Parallel Networks for Machine Vision

Berthold K. P. Horn

The term "parallel networks" in the title of this chapter may appear to be redundant, because the computations at different nodes of an analog network naturally proceed in parallel. However, in several of the examples explored here a number of different interacting networks are used, and these do indeed operate "in parallel." We have to try and understand the kinds of computations that simple networks can perform and then use them as components in more complex systems designed to solve early vision problem.

Some of the ideas are first developed in continuous form, where we deal, for example, with resistive sheets instead of a regular grid of resistors. This is because the analysis of the continuous version is often simpler, and lends itself to well known mathematical techniques. Some thought must, of course, be given to what happens when we approximate this continuous world with a discrete one. This typically includes mathematical questions about accuracy and convergence, but also requires that the network be laid out on a two-dimensional plane, because today's implementations allow only very limited stacking in the third dimension. This can be a problem in the case where the network is inherently three-dimensional, or layered, or where several networks are used cooperatively. There are four major topics addressed here:

1 A Gaussian convolver for smoothing that operates continuously in time.
2 Coupled resistive networks for interpolation of image derived data.
3 Moment calculation methods for determining position and orientation.
4 Systems for recovering motion and shape from time-varying images.

In the process we touch on several important subtopics, including:

- Interlaced arrangements of the cells of the layers of a multi-resolution network on a two-dimensional surface.
- Tradeoffs between closed form solutions favored on serial computers and iterative or feedback methods better suited for analog networks.
- Laying out time as an extra spatial dimension so as to build a system in which information flows continuously.
- An equivalence between two apparently quite differently uses of a resistive network.
- Feedback methods for solving constrained optimization problems using gradient projection, normalization, and penalty functions.

Note, that the four sections of this chapter are fairly independent and not arranged in any particular order—for additional details see Horn [1988].

A Non-Clocked Gaussian Convolver for Smoothing

Gaussian convolution is a useful smoothing operation, often used in early vision, particularly in conjunction with discrete operators that estimate derivatives. There exist several digital hardware implementations, including one that exploits the separability of the two-dimensional Gaussian operator into the convolution of two one-dimensional Gaussian operators [Larson *et al.* 1981]. Analog implementations have also been proposed that use the fact that the solution of the heat-equation at a certain time is the convolution of a Gaussian kernel with the initial temperature distribution [Knight 1983].

One novel feature of the scheme described here is that data flows through continuously, with output available at any time. Another is an elegant way of interlacing the nodes of layers at several resolutions. First comes a brief review of why there is interest in Gaussian convolution.

Edge detection

The detection of step-edge transitions in image brightness involves numerical estimation of derivatives. As such it is an ill-posed problem [Poggio & Torre 1984; Torre & Poggio 1986]. All but the earliest efforts (see, for example, Roberts [1965]) employed a certain degree of smoothing before or after application of finite difference operators in order to obtain a more stable estimate. Equivalently, they used computational molecules of large support (see, for example, Horn [1971]). While most of the early work focused on the image brightness gradient, that is, the first partial derivatives of image brightness, there where some suggestion that second-order partial

derivatives might be useful. Rotationally symmetric ones appeared particularly appealing and it was noted that the Laplacian is the lowest order linear operator that (almost) allows recovery of the image information from the result [Horn 1972, 1974].

It was also clear early on that smoothing filters should be weighted so as to put less emphasis on points further away than those nearby.[1] The Gaussian was popular for smoothing because of a number of its mathematical properties, including the fact that the two-dimensional Gaussian can be viewed as the product of two one-dimensional Gaussians, and, much more importantly, as the convolution of two one-dimensional Gaussians [Horn 1972]. This gave rise to the hope that it might be computed with reasonable efficiency, an important matter when one is dealing with an image containing hundreds of thousands of picture cells. Note that the Gaussian is the only function that is both rotationally symmetric and separable in this fashion [Horn 1972]. The separability property, which was the original impetus for choosing the Gaussian as a smoothing filter, was forgotten at times when proposals where made later to build hardware convolvers (but, see Larson *et al.* [1981]).

Multi-resolution techniques

There are other reasons for smoothing a discretized image, including suppression of higher spatial frequency components before subsampling. Subsampling of an image produces an image of lower resolution, one that contains fewer picture cells. Ideally, one would hope that this smaller image retains all of the information in the original higher resolution image, but this is, of course, in general not possible. The original image can be reconstructed only if it happens not to contain spatial frequency components that are too high to be represented in the subsampled version. This suggests suppressing higher frequency components before subsampling in order to avoid *aliasing* phenomena. An ideal low-pass filter should be used for this purpose.[2] While the Gaussian filter is a poor approximation to a low pass

[1] There was, however, intense disagreement about whether the composite edge operator should have a sharp transition in the middle or not. Some argued that the transition should be rapid, because a matched filter has an impulse response equal to the signal being detected, which in this case was assumed to be an ideal step transition. Others claimed that the aim was to suppress higher spatial frequencies to improve the signal to noise ratio. This latter argument took into account the fact that the signal drops off at higher frequencies while the noise spectrum tends to be fairly flat. The view of the edge operator as a composition of a smoothing filter and a finite difference approximation of a derivative finally reinforced the latter view.

[2] For an excellent finite support approximation to a low-pass filter look in Rifman and McKinnon [1974], Bernstein [1976], Abdou and Wong [1982].

filter, it has the advantage that it does not have any over- or undershoot in either the spatial or the frequency domain. Consequently, the Gaussian smoothing operator has been used in several multi-scale schemes, despite the fact that it is not a good approximation to a low-pass filter.

The difference of two spatially displaced Gaussians was used quite early on in edge detection [MacLeod 1970a, 1970b]. The idea of working at multiple scales occurred around about this time also (Rosenfeld and Thurston [1971, 1972] and Rosenfeld, Thurston and Lee [1972]). An elegant theory of edge detection using zero-crossings of the Laplacian of the Gaussian at multiple scales was developed by Marr and Hildreth [1980] and Hildreth [1980, 1983]). This reversed an earlier suggestion that a directional operator may be optimal [Marr 1976].

It has been shown, then, that the rotationally symmetric operators do have some drawbacks, including greater inaccuracy in edge location when the edge is not straight, as well as higher sensitivity to noise than directional operators (see, for example, Berzins [1984] and Horn [1986]). Operators for estimating the second derivative in the direction of the largest first derivative (the so-called *second directional derivative*) have been proposed Haralick [1984] (see also Hartley [1985] and Horn [1986]).[3] Recently, Canny developed an operator that is optimal (in a sense he defines) in a one-dimensional version of the edge detection problem [Canny 1983]. His operator is similar, but not equal to, the first derivative of a Gaussian. A straightforward (although *ad hoc*) extension of this operator to two-dimensions has recently become popular.

If we view the problem as one of estimating the derivatives of a noisy signal, we can apply Wiener's optimal filtering methods [Wiener 1966; Anderson & Moore 1979]. Additive white noise is uncorrelated and so has a flat spectrum, while images typically have spectra that decrease as some power of frequency, starting from a low-frequency plateau [Ahuja & Schachter 1983]. The magnitude of the optimal filter response ends up being linear in frequency at low frequencies, then peaks and drops off as some power of frequency at higher frequencies. Under reasonable assumptions about the spectra of the ensemble of images being considered, this response may be considered to match (very roughly) the transform of the derivative of a Gaussian.

All this suggests that while there is nothing really magical about the Gaussian smoothing filter, it has been widely accepted and has many desirable mathematical properties (although only a few of these were discussed here). It is thus of interest to find out whether convolutions with Gaussian kernels can be computed directly by simple analog networks. It is also desirable to find out whether the Laplacian of the convolution with a Gaussian, or the directional derivatives, can be computed directly.

[3] While the second directional derivative is a non-linear operator, it is coordinate-system independent, as is the Laplacian operator.

Binomial filters

In practice, we usually have to discretize and truncate the signal, as well as the filters we apply to it. If we sample and truncate a Gaussian, it loses virtually all of the interesting mathematical properties discussed above. In particular, truncation introduces discontinuities that assure that the transform of the filter will fall off only as the inverse of frequency at high frequencies, not nearly as fast as the transform of the Gaussian itself. Furthermore, while the transfer function of a suitable scaled Gaussian lies between zero and one for all frequencies, the transfer function of a truncated version will lie outside this range for some frequencies. These effects are small only when we truncate at a distance that is large compared to the spatial scale of the Gaussian.

In addition, when we sample, we introduce aliasing effects, because the Gaussian is not a low-pass waveform. The aliasing effects are small only when we sample frequently in relation to the spatial scale of the Gaussian. It makes little sense to talk about convolution with a "discrete Gaussian" obtained by sampling with spacing comparable to the spatial scale, and by truncating at a distance comparable to the spatial scale of the underlying Gaussian. The resulting filter weights could have been obtained by sampling and truncating many other functions and so it is not reasonable to ascribe any of the interesting qualities of the Gaussian to such a set of weights.

Instead, we note that the appropriate discrete analog of the Gaussian is the binomial filter, obtained by dividing the binomial coefficients of order n by 2^n so that they conveniently sum to one. Convolution of the binomial filter of order n with the binomial filter of order m yields the binomial filter of order $(n + m)$, as can be seen by noting that multiplication of polynomials corresponds to convolution of their coefficients. The simplest binomial smoothing filter has the weights:

$$\left\{\frac{1}{2}, \frac{1}{2}\right\}.$$

Higher order filters can be obtained by repeated convolution of this filter with itself:

$$\left\{\frac{1}{4}, \frac{2}{4}, \frac{1}{4}\right\} \otimes \left\{\frac{1}{2}, \frac{1}{2}\right\} = \left\{\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}\right\}.$$

The transform of the binomial filter of order n is simply $\cos^n \omega/2$, because the transform of the simple filter with two weights is just $\cos \omega/2$. This shows that the magnitude of the transform is never larger than one for any frequency, a property shared with a properly scaled Gaussian. Such a filter cannot amplify any frequency components, only attenuate them.

Analog implementation of binomial filters

Binomial filters can be conveniently constructed using charge coupled device technology [Sage 1984; Sage & Lattes 1987]. It is also possible to use potential dividers to perform the required averaging. Consider, for example, a uniform one-dimensional chain of resistors with inputs applied as potentials on even nodes and results read out as potentials on odd nodes. The potentials on the odd nodes clearly are just averages of the potentials at neighboring even nodes.[4]

One such resistive chain can be used to perform convolution with the simple two-weight binomial filter. To obtain convolution with higher-order binomial filters, we can reuse the same network, with inputs and outputs interchanged, provide that we have clocked sample-and-hold circuits attached to each node. At any particular time one half of the sample-and-hold circuits are presenting their potentials to the nodes they are attached to, while the other half are sampling the potentials on the remaining nodes.

But we may be more interested in non-clocked circuits, where outputs are available continuously. The outputs of one resistive chain can be applied as input to another, provided that buffer amplifiers are interposed to prevent the second chain from loading the first one. We can cascade many such resistive chain devices to obtain convolutions with binomial filters of arbitrary order.

It is possible to extend this idea to two dimensions. Consider nodes on a square grid, with each node connected to its four edge-adjacent neighbors by a resistor. Imagine coloring the nodes red and black, like the squares on a checker-board. Then the red nodes may be considered the inputs, where potentials are applied, while the black nodes are the outputs, where potentials are read out. Each output potential is the average of four input potentials, and each input potential contributes to four outputs.

Unfortunately, the spatial scale of the binomial filter grows only with the square root of the number of stages used. Thus, while a lot of smoothing happens in the first few stages, it takes many more stages later in the sequence to obtain significantly additional smoothing. Also, the smoothed data has lost some of its high frequency content and so can perhaps be represented by fewer samples. These considerations suggest a multi-scale approach, where the number of nodes decreases from layer to layer. Averaging of neighbors at a later layer involves connections between nodes corresponding to points that are far apart in the original layer. Thus the smoothing that results in one of the later layers is over a larger spatial scale. We discuss later how to efficiently interlace the nodes of several such layers of different resolution on a two-dimensional surface.

[4]The outputs in this case are offset by one half of the pixel spacing from the inputs, but this is not a real problem. In particular, an even number of such filtering stages produces results that are aligned with the original data.

The kind of network we are discussing here can also be approached in a different way, starting from the properties of continuous resistive sheets, and analog methods for solving the heat equation. For details of this approach see Horn [1988].

Multiple scales

The information is smoothed out more and more as it flows through the layers of this system. Consequently we do not need to preserve full resolution in layers further from the input. Very roughly speaking, the information is low-pass filtered and so fewer samples are required to represent it. This suggests that successive sheets could contain fewer and fewer nodes.

Note also that it would be difficult indeed to superimpose, in two dimensions, multiple layers of the three dimensional network described above, if each of them contained the same (large) number of nodes. Now if, instead, a particular layer contains only $1/k$ times as many nodes as the previous layer then the total number of nodes is less than $k/(k-1)$ times the number of nodes in the first layer, as can be seen by summing the appropriate geometric series. If, for example, we reduce the number of nodes by one half each time, then a network containing a finite number of layers has less than twice the number of nodes that the first layer requires. (If we reduce the number of nodes to a quarter each time, then the whole network has less than 4/3 times as many as the first layer.)

Growth of standard deviation with number of layers

If we define the *width* of the binomial filter as the standard deviation from its center position, while the *support* is the number of non-zero weights, then it turns out that width grows only as the square-root of the support. So another argument for subsampling is that, if all the layers and the interconnections are the same, then the width of the binomial filter grows only with the square root of the number of layers. One way to obtain more rapid growth of the width of the smoothing filter is to arrange for successive layers to contain fewer nodes. This can be exploited to attain exponential growth of the effective width of the smoothing filter with the number of layers.

In the case of a square grid of nodes, a simple scheme would involve connecting only one cell out of four in a given layer to the next layer. This corresponds to a simple subsampling scheme. Sampling, however, should always be preceded by low-pass filtering (or at least some sort of smoothing) to limit aliasing. A better approach therefore involves first computing the average of four nodes in a given 2×2 pattern in order to obtain a smoothed

result for the next layer.[5] Each cell in the earlier layer contributes to only one of the averages being computed in this scheme.

The average could be computed directly using four resistors, but these would load down the network. The average can be computed instead using resistors connected to buffer amplifiers. Each cell in the earlier layer feeds a buffer amplifier and the output of the amplifier is applied to one end of a resistor. The other ends are tied together in groups of four and connected to the nodes in the next layer. Note that the nodes of the latter sheet should be thought of as corresponding to image locations between those of the earlier sheet, rather than lying on top of a subset of these earlier nodes. But this subtlety does not present any real problems.

Layout of interlaced nodes

A four-to-one reduction in number of nodes is easy to visualize and leads to rapid reduction in the number of nodes in successive layers, but it does not yield a very satisfactory subsampling operation. Aliasing can be reduced if the number of nodes is reduced only by a factor of two. Note that in this case the total number of nodes in any finite number of layers is still less than twice the number of nodes in the first layer. An elegant way of achieving the reduction using a square grid of nodes is to think of successive layers as scaled spatially by a factor of $\sqrt{2}$ and also rotated 45° with respect to one another. Once again, each of the new nodes is fed a current proportional to the difference between the average potential on four nodes in the earlier layer and the potential of the node itself. This time, however, each of the earlier nodes contribute to two of these averages rather than just one, as in the simple scheme described in the previous section. A node receives contributions from four nodes that are neighbors of its ancestor node in the earlier layer, but it receives no contribution directly from that ancestor.

An elegant partitioning of a square tessellation into subfields may be used in the implementation of this scheme in order to develop a satisfactory physical layout of the interlaced nodes of successive layers of this network (Robert Floyd drew my attention to this partitioning in the context of parallel schemes for producing pseudo grey-level displays on binary image output devices [Floyd 1987]). This leads to the interlaced pattern shown in figure 1, where each cell is labeled with a number indicating which layer it belongs to.

This scheme leads to an arrangement where the nodes of the first layer are thought of as the black cells in a checkerboard. The red cells form a

[5]Naturally, because this is not an ideal low-pass filter, some aliasing effects cannot be avoided. In fact, the resulting transfer function goes through zero not at the Nyquist frequency, but only at twice that frequency, but this is much better than not doing any smoothing at all.

0	1	3	1	5	1	3	1	7	1	3	1	5	1	3	1	9
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
3	1	4	1	3	1	4	1	3	1	4	1	3	1	4	1	3
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
5	1	3	1	6	1	3	1	5	1	3	1	6	1	3	1	5
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
3	1	4	1	3	1	4	1	3	1	4	1	3	1	4	1	3
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
7	1	3	1	5	1	3	1	8	1	3	1	5	1	3	1	7
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
3	1	4	1	3	1	4	1	3	1	4	1	3	1	4	1	3
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
5	1	3	1	6	1	3	1	5	1	3	1	6	1	3	1	5
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
3	1	4	1	3	1	4	1	3	1	4	1	3	1	4	1	3
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
9	1	3	1	5	1	3	1	7	1	3	1	5	1	3	1	6

Figure 1. A way to interlace nodes of several layers of a multi-scale network so they can be laid out on a two-dimensional surface. The network containing nodes labeled $(n + 1)$ has half as many nodes as the network whose nodes are labeled n. The total number of nodes is less than twice the number of nodes in the finest layer.

diagonal pattern with $\sqrt{2}$ times the spacing of the underlying grid. We can now consider this new grid as a checkerboard, turned 45° with respect to the first. The black cells in this checkerboard belong to the second layer. The remaining red cells form a square grid aligned with the underlying grid but with twice the spacing between nodes. Considering this as a checkerboard in turn, we let the black cells be the nodes of the third layer, and so on ....

Note that one half of the cells are labeled 1, one quarter are labeled 2, one eighth are labeled 3 and so on. The top left node, labeled 0, does not belong to any of the partitions. If we consider the nodes labeled with their row number i and there column number j, both starting at zero at the top left node, we find that a node belongs to layer k if the binary representation of $i^2 + j^2$ has $k - 1$ trailing zeros!

Coupled Poisson's Equation for Interpolation

Uniform resistive networks that solve Poisson's and Laplace's equations have many other applications. One is in interpolation, where data may be provided on just a few contours, as happens in the edge matching approach

to binocular stereo [Grimson 1981].[6] Many modern interpolation methods are based on physical models of deformation of elastic sheets or thin plates. So these are briefly reviewed here first.

Mathematical physics of elastic membranes

An elastic membrane takes on a shape that minimizes the stored elastic energy. In two dimensions the stored energy is proportional to the change in area of the membrane from its undisturbed shape, which we assume here is flat. The area is given by

$$\iint_D \sqrt{1 + z_x^2 + z_y^2}\, dx\, dy\,,$$

where $z(x, y)$ is the height of the membrane above some reference plane. If the slope components z_x and z_y are small,

$$\sqrt{1 + z_x^2 + z_y^2} \approx 1 + \frac{1}{2}\left(z_x^2 + z_y^2\right)\,.$$

Thus the membrane minimizes

$$\iint_D (z_x^2 + z_y^2)\, dx\, dy\,,$$

provided that the partial derivatives z_x and z_y are small. A unique minimum exists if the sheet is constrained to pass through a simple closed curve ∂D on which the height is specified. The Euler equation for this calculus of variation problem yields

$$z_{xx} + z_{yy} = 0 \quad \text{or} \quad \Delta z = 0\,,$$

except on the boundary where the height $z(x, y)$ is specified [Courant & Hilbert 1953].

Interpolation by means of a thin plate

The above equation has been proposed as a means of interpolating from sparse data specified along smooth curves, not necessarily simple closed

[6] The problem of interpolation is harder if data is given only on a sparse set of points, as opposed to contours. Consider, for example, Laplace's equation with some constant value specified on a simple closed curve with a different value given at a single point inside the curve. The solution minimizes the integral of the sum of squares of the first partial derivatives. It turns out that this is not a well-posed problem, because there is not a unique solution. One of the "functions" that minimizes the integral takes on the value specified on the boundary everywhere except at the one point inside where a different value is given. Clearly no "interpolation" is occurring here. This problem is not widely discussed, in part because the discrete approximation does not share this pathological behavior [Grzywacs & Yuille 1987].

contours. We explored the use of this idea, for example, in generating digital terrain models from contour maps in our work on automated hill-shading (Strat [1977] and Horn [1979, 1981, 1983]) as well as in remote sensing (Bachmann [1977], Horn and Bachmann [1978], and Sjoberg and Horn [1983]). Some undergraduate research project work was based on this idea [Mahoney 1980], as were the bachelor's theses of Goldfinger [1983], and Norton [1983]. Recently, a 48 × 48 cell analog chip has been built to do this kind of interpolation [Luo, Koch & Mead 1988].

The result of elastic membrane interpolation is not smooth, however, while height in the result is a continuous function of the independent variables, slope is not. Slope discontinuities occur all along contour lines, and the tops of hills and bottoms of pits are flat.[7]

This is why we decided to use thin plates for interpolation from contour data instead. The potential energy density of a thin plate is

$$A\left(\frac{1}{\rho_1^2}+\frac{1}{\rho_2^2}\right)+\frac{2B}{\rho_1\rho_2},$$

where A and B are constants determined by the material of the plate, while ρ_1 and ρ_2 are the principal radii of curvature of the deformed plate [Courant & Hilbert 1953]. Again, assuming that the slopes z_x and z_y are small, we can use the approximations

$$\frac{1}{\rho_1}+\frac{1}{\rho_2}\approx(z_{xx}+z_{yy}) \quad \text{and} \quad \frac{1}{\rho_1\rho_2}\approx z_{xx}z_{yy}-z_{xy}^2.$$

This allows us to approximate the potential energy of the deformed plate by a multiple of

$$\iint_D \left((z_{xx}+z_{yy})^2-2(1-\mu)(z_{xx}z_{yy}-z_{xy}^2)\right)\,dx\,dy,$$

where $\mu = B/A$. If the material constant μ happens to equal one, this simplifies to the integral of the square of the Laplacian:

$$\iint_D (\Delta z)^2\,dx\,dy.$$

The Euler equations for this variational problem lead to the bi-harmonic equation

$$\Delta(\Delta z)=0,$$

except where the plate is constrained. This fourth-order partial differential equation has a unique solution when the height $z(x,y)$, as well as the normal derivative of $z(x,y)$ are specified on a simple closed boundary ∂D.

It turns out that the same Euler equation applies when the material constant μ is not equal to one, because $(z_{xx}z_{yy}-z_{xy}^2)$ is a divergence expression [Courant & Hilbert 1953]. Solution of the bi-harmonic equation, while

[7] Discontinuities in slope are not a problem for many applications of interpolated depth or range data. Shaded views of the surfaces, however, clearly show the discontinuities, because shading depends on surface orientation.

involving considerably more work than Laplace's equation, produces excellent results in interpolation from contours. Iterative methods for solving these equations are available (see for example Horn [1986]). Some obvious implementations may not be stable, particularly when updates are executed in parallel, so care has to be taken to ensure convergence. The problem is that computational molecules or stencils with negative weights are needed, and these can amplify errors with some spatial frequencies rather than attenuate them.[8] This issue is not pursued any further here. The proper way of dealing with boundary conditions is also not discussed here, for details, see the cited references.

The same methods were used in interpolation of surface depth from stereo data along brightness edges [Grimson 1981, 1982, 1983]. Grimson observed that the null-space of the quadratic variation $(z_{xx}^2 + 2z_{xy}^2 + z_{yy}^2)$ is smaller than that of the squared Laplacian $(\Delta z)^2$, and so decided to use the quadratic variation as the basis for his binocular stereo interpolation scheme. This corresponds to choosing $\mu = 0$. Note that this affects only the treatment of the boundary; one still solves the bi-harmonic equation inside the boundary.

The methods discussed here rapidly get rid of high spatial frequency components of the error, but may take many iterations to reduce the low frequency components. The number of iterations required grows quadratically with the width of the largest gap between contours on which data is available.

Efficient multiresolution algorithms were developed to speed up the iterative computation of a solution [Terzopoulos 1983]. This approach has also been applied these to variational problems other than interpolation [Terzopoulos 1984].

Resistive networks for the bi-harmonic equation

It is clear then that methods for solving the bi-harmonic equations are important in machine vision. Unfortunately, simple networks of (positive) resistances cannot be constructed to solve discrete approximations of this equation. Computational molecules or stencils [Horn 1986] for the bi-harmonic operator involve negative weights and connections to nodes two steps away.

It is of interest then to discover ways of using methods for solving Poisson's equation

$$\Delta z(x, y) = f(x, y)$$

in the solution of the bi-harmonic equation, because simple resistive networks can be constructed to solve Poisson's equation. One simple idea is

[8] The corresponding system of linear equations is not diagonally dominant.

to use the coupled system,

$$\Delta z(x,y) = u(x,y) \quad \text{and} \quad \Delta u(x,y) = f(x,y)\,,$$

because here

$$\Delta(\Delta z) = \Delta u = f(x,y)\,.$$

The constraints on $z(x,y)$ can be handled easily in this formulation, but constraints on the partial derivatives of $z(x,y)$ are harder to incorporate. This idea will not be pursued further here.

An alternative explored recently by Harris [1986] involves minimization of the functional

$$\iint_D \left((z_x - p)^2 + (z_y - q)^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2)\right) dx\,dy\,.$$

The Euler equations for this calculus of variation problem yield

$$\begin{aligned} \Delta z &= p_x + q_y\,, \\ \lambda \Delta p &= p - z_x\,, \\ \lambda \Delta q &= q - z_y\,. \end{aligned}$$

In this scheme, three coupled Poisson's equations are used, each of which can be solved using a resistive network. Constraints on both $z(x,y)$ as well as z_x and z_y can be incorporated.

The relationship to the problem of solving the bi-harmonic equation can be seen by expanding

$$\Delta(\Delta z) = \Delta(p_x + q_y)\,,$$

and noting that differentiation and application of the Laplacian are linear operations, so that they can be interchanged:

$$\begin{aligned} \Delta(p_x) &= (\Delta p)_x = (1/\lambda)(p - z_x)_x = (1/\lambda)(p_x - z_{xx}) \\ \Delta(q_y) &= (\Delta q)_y = (1/\lambda)(q - z_y)_y = (1/\lambda)(q_y - z_{yy}) \end{aligned}$$

and finally

$$\Delta(\Delta z) = (1/\lambda)\big((p_x + q_y) - (z_{xx} + z_{yy})\big) = 0\,,$$

since $\Delta z = (p_x + q_y)$. (Note that this does not necessarily imply that $p = z_x$ and $q = z_y$.)

This scheme is reminiscent of the one developed by Horn for recovering depth $z(x,y)$, given dense estimates of the components p and q of the gradient of the surface (as used in Ikeuchi [1984], and described in Horn and Brooks [1986]). There one minimizes

$$\iint_D (z_x - p)^2 + (z_y - q)^2\, dx\,dy\,,$$

for which the Euler equation yields

$$\Delta z = p_x + q_y\,,$$

where p and q are here the given estimates of the components of the surface gradient. In Harris's scheme we do not have these estimates at all points,

instead we are given z at some points, and some linear combination of p and q at some other points.

The above ideas can be extended to solving the *image irradiance equation* that occurs in the *shape from shading* problem [Horn 1989; Horn & Brooks 1989]. See Harris [1989] for other ideas on an analog implementations of the thin plate interpolation method.

Moment Calculations for Position and Orientation

Calculations of sums of products of image coordinates and functions of the picture cell grey-levels are useful in several early machine vision algorithms. These moments are easily calculated using many different architectures, including bit-sliced, pipelined, analog networks, and by means of charge coupled devices. Such methods have several applications. A new technique for directly estimating motion of the camera from first derivatives of image brightness, for example, depends on the calculation of such moments (as discussed in the next section).

In addition, a large fraction of all binary image processing methods involve the computation of the zeroth, first and second moments of the regions of the image considered to be the image of one object. Presently, most commercially available machine vision systems have only rudimentary mechanisms for dealing with grey-level images and are aimed mainly at binary images. These systems typically have digital means for computing the moments. While such systems are restricted in their application, they are widely available and well understood. They can be used, for example, to determine the position and orientation of an isolated, contrasting workpiece lying flat on a conveyor belt (see, for example, Chapter 3 in Horn [1986]). Once the position and orientation of the object is known, a robot hand with the appropriate orientation may be sent to the indicated position to pick up the part. A device that finds the centroid of a spot of light in the image can also be used as a high-resolution light-pen and a means of tracking a light source, such as a light bulb attached to an industrial robot arm.

A variety of methods is available for efficiently computing the zeroth- and first-order moments, including methods for working with projections of the image or run-length coded versions of the image. Less appears to be known about how to easily compute second- and higher-order moments, except that iterated summation can be used to avoid the implied multiplications.

Such ideas are used in special purpose digital chips that have been built for finding moments [Hatamian 1986, 1987]. We nevertheless explore analog networks for this task, partly to see whether they may have advantages over existing digital implementations and partly because they constitute a

stepping stone on the way to some types of networks used in the recovery of motion from time-varying images [Horn 1988].

In this section several different methods are explored for computing moments using analog networks. It will be shown that some elegant methods exist that make it possible to obtain these moments using networks with relatively few components.

Use of first moments for position

Suppose that we have a characteristic function that indicates places in the image where the object region is thought to be. That is,

$$b(x,y) = \begin{cases} 1, & \text{if } (x,y) \text{ is in the region;} \\ 0, & \text{otherwise.} \end{cases}$$

Under favorable circumstances, such a characteristic function can be obtained by thresholding a grey-level image. The area of the object is obviously just the zeroth-order moment

$$A = \iint_D b(x,y)\,dx\,dy\,,$$

where the integral is over the whole image.

The position of the object can be considered to be the location $(\bar{x},\bar{y})$ of its center of area, defined in terms of the two first-order moments as follows:

$$A\,\bar{x} = \iint_D x\,b(x,y)\,dx\,dy \quad \text{and} \quad A\,\bar{y} = \iint_D y\,b(x,y)\,dx\,dy\,.$$

The center of area, or *centroid* is independent of the choice of coordinate system.[9]

Use of second moments for orientation

There are three second-order moments, and these can be used to define the orientation of the object as well as a shape factor. The orientation of the object may be taken to be specified by the direction of the axis of least inertia, which is independent of the choice of coordinate system axes.[10]

The inertia of a particle relative to a given axis is the product of the mass of the particle and the square of the perpendicular distance of

[9] That is, the position of the centroid relative to the object does not depend on the choice of coordinate used in the calculation.

[10] If we rotate the coordinate system, we find that the axis of least inertia determined in the new coordinate system is just the rotated version of the axis of least inertia in the original coordinate system.

the particle form the axis. So the inertia of an extended object about an arbitrary axis in the image plane can be defined as

$$I = \iint_D r^2(x,y)\, b(x,y)\, dx\, dy\,,$$

where

$$r(x,y) = x\sin\theta - y\cos\theta + \rho$$

is the distance of the image point (x,y) from the line with inclination θ (measured anti-clockwise form the x-axis) and perpendicular distance ρ from the origin.

It is easy to show that the axis of least inertia passes through the center of area, so it is convenient to compute the second-order moments with respect to the center of area (see, for example, Chapter 3 in Horn [1986]). Let

$$a' = \iint_D x'^2\, b(x,y)\, dx\, dy\,,$$

$$b' = \iint_D x'y'\, b(x,y)\, dx\, dy\,,$$

$$c' = \iint_D y'^2\, b(x,y)\, dx\, dy\,,$$

where $x' = (x - \bar{x})$ and $y' = (y - \bar{y})$. The inertia can then be expressed as a function of the angle of inclination of the axis in the form

$$I = \frac{1}{2}(a' + c') + \frac{1}{2}(c' - a')\cos 2\theta - b'\sin 2\theta\,.$$

Differentiating this with respect to θ and setting the result equal to zero yields

$$(c' - a')\sin 2\theta_0 + 2\,b'\cos 2\theta_0 = 0\,,$$

for the inclinations of the axes corresponding to extrema of inertia. Note that we do not actually need all three of the second-order moments to compute θ_0, only the combination $(c' - a')$ and b' are required. This observation is exploited later in a circuit designed to find the orientation of the axis of least inertia.

There is a two-way ambiguity here, since the equation is satisfied by $(\theta_0 + \pi)$ if it is satisfied by θ_0. This is to be expected, because we are only finding the line about which the region has least inertia. Higher order moments can be used to resolve this ambiguity, but we will not pursue this subject any further here.

The axis through the center of area yielding maximum inertia lies at right angles to the axis yielding minimum inertia. The maximum and minimum inertia themselves are given by

$$I_{\max} = \frac{1}{2}(a' + c') + \frac{1}{2}\sqrt{b'^2 + (c' - a')^2}\,,$$

$$I_{\min} = \frac{1}{2}(a' + c') - \frac{1}{2}\sqrt{b'^2 + (c' - a')^2}\,.$$

The ratio of $I_{\min}$ to $I_{\max}$ is a factor that depends on the shape of the object. It will be equal to one for a centrally symmetric object like a circular disc and near zero for a highly elongated object. Note that we need all three second-order moments to compute a "shape factor."

So-called *moment invariants* are combinations of moments that are independent of translation and rotation of the object region in the image [Cagney & Mallon 1986]. The second order moment invariants are all combinations of the minimum and maximum inertia. There are thus only two degrees of freedom. One may choose any convenient combinations, such as

$$I_{\max} + I_{\min} = a' + c' ,$$

$$(I_{\max} - I_{\min})^2 = b'^2 + (c' - a')^2 .$$

These invariants are sometimes used in recognition.

Additional comments and higher moments

In practice the double integrals that apply in the continuous domains are replaced by double sums, in the obvious way. So the area, for example, is just a multiple of

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} b_{i,j} .$$

The second-order moments a', b', and c', relative to the centroid $(\bar{x}, \bar{y})$, can be computed from the moments a, b, and c relative to the (arbitrary) origin of the coordinate system, provided that the zeroth and first-order moments are known:

$$a' = a - A\,\bar{x}^2, \quad b' = b - A\,\bar{x}\bar{y}, \quad \text{and} \quad c' = c - A\,\bar{y}^2 .$$

Still higher moments may be used to get more detailed descriptions of the shape. Also, as noted, the axis of least inertia leaves an ambiguity in orientation. The third-order moments can be used to disambiguate the two possibilities.

We have assumed so far that $b(x, y)$ can only take on two values. It should be obvious that the same analysis holds when $b(x, y)$ is not binary (yet independent of accidents of lighting and viewing geometry). This may be advantageous, for example, when one has a coarsely sampled image, in which case the position and orientation of the part may not be determined very accurately from a mere binary image because of aliasing problems. Intermediate grey-levels on the boundary of the object can provide information that allows one to determine the position and orientation to much higher precision.

Resistive networks for moment calculation

If area and center of area are all we are computing, then a very simple scheme can be used. Consider first a regular one-dimensional chain of N resistors each of resistance R. We can use such a simple resistive chain to generate potentials at each node linearly related to the position. This potential can then be used in further calculation—to generate a current injected into a global buss [Horn 1988]. Now consider a different way of using the very same chain. Suppose that the chain is grounded at each end, and that we can measure the currents I_l and I_r flowing into the ground at these points. There are k resistors to the left and $(N-k)$ to the right of the k-th node. Suppose a potential V develops at the k-th node when we inject a current I there. Clearly

$$I_l = \frac{V}{kR} \quad \text{and} \quad I_r = \frac{V}{(N-k)R},$$

while the total current is

$$I = I_l + I_r = \frac{N}{k(N-k)}\frac{V}{R},$$

so that

$$\frac{I_l}{I} = \frac{N-k}{N} \quad \text{and} \quad \frac{I_r}{I} = \frac{k}{N}.$$

We can compute the "centroid" of these two currents:

$$\bar{x} = x_l\frac{I_l}{I} + x_r\frac{I_r}{I} = x_l + \frac{k}{N}(x_r - x_l),$$

which is the x coordinate of the place where the current was injected. If we inject currents at several nodes, we can show, using superposition, that the computation above yields the centroid of the injected currents.

Now imagine a regular two-dimensional resistive grid grounded on the boundary. Current is injected at each picture cell where $b(x,y) = 1$. The currents to ground on the boundary from the network are measured. The total current obviously is proportional to the area, that is, the number of picture cells where $b(x,y) = 1$. More importantly, the center of area of the current distribution on the boundary yields the center of area of the injected current distribution.

To see why, consider a uniform resistive sheet covering the region D, grounded on the boundary ∂D. Current $i(x,y)$ per unit area is injected into the sheet at the point (x,y), where the potential is $v(x,y)$. The potential satisfies Poisson's equation

$$\Delta v(x,y) = -\rho\, i(x,y),$$

where ρ is the resistivity (per unit square). Now consider the current density per unit length extracted from the sheet at the boundary:

$$j(x,y) = -\rho\frac{\partial v}{\partial n},$$

where the normal derivative of the potential can be defined by

$$\frac{\partial v}{\partial n} = \frac{\partial v}{\partial x}\frac{dy}{ds} - \frac{\partial v}{\partial y}\frac{dx}{ds},$$

with the tangent to the boundary given by

$$\left(\frac{dx}{ds}, \frac{dy}{ds}\right)^T.$$

It is clear that the total current injected into the sheet must equal the total current leaving through the boundary. We can show this formally using the two-dimensional version of Green's formula [Korn & Korn 1968]:

$$\iint_D (u\Delta v - v\Delta u)\, dA = \int_{\partial D} \left(u\frac{\partial v}{\partial n} - v\frac{\partial u}{\partial n}\right) ds,$$

with $v = v(x, y)$ and $u(x, y) = 1$. We obtain

$$\iint_D \Delta v\, dA = \int_{\partial D} \frac{\partial v}{\partial n}\, ds,$$

or

$$\iint_D i(x, y)\, dA = \int_{\partial D} j(x, y)\, ds.$$

This works, of course, even when the boundary is not grounded.

Now, if we instead use $u(x, y) = x$ in Green's formula, we obtain

$$\iint_D x\, \Delta v\, dA = \int_{\partial D} \left(x\frac{\partial v}{\partial n} - v\frac{\partial x}{\partial n}\right) ds,$$

which, since $v(x, y) = 0$ on the boundary, becomes just

$$\iint_D x\, \Delta v\, dA = \int_{\partial D} x\frac{\partial v}{\partial n}\, ds,$$

so that

$$\iint_D x\, i(x, y)\, dA = \int_{\partial D} x\, j(x, y)\, ds.$$

So the first-order moment in the x-direction of the boundary current is equal to the first-order moment in the x-direction of the injected current. Similarly,

$$\iint_D y\, i(x, y)\, dA = \int_{\partial D} y\, j(x, y)\, ds.$$

The same trick can be used with any harmonic function $u(x, y)$, that is, a function for which $\Delta u = 0$.

It is easy to see that xy and $(y^2 - x^2)$ are harmonic functions, so we can compute their integrals in this fashion also:

$$\iint_D (y^2 - x^2)\, i(x, y)\, dA = \int_{\partial D} (y^2 - x^2)\, j(x, y)\, ds,$$

and

$$\iint_D xy\, i(x, y)\, dA = \int_{\partial D} xy\, j(x, y)\, ds.$$

Now the first of these integrals corresponds to $(c - a)$, while the second corresponds to b in the calculation of orientation. This means that we can obtain the position and orientation of a region just from the currents on the boundary of the resistive network.

Note, however, that we cannot obtain all three second-order moments *independently* from the boundary currents. We only obtain one of the two second order moment invariants. Consequently, we also cannot compute a shape factor from the boundary currents.

The two-dimensional Laplacian operator can be written in polar form as

$$\Delta u = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2},$$

so we see that

$$u_k = r^k \cos(k\theta) \quad \text{and} \quad v_k = r^k \sin(k\theta),$$

are two families of harmonic functions. We have used the first few members of these sets already, namely,

$$1, \quad x = r\cos\theta, \quad y = r\sin\theta, \quad x^2 - y^2 = r^2\cos 2\theta, \text{ and } 2xy = r^2 \sin 2\theta.$$

The next pair of harmonic functions one could use are the monkey-saddle functions

$$x^3 - 3xy^2 \quad \text{and} \quad 3x^2y - xy^2.$$

Continuing in this way, we see that one can compute two combinations of each of the $(n + 1)$ moments of n-th order from the boundary currents. We cannot compute all of the moments independently. For purposes of determining the position and orientation, however, we only need the first few.

Implementation details and previous work

To obtain the required combinations of moments, we have to integrate the product of the boundary current with

$$1, \quad x, \quad y, \quad (x^2 - y^2) \quad \text{and} \quad 2xy.$$

The first is just the total current flowing out of the resistive network. The computation of the rest will be affected somewhat by the shape chosen for the resistive network. In the case of a circular image region, for example, we multiply the currents by weights that vary as

$$1, \quad \cos\theta, \quad \sin\theta, \quad \cos 2\theta \quad \text{and} \quad \sin 2\theta,$$

where θ is the angle measured from the center of the image. Note that the weights are fixed for each point on the boundary. The computation may be simplified by using a square boundary, but at the cost of loss of rotational symmetry.

There has been considerable work on finding moments using digital means. Special purpose systems have been developed for tracking objects using these schemes [Gilbert *et al.* 1980; Gilbert 1981]. Also, a number of special purpose digital signal processing systems have been built to compute moments. Some of these systems have much of the required circuitry on a single digital chip [Hatamian 1986, 1987]. Furthermore, a discrete analog chip has been built that determines the centroid using a gradient descent method [DeWeerth & Mead 1988]. With considerable increase in circuit complexity this could perhaps be extended to also determine orientation using the approach described in the first part of this section.

There also exists a continuous analog light-spot position sensor that uses a method similar to the one described above (Selspot Systems). It consists of a single, large, square photo-diode and some electronics. Electrodes are attached on four edges of the "lateral effect" photo-diode and four operational amplifiers are used to measure the short-circuit current out of each of the four edges. The total current is just the integral of the signal. The ratio of the difference to the sum of the currents on opposite edges gives the position of the centroid in one direction. The currents in the other two edges give the other component of the centroid.

Apparently the possibility of computing combinations of higher moments from the boundary currents, and thus determining orientation also, has not previously been noted.

A network equivalence theorem

In the above we have discussed two apparently quite different ways of using a simple resistive network:

- Apply a given potential distribution along the edge of the network and use the open-circuit potentials at interiors nodes in further calculation, and
- Inject currents at interior nodes and use the measured short-circuit currents on the edge in further calculation.

There is an intimate relationship between these two ways of using a resistive network. In some cases one of the two schemes leads to much simpler implementation than the other, so it is important to understand the equivalence. This will now be explored in more detail for arbitrary networks of resistors.

Consider a resistive network with external nodes segregated into two sets A and B of size N and M respectively. Now perform two experiments:

1 Connect the nodes in group A to voltage sources with potentials V_n for $n = 1, 2, \ldots, N$ and measure the resulting open-circuit potentials on the nodes in group B. Let these be called v_m, for $m = 1, 2, \ldots, M$.

2 Connect the nodes in group B to current sources with currents i_m, for $m = 1, 2, \ldots, M$, and measure the short-circuit currents in the nodes of group A. Let these be called I_n for $n = 1, 2, \ldots, N$.

Then

$$\boxed{\sum_{n=1}^{N} I_n V_n = \sum_{m=1}^{M} i_m v_m}$$

Proof: Consider in case 1 that we apply a potential only to node n in group A, that is, $V_k = 0$ for $k \neq n$. Let the resulting open-circuit potential on node m in group B be called $v_{m,n}$. We note that superposition tells us that the potential on node m in group B when potentials are applied to *all* of the nodes in group A is

$$v_m = \sum_{n=1}^{N} v_{m,n} .$$

Next, consider in case 2 that we inject current only at node m in group B, that is $i_l = 0$ for $l \neq m$. Let the resulting short-circuit current at node n in group A be called $I_{n,m}$. We note that superposition tells us that the current in node n of group A when currents are injected into *all* of the nodes of group B is

$$I_n = \sum_{m=1}^{M} I_{n,m} .$$

The reciprocity theorem tells us that

$$I_{n,m} V_n = i_m v_{m,n} .$$

Now sum over all of the nodes in group A:

$$\sum_{n=1}^{N} I_{n,m} V_n = \sum_{n=1}^{N} i_m v_{m,n} ,$$

or

$$\sum_{n=1}^{N} I_{n,m} V_n = i_m \sum_{n=1}^{N} v_{m,n} = i_m v_m .$$

Then sum over all of the nodes in group B:

$$\sum_{m=1}^{M} \sum_{n=1}^{N} I_{n,m} V_n = \sum_{m=1}^{M} i_m v_m ,$$

or

$$\sum_{n=1}^{N} \sum_{m=1}^{M} V_n I_{n,m} = \sum_{n=1}^{N} V_n \sum_{m=1}^{M} I_{n,m} = \sum_{m=1}^{M} i_m v_m ,$$

or, finally

$$\sum_{n=1}^{N} I_n V_n = \sum_{m=1}^{M} i_m v_m .$$

Application

One application of this theorem is in the simplification of circuits for the analog computation of some weighted average. Suppose that we have a resistive network that is used to compute some quantities v_m (for example, a potential representing the x position in an image) from some fixed inputs V_n (for example, potentials representing x on the edge of the resistive network). These potentials are then used to compute a weighted average like

$$\overline{v} = \frac{\sum_{m=1}^{M} i_m v_m}{\sum_{m=1}^{M} i_m} ,$$

where the quantities i_m are the weights (for example, image brightness).

Then an equivalent way of obtaining the same result is to inject currents proportional to i_m into the resistive network, now grounded in the places where inputs where applied earlier. Let the currents at the places where the network is grounded be I_n. Then the same weighted average can be obtained by computing instead

$$\overline{V} = \frac{\sum_{n=1}^{N} I_n V_n}{\sum_{n=1}^{N} I_n} .$$

Which of the two schemes is simpler depends on details of the implementation, including the relative sizes of N and M.

Example

In the one dimensional version of the centroid-finding chip, a potential representing x is generated from two fixed input potentials applied at either end of a uniform resistive chain. An output current proportional to the product of the light intensity at a picture cell and the local value of x is injected into a global bus. The weighted average of the potentials at the picture cells can then be computed from this current and a current proportional to the total brightness

$$\overline{v} = \frac{\sum_{m=1}^{M} v_m i_m}{\sum_{m=1}^{M} i_m} .$$

This allows us to determine the x position of the centroid of the light spot

$$\overline{x} = \frac{(V_2 - \overline{v})x_1 + (\overline{v} - V_1)x_2}{V_2 - V_1} ,$$

where x_1 and x_2 are the coordinates at either end of the resistive chain, at the points where the potentials V_1 and V_2 are applied.

The computation can also be performed by injecting currents proportional to the brightness at each picture cell into the same uniform linear resistive chain now grounded at either end. The centroid can be computed from the currents flowing into ground at the ends:

$$\overline{x} = \frac{x_1 I_1 + x_2 I_2}{I_1 + I_2} .$$

In this particular case, the second scheme appears to be simpler.

For generalizations of these ideas to the continuous domain, see Horn [1988].

Short Range Motion Vision Methods

Attacks on the motion vision problem can be categorized in a number of ways. First of all, there is the question of how large a change between successive images the method is meant to deal with. Feature-based methods appear to be best suited for the so-called *long-range* motion vision problem, where there is a relatively large change between images. Conversely, these methods generally are not good at estimating motions with subpixel accuracy. Feature-based methods essentially solve the correspondence problem, which is the central problem in binocular stereo. Unfortunately, the problem in motion vision is typically even harder than the binocular stereo problem, because the search for a match is not confined to an epipolar line.

Gradient-based methods are better suited to situations where the motion between successive images is fairly small, that is, the *short-range* motion vision problem. Correlation methods appear to fall somewhere in between, because they cannot deal with significant changes in foreshortening or photometric changes, yet are not able to easily produce displacement estimates with subpixel accuracy.

There are several different approaches to the short-range motion vision problem. Here we briefly list some based directly on brightness derivatives rather than matching of isolated features or correlation. We first discuss several methods for recovering optical flow and then go on to methods for recovering rigid body motion directly, without using optical flow as an intermediate result.

All methods for recovering motion implicitly make some assumptions about how images change when the viewer moves with respect to the scene. Simple correlation methods, for example, assume that changes in foreshortening can be ignored. This is not a good assumption in wide-baseline binocular stereo nor in some long-range motion vision applications. Feature-based methods and correlation methods also assume that

the brightness pattern does not change drastically with viewpoint. Fortunately, the brightness of many real surfaces does not depend significantly on the viewing direction for a fixed illumination geometry.

Methods based on brightness gradients implicitly assume that the variations in brightness at a particular point in the image due to motion are much larger than the brightness fluctuations induced by changes in viewpoint. This is a reasonable assumption unless the surface lacks markings and is illuminated by rapidly moving light sources. Most methods will be "fooled" by the motion of virtual images resulting from specular or glossy reflections of point light sources.

Recovering optical flow from brightness derivatives

The *motion field* is the projection in the image of velocities of points in the environment with respect to the observer. Observer motion and object shapes can be estimated from the motion field. The *optical flow* is a vector field in the image that indicates how brightness patterns move with time. The optical flow field is not unique, because the matching of points along an isophote in one image with an isophote of the same brightness in the other image is not unique. Additional constraints have to be introduced in order to select a particular "optical flow." Under favorable circumstances the optical flow so computed is a good estimate of the motion field. There are several algorithms of different complexity and robustness for estimating optical flow. At one end of the spectrum we have algorithms that assume the flow is constant over the image, at the other, there are algorithms that can deal with depth discontinuities. Many of the interesting variations are listed here in order of increasing complexity:

1 **Constant Optical Flow** [Nagel 1984; Weldon 1986]: Here the flow velocity, (u, v), is assumed to be constant over the image patch. This may be a good approximation for a small field of view. Several cameras aimed in different directions (spider head) could yield flow vectors that provide the information necessary to solve for the observer motion. Alternatively, this computation may be applied to (possibly overlapping and weighted) patches of one image. A basic least squares analysis leads to a simple algorithm. All that is required is:

   a Estimation of the brightness derivatives E_x, E_y, and E_t.

   b Accumulation of the sums of the products E_x^2, E_xE_y, E_y^2, E_xE_t, and E_yE_t, and,

   c Solution of two linear equations in the two unknowns u and v. This last step could be done off-chip, using the totals accumulated on-chip. Alternatively, the computation can be done in an iterative or feedback mode on chip (as it is in

Tanner and Mead [1987]). The bandwidth going off-chip is very low in either case.

If the computation is done for many (possibly overlapping and weighted) image windows, then an optical flow vector field results (at resolution less than the full image resolution). Such a vector field can then be processed off-chip to yield camera motion and scene structure using a least-squares method (*a lá* Bruss and Horn [1983]).

2 **Basic Optical Flow** [Horn & Schunck 1981]: Here the velocity field is allowed to vary from place to place in the image, but is assumed to vary smoothly. Depth discontinuities are not treated, but elastic deformations, fluid flows and rigid body motions yield reasonable results. The calculus of variation problem here leads to a coupled pair of Poisson's equations for $u(x, y)$ and $v(x, y)$, the components of the optical flow. The right-hand sides of these equations (that is, parts not involving u and v) are computed from the brightness derivatives. One needs to be able to compute values such as $(\alpha^2 + E_x^2 + E_y^2)$ (or approximations thereto). The partial differential equations themselves, of course, can be conveniently solved on two interlaced resistive networks. The inputs may be currents injected at nodes, while the outputs are the potentials there. The boundaries have to be treated carefully. The algorithm is robust with respect to small random errors in the resistive network. (It is not robust against round-off error in the digital version, common when the number of bits available to representing u and v are limited). As usual, there is some small advantage to working on a hexagonal grid.

3 **Optical Flow with Multiplier** [Gennert & Negahdaripour 1987]: The basic optical flow algorithm is based on the assumption that the brightness of a small patch of the surface does not change as it moves. In practice there are small brightness changes, because the shading on the surface may change slowly as a patch moves into areas that are illuminated differently. When the surface is highly textured, brightness variations at a point in the image resulting from motion are much larger than those due to changes in shading and illumination, and so these can be safely ignored. If there is no strong texture on the surface, somewhat better results can be obtained if one takes account of these small changes in shading. One can do this using a simple multiplier model. Here the brightness of a patch in a frame of an image sequence is assumed to be a multiple of the brightness of the same patch in the previous frame. The multiplier (assumed to be near unity) is allowed to vary from point to point in the image, but is assumed to vary slowly with position. The resulting calculus of variation problem now leads to three coupled partial differential equations. The new algorithm is

not much more complex (about 50% more work) than the basic one, yet yields better results.

4 **Optical Flow with Discontinuities** [Koch, Marroquin & Yuille 1986; Gamble & Poggio 1987; Hutchinson, Koch, Luo & Mead 1987; Murray & Buxton 1987]: The notion of a *line process* for dealing with discontinuities in images originated with Geman and Geman [1984]. This idea was later applied to discontinuities in optical flow by Koch, Marroquin and Yuille [1986], Hutchinson, Koch, Luo and Mead [1987], and Murray and Buxton [1987]. To deal with discontinuities in the optical flow, which typically occur at object boundaries, one introduces line processes that cut the solution and prevent smoothing over discontinuities. The resulting penalty function to be minimized is no longer convex and the solution involves more than simply solving a set of coupled partial differential equations. It seemed at first that this approach was doomed to failure, because methods like simulated annealing for solving such nonlinear problems are hopelessly inefficient on an ordinary serial computer. However, a reasonably efficient method results if one gives up the demand for the absolute global minimum and instead is satisfied with a good solution, with cost close to the absolute minimum cost [Blake & Zisserman 1988]. It helps to base the decision about whether to introduce a line process at a particular place only on the local change in the cost of the solution [Geman & Geman 1984]. Further improvements in performance can be had if line processes are allowed only very near to discontinuities in brightness, that is, edges [Gamble & Poggio 1987]. This suggests integrating some edge finding algorithm on the same chip. The approach here leads to an analog network that interacts with some logic circuits implementing the line-process decision making (see figure 5 in Koch, Marroquin and Yuille [1986]).

Often there is a concern about the rate of convergence of simple methods for solving Poisson's equation. Multi-grid methods are suggested as a means of speeding up the process. This is fortunately not so much of a concern here because:

- It is rare to have no inputs (zero right-hand side) over large patches (that is, large patches of uniform brightness are rare).
- The analog networks ought to settle fairly rapidly, even when there are many nodes because the time-constant should be small.
- Excellent starting values are available from the solution for the previous frame.

Because it is difficult to get good estimates of optical flow from noisy image data, there has been a trend recently to go directly to the ultimately desired information, namely observer motion and object shape. Instead of

computing these from a flow field, they are derived directly from image brightness and the partial derivatives of brightness. These methods also lend themselves to implementation in a parallel network (see next section). They do, however, assume rigid body motion. Thus these methods are of little use when we are dealing with elastic deformations and fluid flow. Consequently there is still strong interest in finding rapid, robust methods for estimating the optical flow.

Direct recovery of rigid body motion

It is possible to derive observer motion and object shape directly from brightness gradients using something like a least-squares approach. These methods are not as mature as those for estimating the optical flow, but may ultimately be of more interest. A number of special cases have been solved so far:

1 **Pure Rotation:** [Alomoinos & Brown 1985; Horn & Weldon 1988] In the case of pure rotation, the motion field is particularly simple because it does not depend on the distances of the observer from the objects in the scene. In this case a simple least-squares analysis leads to a set of three linear equations in the three unknown components of the angular velocity vector $\omega = (A, B, C)^T$. The coefficients of these equations are once again sums over the whole image of products of brightness derivatives and image coordinates. The algorithm is remarkably robust with respect to noise in the brightness derivatives, because the problem is so highly overdetermined (three unknowns and hundreds of thousands of measurements).

2 **Pure Translation:** [Horn & Weldon 1988] In the case of pure translation, the task is to recover the direction of the translation vector. The *focus of expansion* is the intersection of this vector with the image plane, that is, it is the image of the point towards which the observer is moving. Once the focus of expansion has been located, relative distances of selected points in the scene (where the brightness gradient is large enough in the direction towards the focus of expansion) can be estimated. (One simply divides the rate of change of brightness in the direction towards the focus of expansion by the time rate of change of brightness.) There are several methods for recovering the direction of translation. The most promising at this point requires eigenvector-eigenvalue decomposition of a 3×3 matrix constructed using sums of products of brightness derivatives and image coordinates. These sums could be computed on-chip, with the final analysis being done off-chip. This algorithm is not nearly as robust as the one for pure rotation, because there are now an enormous number of additional "unknowns,"

namely the distances to the scene at each picture cell. For the same reason this algorithm is much more interesting because it allows us to recover depth and thus obtain surface shape information.

3 **Planar Surface:** [Horn & Negahdaripour 1987] If the scene consists of a single planar surface (perhaps an airport viewed from a landing aircraft), it is possible to compute the direction of translation, the orientation of the plane, the rotational velocity of the observer, as well as the time to impact, directly from certain sums accumulated over the whole image. There is a two-way ambiguity in the result that can be resolved using other sensory information or by waiting for new solutions based on subsequent frames. The sums required are "moments," products of the partial derivatives of brightness (E_x, E_y, and E_z) and the image coordinates x, and y. The final calculation involves eigenvector-eigenvalue decomposition of a 3×3 matrix constructed using these sums, but this can be done off-chip. Both closed form and iterative solutions are known. There are quite a large number of different sums needed, but each is relatively simple to compute.

4 **Other Constraints on Motion:** E. J. Weldon and his students at the University of Hawaii have been investigating a number of other special restrictions on motion. A wheeled vehicle moving in contact with a smooth surface is confined to translation in the local tangent plane and rotation about the local normal. Thus the rotation vector has to be perpendicular to the translation vector. This constraint allows a solution of the motion vision problem that takes a form very similar to the one discussed above. Another interesting special case arises when the vehicle can rotate only about an axis parallel to the translational vector. There is also strong interest in exploiting fixation or tracking. If one fixates on a point in the moving environment, a constraint is introduced between the instantaneous rotational and translational velocities of the observer relative to the environment. This allows one to simplify the motion constraint equation and reduces the problem to something similar to that of pure translation.

The general case (arbitrary surface, both translation and rotation) has not been solved yet. Also, the pure translation solutions are not very robust, suggesting that one needs to continue the solution in time in order to get stable results (all of the methods discussed above work "instantaneously" using two image frames, and do not make much use of information in earlier frames).

In the case of pure translation, depth is recovered only in places where the local brightness gradient is strong enough in the direction towards the focus of expansion. This suggests the need for a smooth interpolation process that fills in the rest. It might take the form of the solution of Poisson's

equation or the bi-harmonic equation. A simple passive network will do for Poisson's equation, of course. If the higher order approach is taken, negative resistances and more connections are required. It is possible, however, as we saw earlier, to decompose the bi-harmonic equation into coupled Poisson's equations. The latter can then be solved using coupled resistive network.

Finally, to deal with depth-discontinuities, one can introduce line-processes once again. Naturally, we are now talking about a very complicated system!

Constant flow velocity

The method that assumes that optical flow is constant in a patch will be considered next, as a simple illustration of the kind of approach taken. First we review the brightness change constraint equation. Image brightness $E(x, y, t)$ is a function of three variables. If the brightness of a small patch does not change as it moves, we can write:

$$\frac{dE}{dt} = 0\,,$$

which can be expanded to yield:

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0\,,$$

or

$$uE_x + vE_y + E_t = 0\,,$$

where E_x, E_y are the components of the brightness gradient, while E_t is the time rate of change of brightness. This so-called *brightness change constraint equation* provides only one constraint on the two components of image flow, u and v. Thus image flow cannot be recovered locally without further information.

Suppose now that the image flow components u and v are constant over a patch in the image. Then we can recover them using a least squares approach: We minimize the total error

$$I = \iint_D (uE_x + vE_y + E_t)^2 \, dx\, dy\,.$$

Differentiation with respect to u and v leads to

$$\frac{dI}{du} = \iint_D (uE_x + vE_y + E_t)\, E_x \, dx\, dy\,,$$

$$\frac{dI}{dv} = \iint_D (uE_x + vE_y + E_t)\, E_y \, dx\, dy\,.$$

Setting these derivatives equal to zero, we obtain

$$u \iint_D E_x^2 + v \iint_D E_x E_y = - \iint_D E_x E_t \,,$$
$$u \iint_D E_y E_x + v \iint_D E_y^2 = - \iint_D E_y E_t \,.$$

These are two linear equations that can be easily solved for u and v.

$$D\,u = \iint_D E_y^2 \iint_D E_x E_t - \iint_D E_x E_y \iint_D E_y E_t \,,$$

and

$$D\,v = \iint_D E_x E_y \iint_D E_x E_t - \iint_D E_y^2 \iint_D E_y E_t \,,$$

where D is the determinant of the coefficient matrix, that is,

$$D = \iint_D E_x^2 \iint_D E_y^2 - \left(\iint_D E_x E_y \right)^2 .$$

The coefficients are easily calculated in parallel, if so desired.

While this closed form solution is very appealing in a sequential digital implementation, it involves division and other operations that are not particularly easily carried out in analog circuitry. In this case, an iterative or feedback strategy may be favored. Using a gradient descent approach, we arrive at

$$\frac{du}{dt} = -\alpha \iint_D (uE_x + vE_y + E_t)\, E_x \, dx\, dy \,,$$
$$\frac{dv}{dt} = -\alpha \iint_D (uE_x + vE_y + E_t)\, E_y \, dx\, dy \,.$$

At each picture cell, we estimate the derivatives of brightness, and compute the error in the brightness change constraint equation

$$e = (uE_x + vE_y + E_t) \,,$$

using global buses whose potentials represent u and v. Currents proportional to $-e\,E_x$ and $-e\,E_y$ are injected into the buses for u and v respectively. This is essentially how the constant flow velocity chip of Tanner [1986] and Tanner Mead [1987] works.

Special purpose direct motion vision systems

We have seen that in short-range motion vision one need not solve the correspondence problem. One can instead use derivatives of image brightness directly to estimate the motion of the camera. The time rate of change of image brightness at a particular picture cell can be predicted if the brightness gradient and the motion of the pattern in the image is known. This two-dimensional motion of patterns in the image, in turn, can be predicted

if the three-dimensional motion of the camera is given. Given these facts, it should be apparent that the motion of the camera can be found by finding the motion that best predicts the time rate of change of brightness (t-derivative) at all picture cells, given the observed brightness gradients (x- and y-derivatives). Once the instantaneous rotational and translational motion of the camera have been found, one can determine the depth at points where the brightness gradient is large and oriented appropriately.

As discussed above, several special situations have already been dealt with, including the case where the camera is known to be rotating only, the case where the camera is translating only, and the case of arbitrary motion where the surface being viewed is known to be planar. The solution in the case of pure rotation is very robust against noise (because there are only three unknowns and thousands of constraints) and so well worth implementing. The solution in the case of arbitrary motion with respect to a planar surface is also quite robust, although it is subject to a two-way ambiguity. In this case there are eight unknowns (the rotational velocity, the translational velocity and the unit surface normal). The solution in the case of pure translation is more sensitive to noise (because there are about as many unknowns as constraints), but of great interest, because depth can be recovered. An elegant solution to the general case has not yet been found. It can, however, be expected that it will not be less robust than the pure translation case (because there are only three more unknowns).

We will now describe in detail a method for the solution of the pure rotation case and a method for the solution of the pure translation case. We saw earlier that if the brightness of a patch does not change as it moves, we obtain the brightness change constraint equation

$$uE_x + vE_y + E_t = 0\,,$$

where E_x, E_y are the components of the brightness gradient, while E_t is the time rate of change of brightness. This equation provides one constraint on the image flow components u and v. Thus image flow cannot be recovered locally without additional constraint.

We are now dealing, however, with rigid body motion, where image flow is heavily constrained. The image flow components u and v dependent on the instantaneous translational and rotational velocities of the camera, denoted $\mathbf{t} = (U, V, W)^T$ and $\boldsymbol{\omega} = (A, B, C)^T$ respectively. It can be shown by differentiating the the equation for perspective projection [Longuett-Higgins & Prazdny 1980], that

$$u = \frac{-U + xW}{Z} + A\,xy - B(1 + x^2) + C\,y\,,$$

$$v = \frac{-V + yW}{Z} + A(1 + y^2) - B\,xy - C\,x\,,$$

where Z is the depth (distance along the optical axis) at the image point (x, y). Combining this with the brightness change constraint equation, we

obtain [Horn & Weldon 1988]

$$E_t + \mathbf{v} \cdot \boldsymbol{\omega} + \frac{1}{Z}\mathbf{s} \cdot \mathbf{t} = 0\,,$$

where

$$\mathbf{v} = \begin{pmatrix} +E_y + y(xE_x + yE_y) \\ -E_x - x(xE_x + yE_y) \\ yE_x - xE_y \end{pmatrix},$$

and

$$\mathbf{s} = \begin{pmatrix} -E_x \\ -E_y \\ xE_x + yE_y \end{pmatrix}.$$

This is called the *rigid body brightness change constraint equation.*

Feedback computation of instantaneous rotational velocity

Horn and Weldon [1988] rediscovered a method apparently first invented by Alomoinos and Brown [1985] for direct motion vision in the case of pure rotation. This method uses integrals of products of first partial derivatives of image brightness and image coordinates and involves the solution of a system of three linear equations in three unknowns. When there is no translational motion, the brightness change constraint equation becomes just

$$E_t + \mathbf{v} \cdot \boldsymbol{\omega} = 0\,.$$

This suggests a least-squares approach, where we minimize

$$I = \iint_D (E_t + \mathbf{v} \cdot \boldsymbol{\omega})^2 \, dx\, dy\,,$$

by suitable choice of the instantaneous rotational velocity $\boldsymbol{\omega}$. This leads to the simple equation

$$\left(\iint_D \mathbf{v}\mathbf{v}^T \, dx\, dy \right) \boldsymbol{\omega} = - \iint_D E_t \mathbf{v} \, dx\, dy\,.$$

This vector equation corresponds to three scalar equations in the three unknown components A, B, and C of the instantaneous rotational velocity vector. The system of linear equations can be solved explicitly, but this involves division by the determinant of the coefficient matrix. When considering analog implementation, it is better to use a resistive network to solve the equations. Yet another attractive alternative is to use a feedback scheme (not unlike the one used to solve for the optical flow velocity components in the case when they are assumed to be constant over the image patch being considered).

Finally, the solution can be obtained by walking down the gradient of the total error. The derivative with respect to $\boldsymbol{\omega}$ of the sum of squares of

errors is just

$$\frac{dI}{d\boldsymbol{\omega}} = 2 \iint_D (E_t + \mathbf{v} \cdot \boldsymbol{\omega})\mathbf{v} \, dx \, dy \, .$$

This suggest a feedback scheme described by the equation

$$\frac{d\boldsymbol{\omega}}{dt} = -\alpha \iint_D (E_t + \mathbf{v} \cdot \boldsymbol{\omega})\mathbf{v} \, dx \, dy \, .$$

The idea revolves around a bus, with potential on three wires proportional to the present estimates of the components A, B and C of the instantaneous angular velocity $\boldsymbol{\omega}$. Estimates of the partial derivatives of image brightness (the components of the brightness gradient and the time rate of change of brightness) are computed at each picture cell. From them, and the position (x, y) of the cell, one can compute $\mathbf{v}$. The coordinates x and y can be made available to each cell using resistive chains that are connected to fixed potentials on the sides of the chip. (It may be useful also to directly supply xy, $(1 + x^2)$ and $(1 + y^2)$, because these are coefficients in the expression for $\mathbf{v}$).

Next, one computes the error term

$$e = E_t + \mathbf{v} \cdot \boldsymbol{\omega} \, ,$$

which, in the absence of noise, is zero when the correct solution has been found. Currents are fed into the bus proportional to

$$-e\mathbf{v} = -(E_t + \mathbf{v} \cdot \boldsymbol{\omega})\mathbf{v} \, .$$

Each of the three bus wires is terminated in a capacitance. We now have a system that obeys an equation like

$$\frac{d\boldsymbol{\omega}}{dt} = -\alpha \iint_D (E_t + \mathbf{v} \cdot \boldsymbol{\omega})\mathbf{v} \, dx \, dy \, ,$$

the steady state solution of which is

$$\iint_D (E_t + \mathbf{v} \cdot \boldsymbol{\omega})\mathbf{v} \, dx \, dy = \mathbf{0} \, ,$$

or

$$\left(\iint_D \mathbf{v}\mathbf{v}^T \, dx \, dy \right) \boldsymbol{\omega} = - \iint_D E_t \mathbf{v} \, dx \, dy \, .$$

The feedback scheme involves considerably less computation than the closed form solution (for example, we do not have to compute the 3×3 matrix $\mathbf{v}\mathbf{v}^T$). Also, the feedback scheme can be shown to be stable (as long as the integral of $\mathbf{v}\mathbf{v}^T$ is not singular, that is, as long as there is sufficient contrast in the image texture).

The elementary components needed are the sensors, differential buffer amplifiers that estimate spatial derivatives, approximate time delays for estimating the temporal derivative, four-quadrant analog multipliers, and current sources. There also will be resistive chains to supply values of x and y at each image location.

Computation of instantaneous translational velocity

While the scheme described above for recovering the rotational velocity is very robust as shown both by sensitivity analysis and experimentation on computers with both synthetic and real images, it is does not allow us to recover depth. This is because there is no dependence of the brightness derivatives on depth when there is no translational motion. We now consider the other extreme, when there is only translational motion.

When there is no rotational motion, the brightness change constraint equation becomes just

$$E_t + (\mathbf{s} \cdot \mathbf{t})\frac{1}{Z} = 0 .$$

Note that multiplying both Z and $\mathbf{t}$ by a constant does not perturb the equality. This tells us right away that there will be a scale factor ambiguity in recovering motion and depth. We take care of this by attempting only to recover the direction of motion. That is, we will treat $\mathbf{t}$ as a unit vector.

We can solve the constraint equation above for the depth Z in terms of the unknown motion parameters. We obtain

$$Z = -\frac{\mathbf{s} \cdot \mathbf{t}}{E_t} .$$

If our estimate of the instantaneous translational motion $\mathbf{t}$ is incorrect, we will obviously obtain incorrect values for the depth from this equation. Some of these values may be negative (which correspond to points on objects behind the camera), while others will be unexpectedly large. Some methods have been explored that to find a direction of translational motion that yields the smallest number of negative depth values when applied to the image brightness gradients [Horn & Weldon 1988]. Although these methods work, they have yet to show promise in terms of computational expediency. We consider another approach next.

In many cases, particularly in industrial robotics, the depth range is bounded and the occurrence of very large depth values is not normally anticipated. One method for estimating the instantaneous translation velocity makes use of this observation.[11] We essentially look for a translational velocity $\mathbf{t}$ that keeps Z small at most points in the image. Suppose, for example, that we find the translational velocity that minimizes

$$I = \iint_D Z^2 \, dx \, dy = \iint_D \frac{(\mathbf{s} \cdot \mathbf{t})^2}{E_t^2} \, dx \, dy ,$$

subject to the constraint that $\mathbf{t}$ be a unit vector. We cannot measure brightness exactly, so there will be some error in our estimate of E_t. To avoid problems due to noise in places where E_t is almost zero, we may

[11] The derivation of the method in terms of a minimization of the integral of Z^2 is merely an explanatory artifice. There is a way of arriving at the same result in a way that does not appear to be this *ad hoc* [Horn & Weldon 1988].

introduce an offset in the denominator as follows:

$$I = \iint_D w(E_t)\,(\mathbf{s}\cdot\mathbf{t})^2\,dx\,dy\,,$$

where $w(E_t) = 1/(E_t^2 + \epsilon^2)$. This integral can also be written in the form

$$I = \mathbf{t}^T \left(\iint_D w(E_t)\,\mathbf{s}\mathbf{s}^T\,dx\,dy \right) \mathbf{t} = \mathbf{t}^T\,S\,\mathbf{t}\,,$$

where S is a 3×3 matrix. The expression for I is clearly a quadratic form in $\mathbf{t}$. Given the constraint that $\mathbf{t}$ be a unit vector, such a quadratic form attains its minimum when $\mathbf{t}$ is the eigenvector of the matrix S corresponding to the smallest eigenvalue (see the discussion of Raleigh's quotient in Korn and Korn [1968]). Analog circuits can be devised to compute these eigenvectors [Horn 1988].

It is also possible to deal with a situation where motion can be arbitrary (that is, both rotation and translation), but the surface shape is constrained. While this problem has a closed form solution [Horn & Negahdaripour 1987], it turns out to be much easier to use gradient descent [Horn 1988]. The circuitry for this begins to be more complex, with several four-quadrant multipliers needed at each picture cell.

Gradient descent methods in general are very appealing when one is thinking about analog implementation. Fortunately it is possible to deal with *constrained* minimization as well as unconstrained minimization using either gradient projection [Horn 1988] or a reversal of the gradient component corresponding to the Lagrangian multiplier, combined with additional penalty functions [Platt & Barr 1988].

Summary and Conclusions

A number of problems in early vision have been explored here and shown to lead to interesting analog networks. The focus was on implementations involving resistive networks, perhaps with capacitors and analog multipliers, as well as simple amplifiers. In several cases, feedback schemes were shown to be considerably simpler to implement than circuits based on the closed form solutions usually sought for in digital implementations. Simple feedback networks with local connections can invert local operations [Horn 1974]. This is of interest because the inverses of local operations typically are global, and direct implementation of these inverses would require unimplementably high wiring densities.

A theorem giving an equivalence between two apparently quite different ways of using the same resistive network sometimes allows one to find a way of implementing a particular computation that is much simpler than the obvious direct implementation. Gradient projection was mentioned as a way of solving constrained minimization problems, although

in several cases it was possible to avoid this added complication through judicious normalization of the terms to be minimized and addition of a penalty term.

Also described here is a novel way of interlacing the nodes of a three-dimensional multi-resolution network in a two-dimensional tessellation. The number of nodes decreases from layer to layer by subsampling after low-pass filtering. Each layer contains half the number of nodes in its predecessor.

Using a spatial dimension to represent time in a partial differential equation was shown to lead to new ways of implementing certain convolutional algorithms that would otherwise require a clocked architecture. In this alternate scheme, image data flows in continuously on one end, while processed information flows continuously out the other end.

It is clear that many early vision problems lend themselves to implementation in parallel analog networks. This applies particularly to so-called *direct* methods, as opposed to *feature-based* methods, because the direct methods deal mostly with quantities connected to measurements at individual picture cells as well as their relationship to values at neighboring picture cells. Work on analog methods for early vision, started more than twenty years ago. It has now received a strong new impetus from the more general availability of facilities for integrated circuit design and fabrication. This renewed interest is reflected in the pioneering work at Caltech in Carver Mead's group [Mead 1989]. But no one should think that the methods explored there, or the ideas collected here, comprise anything more than an extremely sparse sampling of what is yet to come.

References

Abdou, I. E., and K. Y. Wong [1982], "Analysis of Linear Interpolation Schemes for Bi-Level Image Applications," *IBM Journal of Research and Development*, vol. 26, no. 6, pp. 667–686, (see Appendix).

Ahuja, N., and B. J. Schachter [1983], *Pattern Models*, John Wiley, New York, NY.

Alomoinos, Y., and C. Brown [1984], "Direct Processing of Curvilinear Motion from Sequence of Perspective Images," *Proceedings of Workshop on Computer Vision Representation and Control*, Annapolis, Maryland.

The author wishes to acknowledge helpful discussions with Robert Floyd, John Harris, Christof Koch, Jim Little, Carver Mead, Tomaso Poggio, David Standley, and John Wyatt.

This chapter describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided by a grant from the National Science Foundation, Number MIP-8814612 and by Du Pont Corporation.

Anderson, B. O., and J. B. Moore [1979], *Optimal Filters*, Prentice-Hall, Englewood Cliffs, NJ.

Bachmann, B. L. [1977], "Computer Correlation of Real and Synthetic Terrain Photographs," B.S. Thesis, Department of Electrical Engineering and Computer Science.

Bernstein, R. [1976], "Digital Image Processing of Earth Observation Sensor Data," *IBM Journal of Research and Development*, pp. 40-57, (see Appendix).

Berzins, V. [1984], "Accuracy of Laplacian Edge Detectors," *Computer Vision, Graphics and Image Processing*, vol. 27, no. 2, pp. 195–210.

Blake, A., and A. Zisserman [1988], *Visual Reconstruction*, MIT Press, Cambridge, MA.

Bruss, A. R., and B. K. P. Horn [1983], "Passive Navigation," *Computer Vision, Graphics, and Image Processing*, vol. 21, no. 1, pp. 3–20.

Cagney, F., and J. Mallon [1986], "Real-Time Feature Extraction using Moment Invariants," *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision*, October 28–31, Cambridge, MA, vol. 726, pp. 120–124.

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

Courant, R., and D. Hilbert [1953], *Methods of Mathematical Physics*, vol. I, John Wiley and Sons, New York, NY.

DeWeerth, S. P., and C. A. Mead [1988], "A Two-Dimensional Visual Tracking Array," *Proceedings of the 1988 MIT Conference on Very Large Scale Integration*, MIT Press, Cambridge, MA, pp. 259–275.

Floyd, R. W. [1987], private communication, June.

Gamble, E., and T. A. Poggio [1987], "Visual Integration and Detection of Discontinuities: The Key Role of Intensity Edges," Report AIM-970, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Geman, S., and D. Geman [1984], "Stochastic Relaxation, Gibbs' Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721–741.

Gennert, M. and S. Negahdaripour [1987], "Relaxing the Constant Brightness Assumption in Computing Optical Flow," Report AIM-975, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Gilbert, A. L. [1981], "Video Data Conversion and Real-Time Tracking," *IEEE Computer*, pp. 50–56.

Gilbert, A. L., M. K. Giles, G. M. Flachs, R. B. Rogers, and Y. H. U [1980], *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 2, no. 1, pp. 47–56.

Goldfinger, A. M. [1983], "Smooth Interpolation of Digital Terrain Models from Contour Maps," B.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Grimson, W. E. L. [1981], *From Images to Surfaces—A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA.

Grimson, W. E. L. [1982], "A Computational Theory of Visual Surface Interpolation," *Philosophical Transactions of the Royal Society B*, vol. 298, pp. 395–427.

Grimson, W. E. L. [1983], "An Implementation of a Computational Theory of Visual Surface Interpolation," *Computer Vision, Graphics and Image Processing*, vol. 22, pp. 39–69.

Grzywacs, N., and A. Yuille [1987], "Massively Parallel Implementations of Theories for Apparent Motion," Report AIM-888, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Hartley, R. [1985], "A Gaussian-Weighted Multi-Resolution Edge Detector," *Computer Vision, Graphics and Image Processing*, vol. 30, no. 1, pp. 70–83.

Haralick, R. M. [1984], "Digital Step Edges from Zero Crossings of Second Directional Derivatives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 1, pp. 113–129.

Harris, J. G. [1986], "The Coupled Depth/Slope Approach to Surface Reconstruction," Report AIM-908, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA. Also [1987] *Proceedings of the IEEE International Conference on Computer Vision*, London, England, pp. 277–283.

Harris, J. G. [1989], "An Analog VLSI Chip for Thin-Plate Surface Interpolation," *Proceedings of IEEE Neural Information Processing Systems Conference*, Denver, CO.

Hatamian, M. [1986], "A Real-Time Two-Dimensional Moment Generating Algorithm and Its Single Chip Implementation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 3, pp. 546–553.

Hatamian, M. [1987], "A Fast Moment Generating Chip," *Proceedings of the International Conference on Digital Signal Processing*, Florence, Italy, pp. 230–234.

Hildreth, E. [1980], "Implementation of A Theory of Edge Detection," Report AIM-579.

Hildreth, E. [1983], "The Detection of Intensity Changes by Computer and Biological Vision Systems," *Computer Vision, Graphics and Image Processing*, vol. 22, no. 1, pp. 1–27.

Horn, B. K. P. [1971], "The Binford-Horn Linefinder," Report AIM-285, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P. [1972], "VISMEM: A bag of 'robotics' formulae," Report AIW-34, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P. [1974], "Determining Lightness from an Image," in *Computer Graphics and Image Processing*, vol. 3, no. 1, pp. 277–299.

Horn, B. K. P. [1979], "Automatic Hill-Shading and the Reflectance Map," *Proceedings of the Image Understanding Workshop*, Palo Alto, CA, pp. 79–120. Also AD-A098261 available from National Technical Information Service. Also SAI-80-895-WA available from Science Application Incorporated.

Horn, B. K. P. [1981], "Hill Shading and the Reflectance Map," *Proceedings of the IEEE*, vol. 69, no. 1, pp. 14–47. Also, same title [1982] *Geo-Processing*, vol. 2, pp. 65-146.

Horn, B. K. P. [1983], "The Least Energy Curve," *ACM Transactions on Mathematical Software*, vol. 9, no. 4, pp. 441–460.

Horn, B. K. P. [1986], *Robot Vision*, MIT Press, Cambridge, MA and McGraw-Hill, New York, NY.

Horn, B. K. P. [1988], "Parallel Networks for Machine Vision," Report AIM-1071, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P. [1989], "Height and Gradient from Shading," Report AIM-1150, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P., and B. L. Bachmann [1978], "Using Synthetic Images to Register Real Images with Surface Models," *Communications of the ACM*, vol. 21, no. 11, pp. 914–924.

Horn, B. K. P., and M. J. Brooks [1986], "The Variational Approach to Shape from Shading," *Computer Vision, Graphics and Image Processing*, vol. 33, no. 2, pp. 174–208. Also [1985] Report AIM-813, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Horn, B. K. P., and M. J. Brooks [1989], *Shape from Shading*, MIT Press, Cambridge, MA.

Horn, B. K. P., and S. Negahdaripour [1987], "Direct Passive Navigation: Analytical Solution for Planes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 1, pp. 168–176.

Horn, B. K. P., and B. G. Schunck [1981], "Determining Optical Flow," *Artificial Intelligence*, vol. 16, no. 1–3, pp. 185–203.

Horn, B. K. P., and E. J. Weldon Jr. [1988], "Direct Methods for Recovering Motion," *International Journal of Computer Vision*, vol. 2, no. 1, pp. 51–76.

Hutchinson, J., C. Koch, J. Luo, and C. A. Mead [1988], "Computing Motion using Analog and Binary Resistive Networks," *IEEE Computers*, vol. 21, pp. 52-63.

Ikeuchi, K. [1984], "Reconstructing a Depth Map from Intensity Maps," *International Conference on Pattern Recognition*, Montreal, Canada, pp. 736–738. Also "Constructing a Depth Map from Images," Report AIM-744, Artificial

Intelligence Laboratory, Also AD-A135679 available from National Technical Information Service.

Knight, T. [1983], "Design of an Integrated Optical Sensor with On-Chip Pre-Processing," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Koch, C., J. Marroquin, and A. Yuille [1986], "Analog 'Neuronal' Networks in Early Vision," *Proceedings National Academy of Sciences, USA* (Biophysics), vol. 83, pp. 4263–4267. Also [1985] Report AIM-751, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Korn, G. A., and T. M. Korn [1968], *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill.

Larson, N. G., K. Nishihara and B. K. P. Horn [1981], "Digital Gaussian Convolver," Patent Application, Registry no. 26192, April 22.

Longuett-Higgins, H. C., and K. Prazdny [1980], "The Interpretation of a Moving Retinal Image," *Proceedings of the Royal Society of London B*, vol. 208, pp. 385–397.

Luo, J., C. Koch, and C. Mead [1988], "An Experimental Subthreshold, Analog CMOS two-dimensional Surface Interpolation Circuit," *Proceedings of IEEE Neural Information Processing Systems Conference*, Denver.

MacLeod, I. D. G. [1970a], "A Study in Automatic Photo-Interpretation," Ph.D. Thesis, Department of Engineering Physics, Australian National University, Canberra, Australia.

MacLeod, I. D. G. [1970b], "On Finding Structure in Pictures," in *Picture Language Machines*, edited by S. Kaneff, Academic Press, London, England, pp. 231–256.

Mahoney, J. V. [1980], "Interpolation of a Contour Map of the Island of Mauritius using Elastic Membranes and Thin Plates," unpublished work in Undergraduate Research Opportunities Program, Massachusetts Institute of Technology.

Marr, D. [1976], "Early Processing of Visual Information," *Philosophical Transactions of the Royal Society B*, vol. 275, pp. 1377–1388.

Marr, D., and E. Hildreth [1980], "Theory of Edge Detection," *Proceedings of the Royal Society B*, vol. 207, pp. 187–217.

Mead, C. A. [1989], *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, MA.

Murray, D. W., and B. F. Buxton [1987], "Scene Segmentation from Visual Motion Using Global Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 2, pp. 147–163.

Nagel, H. H. [1984], Unpublished Internal Report, University of Hamburg.

Norton, S. W. [1983], "Information Theoretic Surface Estimation Using Elevation Data," B.S. Thesis, Department of Electrical Engineering and Computer Science.

Platt, J. C., and A. H. Barr [1988], "Constrained Differential Optimization for Neural Networks," Technical Report TR-88-17, Computer Science Department, California Institute of Technology, Pasadena, CA. Also [1987] *Proceedings of IEEE Neural Information Processing Systems Conference.*

Poggio, T. A., and V. Torre [1984], "Ill-Posed Problems and Regularization Analysis in Early Vision," Report AIM-773, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Rifman, S. S., and D. M. McKinnon [1974], "Evaluation of Digital Correction Techniques—for ERTS Images," Report Number E74-10792, TRW Systems Group, July 1974 (see Chapter 4). Also Final Report TRW 20634-6003-TU-00, NASA Goddard Space Flight Center.

Roberts, L. G. [1965], "Machine Perception of Three-Dimensional Solids," in *Optical and Electro-Optical Information Processing*, edited by J. T. Tippet *et al.*, MIT Press, Cambridge, MA, pp. 159–197.

Rosenfeld, A. and M. Thurston [1971], "Edge and Curve Detection for Visual Scene Analysis," *IEEE Transactions on Computers*, vol. 20, no. 5, p. 562–569.

Rosenfeld, A., M. Thurston, and Y. H. Lee [1972], "Edge and Curve Detection: Further Experiments," *IEEE Transactions on Computers*, vol. 21, no. 7, p. 677–715.

Sage, J. P. [1984], "Gaussian Convolution of Images Stored in a Charge-Coupled Device," Solid State Research, Quarterly Technical Report for period from 1 August to 31 October 1983, MIT Lincoln Laboratory, pp. 53–59.

Sage, J. P., and A. L. Lattes [1987], "A High-Speed Two-Dimensional CCD Gaussian Image Convolver," Solid State Research, Quarterly Technical Report for period from 1 August to 31 October 1986, MIT Lincoln Laboratory, pp. 49–52.

Sjoberg, R. J., and B. K. P. Horn [1983], "Atmospheric Effects in Satellite Imaging of Mountainous Terrain," *Applied Optics*, vol. 22, no. 11, pp. 1702–1716.

Strat, T. M. [1977], "Automatic Production of Shaded Orthographic Projections of Terrain," B.S. Thesis, Department of Electrical Engineering and Computer Science.

Tanner, J. E. [1986], "Integrated Optical Motion Detection," Ph.D. Thesis, Computer Science Department, California Institute of Technology, Pasadena, CA. Technical Report 5223:TR:86

Tanner, J. E., and C. A. Mead [1987], "An Integrated Optical Motion Sensor," *VLSI Signal Processing II, (Proceedings of the ASSP Conference on VLSI Signal Processing)*, UCLA, pp 59–76.

Terzopoulos, D. [1983], "Multilevel Computational Processes for Visual Surface Reconstruction," *Computer Vision, Graphics and Image Processing*, vol. 24, no. 1, pp. 52–96.

Terzopoulos, D. [1984], "Efficient Multiresolution Algorithms for Computing Lightness, Shape from Shading, and Optical Flow," *International Joint Conference on Artificial Intelligence*, University of Texas, Austin, TX, pp. 314–317.

Torre, V., and T. A. P gio [1986], "On Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 147–163, Also [1984] Report AIM-768,Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Weldon, E. J., Jr. [1986], Unpublished Internal Report, University of Hawaii, private communication.

Wiener, N. [1966], *Extrapolation, Interpolation, and Smoothing of Stationary Time Series with Engineering Applications*, MIT Press, Cambridge, MA.

Woodham, R. J. [1977], "A Cooperative Algorithm for Determining Surface Orientation from a Single View," *International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 635–641.

43

When AI people first wrote image-understanding programs, processing a 256 × 256 pixel image with even the simplest program would often take an hour. This limited not only what people did, but also what people thought about. Back in the days when the PDP-10 was the dominant AI computer, it was natural to think in terms of 3 × 3 filters and 4 × 4 filters and not as natural to think in terms of the massive computations that the human eye can deploy.

On the other hand, knowing that a blazingly-fast device can be built enables you to think about what you would do if you had it even before fully functional hardware takes shape. Once Knight had built a small 8 × 10 pixel chip for sensing images and producing the difference-of-Gaussian computation on them, vision researchers could get on with using difference-of-Gaussian computations. Even though that computation takes a long time on a serial machine, vision researchers realized that practical devices could be built if and when the difference-of-Gaussian computation proved to have important practical applications.

Now further advances in VLSI *technology have enabled a new round of thinking about what sorts of image-oriented computations can be done on chips. The collection of ideas explained by Horn in this chapter, while not intended to be a thorough review, demonstrates nevertheless that it is reasonable to think in terms of silicon retinas without seriously overstressing the metaphorical sense of the word* retina.

ARTIFICIAL INTELLIGENCE AT MIT
EXPANDING FRONTIERS

edited by Patrick Henry Winston
with Sarah Alexandra Shellard

Volume 2