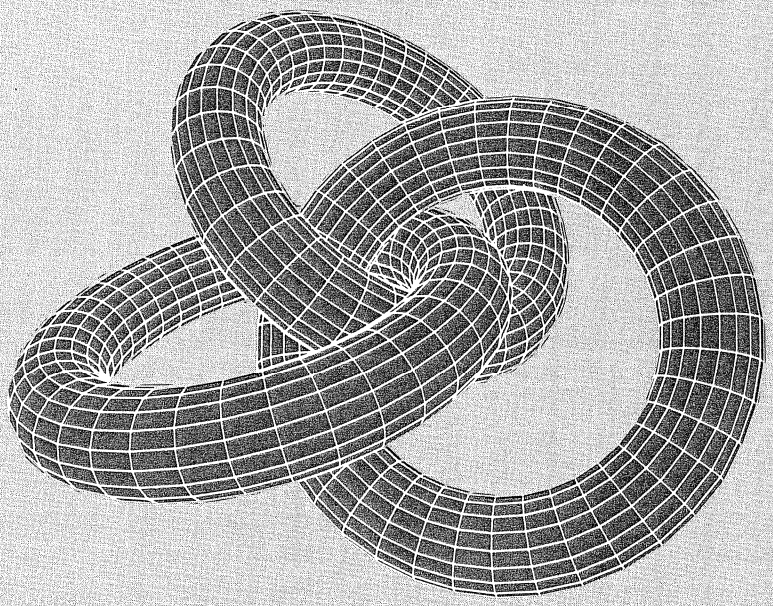
THE INTERNATIONAL JOURNAL OF

SUPERCOMPUTER APPLICATIONS

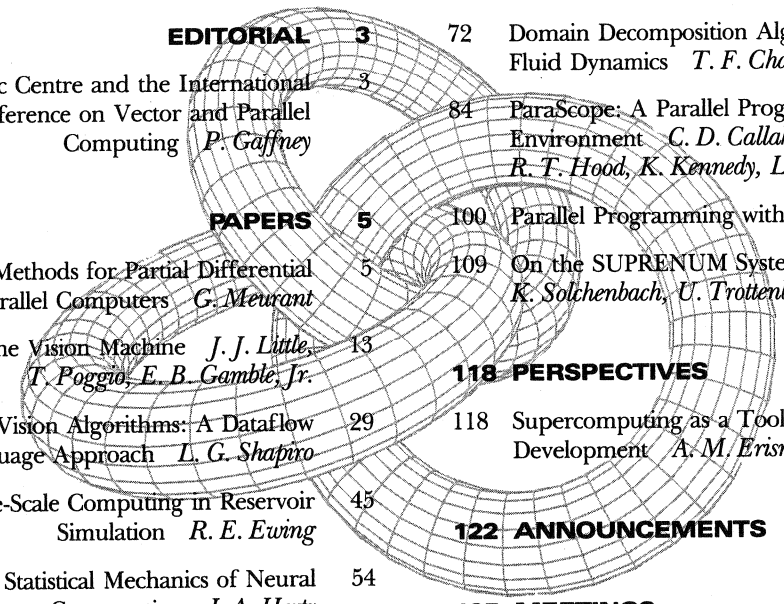


VOLUME 2 NUMBER 4 WINTER 1988

ISSN 0890-2720

THE INTERNATIONAL JOURNAL OF

SUPERCOMPUTER APPLICATIONS



VOLUME 2 NUMBER 4 WINTER 1988

ISSN 0890-2720

SEEING IN PARALLEL: THE VISION MACHINE

James J. Little, Tomaso Poggio, and Edward B. Gamble, Jr.

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS 02139

Summary

Computer algorithms have been developed for early vision processes that give separate cues to the distance from the viewer of three-dimensional surfaces, their shape, and their material properties. The MIT Vision Machine is a computer system that integrates several early vision modules to achieve high-performance recognition and navigation in unstructured environments. It is also an experimental environment for theoretical progress in early vision algorithms, their parallel implementation, and their integration. The Vision Machine consists of a movable, two-camera Eye-Head input device and an 8K Connection Machine. We have developed and implemented several parallel early vision algorithms that compute edge detection, stereopsis, motion, texture, and surface color in close to real time. The integration stage, based on coupled Markov random field models, leads to a cartoon-like map of the discontinuities in the scene, with partial labeling of the brightness edges in terms of their physical origin.

The International Journal of Supercomputer Applications, Volume 2, No. 4, Winter 1988, pp. 13–28.

Introduction

It is increasingly clear that one of the keys to the reliability, flexibility, and robustness of biological vision systems in unconstrained environments is their ability to integrate several visual cues. For this reason we are developing the Vision Machine system to explore the issue of the integration of early vision modules. The system also serves the purpose of developing parallel vision algorithms, since its main computational engine is a parallel supercomputer—the Connection Machine (Hillis, 1985).

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 suggests 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 perfect. 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 (Poggio, 1985).

There are several different approaches to the problem of how to integrate visual cues (Brooks, 1987; Ullman, 1984; Hurlbert and Poggio, 1986; Mahoney, 1986). Our approach (Gamble and Poggio, 1987, Poggio et al., 1988) assumes that 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 2½-dimensional sketch (Marr, 1982), 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.

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 pro-

cessing: it is a laboratory for our theories and algorithms.

In this paper we describe the Vision Machine system and review the present hardware of the Vision Machine: the Eye-Head system and the Connection Machine. We introduce the vector model, a model of computation for fine-grain machines such as the Connection Machine, and 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 is discussed. Finally, we analyze some results and illustrate the merits and the pitfalls of our present system.

1. THE VISION MACHINE SYSTEM

The overall organization of the system is shown in Figure 1. The image or images are processed through independent algorithms or modules corresponding to different visual cues, in parallel. Edges are extracted using Canny's edge detector (Canny, 1986). Stereo (Drumheller and Poggio, 1986) computes disparity from the left and right images. The motion module (Little, Bülthoff, and Poggio, 1988) estimates 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 and Poggio, 1988). 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 Poggio and staff (1985).

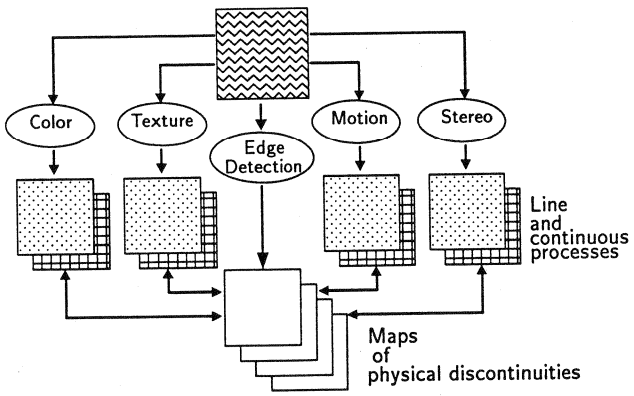


Fig. 1 Overall organization of the integration stage The output of each early visual cue (or algorithm)—stereo, motion, texture, and color—is coupled to its own line process (the crosses), that is, its discontinuities. Each is also coupled to the discontinuities in the surface properties—occluding edges (both extremal edges and blades), orientation discontinuities, specular edges, texture marks (including albedo discontinuities), shadow edges. The image data—and especially the sharp changes in brightness labeled here as edges—are input to the lattices that represent the discontinuities in the physical properties of the surfaces. Before integration the brightness edges may be completed (in some cases this may lead to "subjective contours") by the equivalent of a higher order MRF that reflects long-range constraints of colinearity and continuation and even hypotheses from the recognition stage, supposed to use the set of discontinuities at the bottom as its main input. Our present implementation does not distinguish the different types of physical discontinuities: sharp changes in brightness are directly coupled to the line processes of each of the cues. The individual modules are therefore integrated with each other only indirectly, through the brightness edges.

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 a stage of approximation and restoration of data, performed by 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), shadow edges—and coupling them with each other and back to the discontinuities in the visual cues, as illustrated in Figure 1. So far we have implemented only the coupling of brightness edges to each of the cues provided by the early algorithm. As we discuss below, the technique we use to 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. In our implemented version of the system we find discontinuities in disparity, motion, texture, and color. These discontinuities, taken together, represent a "cartoon" of the original scene which can be used for

recognition and navigation (along with, if needed, interpolated depth, motion, texture, and color fields).

2. EYE-HEAD SYSTEM

The Eye-Head system consists of two cameras ("eyes") mounted on a variable-attitude platform ("head"). The apparatus allows the cameras to be moved as a unit, analogous to head movement. It also allows the lines of sight of the cameras to be pointed independently, analogous to eye movement. Each camera is equipped with a motorized zoom lens (*F*1.4, focal length from 12.5 to 75 mm), allowing control of the iris, focus, and focal length by the host computer (currently a Symbolics 3645 Lisp Machine). Other hardware allows for repeatable calibration of the entire apparatus.

3. THE COMPUTATIONAL ENGINE: THE CONNECTION MACHINE

The Connection Machine (Hillis, 1985) is a fine-grained, single instruction, multiple data (SIMD) parallel computer with between 8K and 64K processors (see Fig. 2). Each processor is a one-bit serial processor, with 64K bits of local memory, an 8 MHz clock, and optional floating point hardware. All the processors are controlled by a microcontroller, which receives macro instructions from a front-end computer. The processors are organized into a hypercube network with 16 processors at each node (corner) of the hypercube. The wires of the hypercube network support the communication primitives discussed in the following sections.

3.1 VECTOR MODEL

Early vision modules have been implemented using a set of primitive parallel operations, including general permutations, grid permutations, and the scan operation—a restricted form of the prefix computation.

The model we use to describe the algorithms is a vector model of computation (Blelloch, 1988). In a vector model all the primitive operations are defined to work on a vector of values. For example, we might have a primitive that elementwise adds two vectors, or a primitive that sorts a vector. A vector model is defined in terms of a set of primitive operations that require vectors as input and return vectors as output. A vector is

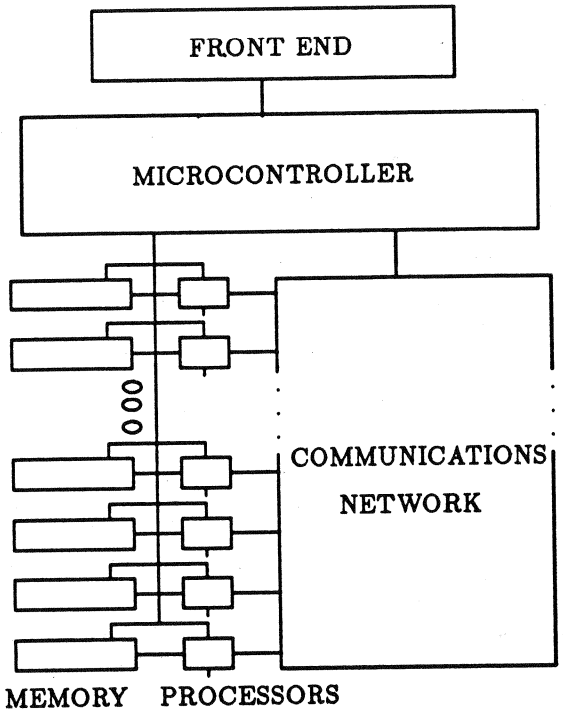


Fig. 2 Block diagram of the Connection Machine

an ordered set of simple values, such as integers, floating point, or Boolean values. Each element of a vector has an index.

Each elementwise primitive operates on equal-length vectors, producing a result vector of equal length. The element i of the result is an elementary arithmetic or logical primitive (such as $+$, $-$, $*$, **or** and **not**) applied to element i of each of the input vectors.

The *permutation* primitive takes two vector arguments: a *data vector* and an *index vector*. The permutation primitive permutes each element in the data vector to the location specified in the index vector. It is an error for more than one element to have the same index—the permutation must be one-to-one. An example of a permutation:

Index	= [0 1 2 3 4 5 6 7 8 9]
Data	= [g r o t i s h m a l]
I	= [2 4 3 6 5 9 7 8 0 1]
permute(Data, I)	= [a l g o r i t h m s]

The *combining* primitives also take a data vector and an index vector, but they allow many-to-one mappings: the indices need not be unique. Values with the same index are combined using a binary associative operator. Valid binary operations include +, **maximum, minimum,** and **or,** generating combine operations **+-combine, max-combine, min-combine,** and **or-combine.** The following example diagrams some combining primitives:

```
Index                    = [0 1 2 3 4 5 6 7]
Data                     = [5 1 3 4 3 9 2 6]
I                        = [2 5 4 3 1 6 3 5]

+-combine(Data, I)       = [0 3 5 6 3 7 9 0]
max-combine(Data, I)     = [0 3 5 4 3 6 9 0]
```

To allow communication between vectors of different sizes, we include a version of the **permute** primitive that returns a vector of different length than the source vectors. This version takes two extra arguments: one that specifies the length of the destination, and another vector that specifies which elements appear in the result.

A grid permutation maps a vector onto a grid and permutes elements to the closest neighbor in some direction on the grid.

3.2 SCAN PRIMITIVES

The scan primitives execute a scan operation, sometimes called a prefix computation (Ladner and Fischer, 1980; Kruskal, Rudolph, and Snir, 1985), on a vector. The scan operation takes a binary associative operator $\oplus$, and a vector $[a_0, a_1, \ldots, a_{n-1}]$ of n elements, and returns the vector $[a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-1})]$. The operators for the scan primitives include **maximum, minimum, or,** and **and,** generating scan operations termed **+-scan, max-scan, min-scan, or-scan,** and **and-scan.** Some examples:

```
A                = [5 1 3  4  3  9  2  6]
+-scan(A)        = [5 6 9 13 16 25 27 33]
max-scan(A)      = [5 5 5  5  5  9  9  9]
```

The grid scan primitives are analogous to the vector scans but operate on a grid ordering instead of the vector ordering: each column or row can be viewed as a vector.

3.3 GLOBAL PRIMITIVES

The global primitives reduce the values in a vector using a binary associative operator. As with the scan and combine primitives, we only use the operators +, **maximum, minimum, or, and,** and **first.** Some examples:

```
A                = [5 1 3 4 3 9 2 6]
+-reduce(A)      = 33
max-reduce(A)    = 9
```

The global primitive can be implemented with either a scan or a combine operation.

3.4 SEGMENTED PRIMITIVES

A vector can be broken into contiguous segments in which the beginning of each segment is marked with a flag. For example:

```
A                = [5  1  3  4  3  9  2  6]
Segment-Flags    = [T  F  T  F  F  F  T  F]
                   [5  1] [3  4  3  9] [2  6]
```

We can define segmented versions of both the permutation primitives and the scan primitives that work independently within each segment. A description of the use of segments can be found in Blelloch (1988).

3.5 PRIMITIVES ON THE CONNECTION MACHINE

All primitives just described are implemented on the Connection Machine. The wires of the hypercube in the Connection Machine are shared by the grid permutation primitives, the scan primitives, and the permutation and combine primitives.

The Connection Machine (CM) supports a virtual vector machine by distributing the elements of a vector across the processors. When using a vector of length m on an n processor CM, $[m/n]$ vector elements are placed in the memory of each processor. Each processor is responsible for the vector elements in its memory. The

mapping of vectors onto processors is supported in microcode and is transparent to the user.

The elementwise vector primitives are implemented on the CM by loading an element of each argument vector from the memory of each processor into the processor, operating on these elements, and storing the result back to memory. Elementwise primitives never require communication among processors. When there are many elements per processor, each processor loops over all of its elements.

The vector permutation primitives and the vector combining primitives are supported by the routing hardware of the Connection Machine. The router uses a packet-switched message-routing scheme that directs messages along the hypercube wires to their destinations. The combine primitives are supported with combining router switches similar to those suggested for the NYU Ultracomputer (Gottlieb, Lubachevsky, and Rudolph, 1983).

Vector permutations, which take $O(1)$ time in the vector model, are simulated on the Connection Machine in $O(\log n)$ time with high probability. As with the elementwise primitives, when there are more vector elements than processors, the CM loops over the elements in each processor.

The grid permutations are supported on the CM using the same hypercube wires as the router. Each dimension of a grid is placed in gray code order with respect to the hypercube ordering, so that neighboring grid elements are separated by a single wire. Because of this ordering, the grid permutation primitives are significantly faster than the general permutation primitive.

The scan vector primitives are implemented on the CM using a binary tree algorithm. This algorithm uses the same wires as the router but does not use the routing hardware. The scan primitives take $O(\log n)$ time and, in practice, are usually faster than the general permutation primitive. Grid scans are implemented in a similar way: a tree is used for each row or each column.

3.6 ROUTINES AND ALGORITHMS IN THE VECTOR MODEL

The primitives described above simplify implementing more complex manipulations of data structures. We have devised a set of routines, functions composed of the various primitives, that themselves form important building blocks for vision algorithms. These routines include pointer jumping, ordering, region summation, outer product, and histogram computation. More details on the structure of the primitives, the routines, and their use in general early and middle vision algorithms can be found in Little, Blelloch, and Cass (1987a, 1987b, 1989).

For example, one simple routine, termed *region summation,* uses grid scans. Region summation sums, at each pixel in a grid, a $(2m + 1) \times (2m + 1)$ square region around the pixel. This procedure can be implemented using a constant number of grid scans and permutations in the vector model. Several iterative local operations, such as boxcar convolution, can be efficiently implemented using region summation; the stereo and motion modules also use region summation.

In early visual processing, the image maps directly onto the grid coordinate system in the vector model. This processing involves many operations that use either elementwise parallel or grid operations, where the computation at each pixel depends only on points adjacent in the grid. The classic example is filtering and convolution.

The data structures of middle vision are dependent on image content, for example, linked edges, and therefore usually depend more on arbitrary communication patterns. Computation of image features often requires communication patterns that follow arbitrary paths through the image grid, depending on local image structure. To link edge elements, for example, a procedure must access elements in a direction perpendicular to the image brightness gradient. Permutation operations fit this task well by facilitating efficient algorithms based on pointer jumping. High-level vision processes, such as recognition, utilize more abstract representations that do not necessarily operate in the image coordinate system. They typically use permutation and scan primitives.

4. EARLY VISION MODULES

4.1 STEREO

The parallel stereo algorithm of Drumheller and Poggio (1986) is implemented in the Vision Machine. Disparity

data produced by the algorithm constitutes one of the inputs to the MRF-based integration stage of the Vision Machine. The stereo algorithm runs on the Connection Machine system with good results on natural scenes in times that are typically on the order of 1 second.

Stereo matching is an ill-posed problem (Bertero, Poggio, and Torre, 1987) that cannot be solved without taking advantage of natural constraints. 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 that may, however, be a satisfactory *local* approximation in many cases (it can be extended to more general surface assumptions in a straightforward way but at high computational cost). Let $E_L(x,y)$ and $E_R(x,y)$ represent the left and the right images 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

$$\|E_L(x,y) - E_R(x + d(x,y),y)\|_{P(x,y)} \quad (1)$$

where the norm is a summation over a local neighborhood patch, $P_{(x,y)}$, centered at each location (x,y); $d(x,y)$ is assumed constant in the neighborhood. Equation 1 implies that we should look at each (x,y) for $d(x,y)$ such that

$$\int_{P(x,y)} (E_L(x,y)E_R(x + d(x,y),y))^2 dxdy \quad (2)$$

is maximized.

The algorithm that we have implemented on the Connection Machine is actually somewhat more complicated, since it involves geometric constraints that affect the way the maximum operation is performed (Drumheller and Poggio, 1986). The implementation currently used in the Vision Machine at the MIT Artificial Intelligence Laboratory uses maps of Canny's edges (Canny, 1986) 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.

The computational requirements of the stereo algorithm are essentially local. Feature computation is elementwise, or depends only on filtering, which is local. The stereo matching problem is one-dimensional (when epipolar lines coincide with image rows); matches are found while sliding the right image over the left image horizontally, computing a set of potential match planes, one for each horizontal disparity. To determine the degree of continuity around each potential match, we compute a local support score, using region summation. Finally, we select matches by comparing the local match sums, over a set of possible matches determined by geometric constraints.

4.2 MOTION

Computing the displacement in the image of moving elements is very similar to stereo computation, but the range of displacements is two-dimensional, not one-dimensional. We use a parallel motion algorithm (Little et al., 1988) modeled on the Drumheller-Poggio stereo algorithm. The algorithm produces dense displacement fields from the moving images, together with direct cues for the location of occluding boundaries of scene objects. Its computational requirements are similar to those of stereopsis, although necessarily higher, since the number of displacements considered can be larger.

4.3 TEXTURE

The texture algorithm is a greatly simplified parallel version of the texture algorithm developed by Voorhees and Poggio (1988). It measures the level density of "blobs" extracted from the image through a filtering process involving center-surround filters with appropriate size and threshold. It utilizes only filtering operations to extract features, "blobs," and region summation operations to estimate densities.

4.4 COLOR

The color algorithm provides a local measure of hue, $H = (R_{i,j})/(R_{i,j} + G_{i,j})$, where R and G are the measurements in the red and green channels, respectively, of a digital color camera. Under certain conditions (Hurlbert, see Poggio and staff, 1985), this ratio is independent of illumination and three-dimensional shape. The color computation uses only local computations.

5. INTEGRATION

It is reasonable to assume that combining the evidence provided by multiple visual cues—for example, edge detection, stereo, and color—should provide a more reliable map of the objects in a visual scene than any single cue alone, but it is not obvious how to accomplish this integration. One of the most important constraints for recovering surface properties from each of the individual cues is that the physical processes underlying image formation, such as depth, orientation, and reflectance of the surfaces, change slowly in space (adjacent points on a surface are not at random depths, for instance). Standard regularization (Poggio and Torre, 1984; Bertero, Poggio, and Torre, 1987; Poggio, Torre, and Koch, 1985), on which many examples of the early vision algorithms are based, captures these smoothness properties well. The physical properties of surfaces, however, are smooth *almost* everywhere, but not at discontinuities. Reliable detection of discontinuities of the physical properties of surfaces is critical for a vision system, since discontinuities are often the most important locations in a scene: depth discontinuities, for example, normally correspond to the boundaries of an object. Thus, the output of each vision module has to be *smoothed* and interpolated (that is, "filled-in"), since it is noisy and often sparse; at the same time discontinuities must be simultaneously detected.

Discontinuities can also be used effectively to fuse information between different visual cues (Poggio, 1985; Gamble and Poggio, 1987; Hutchinson et al., 1988; Poggio and staff, 1987) and the image data (see also Chou and Brown, 1987a, 1987b, 1988). For instance, a depth discontinuity usually produces a sharp change of brightness in the image (usually called a brightness *edge*); and a motion boundary often corresponds to a depth discontinuity (and a brightness edge) in the image. The idea is thus to couple different cues—stereo, motion, texture, and color—to the image data (in particular to the sharp changes of brightness in the image) through the discontinuities in the physical properties of the surfaces (see Fig. 1) (Barrow and Tenenbaum, 1978). The final goal of this approach is to use information from several cues simultaneously to refine the initial estimation of surface discontinuities. We will describe a first step in this direction that combines brightness edges with discontinuities in each of the modules separately.

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). (For alternative approaches, see Hoff and Ahuja, 1987; Blake and Zisserman, 1987; Aloimonos and Brown, 1987; Cohen and Cooper, 1983.) Consider the prototypical problem of approximating a surface given sparse and noisy data (depth data), on a regular two-dimensional lattice of sites (Fig. 3). We first define the prior probability of the class of surfaces in which we are interested. 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 has the Gibbs form:

$$P(f) = \frac{1}{Z} e^{-\frac{U(f)}{T}}, \tag{3}$$

where Z is a normalization constant, T is called temperature, and $U(f) = \Sigma_i U_i(f)$ is an energy function that can be computed as the sum of local contributions from each lattice site i. The energy at each lattice site $U_i(f)$ is, itself, a sum of the *potentials*, $U_C(f)$, of each site'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 (Gamble and Poggio, 1987; Marroquin, Mitter, and Poggio, 1985). As a simple example, when the surfaces are expected to be smooth (like a membrane), the prior energy can be given in terms of

$$U_i(f) = \sum_j (f_i - f_j)^2, \tag{4}$$

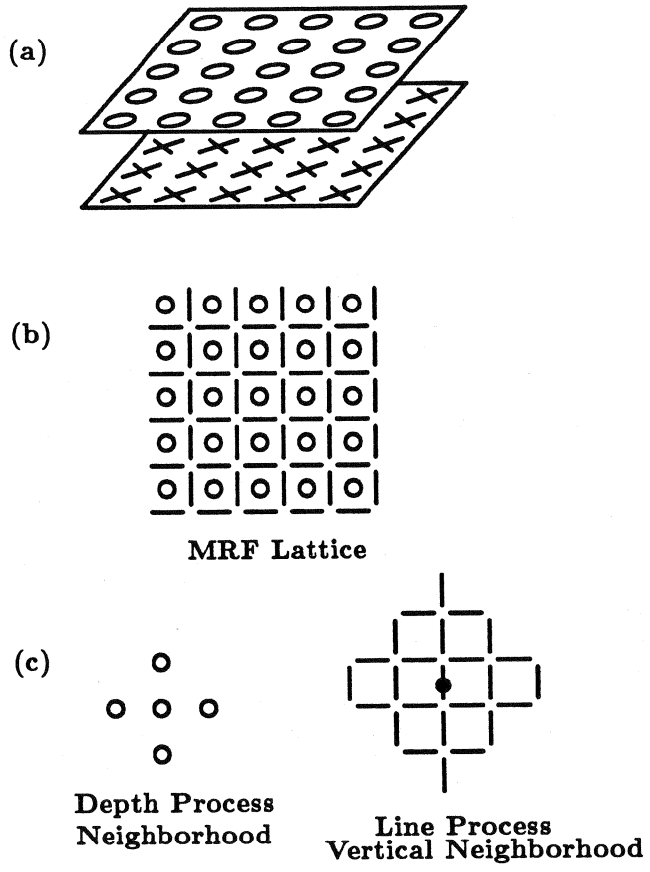


Fig. 3 Coupled MRF lattices The circles represent the continuous process (either depth, motion, color, or texture), and the lines represent the associated line process, that is, the discontinuities. The neighborhoods of the depth process and of the line process are also shown. The cost of an isolated line process is much higher than that of a continuous line.

where j is a neighboring site to i (that is, i and j belong 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's theorem then allows one to write the posterior distribution:

$$P(f|g) = \frac{1}{Z} e^{-\frac{U(f|g)}{T}} \tag{5}$$

In the example of Eq.(4), we have (for Gaussian noise):

$$U_i(f|g) = \sum_j (f_i - f_j)^2 + \alpha_i \gamma_i (f_i - g_i)^2, \tag{6}$$

where $\gamma_i = 1$ only where data are available, and otherwise $\gamma_i = 0$. More complicated cases can be handled in a similar manner (Gamble and Poggio, 1987).

The maximum of the posterior distribution or other related estimates cannot be computed analytically, but sample distributions with the probability distribution of Eq.(5) can be obtained by means of Monte Carlo techniques such as the Metropolis algorithm (Metropolis et al., 1953). 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, the Connection Machine 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 that can be obtained more efficiently simply by finding the average value of f at each lattice site.

One of the main attractions of MRF models 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 (see Fig. 3), and representing explicitly the presence or absence of discontinuities that break the smoothness assumption (Eq.4). The associated prior energy then becomes

$$U_i(f,l) = \sum_j (f_i - f_j)^2 (1 - l_i^j) + \beta V_C(l_i^j), \tag{7}$$

where l_i^j is a binary line element between site i, j. The term $V_C(l_i^j)$ reflects the fact that certain configurations of the line process are more likely than others to occur. Depth discontinuities are usually themselves continuous, nonintersecting, and rarely isolated points. These properties of physical discontinuities can be enforced locally by defining an appropriate set of energy values $V_C(l_i^j)$

for different configurations of the line process (Gamble and Poggio, 1987; Geman and Geman, 1984; Marroquin et al., 1985).

It is possible to extend the energy function of Eq.(7) to accommodate the interaction of more processes and of their discontinuities. In particular, we have extended the energy function to couple several of the early vision modules (depth, motion, texture, and color) to sharp changes of brightness in the image. This is a central point in our integration scheme: here we assume that changes of brightness 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 primary role of the gradient of brightness, as conjectured here, is that changes in surface properties usually produce large brightness gradients in the image.

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

$$V(l,e) = g(e_i^j, l_i^j), \qquad (8)$$

where e_i^j represents a measure of the strength of the brightness gradient (that is, of a 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; for instance, $g(e_i^j, l_i^j) = e_i^j(1 - l_i^j)$. This term facilitates formation of discontinuities (that is, $l_i^j = 1$) at the locations of sharp brightness changes, without restricting them to brightness edges. High values of the brightness gradient (together with image data in the neighborhood) activate with different probabilities the different types of surface discontinuities (see Fig. 1), which in turn are coupled to the output of stereo, motion, color, texture, and possibly other early vision algorithms.

6. AN EXAMPLE

We have been using the MRF machinery with prior energies like those given in Eq.(7) and Eq.(8) (see also Fig. 1) to integrate edge brightness data with stereo, motion, color, and texture information on the MIT Vision Machine system.

Figure 4 shows the inputs to the integration stage: the gray-level image and the associated brightness edges. Figure 5 shows the results of integrating brightness edges with a parallel stereo algorithm (Drumheller and Poggio, 1986). In a similar way (Fig. 6), the optical flow and its boundary from the same scene are computed from motion data (Little et al., 1988) and brightness edges (Gamble and Poggio, 1987; Hutchinson et al.,

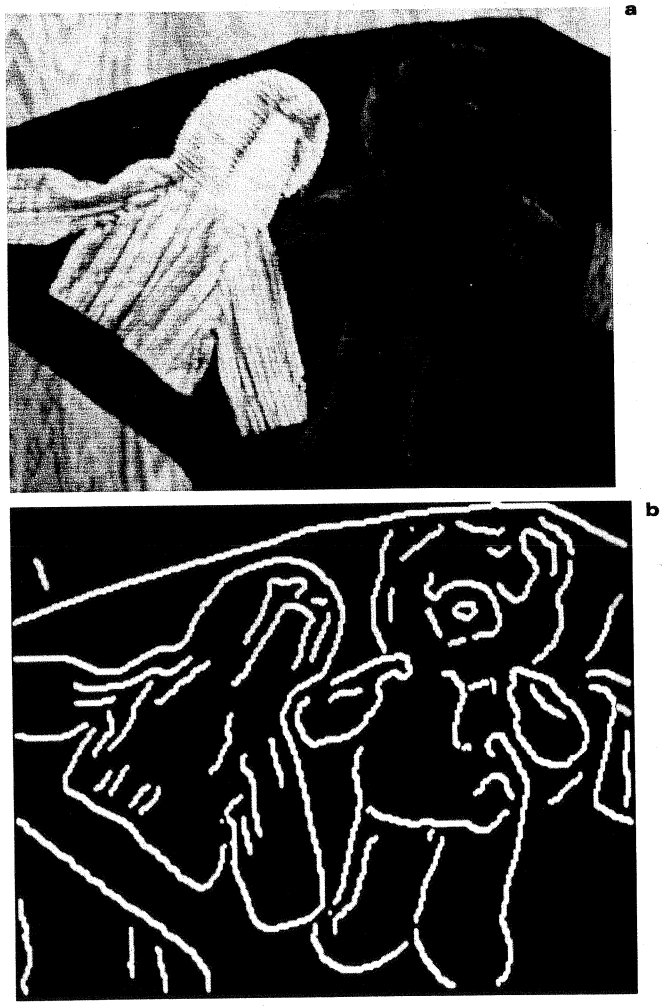


Fig. 4 (a) Gray-level image; (b) associated brightness edges as computed with a parallel implementation of Canny's algorithm

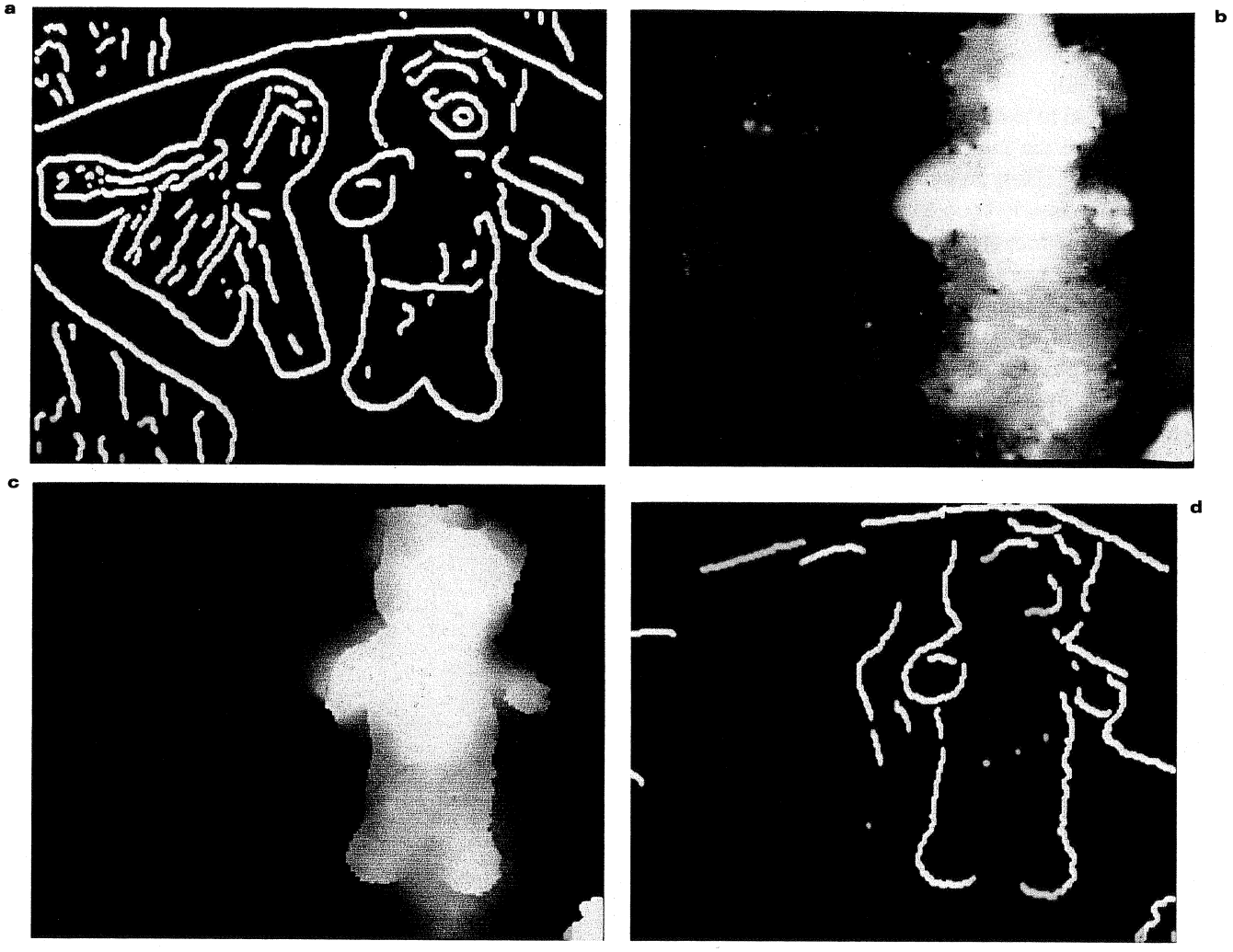


Fig. 5 (a) Brightness edges used with stereo; (b) stereo data; (c) reconstructed surface depth; (d) depth discontinuities found by the MRF integration scheme using brightness edges

1988; Murray and Buxton, 1987; Yuille, 1987). Figures 7 and 8 show simple examples of a similar integration performed with texture and color data. A MRF model that enforces local constancy of the hue H uses the dense but noisy data from the color module to segment the image into regions of different constant reflectance (Poggio et al., 1988). The coupling with brightness edges facilitates finding the boundaries: usually sharp changes in the ratio H correspond to a subset of the brightness edges.

The union in Figure 9 of the discontinuities in depth, motion, and texture for the scene of Figure 4 gives a "cartoon" of the original scene. Notice that this "cartoon" represents discontinuities in the physical properties of three-dimensional surfaces that are well defined, whereas brightness "discontinuities" are not.

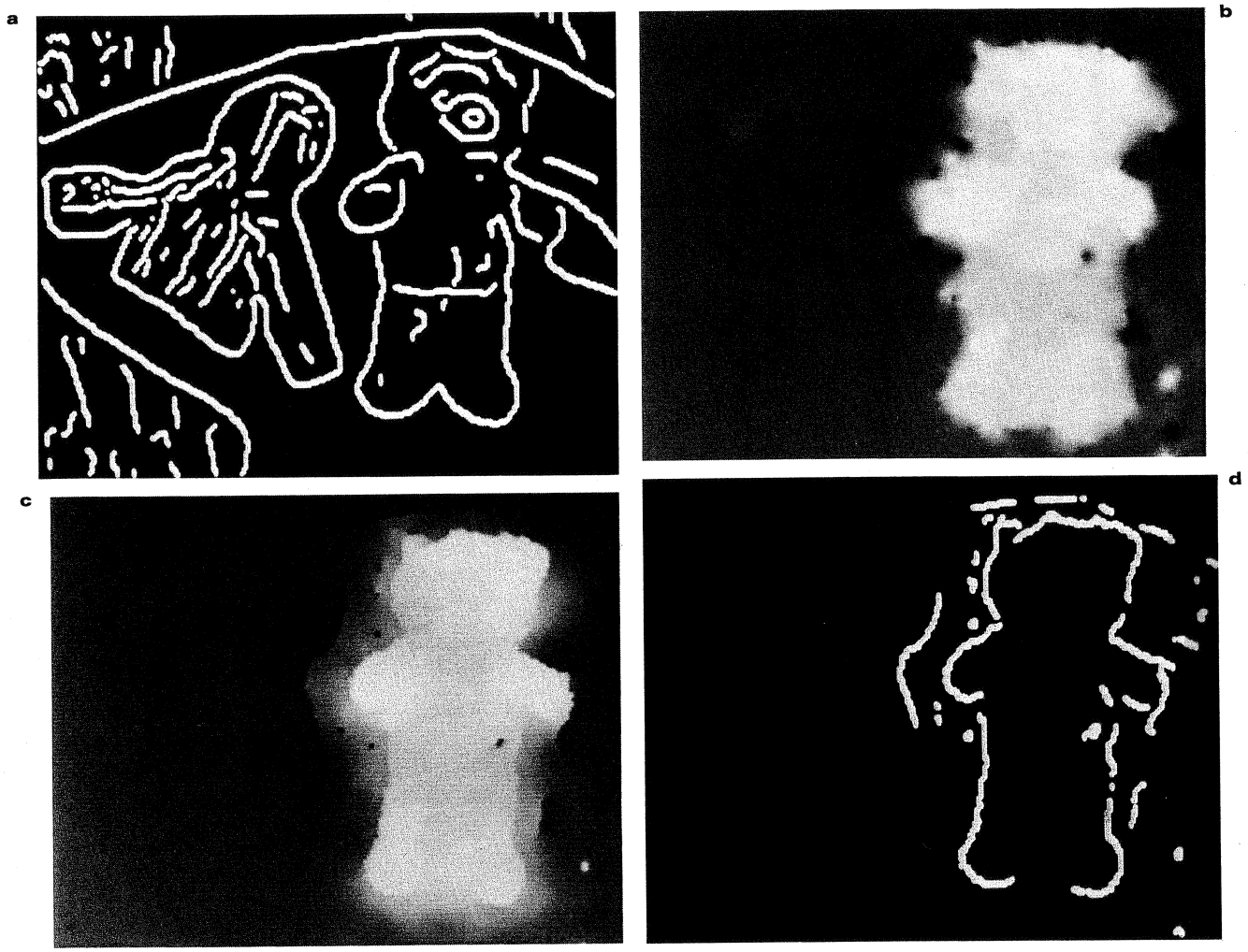


Fig. 6 (a) Brightness edges used with motion; (b) motion data; (c) the MRF reconstructed flow; (d) motion discontinuities

Discussion

Our integration algorithm achieves a preliminary classification of the brightness edges in the image, in terms of their physical origin. A more complete classification may be achieved by implementing the full scheme of Figure 1. The lattices at the top classify the different types of discontinuities in the scene: depth discontinuities, orientation discontinuities, albedo edges, specular edges, and shadow edges. The set of such discontinuities in the various physical processes seems to represent a good set of data for later recognition. In preliminary experiments we have successfully used a parallel, model-based recognition system (Cass, 1988) on the discontinuities (stereo and motion) provided by our MRF scheme. We plan to exploit the labeling of discontinuities in future recognition experiments. In addition, our integration scheme allows us to segment the scene into different depth planes, for instance, thereby considerably reducing the combinatorics of model-based recognition.

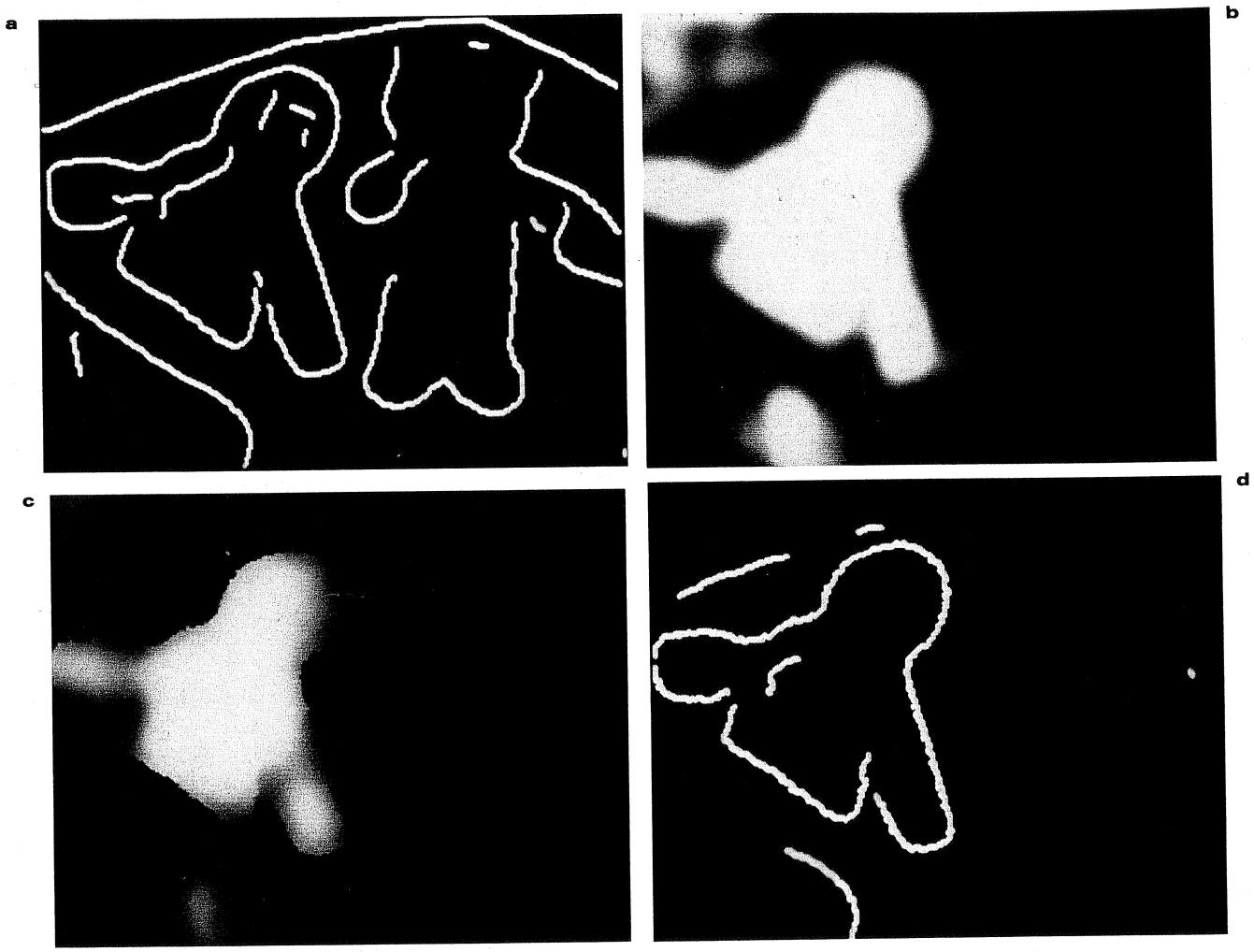


Fig. 7 (a) Brightness edges used with texture; (b) texture data; (c) reconstructed uniform texture regions; (d) texture discontinuities

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. As described elsewhere (Poggio et al., 1988; Gamble and Poggio, 1987), 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.

Our formulation of the integration problem in terms of MRF does not imply that the algorithms are necessarily stochastic. Deterministic approximations to the more general stochastic schemes may work quite well, especially in situations where redundant and contradictory data from several sources effectively set the initial state of the system close to the solution. We have, in fact, found that gradient descent in the space of the depth and the line process often works well. We

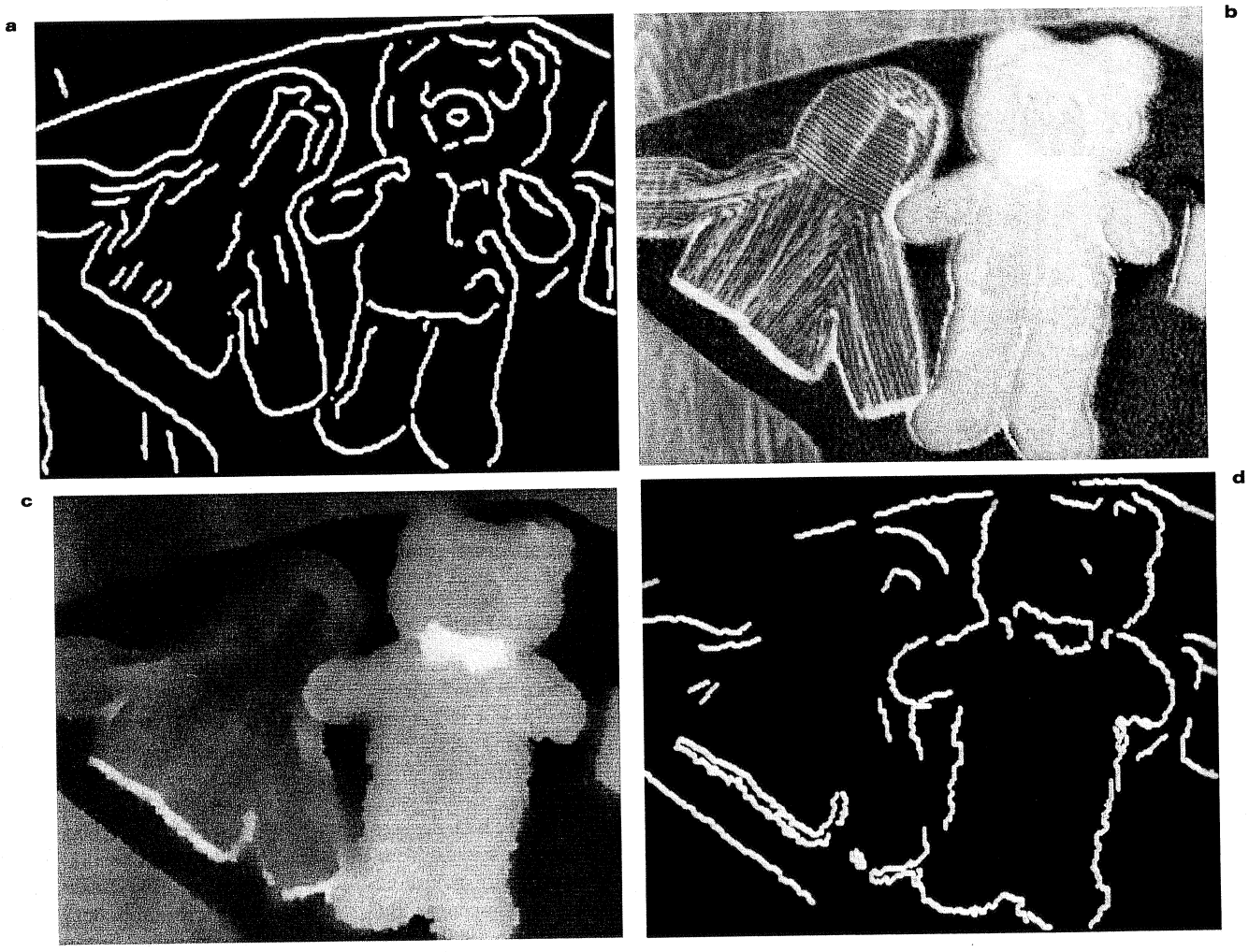


Fig. 8 (a) Brightness edges used with color; (b) color data (hue); (c) the MRF segmentation in terms of constant reflectance regions; (d) color boundaries

routinely use a mixed deterministic and stochastic strategy (Marroquin et al., 1985) in which the continuous (depth) process is deterministically updated while the line process is updated stochastically. Other strategies may also be effective (Chou and Brown, 1987a), such as space-variant filtering, coupled with edge detection. In addition, time-dependent schedules of the coupling parameters can be useful. They are somewhat similar to simulated annealing, which can also be effectively used, though it is quite slow.

The highly parallel algorithms we have described map quite naturally onto an architecture such as the Connection Machine. The vector model used on the Connection Machine elucidates the structure of the computations in all the algorithms. The MRF integration stage, in particular, has a quite simple structure. The energies computed at each step of the process depend only on the local configuration of the data and the line processes, and thus require little data communication. The implementation of the integration stage on the

Fig. 9 Union of the depth, motion, and texture discontinuities: a preliminary classification of the brightness edges in the image, in terms of their physical origin

Connection Machine is quite efficient. The same algorithms also map onto VLSI architectures of fully analog elements (we have successfully experimented with a version of Eq.(7) and Eq.(8), in which l is a continuous variable), mixed analog and digital components, and purely digital processors (similar to a much simplified and specialized Connection Machine).

A plausible organization of visual integration as sketched in Figure 1 may be found ultimately by theory and by computer experiments of the type described here. We believe that psychophysical and physiological data about the integration stage in the mammalian visual system may be helpful in guiding our theoretical and computational work. The system described here has already triggered a series of psychophysical experiments in order to establish whether and how brightness edges aid human computation of surface discontinuities (Heinrich Bülthoff, personal communication).

ACKNOWLEDGMENT

This report describes research done within the Artificial Intelligence Laboratory. Support for the laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76-85-C-0010 and in part under Office of Naval Research (ONR) contract N00014-85-K-0124. Support for this research is also provided by a grant from ONR, Engineering Psychology Division, and by a gift from the Artificial Intelligence Center of Hughes Aircraft Corporation to T. Poggio. Portions of this paper and Figures 1, 3, and 4–9 appeared in *Science,* 21 October 1988, © 1988 by the AAAS. See Poggio, Gamble, and Little (1988).

BIOGRAPHIES

James J. Little was born in Boston, in 1951. He received the A.B. degree in engineering and applied physics from Harvard University in 1972 and the M.S. and Ph.D. degrees in computer science from the University of British Columbia, Vancouver, B.C., Canada, in 1980 and 1985, respectively. From 1972 to 1975, he was with the Laboratory for Computer Graphics and Spatial Analysis at Harvard University. He was a research associate with the Department of Geography, Simon Fraser University, Burnaby, B.C., Canada, from 1975 to 1978. From 1985 to 1988, he was a research scientist at the Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Presently, he is an assistant professor of computer science at the University of British Columbia, Vancouver, where he is affiliated with the Centre for Integrated Computer Systems Research. His research interests include computer vision, robotics, computational geometry, and parallel algorithms for each of these areas. Dr. Little is a member of the American Association for Artificial Intelligence, the IEEE Computer Society, and the Canadian Society for the Computational Study of Intelligence.

Tomaso Poggio was born in Genoa, Italy, on September 11, 1947. He received the Ph.D. degree in theoretical physics from the University of Genoa in 1970. From 1971 to 1982, he was Wissenschaftlicher Assistant at the Max-Planck Institut für Biologische Kybernetik in Tübingen, West Germany. Since 1982, he has been a professor at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. In 1984, he was also appointed professor at MIT's Whitaker College of Health Sciences and Technology, and was named the first director of the Center for Biological Information Processing. In 1988, he was named to the Uncas and Helen Whitaker Professorship. Professor Poggio has authored over 100 papers in areas ranging from psychophysics and biophysics to information processing in man and machine, artificial intelligence, and machine vision. He is on the

editorial board of *Biological Cybernetics, Advances in Applied Mathematics,* and several other interdisciplinary journals.

Edward B. Gamble, Jr., received the B.S. and M.S. degrees in electrical engineering from the University of California, Los Angeles, during 1981. He is currently seeking a Ph.D. from the Massachusetts Institute of Technology while working in MIT's Artificial Intelligence Laboratory. His research interests include computer and human vision.

CORRESPONDING AUTHOR

James J. Little, Department of Computer Science, University of British Columbia, Vancouver, BC V6T 1W5, Canada. (604)228-4830.

REFERENCES

Aloimonos, J., and Brown, C. M. 1987. Robust computation of intrinsic images from multiple cues. In *Advances in computer vision,* edited by C. Brown. Hillsdale, N.J.: Erlbaum.

Barrow, H. G., and Tenenbaum, J. M. 1978. Recovering intrinsic scene characteristics from images. In *Computer vision systems,* edited by A. R. Hanson and E. M. Riseman. New York: Academic Press, pp. 3–26.

Bertero, M., Poggio, T., and Torre, V. 1987. Ill-posed problems in early vision. A.I. Memo No. 924. Cambridge: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Blake, A., and Zisserman, A. 1987. *Visual reconstruction.* Cambridge: MIT Press.

Blelloch, G. E. 1988. Scan primitives and parallel vector models. Ph.D. dissertation, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Brooks, R. A. 1987. A robust layered control system for a mobile robot. *IEEE J. Robotics Automation* RA-2:14–23.

Canny, J. F. 1986. A computational approach to edge detection. *IEEE Trans. Pattern Analysis Machine Intelligence* PAMI-8(6):679–698.

Cass, T. A. 1988. Robust parallel computation of 2d model-based recognition. In *Proceedings image understanding workshop.* San Mateo, Calif.: Morgan Kaufmann.

Chou, P. B., and Brown, C. M. 1987a. Multi-modal segmentation using Markov random fields. In *Proceedings image understanding workshop.* San Mateo, Calif.: Morgan Kaufmann, pp. 663–670.

Chou, P. B., and Brown, C. M. 1987b. Probabilistic information fusion for multi-modal image segmentation. In *Proceedings IJCAI.* San Mateo, Calif.: Morgan Kaufmann, pp. 779–782.

Chou, P. B., and Brown, C. M. 1988. Multimodal reconstruction and segmentation with Markov random fields and HCF optimization. In *Proceedings image understanding workshop.* San Mateo, Calif.: Morgan Kaufmann, pp. 214–221.

Cohen, F. S., and Cooper, D. B. 1983. Real time textured image segmentation based on noncausal Markovian random field models. In *Proceedings of SPIE conference on advances in intelligent robotics systems.* Bellingham, Wash.: SPIE.

Drumheller, M., and Poggio, T. 1986. On parallel stereo. In *Proceedings of IEEE conference on robotics and automation.* Washington, D.C.: IEEE, pp. 1439–1448.

Gamble, E., and Poggio, T. 1987. Visual integration and detection of discontinuities: the key role of intensity edges. A.I. Memo No. 970. Cambridge: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Geman, S., and Geman, D. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis Machine Intelligence* PAMI-6:721–741.

Gottlieb, A., Lubachevsky, B. D., and Rudolph, L. 1983. Basic techniques for the efficient coordination of very large numbers of cooperating sequential processors. *ACM Trans. Programming Languages Systems* 5(2).

Hillis, W. D. 1985. *The connection machine.* Cambridge: MIT Press.

Hoff, W., and Ahuja, N. 1987. Extracting surfaces from stereo images: an integrated approach. In *Proceedings of the international conference on computer vision.* Washington, D.C.: IEEE, pp. 284–294.

Hurlbert, A., and Poggio, T. 1986. Do computers need attention? *Nature* 321(12).

Hutchinson, J., Koch, C., Luo, J., and Mead, C. 1988. Computing motion using analog and binary resistive networks. *IEEE Computer Magazine* 21:52–64.

Kruskal, C. P., Rudolph, L., and Snir, M. 1985. The power of parallel prefix. In *Proceedings int. conf. on parallel processing.* University Park: Pennsylvania State University Press, pp. 180–185.

Ladner, R. E., and Fischer, M. J. 1980. Parallel prefix computation. *J. ACM* 27(4):831–838.

Little, J. J., Blelloch, G. E., and Cass, T. A. 1987a. How to program the Connection Machine for computer vision. In *Proc. 1987 workshop on comp. arch. for patt. anal. and mach. intell.* Washington, D.C.: IEEE, pp. 11–18.

Little, J. J., Blelloch, G. E., and Cass, T. A. 1987b. Parallel algorithms for computer vision on the Connection Machine. In *Proceedings of the international conference on computer vision.* Washington, D.C.: IEEE, pp. 587–591.

Little, J. J., Blelloch, G. E., and Cass, T. 1989. Algorithmic techniques for vision on a fine-grained parallel machine. *IEEE Trans. Pattern Analysis Machine Intelligence,* in press.

Little, J. J., Bülthoff, H. H., and Poggio, T. 1988. Parallel optical flow using local voting. In *Proceedings of the international conference on computer vision.* Washington, D.C.: IEEE.

Mahoney, J. V. 1986.

Image chunking: defining spatial building blocks for scene analysis. Master's thesis, Massachusetts Institute of Technology.

Marr, D. 1982. *Vision: a computational investigation into the human representation and processing of visual information.* San Francisco: W. H. Freeman and Company.

Marroquin, J. L., Mitter, S., and Poggio, T. 1985. Probabilistic solution of ill-posed problems in computational vision. In *Proceedings image understanding workshop.* McLean, Va.: Scientific Applications International Corporation, pp. 293–309.

Marroquin, J. L. 1985. Probabilistic solution of inverse problems. Ph.D. dissertation, Massachusetts Institute of Technology.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. 1953. Equation of state calculations by fast computing machines. *J. Phys. Chem.* 21:1087.

Murray, D. W., and Buxton, B. F. 1987. Scene segmentation from visual motion using global optimization. *IEEE Trans. Pattern Analysis Machine Intelligence* PAMI-9(2):220–228.

Poggio, T. 1985. Integrating vision modules with coupled MRF's. Working Paper No. 285. Cambridge: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Poggio, T., and staff. 1985. MIT progress in understanding images. In *Proceedings image understanding workshop.* McLean, Va.: Scientific Applications International Corporation, pp. 25–39.

Poggio, T., and staff. 1987. MIT progress in understanding images. In *Proceedings image understanding workshop.* San Mateo, Calif.: Morgan Kaufmann, pp. 41–54.

Poggio, T., and Torre, V. 1984. Ill-posed problems and regularization analysis in early vision. Technical Report AIM-773, C.B.I.P. Paper No. 001. Cambridge: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Poggio, T., Torre, V., and Koch, C. 1985. Computational vision and regularization theory. *Nature* 317:314–319.

Poggio, T., Gamble, E., and Little, J. J. 1988. Parallel integration of vision modules. *Science,* 242(4877):436–440.

Poggio, T., Little, J., Gamble, E., Gillett, W., Geiger, D., Weinshall, D., Villalba, M., Larson, N., Cass, T., Bülthoff, H., Drumheller, M., Oppenheimer, P., Yang, W., and Hurlbert, A. 1988. The MIT Vision Machine. In *Proceedings image understanding workshop.* San Mateo, Calif.: Morgan Kaufmann.

Ullman, S. 1984. Visual routines. *Cognition* 18.

Voorhees, H., and Poggio, T. 1988. Computing texture boundaries from images. *Nature* 333(6171):364–367.

Yuille, A. L. 1987. Energy functions for early vision and analog networks. Technical Report AIM-987. Cambridge: Massachusetts Institute of Technology, Artificial Intelligence Laboratory.