental fashion, including pathways not shown in figure 1, such as feedback from the results of integration into the matching stage of the stereo and motion algorithms. Examples can be found in Poggio, Gamble and Little [1988] and in Poggio and The Staff [1987].

◇ Algorithms: Deterministic and stochastic

We have chosen to use MRF models because of their generality and theoretical attractiveness. This does not imply that stochastic algorithm must be used. For instance, in the cases in which the MRF model reduces to standard regularization [Marroquin 1987] and the data are given on a regular grid, the MRF formulation leads not only to a purely deterministic algorithm, but also to a convolution filter. Recent work in color [Hurlbert & Poggio 1989] shows that one can perform integration similar to the MRF-based scheme using a deterministic update. Geiger and Girosi [1989] have shown that there is a class of deterministic schemes that are the mean-field approximations of the MRF models. These schemes have a much higher speed than the Montecarlo schemes we used so far, while promising similar performance.

Recognition

The output of the integration stage provides a set of edges labeled in terms of physical discontinuities of the surface properties. They represent a good input to a model-based recognition algorithm like the ones described by Huttenlocher and Cass [1988]. In particular, we have interfaced the Vision Machine as implemented so far with the Cass algorithm. We have used only discontinuities for recognition; later we will also use the information provided by the MRFs on the surface properties *between* discontinuities.

We have more ambitious goals for the recognition stage of the Vision Machine. In an unconstrained environment the library of models that a system with human-level performance requires is in the order of many thousands. Thus, the ability to learn from examples appears to be essential for the achievement of high performance in real-world recognition tasks. Learning the models becomes then a primary concern in developing a recognition system for the Vision Machine. This has not been the case in other approaches of the last few years, mainly motivated by a robotic framework.

Learning in a three-stage recognition scheme

Although some of the existing recognition systems incorporate a module for learning object models from examples (for example Tucker's 2D system [Tucker *et al.* 1988]) no such capability exists yet for the more difficult problems of recognizing 3D objects [Huttenlocher & Ullman 1987] or handwriting [Edelman & Ullman to appear 1990]. We believe that incorporating learning into a general-purpose recognition system may be facilitated by breaking down the task of recognition into three distinct but interacting stages: *selection*, *indexing*, and *verification*.

Selection. Selection or segmentation breaks down the image into regions that are likely to correspond to single objects. The utility of an early segmentation of a scene into meaningful entities lies in the great reduction of complexity of scene interpretation. Each of the detected objects can in turn be subjected to separate recognition, by comparing it with object models stored in memory. Without prior segmentation, every possible combination of image primitives such as lines and blobs can in principle constitute an object and must be checked out. The power of early segmentation may be enhanced by integrating all available visual cues, especially if the integration parameters are automatically adjusted to suit the particular scene in question.

Indexing. By indexing we mean defining a small set of candidate objects that are likely to be present in the image. Although one cannot hope to achieve an ideal segmentation in real-world situations, partial success is sufficient if the indexing process is robust. Assuming that most objects in the real world are redundantly specified by their local features, a good indexing mechanism would use such features to overcome changes in viewpoint and illumination, occlusion and noise.

What kind of feature is good for indexing? Reliably detected lines provided by the integration of several low-level cues in the process of segmentation may suffice in many cases. We conjecture that *simple* viewpoint-invariant combinations of primitive elements, such as two lines forming a corner, parallel lines, and symmetry are also likely to be useful. Ideally, only 2D information should be used for indexing, although it may be augmented sometimes by qualitative 3D cues such as relative depth.

Verification. In the verification stage each of the candidates screened by the indexing process is tested to find the best match to the image. At this stage, the system can afford to perform complicated tests, because the number of candidate objects is small. We conjecture that hierarchical indexing by a small number (two or three) features that are spatially localized in 2D suffices to achieve useful interpretations of most everyday scenes. In general, however, further verification by model dependent routines (if it is a Mercedes it must have a three joint star in front) or precise shape matching, possibly involving 3D information, is required [Ullman 1884; Lowe 1986; Huttenlocher & Ullman 1987; Bolles *et al.* 1983; Ayache & Faugeras 1986; Tucker *et al.* 1988].

Future Developments

The Vision Machine will evolve in several parallel directions:

- Improvement and extensions of its early modules
- Improvement of the integration and recognition stages (recognition is discussed later)
- Use of the eye-head system in an active mode during recognition task by developing appropriate gaze strategies
- Use of the results of the integration stage in order to improve the operation of early modules such as stereo and motion by feeding back the preliminary computation of the discontinuities

Two goals will occupy most of our attention. The first one is the development of the overall organization of the Vision Machine. The system can be seen as an implementation of the *inverse optics* paradigm: it attempts to extract surface properties from the integration of image cues. It must be stressed that we never intended this framework to imply that precise surface properties such as dense, high resolution depth maps, must be delivered by the system. This extreme interpretation of inverse optics seems to be common, but was not the motivation of our project, which originally started with the name *Coarse Vision Machine* to emphasize the importance of computing qualitative, as opposed to very precise, properties of the environment.

Our second main goal in the Vision machine project will be Machine Learning. In particular, we have begun to explore simple learning and estimation techniques for vision tasks. We have succeeded in synthesizing a color algorithm from examples [Hurlbert & Poggio 1987] and in developing

a technique to perform unsupervised learning [Sanger 1988] of other simple vision algorithms such as simple versions of the computation of texture and stereo. In addition, we have used learning techniques to perform integration tasks, such as labeling the type of discontinuities in a scene. We have also begun to explore the connections between recent approaches to learning, such as neural networks, genetic algorithms, and classical methods in approximation theory such as splines, Bayesian techniques, and Markov Random Field models. We have identified some common properties of all these approaches and some of the common limitations, such as sample complexity. As a consequence, we now believe that we can leverage our expertise in approximation techniques for the problem of learning in machine vision. Our future theoretical and computational studies will examine available learning techniques, their properties and limitations and develop new ones for the tasks of early vision, for the integration stage and for object recognition. The algorithms will be tested with the Vision Machine system and eventually incorporated into it. We will also pay attention to parallel network implementations of these algorithms: for this subgoal we will be able to leverage the work we are now doing in developing analog VLSI networks for several of the components of the Vision Machine. Towards the goal of achieving much higher flexibility in the Vision Machine we propose to explore (a) the synthesis of vision algorithms from a set of instances and (b) the refinement and tuning of preprogrammed algorithms, such as edge detection, texture discrimination, motion, color and calibration for stereo. We will also develop techniques to estimate parameters of the integration stage. Much of our effort will be focused on the new scheme for visual recognition of 3D objects, whose key component is the automatic learning of a large database of models. We aim to develop a prototype of a flexible vision system that can, in a limited way, learn from experience.

In the following, we outline some of the other directions of future development.

Labeling the physical origin of edges: Computing qualitative surface attributes

◊ Physical discontinuities

We classify edges according to the following physical events: discontinuities in surface properties, called *mark* or *albedo* edges (for example, changes in the color of the surface); discontinuities in the orientation of the surface patch, called *orientation* edges (for example, an edge in a polyhedron); discontinuities in the illumination, called *shadow* edges; *occluding boundaries*, which are discontinuities in the object space (a different object); and *specular* discontinuities, which exist for non-Lambertian objects.

◊ Integration via labeling with a linear classifier

Gamble, Geiger, Weinshall and Poggio have implemented a part of the general scheme. More specifically, they have used a simple linear classifier to label edges at pixels where there exists an intensity discontinuity, using the output of the line process associated with each low-level vision module. They use the fact that the modules' discontinuities are aligned, having being integrated with the intensity edges before, so that the nonexistence of a module discontinuity at a pixel is meaningful. The linear classifier corresponds to a linear network where each output unit is a weighted linear combination of its inputs (for a similar application to a problem of color vision, see Hurlbert and Poggio [1987]). The input to the network is a pixel where there exists an intensity edge and that feeds a set of qualitatively different input units. The output is a real value vector of each labels' support.

In the system we have so far implemented, we achieve a rather restricted integration, because each module is integrated only with the intensity module, and labeling is done via a simple linear classifier only. It is still unclear how successful labeling can be, using only local information.

Saliency, grouping, and segmentation

A grouping and segmentation module working on the output of the edge detection module is an important part of a vision system: humans can deal with monocular, still, black and white pictures devoid of stereo, motion and color. We are now developing techniques to find salient edges, to group them and thereby segment the image. These algorithms have not been integrated yet in the Vision Machine system.

◊ Saliency measure

Edge maps produced by most current edge detectors are cluttered with edge responses and may have edges caused by noise. This creates difficulties for higher level processing, because the combinatorics of these algorithms often depends on the number of edge primitives being examined. What is needed is a technique to focus attention on the "important" edges in a scene. We call such attention focusing techniques that measure the "importance" of an edge saliency measures. Shimon Ullman [Ullman & Sha'ashua 1988] has proposed two different kinds of saliency measures: local saliency and structural saliency. An edge's local saliency is entirely determined by features of that edge alone. For example, an edge's length, its average gradient magnitude, or the color of a bounding region serve as local saliency measures.

Structural saliency refers to more global properties of an edge—its relationships with other edges. Although two edges may not be locally salient, if there is a "nonaccidental" relationship between them, then the structure becomes salient. Examples of "nonaccidental" relationships, as pointed out by David Lowe, include collinearity, parallelism, and symmetry, among other things.

We have investigated local saliency measures applied to the output of the Canny edge detector [Beymer 1989]. The edge features we have considered include curvature, edge length, and gradient magnitude. The measure favors those edges that have low average curvature, long length, and a high gradient magnitude. The saliency measure eliminates many of the edges due to noise and many of the unimportant edges. The edges that remain are often the long, smooth boundaries of objects and significant intensity changes inside the objects. We expect that the salient edges will help higher level processes such as grouping (structural saliency) and model based recognition by allowing them to focus attention on regions of an image bounded by salient edges.

◇ T junctions: Their detection and use in grouping

In cluttered imagery, imagery containing many objects occluding one another, it is important to group together pieces of the image that come from the same object. In particular, given an edge map produced by the Canny edge detector, we would like to select and group together the edges from a particular object before running high level recognition algorithms on the edge data. This grouping stage helps reduce the combinatorics of the higher level stages, as they are not forced to consider false edge groupings as objects. Considering how occlusion cues can be used in grouping, we have investigated the detection of T junctions and grouping rules arising from the pairing of T junctions. When one object partially occludes another in a cluttered scene, a T junction is formed between the two objects. Beymer has developed algorithms for detecting T junctions as a postprocessing step to the Canny edge detector. The Canny edge detector, while very good at detecting edges, is particularly bad at detecting junctions. Indeed, it was designed to detect one dimensional events. This one dimensional characterization of the image breaks down at junctions because locally there are three or more surfaces in the image. We have investigated how one could use edge curvature and region properties of the image to reconstruct these "broken" junctions. Often the way Canny will fail at junctions is that one of the three curves belonging to the junction will be broken off from the other two. We have modified an existing algorithm and achieved promising results in restoring broken T junctions.

Fast vision: The role of time smoothness

The present version of the Vision Machine processes only isolated frames. Even our motion algorithm takes as input simply a sequence of two images. The reason for this is, of course, limitations in raw speed. We cannot perform all of the processing we do at video rate (say, 30 frames per second), though this goal is certainly within present technological capabilities. If we could process frames at video rate, we could exploit constraints in the time dimension similar to the ones we are already exploiting in the space domain. Surfaces, and even the brightness array itself, do not usually change too much from frame to frame. This is a constraint of smoothness in time, which is valid almost everywhere, but not *across* discontinuities in time. Thus one may use the same MRF technique, applied to the output of stereo, motion, color, and texture, and enforce continuity in time (if there are no discontinuities), that is, exploit the redundancy in the sequence of frames.

We believe that the surface reconstructed from a stereo pair usually does not need to be recomputed completely when the next stereo pair is taken a fraction of a second later. Of course, the role of the MRFs may be accomplished in this case by some more specific and more efficient deterministic method such as, for example, a form of Kalman filtering. Note that space-time MRFs applied to the brightness arrays would yield spatiotemporal interpolation and approximation of a kind already considered [Fahle & Poggio 1980; Poggio, Nielsen & Nishihara 1982; Bliss 1985].

A VLSI Vision Machine?

Our Vision Machine consists mostly of specialized software running on the Connection Machine. This is a good system for the present stage of experimentation and development. Later, once we have perfected and tested the algorithms and the overall system, it will make sense to compile the software into silicon in order to produce a faster, cheaper, and smaller Vision Machine. We are presently planning to use VLSI technologies to develop some initial chips as a first step toward this goal. In this section, we will outline some thoughts about VLSI implementation of the Vision Machine.

Algorithms and Hardware. We realize that our specialized software vision algorithms are not, in general, optimized for hardware implementation. So, rather than directly "hardwiring algorithms" into standard computing circuitry, we will be investigating "algorithmic hardware" designs that utilize the local, symmetric nature of early vision problems. This will be an iterative process, as the algorithm influences the hardware design and as hardware constraints modify the algorithm.

Degree of Parallelism. Typical vision tasks require tremendous amounts of computing power, and are usually parallel in nature. As an example, biological vision uses highly parallel networks of relatively slow components to achieve sophisticated systems. However, when implementing our algorithms in silicon integrated circuits, it is not clear what level of parallelism is necessary. While biology is able to use three dimensions to construct highly interconnected parallel networks, VLSI is limited to $2\frac{1}{2}$ dimensions, making highly parallel networks much more difficult and costly to implement. However, the electrical components of silicon integrated circuits are approximately four orders of magnitude faster than the electrochemical components of biology. This suggests that pipelined processing or other methods of time-sharing computing power may be able to compensate for the lower degree of connectivity of silicon VLSI. Clearly, the architecture of a VLSI vision system may not resemble any biological vision systems.

Signal Representation. Within the integrated circuit, the image data may be represented as a digital word or an analog value. While the advantages of digital computation are its accuracy and speed, digital circuits do not have as high a degree of functionality per device as analog circuits. Therefore, analog circuits should allow much denser computing networks. This is particularly important for the integration of computational circuitry and photosensors, which will help to alleviate the I/O bottleneck typically experienced whenever image data are serially transferred between Vision Machine components. However, analog circuits are limited in accuracy, and are difficult to characterize and design.

Learning and parameter estimation

Using the MRF model involves an energy function which has several free parameters, in addition to the many possible neighborhood systems. The values of these parameters determine a distribution over the configuration-space to which the system converges, and the speed of convergence. Thus rigorous methods for estimating these parameters are essential for the practical success of the method and for meaningful results. In some cases, parameters can be learned from the data: for example, texture parameters [Geman & Graffigne 1987], or neighborhood parameters (for which a cellular automaton model may be the most convenient for the purpose of learning). There are general statistical methods which can be used for parameter estimation:

- A maximum likelihood estimate—one can use the indirect iterative EM algorithm [Dempster *et al.* 1977], which is most useful for maximum likelihood estimation from incomplete data (see Marroquin [1987] for a special case). This algorithm involves the iterative maximization (over the parameter space) of the expected value of the likelihood function given that the parameters take the values of their estimation in the previous iteration. Alternatively, a search constrained by some statistics for a minimum of an appropriate merit function may be employed (see Marroquin [1987]).
- A smoothing (regularization) parameter can be estimated using the methods of cross-validation or unbiased risk, to minimize the mean square error. In cross-validation, an estimate is obtained omitting one data point. The goal is to minimize the distance between the predicted data point (from the estimate above with the point omitted) and the actual value, for all points.

In the case of Markov Random Fields, some more specific approaches are appropriate for parameter estimation:

- Besag [1972] suggested conditional maximum likelihood estimation using coding methods, maximum likelihood estimation with unilateral approximations on the rectangular lattice, or "maximum pseudolikelihood"—a method to estimate parameters for homogeneous random fields (see Geman and Graffigne [1987]).
- For the MPM estimator, where a fixed temperature is yet another parameter to be estimated, one can try to use the physics behind the model to find a temperature with as little disorder as possible and still reasonable time of convergence to equilibrium (for example, away from "phase-transition").

An alternative asymptotic approach can be used with smoothing (regularization) terms: instead of estimating the smoothing parameter, let it tend to 0 as the temperature tends to 0, to reduce the smoothing close to the final configuration (see Geman and Geman [1987]).

In summary, we plan to explore three distinct stages for parameter estimation in the integration stage of the Vision Machine:

- **Modeling** from the physics of surfaces, of the imaging process and of the class of scenes to be analyzed and the tasks to be performed The range of allowed parameter values may also be established at this stage (for example, minimum and maximum brightness value in a scene, depth differences, positivity of certain measurements, distribution of expected velocities, reflectance properties, characteristics of the illuminant, etc.).
- **Estimating** of parameter values from a set of examples in which data and desired solution are given. This is a *learning stage.* We may have to use days of CM time and, at least initially, synthetic images to do this.
- **Tuning** of some of the parameters directly from the data (by using EM algorithm, cross-validation, Besag's work, or various types of heuristics).

The dream is that at some point in the future the Vision Machine will run all the time, day and night, looking about and learning on its own to see better and better.

References

Barrow, H. G., and J. M. Tenenbaum [1978], "Recovering Intrinsic Scene Characteristics from Images," in *Computer Vision Systems*, edited by A. Hanson, and E. Riseman, Academic Press, New York.

Bertero, M., T. Poggio, and V. Torre [1988], "Ill-Posed Problems in Early Vision," Report AIM-924 Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Besag, J. [1972], "Spatial Interaction and the Statistical Analysis of Lattice systems," *J. Roy. Stat. Soc.*, vol. B34, pp. 75-83.

Beymer, David [1989], "Junctions: their detection and use for grouping in images," Massachusetts Institute of Technology, (in press).

Blake, A [1986], "On the Geometric Information Obtainable from Simultaneous Observation of Stereo Contour and Shading," Technical Report CSR-205-86, Dept. of Computer Science, University of Edinburgh.

Blelloch, G. E. [1987], "Scans as Primitive Parallel Operations," *Proc. Intl. Conf. on Parallel Processing*, pp. 355-362.

Bliss, J. [1985] "Velocity Tuned Spatio-Temporal Interpolation and Approximation in Vision," M.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Brooks, R. [1987], "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, pp. 14-23.

Bülthoff H., and H. Mallot [1987], "Interaction of Different Modules in Depth Perception," *Proc. First Intl. Conf. on Computer Vision*, Computer Society of the IEEE, Washington, DC, pp. 295-305.

Bülthoff, H., and H. Mallot [1987], "Interaction of Different Modules in Depth Perception: Stereo and Shading," Report AIM-965, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Bülthoff, Heinrich H., and Hamspeter A. Mallot [1988], "Integration of depth modules: stereo and shading," *J. Opt. Soc. Am.*, vol. 5, pp. 1749-1758.

Bülthoff , Heinrich H. [1988], personal communication.

Bülthoff, Heinrich H., James J. Little, and Tomaso Poggio [1989], "A parallel algorithm for real-time computation of optical flow," *Nature*, vol. 337, pp. 549-553.

Canny, J. F. [1983], "Finding Edges and Lines," Report AI-TR-720, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Cornog, K. H. [1985], "Smooth Pursuit and Fixation for Robot Vision," M.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Dempster, A. P., N. M. Laird, and D. B. Rubin [1977], "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Roy. Stat. Soc.*, vol. B39, pp. 1-38.

Drumheller, M., and T. Poggio [1986], "On Parallel Stereo," *Proc. Intl. Conf. on Robotics and Automation*, IEEE.

Fahle, M., and T. Poggio [1980], "Visual Hyperacuity: Spatiotemporal Interpolation in Human Vision," *Proc. Roy. Soc. Lond. B*, vol. 213, pp. 451-477.

Geman, D., and S. Geman [1987], "Relaxation and Annealing with Constraints," *Complex Systems Technical Report 35*, Division of Applied Mathematics, Brown University, Providence, RI.

Geman, S., and D. Geman [1984], "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 6.

Geman, S., and C. Graffigne [1987], "Markov Random Field Image Models and their Applications to Computer Vision," *Proc. Intl. Congress of Mathematicians*, preprint, edited by A. M. Gleason.

Gamble, E., and T. Poggio [1987], "Integration of Intensity Edges with Stereo and Motion," Report AIm-970, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Grimson, W. E. L. [1981], *From Images to Surfaces*, The MIT Press, Cambridge, MA.

Grimson, W. E. L. [1982], "A Computational Theory of Visual Surface Interpolation," *Phil. Trans. Roy. Soc. Lond. B*, vol. 298, pp. 395-427.

Grimson, W. E. L. [1984], "Binocular Shading and Visual Surface Reconstruction," *Computer Vision, Graphics and Image Processing*, vol. 28, pp. 19-43.

Hildreth, E. C. [1983], *The Measurement of Visual Motion*, The MIT Press, Cambridge, MA.

Hillis, D. [1985], "The Connection Machine," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P. [1986], *Robot Vision*, The MIT Press, Cambridge, MA.

Hurlbert, A. [1989], "The Computation of Color Vision," Ph.D. Thesis, Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA.

Hurlbert, A., and T. Poggio [1986], "Do Computers Need Attention?" *Nature*, vol. 321, p. 12.

Hurlbert, A., and T. Poggio [1987] "Learning a Color Algorithm from Examples," Report AIM-909, Artificial Intelligence Laboratory *Center for Biological Information Processing Paper 25*, Massachusetts Institute of Technology, Cambridge, MA.

Huttenlocher, D., and S. Ullman [1987], "Recognizing Rigid Objects by Aligning them with an Image," Report AIM-937, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Huttenlocher, D., and T. Cass [1988], Proceedings of the Image Understanding Workshop.

Ikeuchi, K., and B. K. P. Horn [1981], "Numerical Shape from Shading and Occluding Boundaries" *Artificial Intelligence*, vol. 17, pp. 141-184.

Kender, J. R. [1979], "Shape from Texture: An Aggregation Transform that Maps a Class of Textures into Surface Orientation," *Proc. Sixth Intl. Joint Conf. on Artificial Intelligence*, Tokyo.

Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi [1983], "Optimization by Simulated Annealing," *Science*, vol. 220.

Kruskal, C. P., L. Rudolph, and M. Snir [1985], "The Power of Parallel Prefix," *Proc. Intl. Conf. on Parallel Processing*, pp. 180-185.

Lim, W. [1986] "Fast Algorithms for Labelling Connected Components in 2D Arrays," *Thinking Machines Corp.Technical Report NA86-1*, Cambridge, MA.

Little, J., G. E. Blelloch, and T. Cass [1987], "Parallel Algorithms for Computer Vision on the Connection Machine," *Proceedings Intl. Conf. on Computer Vision*, Los Angeles, pp. 587-591.

Little, J., H. Bülthoff, and T. Poggio [1987], "Parallel Optical Flow Computation," *Proc. Image Understanding Workshop*, edited by L. Bauman, Science Applications International Corp., McLean, VA, pp. 915-920.

Little, J., H. Bülthoff, and T. Poggio [in preparation], "Parallel Optical Flow Using Winner-Take-All Scheme," 1989.

Mahoney, J. V. [1987], "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," M.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA. Published as AI-TR-980, Artificial Intelligence Laboratory , Technical Report , Cambridge, MA.

Marr, D. [1982], *Vision*, Freeman, San Francisco.

Marr, D., and E. Hildreth [1980], "Theory of Edge Detection," *Proc. Roy. Soc. Lond. B*, vol. 207, pp. 187-217.

Marr, D., and T. Poggio [1976], "Cooperative Computation of Stereo Disparity," *Science*, vol. 194, pp. 283-287.

Marr, D., and T. Poggio [1979], "A Computational Theory of Human Stereo Vision," *Proc. Roy. Soc. Lond. B*, vol. 204, pp. 301-328.

Marroquin, J. L. [1987] "Deterministic Bayesian Estimation of Markov Random Fields with Applications to Computational Vision," *Proc. First Intl. Conf. on Computer Vision*, Computer Society of the IEEE, Washington, DC.

Marroquin, J. L. [1985], "Probabilistic Solutions of Inverse Problems," Report AI-TR-860, Artificial Intelligence Laboratory Technical Report , Massachusetts Institute of Technology, Cambridge, MA.

Marroquin, J. L. [1984], "Surface Reconstruction Preserving Discontinuities," Report AIM-792, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Marroquin, J. L., S. Mitter, and T. Poggio [1986], "Probabilistic Solution of Ill-Posed Problems in Computational Vision," *Proc. Image Understanding Workshop*, edited by L. Bauman, Scientific Applications International Corp., McLean, VA, 1986. A more complete version appears in *J. Amer. Stat. Assoc.*, vol. 82, pp. 76-89, 1987.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller [1953], "Equation of State Calculations by Fast Computing Machines," *J. Phys. Chem.*, vol. 21.

Nishihara, H. K. [1984]. "PRISM: A Practical Real-Time Imaging Stereo Matcher," Report AIM-780, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Nishihara, H. K., and P. A. Crossley [1988], "Measuring Photolithographic Overlay Accuracy and Critical Dimensions by Correlating inarized Laplacian of Gaussian Convolutions," *IEEE Trans. Pattern Matching and Machine Intell.*, vol. 10.

Poggio, T., K. R. K. Nielsen, and H. K. Nishihara [1982], "Zero-Crossings and Spatiotemporal Interpolation in Vision: Aliasing and Electrical Coupling Between Sensors," Report AIM-675, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Poggio, G., and T. Poggio [1984], "The analysis of stereopsis," *Ann. Rev. Neurosci.*, vol. 7, pp. 379-412.

Poggio, T. [1985], "Early Vision: From Computational Structure to Algorithms and Parallel Hardware," *Computer Vision, Graphics, and Image Processing*, vol. 31.

Poggio, T. [1985], "Integrating Vision Modules with Coupled MRFs," Report AIW-285, Artificial Intelligence Laboratory Working Paper , Massachusetts Institute of Technology, Cambridge, MA.

Poggio, T., and staff [1985], "MIT Progress in Understanding Images," *Proc. Image Understanding Workshop*, edited by L. Bauman, Scientific Applications International Corp., McLean, VA.

Poggio T., and staff [1987], "MIT Progress in Understanding Images," *Proc. Image Understanding Workshop*, edited by L. Bauman, Scientific Applications International Corp., McLean, VA.

Poggio, T., H. L. Voorhees, and A. L. Yuille [1984], "Regularizing Edge Detection," Report AIM-776, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Poggio, T., V. Torre, and C. Koch [1985], "Computational Vision and Regularization Theory," *Nature*, vol. 317, pp. 314-319.

Richards, W., and D. D. Hoffman [1985], "Codon Constraints on Closed 2D Shapes," *Computer Vision, Graphics, and Image Processing*, vol. 32, pp. 265-281.

Reichardt W., and T. Poggio [1976], "Visual Control of Orientation in the Fly: I. A Quantitative Analysis," *Quart. Rev. Biophysics*, vol. 3, pp. 311-375.

Reichardt W., and T. Poggio [1976], "Visual Control of Orientation in the Fly: II. Towards the Underlying Neural Interactions," *Quart. Rev. Biophysics*, vol. 9, pp. 377-439.

Rock, I. [1973], *Orientation and Form*, Academic Press, New York.

Terzopoulos, D. [1986], "Integrating Visual Information From Multiple Sources," in *From Pixels to Predicates*, edited by A. P. Pentland, Ablex Publishing Corp., Norwood, NJ.

Tikhonov, A. N., and V. Y. Arsenin [1977], *Solution of Ill-Posed Problems*, Winston and Wiley Publishers, Washington, DC.

Torre, V., and T. Poggio [1984], "On Edge Detection," Report AIM-768, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Ullman S. [1979], *The Interpretation of Visual Motion*, The MIT Press, Cambridge, MA.

Ullman, S. [1984], "Visual Routines," *Cognition*, vol. 18.

Verri, A., and T. Poggio [1986], "Motion Field and Optical Flow: Qualitative Properties," Report AIM-917, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge, MA.

Voorhees, H. L., and T. Poggio [1987], "Detecting Textons and Texture Boundaries in Natural Images," *Proc Intl. Conf. on Computer Vision*, Computer Society of the IEEE, Washington, DC.

Waxman, A. [1987], "Image Flow Theory: A Framework for 3-D Inference from Time-Varying Imagery," in *Advances in Computer Vision*, edited by C. Brown, Lawrence Erlbaum Assocs, NJ.

Wyllie, J. C. [1979], "The Complexity of Parallel Computations," Technical Report pp. 79-387, Department of Computer Science, Cornell University, Ithaca, NY.