MASSACHUSETTS INSTITUTE OF TECHNOLOGY
THE ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1071 December 1988

Parallel Networks for Machine Vision

Berthold K.P. Horn

Abstract: The amount of computation required to solve many early vision problems is prodigious, and so it has long been thought that systems that operate in a reasonable amount of time will only become feasible when parallel systems become available. Such systems now exist in digital form, but most are large and expensive. These machines constitute an invaluable test-bed for the development of new algorithms, but they can probably not be scaled down rapidly in both physical size and cost, despite continued advances in semiconductor technology and machine architecture.

Simple analog networks can perform interesting computations, as has been known for a long time. We have reached the point where it is feasible to experiment with implementation of these idea in VLSI form, particularly if we focus on networks composed of locally interconnected passive elements, linear amplifiers, and simple nonlinear components. While there have been excursions into the development of ideas in this area since the very beginnings of work on machine vision, much work remains to be done. Progress will depend on careful attention to matching of the capabilities of simple networks to the needs of early vision.

Note that this is not at all intended to be anything like a review of the field, but merely a collection of some ideas that seem to be interesting.

Key Words: Analog networks, Early vision, Coupled networks, Networks with feedback, Resistive networks, Layered networks, Relaxation, Cooperative computation, Gaussian convolution, Edge detection, Multiple scales, Position and orientation, Interpolation, Motion vision, Direct motion vision.

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

0. Introduction

The term "parallel networks" in the title may appear to be redundant, since the computations at different nodes of an analog network naturally proceed in parallel. In several of the examples explored here, however, 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, since 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 addresses here:

1. A Gaussian convolver for smoothing that operates continuously in time.
2. Coupled resistive networks for interpolation.
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:

- Feedback methods for solving constrained optimization problems using gradient projection, normalization and penalty functions.
- 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.

Note, by the way, that the four sections of this memo are fairly independent and not arranged in any particular order.

1. 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.* 81]. 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 83].

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.

1.1. 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 84] [Torre & Poggio 86]. All but the earliest efforts (see, for example, [Roberts 65]) 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 71]). 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 72, 74].

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

[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, since 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.

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 72]. 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 72]. 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.* 81]).

Multi-Resolution Techniques

There are other reasons for smoothing a discretized image, including suppression of higher spatial frequency components before sub-sampling. Sub-sampling 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 sub-sampled version. This suggests suppressing higher frequency components before sub-sampling 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 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 70a, 70b]. The idea of working at multiple scales occurred around about this time also ([Rosenfeld & Thurston 71, 72] and [Rosenfeld, Thurston & Lee 72]). An elegant theory of edge detection using zero-crossings of the Laplacian of the Gaussian at multiple scales was developed by Marr and Hildreth ([Marr & Hildreth 80] and [Hildreth 80, 83]). This reversed an earlier suggestion that a directional operator may be optimal [Marr 76].

Since then, it has been shown that the rotationally symmetric operators

[2]For an excellent finite support approximation to a low-pass filter look in [Rifman & McKinnon 74] [Bernstein 76] [Abdou & Wong 82].

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 84] and [Horn 86]). Operators for estimating the second derivative in the direction of the largest first derivative (the so-called *second directional derivative*) have been proposed by Haralick [Haralick 84] (see also [Haralick 80] [Hartley 85] [Horn 86])[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 83]. 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 66] [Anderson & Moore 79]. Additive white noise is uncorrelated and so has a flat spectrum, while images typically have spectra that decrease as some power of frequency, starting form a low-frequency plateau [Ahuja & Schachter 83]. 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 transofmr of the derivative of a Gaussian.

The above 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.

1.2. 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

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

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, since 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

$$\cos^n \omega/2,$$

since 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.

1.3. Analog Implementation of Binomial Filters

Binomial filters can be conveniently constructed using charge coupled device technology [Sage 84] [Sage & Lattes 87]. 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 are here 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 neighbours 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 neighbours 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. But first we will approach this smoothing method from the point of view of the equivalent continuous system. For this reason, we next review some properties of resistive sheets.

---

[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 alogned with the original data.

1.4. Resistive Sheets

Continuous resistive sheets solve Poisson's equation

$$\Delta u(\mathbf{x}) = -\rho\, i(\mathbf{x}),$$

where the output $u(\mathbf{x})$ is the potential on the sheet at the point $\mathbf{x}$, while the input $i(\mathbf{x})$ is the current density injected and ρ is the resistivity "per square." Viewed this way, one can think of the solution of this second-order partial differential equation for $u(\mathbf{x})$, given $i(\mathbf{x})$, as application of the "inverse of the Laplacian operator." In image processing, we are usually concerned with the two-dimensional case

$$u_{xx}(x,y) + u_{yy}(x,y) = -\rho\, i(x,y).$$

Discrete arrangements of resistors can be designed to solve difference approximations of Poisson's equation. These resistive networks are remarkably robust against small changes in individual resistances and even errors in interconnection. This has all been known for a very long time, used in analog computer simulations of steady state heat flow, for example, and has even been exploited in machine vision (see, for example, [Horn 74]).

Discrete approximations of Poisson's equation can also be solved by a simple network of operational amplifiers and resistors [Horn 74]. The Laplacian of a function can be considered to be the limit of the convolution of the function with a rotationally symmetric center-surround operator of local support, as the scale of this operator shrinks to zero. We can, for example, think of application of the Laplacian as the limit of convolution with

$$L_\epsilon(x,y) = \begin{cases} -2/\pi\epsilon^4, & \text{for } 0 \le x^2+y^2 < \epsilon^2; \\ +2/3\pi\epsilon^4, & \text{for } \epsilon^2 \le x^2+y^2 < 4\epsilon^2; \\ 0, & \text{for } 4\epsilon^2 \le x^2+y^2. \end{cases},$$

as ϵ tends to zero [Horn 86]. This view of the Laplacian is implicit in the usual discrete approximations of the Laplacian. A resistive network with operational amplifiers can compute the inverse of convolution with such a center-surround operator. Such networks actually can be used to easily invert any convolution with local support, and so are more general than resistive sheets. They can accommodate spatially varying linear operators also, since the feedback arrangement merely has to mimic the forward operation [Horn 74].

Note that the inverse operation can, of course, not recover any spatial frequency components removed by the forward operation. In this case boundary conditions are needed to arrive at a unique stable solution of the inverse problem. Similarly, in the presence of noise, spatial frequency components that are strongly attenuated by the forward operation will be recovered inaccurately by the inverse operation. The inverses of local operations are in general global, but they have a special structure that makes it possible to

compute them using local feedback methods, as described. They are at times called *quasi-local* or *pseudo-local* for this reason.

1.5. Solving the Heat Equation

By adding capacitance, and removing the input, a resistive sheet can be used to solve the heat equation

$$\Delta u(\mathbf{x}) = \kappa\, u_t(\mathbf{x}),$$

which in two dimensions reads,

$$u_{xx}(x,y,t) + u_{yy}(x,y,t) = \kappa\, u_t(x,y,t),$$

where κ is the product of the resistivity (resistance "per square") and the capacitance per unit area. The steady state of such a sheet is given by the solution of Laplace's equation, since $u_t = 0$ in the steady state.

If the potential at time $t = 0$ is forced to equal some given input function, $U(x,y)$ say, we obtain, at a later time, the convolution of the input function with some Gaussian kernel [Courant & Hilbert 62]. The result can be written in the form

$$u(x,y,t) = \iint_D U(\xi,\eta)\frac{1}{4\pi\kappa t}e^{-\frac{(x-\xi)^2+(y-\eta)^2}{4\kappa t}}\,d\xi\,d\eta,$$

as can easily be verified by taking the required partial derivatives, and noting that the Gaussian

$$\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

becomes the unit impulse function as σ tends to zero. Here the standard deviation of the Gaussian is given by

$$\sigma = \sqrt{2\kappa t}.$$

Convolutions with Gaussians of different widths can be obtained by waiting varying amounts of time. This is the observation exploited by Thomas Knight in his Gaussian convolver chip design [Knight 83]. Note that the spatial scale (σ) of the Gaussian only grows with the square root of time.

In an edge detector, one is actually looking for derivatives of the convolved output, or combinations of derivatives, such as, for example, the Laplacian[5]. In the scheme above, the partial derivatives have to be computed in a separate step. Since differentiation is a linear shift-invariant operation, it commutes with the Gaussian convolution step. Consequently these operations can be

[5]The Laplacian of the Gaussian can be approximated by the difference of two Gaussians of different widths. This, however, is a good approximation only when the two Gaussians have almost exactly the same width. The results of convolutions with two Gaussians of almost the same width will be very similar, and so numerical problems arise when the results are subtracted.

performed in either order. A subtlety often overlooked is that the result of convolution with a Gaussian is likely to be very smooth and so differences of quantities that are nearly equal have to be taken when derivatives of the result are to be found. If the intermediate result is quantized (or noise added), the differenced output may in extreme cases be almost totally worthless. It is better then to apply the discrete difference operator first, and then smooth the result. This approach, however, increases the amount of computation required when more than one directional derivative is to be estimated, since the Gaussian convolution has to be repeated for each derivative.

1.6. Non-clocked Approach

The approach given in the previous section for obtaining the convolution of an image with a Gaussian requires that the input be loaded into the network at a certain time, the input then disconnected and the output read a fixed time later. It may be more convenient to have a non-clocked system, where the input is applied continuously to one end of a three-dimensional, layered resistive network, while the output is available continuously at the other end. If Gaussians of differing width are needed, these can be read out from intermediate layers of the network. Ideally, it should also be possible to directly read out the Laplacian of the Gaussian at any stage.

These objectives can be met by a system composed of several layers of resistive sheets, where time is in essence laid out as another spatial dimension. Each sheet solves Poisson's equation with an input current density that is forced to be proportional to the difference in potential at corresponding points in successive sheets. Successive layers correspond to different times in the solution of the heat equation by the sheet mentioned earlier, so the potential difference between layers, $u^{(n+1)}(x,y) - u^{(n)}(x,y)$, is a discrete approximation of (a multiple of) u_t, the time derivative.

Now a resistor draws a current that is proportional to the difference of potentials on its ends. So resistors connected between corresponding nodes on two successive layers would appear to inject the desired currents into the second layer. But we cannot just interconnect adjacent sheets using a layer of resistors, since the current they inject into one sheet is extracted from the other sheet. This disturbs the solution on the earlier sheet. In fact, such a network solves Laplace's equation in three dimensions, rather than the heat equation in two dimensions. The result in this case can be written in the form

$$u(x,y,z) = \iint_D U(\xi,\eta)\,\frac{1}{2\pi z^2}\,\frac{1}{\left(1+\frac{(x-\xi)^2+(y-\eta)^2}{z^2}\right)^{3/2}}\,d\xi\,d\eta,$$

as can easily be verified by taking the required partial derivatives, and noting that

$$\frac{1}{2\pi z^2} \frac{1}{\left(1+\frac{x^2+y^2}{z^2}\right)^{3/2}}$$

becomes the unit impulse function as z tends to zero. This is not what we want. For one thing, this smoothing function falls off much more slowly with radial distance than the Gaussian.

All that we need to make the basic idea work is to add a layer of buffer amplifiers that copy the potential of one sheet without drawing any current from it. The outputs of these amplifiers are then connected to the next sheet by means of suitably chosen resistors (equivalently, we can use a differential transconductance amplifier with high input impedance, that produces a current at its output proportional to the potential difference between input and output [Mead 89]).

1.7. Convolution with the Laplacian of the Gaussian

If we wish to extract the Laplacian of the Gaussian, we can read out the currents in the resistors (or the buffer amplifiers) connecting successive layers. The reason is that these correspond to the time derivative u_t in the heat equation and hence are proportional to the Laplacian, since

$$u_{xx}(x,y,t) + u_{yy}(x,y,t) = \kappa\, u_t(x,y,t).$$

By reading out the currents in different interconnecting layers, we can obtain convolutions with the Laplacians of Gaussians of different widths. Usually we are only interested in the zero-crossings of the result, so we may not need to extract all of these measurements. Instead, neighborhoods are located where some currents are positive while others are negative.

Actually, interpolation can be used to recover the location of the edge fragments to considerably better accuracy than the spacing between nodes in the network of resistors. In digital simulations one finds almost an order of magnitude improvement in resolution when the signal is reasonably free of noise. There is then a tradeoff between a network with very many simple nodes versus a network with fewer complex nodes capable of supporting the interpolation process. This may be an issue of considerable importance if there is a limit on the total number of nodes that can be conveniently constructed using a particular fabrication technology.

Effects of Discretization

In practice we typically have to discretize the continuous analog system that

computes the desired convolution. For a start, we are now using discrete layers in the z-direction (which was the time direction before). This means that the convolution we obtain is actually not with a Gaussian, but the zero-th order modified Bessel function, $K_0(r)$. The result can be written in the form

$$u^{(n+1)}(x,y) = \iint_D u^{(n)}(\xi,\eta)\,\frac{\rho g}{2\pi}\,K_0\left(\sqrt{\rho g}\sqrt{(x-\xi)^2+(y-\eta)^2}\right)\,d\xi\,d\eta,$$

where g is the conductance per unit area of the material connecting the (buffered) output of the n-th layer to the $(n+1)$ layer. The zero-th order modified Bessel function is not a particularly good approximation to the Gaussian—for one thing, it has a singularity at the origin[6]. Fortunately, if we repeat this convolutional operation many times we obtain an effective overall response that is close to Gaussian, as a consequence of the central limit theorem[7].

The convolutional operator changes once again when the continuous resistive sheet is replaced by a regular discrete grid of resistors. No closed form solution is known in this case, for either square or hexagonal tesselations, although the response can be estimated readily using numerical techniques.

It is also interesting to compare this scheme, derived by mapping the time dimension into a third spatial dimension and then discretizing, with the binomial filter scheme discussed earlier. One of the differences between the two schemes is that there are no resistors between layers in the binomial filter scheme, connections are made directly to the outputs of the buffer amplifiers from the previous layer. Another difference is that the output nodes are distinct from the input nodes in the binomial filter scheme, whereas all nodes act as both inputs and outputs in the scheme discussed here. In other respects the two methods are similar.

1.8. Multiple Scales

The information is smoothed out more and more as it flows through the layers of such as 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,

[6] $K_0(r) \approx -\log r$ for small r.

[7] Another possibility is to use networks that solve the discrete analog of the biharmonic equation. In this case the resulting convolutional kernel is a better approximation to the Gaussian—for one thing, the kernel does not have a singularity at the origin [Poggio *et al.* 85] [Harris 89].

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

$$\frac{k}{k-1}$$

times the number of nodes in the first layer. 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.)

1.9. Growth of Standard Deviation with Number of Layers

Another argument for sub-sampling is that, if all the layers and the interconnections are the same, then the width of the Gaussian grows only with the square root of the number of layers, as can be seen from the form of the explicit solution of the heat equation given earlier. This suggests that arrangements be made to ensure that κ varies from layer to layer. We can increase κ either by decreasing the resistances in the sheets themselves, or by decreasing the conductance in the interconnecting layers. But note that if successive layers contain fewer nodes, while the resistances between nodes are kept the same, then κ in effect is increased automatically. This can be exploited to attain exponential growth of the effective width of the Gaussian 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 sub-sampling 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[8]. 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 group of four and connected to

[8]Naturally, since 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.

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.

1.10. 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 discrete approximation to the original continuous domain equation. A better approximation can be attained 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 sub-fields 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 87]). This leads to the interlaced pattern shown in Figure 1, where each cell is labelled with a number indicating what 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 white cells form a 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 white 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, an so on ...

Note that one half of the cells are labelled 1, one quarter are labelled 2, one eigth are labelled 3 and so on. The top left node, labelled 0, does not

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 labelled $(n+1)$ has half as many nodes as the network whose nodes are labelled n. The total number of nodes is less than twice the number of nodes in the finest layer.

belong to any of the partitions. If we consider the nodes labelled 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!

2. 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 81][9]. Many modern interpolation methods are based on physical models of deformation of elastic sheets or thin plates. So these are briefly reviewed here first.

2.1. Mathematical Physics

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 53].

---

[9]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, since 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 behaviour [Grzywacs & Yuille 87].

2.2. Interpolation from Sparse Contours and Isolated Points

The above equation has been proposed as a means of interpolating from sparse data specified along smooth curves, not necessarily simple closed 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 77] and [Horn 79, 81, 83]) as well as in remote sensing ([Bachmann 77], [Horn & Bachmann 78] and [Sjoberg & Horn 83]). Some undergraduate research opportunities project work was based on this idea [Mahoney 80], as were the bachelor's theses of [Goldfinger 83], and [Norton 83]. Recently, a 48×48 cell analog chip has been built to do this kind of interpolation [Luo, Koch & Mead 88].

The result of elastic membrane interpolation is not smooth, however, since, 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[10].

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 53]. 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 one 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

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

[10] 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, since shading depends on surface orientation.

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 53]. Solution of the bi-harmonic equation, while 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 86]). 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. (The corresponding system of linear equations is not diagonally dominant.) 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 where used in interpolation of surface depth from stereo data along brightness edges [Grimson 81, 82, 83]. 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 [Terzopolous 83]. Terzopolous also applied these ideas to variational problems other than interpolation [Terzopolous 84].

2.3. Resistive Networks for the Bi-Harmonic Equation

It is clear that methods for solving the bi-harmonic equations are important in machine vision. Unfortunately, simple networks of (positive) resistances can not be constructed to solve discrete approximations of this equation. Computational molecules or stencils [Horn 86] 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 Pois-

son's equation

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

in the solution of the bi-harmonic equation, since simple resistive networks can be constructed to solve Laplace's equation. One simple idea is to use the coupled system,

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

since 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 [Harris 86] 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

$$\Delta z = p_x + q_y,$$
$$\lambda \Delta p = p - z_x,$$
$$\lambda \Delta q = q - z_y.$$

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:

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

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 84], and described in [Horn & Brooks 86]). 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.

2.4. Application to Shape from Shading

Solution of the shape from shading problem, in the special case when the light sources and the viewer are far away in relation to the size of the object being viewed, revolves around the nonlinear first-order partial differential equation

$$E(x,y) = R\big(z_x(x,y), z_y(x,y)\big),$$

the so-called *image irradiance equation* [Horn 86]. Solutions may be obtained by the method of characteristic strip expansion [Garabedian 64]. The results can be improved considerably by solving characteristics in parallel so as to permit application of a *sharpening* method to the propagating solution wavefront [Horn 70]. This method adjusts estimated surface orientations according to local brightness measurements and so reduces the propagation of errors inherent in methods based on brightness gradients.

None of these methods lend themselves to parallel implementation on a regular grid registered with the picture cells [Horn 70]. Alternatives have been explored that lead to methods similar to those used for solving second-order elliptic partial differential equations [Woodham 77] [Strat 79] [Ikeuchi & Horn 81]. It is difficult to come up with a convergent iterative scheme that directly gives height $z(x,y)$ above some reference plane [Brooks & Horn 85] [Horn & Brooks 86]. Most methods instead compute estimates of surface orientation. These may not be *integrable* in the sense that they may not be consistent with any underlying surface. For example, if the method recovers estimates of $p = z_x$ and $q = z_y$, it may be that $p_y \neq q_x$. This problem can be solved by finding the "nearest" integrable surface after each iteration [Frankot & Chellappa 87].

The nearest integrable surface may be found using a set of basis functions that are integrable, or by the method described above for recovering $z(x,y)$ from $p(x,y)$ and $q(x,y)$. Numerical estimates of the derivatives of the computed $z(x,y)$ are then used as starting values for the next iteration.

It is possible to combine the iterative scheme for solving the shape form shading problem with that for projecting the solution onto the space of integrable solutions. Suppose for example that we wish to minimize the following

functional:

$$\iint \big(E - R(p,q)\big)^2 + \lambda(p_x^2 + p_y^2 + q_x^2 + q_y^2) + \mu\big((z_x - p)^2 + (z_y - q)^2\big)\, dx\, dy.$$

The Euler equations for this calculus of variations problem are

$$\lambda \Delta p = -(E - R)R_p + \mu(p - z_x),$$
$$\lambda \Delta q = -(E - R)R_q + \mu(q - z_y),$$
$$\Delta z = p_x + q_y.$$

This illustrates that the methods discussed here for interpolation from sparse data have applications in other areas as well.

3. 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 86]). 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 that have been built for finding moments [Hatamian 86, 87]. We nevertheless explore analog networks for this task, partly to see whether they may have advantages over existing digital implementations, but mostly because they constitute a stepping stone on the way to networks for the recovery of motion from time-varying images. Analog circuitry for the motion vision task share many of the features of the simple moment generating circuits, but are more complex.

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.

3.1. 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[11].

3.2. 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[12].

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 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.

---

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

[12] 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

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 86]). 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 don't 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, by the way, 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, since 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 86]. 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.

3.3. 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 high precision.

3.4. Methods for Computing Moments

There are many methods for efficiently computing moments. It is possible, for example, to avoid the multiplications appearing in our simple definition of the moments by repeated summation. Note, for example, that

$$\sum_{i=1}^{n} i\, f_i = \sum_{i=1}^{n} \sum_{j=1}^{i} f_i = \sum_{j=1}^{n} \sum_{i=j}^{n} f_i,$$

so that this sum can be computed using the coupled multiplication-free iterative scheme

$$s_i = s_{i+1} + f_i \quad \text{and} \quad S_j = S_{j+1} + s_j,$$

with $s_{n+1} = 0$ and $S_{n+1} = 0$. The total is given by S_1. A similar scheme using two intermediate sums can be used to obtain a second moment and so on.

In serial digital implementation, run-length coding can be used to advantage. Further, projections can dramatically compress the information. A projection at an arbitrary angle θ is given by

$$p_\theta(t) = \iint_D b(x,y)\,\delta(x\cos\theta + y\sin\theta - t)\,dx\,dy,$$

or

$$p_\theta(t) = \int_L b(t\cos\theta - s\sin\theta, t\sin\theta + s\cos\theta)\,ds,$$

where L is the straight line $x\cos\theta + y\sin\theta = t$. It can be shown that all n-th order moments can be computed from $(n+1)$ projections.

Consider, in particular, the vertical and the horizontal projections (where $\theta = 0$, and $\theta = \pi/2$ respectively):

$$v(x) = \int b(x,y)\,dy \quad \text{and} \quad h(y) = \int b(x,y)\,dx.$$

The integral of either projection gives us the area A, while

$$A\,\bar{x} = \int x\,v(x)\,dx \quad \text{and} \quad A\,\bar{y} = \int y\,h(y)\,dy,$$

gives us the center of area. Three projections are needed to calculate the second moments. We can use the two we already have, plus a projection in a diagonal direction (see, for example, Chapter 3 in [Horn 86]):

$$d(t) = \int b\left(\frac{t-s}{\sqrt{2}}, \frac{t+s}{\sqrt{2}}\right) ds.$$

The moments involving x^2 and y^2 can be obtained straightforwardly from $v(x)$ and $h(y)$, while the moment involving xy can be computed as follows:

$$\iint_D xy\,b(x,y)\,dx\,dy = \int t^2\,d(t)\,dt - \frac{1}{2}\int x^2\,v(x)\,dx - \frac{1}{2}\int y^2\,h(y)\,dy.$$

In the case of a hexagonal grid, the three projections can be conveniently taken in directions spaced 120° apart. Working with projections dramatically reduces the amount of arithmetic required in a digital implementation.

3.5. Feedback Method for Computing Center of Area

The computation suggested by the equations above can be embodied in hardware in many different ways. One fairly obvious implementation of the first-order moment calculation uses linear resistive chains to obtain a potential

proportional to x at every picture cell, and a switch that injects a current proportional to x into a global bus wherever $b(x,y) = 1$. The bus is terminated in a resistor; its potential represents $A\,\bar{x}$. A similar arrangement is used to compute $A\,\bar{y}$. A third bus is used to compute a potential proportional to the area A itself. (If $b(x,y)$ is not binary, analog multipliers will need to be used at each picture cell in order to obtain the required products.)

The computation can also be performed by first obtaining the horizontal and vertical projections. (In the case of grey-level information, the number of analog multipliers is drastically reduced by first computing these projections.) A one-dimensional circuit, using analog multipliers, then computes the one-dimensional centroid of each of the two projections. Note that the projections may also be processed off chip and that the projections may be obtained using charge coupled device technology.

Other possibilities abound. A particularly simple method involves a feedback scheme obtained by noting that the inertia about an axis perpendicular to the image plane is minimized when this axis passes through the centroid. That is, we have to find $\bar{x}$ and $\bar{y}$ such that

$$E = \iint_D \left((x-\bar{x})^2 + (y-\bar{y})^2\right)\, b(x,y)\, dx\, dy,$$

is made as small as possible. The derivatives of this integral with respect to $\bar{x}$ and $\bar{y}$ are

$$\frac{dE}{d\bar{x}} = -2\iint_D (x-\bar{x})\, b(x,y)\, dx\, dy \quad \text{and} \quad \frac{dE}{d\bar{y}} = -2\iint_D (y-\bar{y})\, b(x,y)\, dx\, dy.$$

We can use a gradient descent method to solve the least squares problem. This leads to the scheme:

$$\frac{d\bar{x}}{dt} = \alpha \iint_D (x-\bar{x})\, b(x,y)\, dx\, dy \quad \text{and} \quad \frac{d\bar{y}}{dt} = \alpha \iint_D (y-\bar{y})\, b(x,y)\, dx\, dy,$$

where α is a gain factor that controls the speed of adjustment of the estimates of $\bar{x}$ and $\bar{y}$. When the circuit settles, the time derivatives are zero and the extremum has been reached.

A simple implementation of this idea uses a global bus with potential proportional to the present estimate of $\bar{x}$, the x-component of the center of area. Each picture cell injects a current proportional to the difference between its x coordinate and the bus potential $\bar{x}$, provided that it is in the object region, that is, if $b(x,y) = 1$. The current can be generated easily by a resistor that is connected between a potential follower that buffers the x-coordinate potential and the bus for $\bar{x}$. A switch connects this resistor to the buffer amplifier wherever $b(x,y) = 1$. A similar arrangement is used for $\bar{y}$, the y component of the center of area. Both busses are terminated in capacitors

connected to ground[13].

In the equilibrium state, the currents injected into each of the two busses add up to zero. There is no need in this scheme to compute the area separately, if only the center of area is needed. A discrete analog chip has in fact been built that determines the centroid using a method like this [DeWeerth & Mead 88].

3.6. Feedback Schemes for Orientation

The computation of orientation is a little harder, since it involves second-order moments and requires that the center of area be known. The axis of least inertia passes through the centroid and the distance of an image point from the axis is just

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

where $x' = (x - \bar{x})$ and $y' = (y - \bar{y})$. We could try to develop a feedback scheme for computing the angle θ directly, but this would require evaluation of trigonometric functions and some way of letting the representation "wrap around" when θ exceeds $+\pi$ or becomes less than $-\pi$. It is better to use a redundant representation for orientation. One can, for example, use two quantities, c and s, proportional to the sine and cosine of θ. The inertia integral then can be written

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

The only difficulty with a scheme based on this approach is the need to keep the two quantities consistent, that is, some way of ensuring that $c^2+s^2-1 = 0$.

One way of dealing with this constraint is to add a Lagrange multiplier term to obtain the modified inertia integral

$$I' = \iint_D (x's - y'c)^2 \, b(x,y)\, dx\, dy + \lambda\,(c^2 + s^2 - 1).$$

We might then consider differentiating with respect to c, s, and λ to obtain the gradient of the modified inertia integral:

$$\frac{dI'}{dc} = -2 \iint_D (x's - y'c)\, y' \, b(x,y)\, dx\, dy + 2\lambda c,$$

$$\frac{dI'}{ds} = +2 \iint_D (x's - y'c)\, x' \, b(x,y)\, dx\, dy + 2\lambda s,$$

$$\frac{dI'}{d\lambda} = (c^2 + s^2 - 1).$$

[13]It may be sufficient to use the parasitic capacitance of the busses, provided there are no stability problems resulting from unmodeled effects.

By adding c times the first derivative to s times the second derivative, we see that at the stationary point,

$$\lambda = -\iint_D (x's - y'c)^2\, b(x,y)\, dx\, dy = -I.$$

Subtracting s times the first derivative from c times the second derivative, we also see that at the stationary point,

$$sc \iint_D (y'^2 - x'^2)\, dx\, dy + (c^2 - s^2) \iint_D x'y'\, dx\, dy = 0.$$

If we use polar coordinates here for a moment and let $c = \rho\cos\theta$ and $s = \rho\sin\theta$, we note that

$$I' = \rho^2 \iint_D (x'\sin\theta - y'\cos\theta)^2\, b(x,y)\, dx\, dy + \lambda(\rho^2 - 1).$$

Now

$$\frac{dI'}{d\theta} = -2\rho^2 \iint_D \Big((y'^2 - x'^2)\sin\theta\cos\theta + x'y'(\cos^2\theta - \sin^2\theta)\Big)\, dx\, dy,$$

which, of course, is zero at the stationary point. Also

$$\frac{d^2I'}{d\theta^2} = -2\rho^2 \iint_D \Big((y'^2 - x'^2)(\cos^2\theta - \sin^2\theta) - 4x'y'\sin\theta\cos\theta\Big)\, dx\, dy,$$

At the minimum of I, this equals

$$\frac{\iint_D (x'^2 + y'^2)\, b(x,y)\, dx\, dy}{\sqrt{\left(\iint_D x'y'\, b(x,y)\, dx\, dy\right)^2 + \left(\iint_D (y'^2 - x'^2)\, b(x,y)\, dx\, dy\right)^2}}.$$

So the second derivative of I' with respect to θ is positive at the minimum (and it is negative at the maximum). Unfortunately, the second derivative of I' with respect to ρ is zero there, given the value computed above for λ. Also, the second derivative of I' with respect to λ is always zero.

We conclude that the minimum of the original inertia integral, I (a function of the single variable θ), does not correspond to a minimum of the modified inertia integral, I' (a function of the three variables c, s, and λ), but rather some kind of saddle point. This means that we cannot use steepest descent methods directly. We discuss a novel way of dealing with this problem in the next section that requires inverting the sign of the gradient component in the λ direction and the addition of a penalty term proportional to $(c^2 + s^2 - 1)^2$ (see also [Platt & Barr 88]). But in this particular case we can avoid this complication by noting that when the solution has been found,

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

This provides a way of estimating λ at any given stage of the computation that we can use instead of gradient descent for finding a new value of λ.

Another way of dealing with the problem is to make adjustments only in directions that keep the value of $c^2 + s^2$ constant. This can be done by removing the component of the gradient that is along the normal to the constraint curve defined by $c^2+s^2-1 = 0$. This gradient projection method leads to useful feedback schemes for finding the minimum. Gradient projection is discussed in more detail in the next section.

3.7. Normalization of the Inertia Integral

In this particular case here we can use an approach that is a bit simpler than gradient projection, because the integral we are trying to minimize has a particularl scaling property. Note that if we multiply c and s by some constant k, the integral is just multiplied by k^2. This suggest that we can circumvent the difficulty noted above simply by normalizing the integral by dividing by (c^2+s^2). This makes the result independent of the scale of c and s. To force c and s to have the correct scale, we can then add a penalty term proportional to the square of the error in the constraint $c^2 + s^2 - 1 = 0$, so that overall we now have to minimize

$$I'' = \frac{1}{c^2+s^2} \iint_D (x's - y'c)^2\, b(x,y)\; dx\, dy + \mu(c^2 + s^2 - 1)^2.$$

Note that μ is not an unknown parameter, but a quantity we can adjust to control the rate of convergence of the resulting system towards the condition $c^2 + s^2 = 1$. Differentiating I'' with respect to c and s we obtain

$$\frac{dI''}{dc} = \frac{+2s}{(c^2+s^2)^2} \iint_D e(x,y)\, b(x,y)\; dx\, dy + 4\mu(c^2 + s^2 - 1)c,$$

$$\frac{dI''}{ds} = \frac{-2c}{(c^2+s^2)^2} \iint_D e(x,y)\, b(x,y)\; dx\, dy + 4\mu(c^2 + s^2 - 1)s,$$

where

$$e(x,y) = x'y'(c^2 - s^2) + (y'^2 - x'^2)\, sc.$$

This suggests a simple gradient descent method:

$$\frac{dc}{dt} = -\alpha\, s \iint_D e(x,y)\, b(x,y)\; dx\, dy - \beta c(c^2 + s^2 - 1),$$

$$\frac{ds}{dt} = +\alpha\, c \iint_D e(x,y)\, b(x,y)\; dx\, dy - \beta s(c^2 + s^2 - 1),$$

where $\beta = 2\alpha\mu$. Note that if we omit the penalty term, that is, when $\beta = 0$, then the adjustment to (c, s) is in the direction $(-s, c)$, which is orthogonal to (c, s). Thus the magnitude of (c, s) is preserved by this component of the adjustment. The other component, arising from the penalty term, is in the direction (c, s) and thus does not affect the first part of the modified integral, only bringing the magnitude of (c, s) closer to unity.

The above leads to the following feedback scheme: The values of c and s are represented as potentials on global bus lines. The combinations sc and $(c^2 - s^2)$ may be either computed locally, or distributed by other global bus lines, fed by circuits that compute these values based on the values of c and s. At each picture cell we compute the differences $x' = (x - \bar{x})$ and $y' = (y - \bar{y})$, as well as $x'y'$ and $(y'^2 - x'^2)$. Additional circuitry is used to obtain the error term

$$e = x'y'(c^2 - s^2) + (y'^2 - x'^2)sc.$$

Finally, in places where $b(x, y) = 1$, currents proportional to $-e\,s$ and $+e\,c$ are injected into the global busses for c and s respectively. No current is injected at picture cells where $b(x, y) = 0$.

The two global busses for c and s are terminated in capacitors connected to ground[14]. A single separate global feedback circuit is used to ensure that $c^2 + s^2 - 1 = 0$. The error in the magnitude of the sum of c^2 and s^2 is computed and used to inject currents proportional to $-\beta\,c\,(c^2 + s^2 - 1)$ and $-\beta\,s\,(c^2 + s^2 - 1)$ into the capacitors whose potentials represent c and s.

It may appear at first sight that there is an opportunity for instability here, since there are two coupled first-order feedback loops. No such problem arises, however, since the adjustments made to (c, s) by the two systems are in orthogonal directions.

There may also appear to be potential start up problem here, since the adjustments are zero as long as $c = 0$ and $s = 0$. But this is not a serious concern, since this state is an unstable equilibrium and so any small noise current will cause the system to move away from it.

In the scheme as described so far, a considerable number of local computational elements are needed to obtain the factors $x'y'$ and $(y'^2 - x'^2)$. Rather than computing these terms at each picture cell from values of x' and y', one can obtain them using two resistive grids. This is because both of these factors are harmonic functions, that is, they satisfy Laplace's equation, $\Delta f(x, y) = 0$. A uniform resistive sheet solves Laplace's equation when no current is injected into it. If the boundary of the grid is held at a potential proportional to $x'y'$, the interior will also settle to a potential proportional to $x'y'$, since this is the unique solution of Laplace's equations for these boundary conditions. The same holds for $(y'^2 - x'^2)$, which is actually just a scaled version of the same function rotated by $\pi/4$. This idea is explored further later in this section.

In any case, the overall circuit will settle into one of two opposite states, depending on initial conditions, provided that the object imaged is not too symmetrical. (If the object is almost symmetrical, currents generated in one

[14]It may be sufficient to use the parasitic capacitance of the busses, provided there are no stability problems resulting from unmodeled effects.

image area will tend to cancel currents in other image areas and small overall noise currents will drive the result.) We have above, by the way, indirectly solved the problem of finding the eigenvector corresponding to the smallest eigenvalue of a 2×2 symmetric matrix. It may seem surprising that a fairly straightforward analog circuit can do this for us! We explore a generalization of this in the next section.

3.8. Resistive Networks for Moment Calculation

If area and center of area is all we are computing, then even the simple feedback schemes discussed above appear to constitute overkill. Consider first a regular one-dimensional chain of N resistors each of resistance R. Above we used such a simple resistive chain to generate potentials at each node linearly related to the position. This potential was then used in further calculation—to generate a current injected into a global buss. 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. We show this now in the continuous case.

To see this more clearly, 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 68]:

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

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 can also not 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.

3.9. Implementation Details & 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.* 80] [Gilbert 81]. 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 86, 87]. Furthermore, a discrete analog chip has been built that determines the centroid using a gradient descent method [DeWeerth & Mead 88]. 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.

3.10. A Network Equivalence Theorem

In the above we have explored 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

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

3.11. 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.

3.12. 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[15]:

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

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.

3.13. Generalizations

The same ideas can be applied in the continuous domain, where we are dealing with resistive sheets, rather than networks of discrete components. In particular, we can use it to design circuits that compute combinations of various moments of image brightness. Consider a uniform two-dimensional resistive sheet. The potential in the interior satisfies Laplace's equation

$$\Delta v(x, y) = 0.$$

We can obtain a potential distribution proportional to an arbitrary harmonic function $v(x, y)$ simply by applying a potential proportional to $v(x, y)$ to the *boundary* of the sheet.

The functions 1, x, y, xy and $y^2 - x^2$ are harmonic. This suggest that we can use this idea to compute the zeroth, first, and some combinations of the

[15] In the feedback version of this idea, currents are generated proportional to the difference between the potentials representing x and $\overline{x}$. These are injected into an unterminated global bus. In the steady state we have from Kirchhoff's law:

$$\sum_{m=1}^{M} (v_m - \overline{v}) i_m = 0,$$

which leads to the same result.

higher-order moments of image brightness. A current density proportional to the product of brightness at each point in the image and the potential at the corresponding point on the resistive sheet is injected into a global bus. The total current is proportional to the desired combination of moments.

Alternatively, using the method developed above, we can use the same resistive sheet to direct the currents rather than as a way of generating potential distributions. At each point we simply inject a current density proportional to brightness. The edge of the resistive sheet is now grounded and the current density along the boundary is read out. The desired moment is obtained by integrating the product of $v(x,y)$ and the current density on the boundary.

In the continuous domain, with a uniform resistive grid, the equivalence between the two methods described above can be obtained by an application of Green's theorem in two dimensions for converting an integral over a region into an integral along the boundary of the region. We have

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

Now let $u(x,y)$ be the potential on the resistive sheet, while $v(x,y)$ is some chosen harmonic function. Then, if the current density injected into the sheet is $i(x,y)$ and its resistivity ρ, we have

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

while the current density along the boundary is given by

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

Using the fact that $u(x,y) = 0$ on the boundary, and that $v(x,y)$ is harmonic we obtain from Green's theorem:

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

Substituting for $u(x,y)$ in the interior in terms of the injected current density and for the normal derivative of $u(x,y)$ on the boundary, we finally see that

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

The above analysis holds as long as the multiplier is a harmonic function.

The theorem presented earlier is more general, since it applies even to networks that are not uniform and does not require that the multiplier function be harmonic. It also can be used directly on discrete networks and does not involve approximating a continuous resistive sheet with a discrete one.

In the discrete case, there is a definite implementation advantage to the second scheme, since there are few nodes on the boundary in comparison to the number of nodes in the interior of the resistive grid.

4. 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 a 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 sub-pixel 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, since they cannot deal with significant changes in foreshortening or photometric changes, yet are not able to produce displacement estimates with sub-pixel 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 rapdidly moving light sources. Most methods will be fooled by the motion of virtual images resulting from specular or glossy reflections of point light sources.

4.1. 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, since 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 84] & [Weldon 86]: 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 & Mead 87]). 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 & Horn 83]).

2. **Basic Optical Flow** [Horn & Schunck 81]): 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, by the way, 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 87]: 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, since 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 86] [Gamble & Poggio 87] [Hutchinson, Koch, Luo & Mead 87] [Murray & Buxton 87]: The notion of a *line process* for dealing with discontinuities in images originated with [Geman & Geman 84]. This idea was later applied to discontinuities in optical flow by [Koch, Marroquin & Yuille 86], [Hutchinson, Koch, Luo & Mead 87] and [Murray & Buxton 87]. 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, since 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 88]. 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 84]. Further improvements in performance can be had if line processes are allowed only very near to discontinuities in brightness, that is, edges [Gamble & Poggio 87]. 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 & Yuille 86]).

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 since:

- 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 since 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 too 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 a strong interest in finding rapid, robust methods for estimating the optical flow.

4.2. Direct Recovery of Rigid Body Motion

It is possible to derive observer motion and object shape directly from bright-

ness 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 85] [Horn & Weldon 88]: In the case of pure rotation, the motion field is particularly simple since 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, since the problem is so highly overdetermined (three unknowns and hundreds of thousands of measurements).

2. **Pure Translation** [Horn & Weldon 88]: 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, since 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 since it allows us to recover depth and thus obtain surface shape information.

3. **Planar Surface** [Horn & Negahdaripour 87]: 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 par-

tial 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 to 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 Laplace's equation or the bi-harmonic equation. A simple passive network will do for Laplace'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 Laplace 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 pretty complex system!

4.3. Contant 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 favoured. 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 onstraint 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 and Mead works [Tanner 86] [Tanner & Mead 87].

4.4. 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 (since 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 (since there are about as many unknowns as constraints), but of great interest, since 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 (since 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 80], 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 88]

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

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.*

4.5. Feedback Computation of Instantaneous Rotational Velocity

Horn & Weldon [1988] rediscovered a method apparently first invented by Alomoinos & 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 ω. 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 $(1 + y^2)$, since these are coefficients in the expression for $\mathbf{v}$).

Next, one computes the error term

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

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

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

Each of the three bus wires is terminated in a capacitance[16]. We now have a system that obeys an equation like

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

the steady state solution of which is

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

or

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

The feedback scheme involves considerably less computation than the closed form solution (for example, we don't 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 photo-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.

4.6. 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

[16] Unless there are instability problems due to unmodeled effects, one may be able to just rely on the parasitic capacitances.

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 88]. 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[17]. 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 introduce an offset in the denominator as follows:

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

[17] 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 does not appear to be this *ad hoc* [Horn & Weldon 88].

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 [Korn & Korn 68].

We explore in the next section how a circuit can be devised to compute this eigenvector.

4.7. Finding Eigenvectors Using Analog Networks

In one of the direct methods for recovering translational motion, the direction of motion is found to be the eigenvector of a symmetric 3×3 matrix S associated with the smallest eigenvalue. The coefficients of the matrix are sums of products of image coordinates and first derivatives of image brightness. Note that the computation of the eigenvector needs to be done only in one place, using data accumulated over the whole image, rather than at each picture cell. It could potentially be done on a serial computer using the accumulated total obtained. An interesting question is whether this eigenvector can be found using an analog network, hopefully by means of a network that is not too complex. There are actually several ways of doing this.

First of all, note that a dot-product can be computed using three multipliers, while a cross-product takes six. The product of a 3×3 matrix and an arbitrary vector requires nine multipliers. If we were looking for the eigenvector associated with the *largest* eigenvector, we could use the observation that the iteration

$$\mathbf{t}^{n+1} = S\, \mathbf{t}^n,$$

converges to a multiple of this eigenvector given virtually any starting value

$$\mathbf{v} = \alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \alpha_2 \mathbf{e}_3,$$

since

$$S^k(\alpha_1 \mathbf{e}_1 + \alpha_2 \mathbf{e}_2 + \alpha_3 \mathbf{e}_3) = \lambda_1^k \alpha_1 \mathbf{e}_1 + \lambda_2^k \alpha_2 \mathbf{e}_2 + \lambda_2^k \alpha_2 \mathbf{e}_3,$$

where λ_1, λ_2, and λ_3 are the eigenvalues and $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ are the corresponding eigenvectors. As long as the eigenvalues are distinct, the term corresponding to the largest eigenvalue will dominate after a number of iterations (or equivalently, many time constants in a feedback implementation).

For the results of such an iteration to remain within bounds, the result must be renormalized each time. Doing this the obvious way involves division, but a feedback circuit can achieve the same effect using only multiplication as

follows: (a) each of the components of the vector is multiplied by an adjustable positive scale factor, (b) the magnitude squared of the result is computed and (c) the scale factor is adjusted if the magnitude is not equal to one. The scale factor itself equals the inverse of the largest eigenvalue when the system stabilizes. Renormalization may be based on the maximum of the absolute values of the components instead of the sum of squares, if this turns out to be cheaper to compute. There may be some stability questions here, since the normalization is in essence trying to stabilize a positive feedback loop.

We can get the eigenvector associated with the *smallest* eigenvalue if we apply the same idea to the inverse matrix S^{-1}. Of course, inverting the matrix is in itself not trivial. But we can instead set up a network to solve the equation $S\,\mathbf{v} = \mathbf{t}$ for $\mathbf{v}$ given $\mathbf{t}$, and use the iteration

$$S\,\mathbf{t}^{n+1} = \mathbf{t}^n,$$

so that we do not have to explicitly invert the matrix.

Another method is based on a different view of the value being minimized

$$I = \mathbf{t}^T S \mathbf{t}.$$

The gradient is just

$$\frac{dI}{d\mathbf{t}} = 2S\mathbf{t},$$

so we could consider adjusting the present guess for $\mathbf{t}$ in the direction of steepest descent. A problem with this is that we are dealing with a constrained minimization problem. Steepest descent will in fact just lead to the trivial solution $\mathbf{t} = \mathbf{0}$. We have to maintain the condition that $\|\mathbf{t}\|^2 = 1$. One way of doing this is to introduce a Lagrangian multiplier and add a term to the integral above:

$$I' = \mathbf{t}^T S \mathbf{t} + \lambda(\mathbf{t} \cdot \mathbf{t} - 1).$$

We can then take the derivatives with respect to $\mathbf{t}$ and λ:

$$\frac{dI'}{d\mathbf{t}} = 2S\mathbf{t} + 2\lambda\mathbf{t},$$

$$\frac{dI'}{d\lambda} = \mathbf{t} \cdot \mathbf{t} - 1.$$

Unfortunately, the minimum of the original problem corresponds to a saddle point in this modified problem (where we have four instead of three unknown parameters). So descent along the gradient will not get us to the solution (but we could use the method of [Platt & Barr 88]; see later).

One way to cirucmvent this difficulty is to note that in this special case we can compute the value of λ at the extremum:

$$\lambda = -\mathbf{t}^T S \mathbf{t}.$$

This provides us with a way of estimating λ that does not involve gradient descent.

Another approach is to remove the component of the gradient of I' in the direction of the gradient of the constraint function $(\mathbf{t}\cdot\mathbf{t}-1)$. Such gradient projection methods lead to viable feedback schemes, as shown later.

However, in this special case, we can do something simpler. We can normalize the integral by dividing by $(\mathbf{t}\cdot\mathbf{t})$. This makes the result insensitive to changes in the magnitude of $\mathbf{t}$. To then force the result to also satisfy the constraint, we add a term that penalizes departure from the condition $(\mathbf{t}\cdot\mathbf{t}-1)=0$:

$$I''=\frac{1}{\mathbf{t}\cdot\mathbf{t}}\mathbf{t}^T S\mathbf{t}+\mu(\mathbf{t}\cdot\mathbf{t}-1)^2.$$

The gradient now is

$$\frac{dI''}{d\mathbf{t}}=\frac{2}{(\mathbf{t}\cdot\mathbf{t})^2}\big((\mathbf{t}\cdot\mathbf{t})\,S\mathbf{t}-(\mathbf{t}^T S\mathbf{t})\,\mathbf{t}\big)+2\mu(\mathbf{t}\cdot\mathbf{t}-1)\mathbf{t}.$$

This suggests a feedback scheme like

$$\frac{d\mathbf{t}}{dt}=-\alpha\big((\mathbf{t}\cdot\mathbf{t})\,S\mathbf{t}-(\mathbf{t}^T S\mathbf{t})\,\mathbf{t}\big)-\beta(\mathbf{t}\cdot\mathbf{t}-1)\mathbf{t}.$$

The first term above equals

$$-\alpha\big((\mathbf{t}\times S\mathbf{t})\times\mathbf{t}\big),$$

and so is orthogonal to $\mathbf{t}$. We can conclude that the magnitude of $\mathbf{t}$ is not affected by adjustments resulting from this term. It is the second term, arising from the penalty function, that forces the magnitude of $\mathbf{t}$ to approach one as time increases.

At this point we need to remember that

$$S\mathbf{t}=\iint_D(\mathbf{s}\cdot\mathbf{t})\mathbf{s}\,dx\,dy,$$

and

$$\mathbf{t}^T S\mathbf{t}=\iint_D(\mathbf{s}\cdot\mathbf{t})^2\,dx\,dy.$$

So we have to estimate the brightness derivatives E_x, E_y and E_t at every picture cell and compute $\mathbf{s}$, the weighting factor $w(E_t)$, and the dot-products $(\mathbf{s}\cdot\mathbf{t})$ and $(\mathbf{t}\cdot\mathbf{t})$. We then generate currents proportional to

$$-\alpha\,w(E_t)\,(\mathbf{s}\cdot\mathbf{t})\big((\mathbf{t}\cdot\mathbf{t})\mathbf{s}-(\mathbf{s}\cdot\mathbf{t})\mathbf{t}\big)$$

that are injected into a global three wire bus whose potentials represents the components of the translation vector $\mathbf{t}$. The bus wires are terminated in capacitors connected to ground[18].

A separate circuit computes the error in the magnitude squared of $\mathbf{t}$, and produces currents proportional to

$$-\beta\,(\mathbf{t}\cdot\mathbf{t}-1)\,\mathbf{t},$$

[18] Again, the normal parasitic capacitances of the wires may be relied upon unless there are instability problems due to unmodeled effects.

that are injected into the capacitors whose potentials represent the components of $\mathbf{t}$.

If $\mathbf{t}$ minimizes the error integral, so does $-\mathbf{t}$. One thus may arrive at one of two solutions depending on the initial conditions. Also, the adjustments are zero when $\mathbf{t} = \mathbf{0}$. This equilibrium point at the origin is metastable, however, so any small disturbance will lead the system away.

All of the above should, by the way, appear to the reader to be very familiar, since we used similar ideas in the previous section for finding the axis of least inertia of a binary image. The only real difference is that we were dealing there with a 2×2 matrix instead of a 3×3 matrix.

The elementary components of the circuit needed are similar to the ones that are needed for the pure rotation case. One difference is that the vector $\mathbf{s}$, is simpler to compute than $\mathbf{v}$, the vector needed in the case of pure rotation. On the other hand, some nonlinear transfer function is needed to compute the weighting factor $w(E_t)$.

4.8. General Motion with Respect to Planar Surface

Another special case of considerable interest is that of general motion (translation and rotation) with respect to a planar surface. This is the motion vision problem confronting a pilot landing an aircraft (although, of course, a pilot has many additional visual cues available). A planar surface can be described by an equation of the form

$$\mathbf{n} \cdot \mathbf{R} = 1,$$

where $\mathbf{n}$ is a vector parallel to the normal with length equal to the inverse of the perpendicular distance of the plane from the origin. The perspective projection equation is

$$\mathbf{r} = \frac{\mathbf{R}}{\mathbf{R} \cdot \hat{\mathbf{z}}} = \frac{\mathbf{R}}{Z},$$

where $\mathbf{R} = (X, Y, Z)^T$ is the coordinate of a point in the scene, while $\mathbf{r} = (x, y, 1)^T$ is the coordinate of the corresponding image point. In the case of a planar surface, $Z = 1/(\mathbf{n} \cdot \mathbf{r})$, so that the rigid body brightness change constraint equation,

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

becomes

$$E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t}) = 0.$$

Note that the equality is not disturbed if we replace $\mathbf{n}$ by $k\mathbf{n}$ while at the same time replacing $\mathbf{t}$ by $k\mathbf{t}$. This is just a reflection of the scale factor ambiguity already discussed above. We can allow for it by forcing $\mathbf{t}$ (or $\mathbf{n}$) to be a

unit vector. It can also be shown that the equality is not disturbed if we exchange $\mathbf{n}$ and $\mathbf{t}$ and replace $\boldsymbol{\omega}$ with $(\boldsymbol{\omega} + \mathbf{n} \times \mathbf{t})$ [Horn & Negahdaripour 87]. This possibly unexpected result shows that there will be a two-way ambiguity (unless $\mathbf{n}$ happens to be parallel to $\mathbf{t}$).

A least-squares problem suggested by the above analysis has us minimize

$$I = \iint_D \left(E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t})\right)^2 dx\, dy,$$

by suitable choice of $\boldsymbol{\omega}$, $\mathbf{t}$, $\mathbf{n}$. Adding a Lagrange multiplier to help enforce the condition $\mathbf{t} \cdot \mathbf{t} = 1$, leads to

$$I' = \iint_D \left(E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t})\right)^2 dx\, dy + \lambda(\mathbf{t} \cdot \mathbf{t} - 1).$$

Perhaps surprisingly, this problem has a closed-form solution [Horn & Negahdaripour 87]. This solution is, however, complex, involving eigenvector/eigenvalue decomposition of matrices and other operations. Iterative solutions that were discovered earlier suggest more reasonable parallel implementations. We can, for example, once again consider the gradient of the total error.

We note that the derivatives of the total error with respect to $\boldsymbol{\omega}$, $\mathbf{t}$, $\mathbf{n}$, and λ are

$$\begin{aligned}
\frac{dI}{d\boldsymbol{\omega}} &= 2 \iint_D \left(E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t})\right) \mathbf{v}\, dx\, dy, \\
\frac{dI}{\mathbf{t}} &= 2 \iint_D \left(E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t})\right) (\mathbf{n} \cdot \mathbf{r})\, \mathbf{s}\, dx\, dy + 2\lambda \mathbf{t}, \\
\frac{dI}{d\mathbf{n}} &= 2 \iint_D \left(E_t + \mathbf{v} \cdot \boldsymbol{\omega} + (\mathbf{n} \cdot \mathbf{r})(\mathbf{s} \cdot \mathbf{t})\right) (\mathbf{s} \cdot \mathbf{t})\, \mathbf{r}\, dx\, dy, \\
\frac{dI}{d\lambda} &= (\mathbf{t} \cdot \mathbf{t} - 1),
\end{aligned}$$

respectively. Once again, we find that the minimum of I corresponds to a saddle point of I', so that straightforward application of gradient descent will not provide us with the solution. One way to avoid this problem is to derive a closed-form expression for the Lagrange multiplier at the extremum and use this to estimate the multiplier during the iteration, rather than using gradient descent to adjust the estimate. Alternatively, one can use a gradient projection method to arrive at a viable scheme.

One way of doing this is to make sure that the adjustments $\delta\mathbf{t}$ in $\mathbf{t}$ are always orthogonal to $\mathbf{t}$. This can be done by removing the component parallel to $\mathbf{t}$ from the gradient $\mathbf{g}$ as follows

$$\delta\mathbf{t} = -\lambda(\mathbf{g} - (\mathbf{g} \cdot \mathbf{t})\mathbf{t}),$$

where we have assumed that $\mathbf{t}$ really is a unit vector. If it is not, we can use instead

$$\delta\mathbf{t} = -\lambda\big((\mathbf{t}\cdot\mathbf{t})\mathbf{g} - (\mathbf{g}\cdot\mathbf{t})\mathbf{t}\big),$$

This can also be written as a double cross-product:

$$\delta\mathbf{t} = -\lambda(\mathbf{g}\times\mathbf{t})\times\mathbf{t}.$$

Alternatively, we can just adjust $\mathbf{t}$ in the direction of steepest descent and then renormalize the result. For small adjustments the two methods produce similar same result, and the renormalization method is simpler. We then have

$$\mathbf{t}^{n+1} = \frac{\mathbf{t}^n - \lambda\mathbf{g}^n}{\|\mathbf{t}^n - \lambda\mathbf{g}^n\|}.$$

By the way, which of the two possible solutions one arrives at will depend on the initial conditions. Fortunately, the functional relationship between the two solutions is known, so that one can easily compute one from the other. Note that the computation of the local departure from satisfaction of the brightness change constraint equation,

$$E_t + \mathbf{v}\cdot\boldsymbol{\omega} + (\mathbf{n}\cdot\mathbf{r})(\mathbf{s}\cdot\mathbf{t}),$$

need be done only once. Still, there are a considerable number of multiplications at each picture cell, including those needed to compute $\mathbf{s}$ and $\mathbf{v}$. The methods developed for the other special cases should help in the implementation of this more ambitious one.

4.9. Gradient Projection Method

In the above we have designed circuits for solving constrained minimization problems. In several cases the form of the term to be minimized and the form of the constraint made it possible to arrive at a related unconstrained problem, simply by dividing by a suitable normalizing factor (and adding a suitable penalty factor). This cannot always be done. We now discuss in more detail a general method for dealing with these problems.

When implementing algorithms using analog circuitry, it is often convenient to depend on something like steepest descent to solve a minimization problem, even when a closed form solution is available. The reason is that the closed form solution may involve operations (such as matrix inversion) that are difficult to implement directly in analog circuitry. An iterative scheme may take many steps to converge, but if the steps can be performed rapidly, this is not a concern. Similarly, a feedback scheme may take a time that is a large multiple of a basic time constant of the circuitry, but this is not an issue if that time constant is small enough.

A difficulty arises, however, when the minimization problem happens to be constrained. The introduction of Lagrange multipliers, for example, yields

a search for stationary values in a larger space, but the extremum of the original constrained problem now corresponds to a saddle point, and thus can not be found using steepest descent. One possibility is to modify the equations by reversing the sign of the gradient component in the direction corresponding to the Lagrange multiplier and so obtain convergence, at least near the extremum of the original problem [Platt & Barr 88]. Global convergence can often be restored by adding a penalty term that grows quadratical with distance from the constraint surface. This makes the overall scheme fairly complex.

Let us look at the constrained problem in more detail now. Suppose that $f(\mathbf{x})$ is to be minimized, but that the solutions are constrained to lie on the surface $g(\mathbf{x}) = 0$. If we simply follow the gradient

$$f_{\mathbf{x}} = \frac{df}{d\mathbf{x}},$$

we will in general not preserve the condition that $g(\mathbf{x})$ have a constant value. What we can do is to remove the component of the gradient of $f(\mathbf{x})$ in the direction of the gradient of $g(\mathbf{x})$ to obtain:

$$f_{\mathbf{x}} - \frac{f_{\mathbf{x}} \cdot g_{\mathbf{x}}}{g_{\mathbf{x}} \cdot g_{\mathbf{x}}} g_{\mathbf{x}}.$$

Moving in this direction, which is clearly orthogonal to $g_{\mathbf{x}}$, does not alter the value of $g(\mathbf{x})$. We can, of course, us any convenient multiple of this vector to arrive at a feedback scheme. For example, we may wish to use

$$\frac{d\mathbf{x}}{dt} = -\alpha\big((g_{\mathbf{x}} \cdot g_{\mathbf{x}}) f_{\mathbf{x}} - (g_{\mathbf{x}} \cdot f_{\mathbf{x}}) g_{\mathbf{x}}\big).$$

which can also be written in the form

$$\frac{d\mathbf{x}}{dt} = -\alpha\big((g_{\mathbf{x}} \times f_{\mathbf{x}}) \times g_{\mathbf{x}}\big).$$

Clearly, $(d\mathbf{x}/dt) \cdot g_{\mathbf{x}} = 0$, so that $g(\mathbf{x})$ remains constant if we follow this trajectory exactly.

In practice, small errors will lead to departures from a particular surface $g(\mathbf{x}) = c$, and furthermore, we may not start right away on the surface $g(\mathbf{x}) = 0$. So there is a need to introduce a force that pulls the solution towards this constraint surface. We could try to do this by adding a penalty function like

$$\mu\, g(\mathbf{x})^2,$$

to $f(\mathbf{x})$, which would yield a gradient component

$$2\mu\, g(\mathbf{x})\, g_{\mathbf{x}}.$$

This component is directly along the gradient of $g(\mathbf{x})$ and so would be completely removed by the gradient projection scheme discussed above. But there is no reason why we can not add such a term after the gradient projection. This leads to a feedback scheme for solving the constrained minimization

problem like

$$\frac{d\mathbf{x}}{dt} = -\alpha\left((g_{\mathbf{x}} \cdot g_{\mathbf{x}})f_{\mathbf{x}} - (g_{\mathbf{x}} \cdot f_{\mathbf{x}})g_{\mathbf{x}}\right) - \beta\, g(\mathbf{x})g_{\mathbf{x}}.$$

An interesting question is whether this is the gradient of some function, perhaps some modification of $f(\mathbf{x})$. In general it is not, that is, there is no function whose derivative with respect to $\mathbf{x}$ is given by this expression.

Let us consider an example. Suppose we wish to find the minimum of

$$A\,x^2 + 2B\,xy + C\,y^2 \qquad \text{subject to} \quad x^2 + y^2 - 1 = 0.$$

Here we have $\mathbf{x} = (x, y)^T$, so

$$f_{\mathbf{x}} = 2(Ax + By, Bx + Cy)^T \qquad \text{and} \qquad g_{\mathbf{x}} = 2(x, y)^T.$$

Hence

$$g_{\mathbf{x}} \cdot g_{\mathbf{x}} = 4(x^2 + y^2) \qquad \text{and} \qquad g_{\mathbf{x}} \cdot f_{\mathbf{x}} = 4(A\,x^2 + 2B\,xy + C\,y^2).$$

Consequently

$$\left((g_{\mathbf{x}} \cdot g_{\mathbf{x}})f_{\mathbf{x}} - (g_{\mathbf{x}} \cdot f_{\mathbf{x}})g_{\mathbf{x}}\right) = 8\big((C - A)\,xy + B\,(x^2 - y^2)\big)\,(-y, +x)^T.$$

So finally we arrive at a feedback scheme like

$$\frac{d\mathbf{x}}{dt} = -\alpha\big((C - A)\,xy + B\,(x^2 - y^2)\big)\,(-y, +x)^T - \beta(x^2 + y^2 - 1)(x, y)^T.$$

The first term drives the state along circular paths towards one of the two point where the circle intersects a line making an angle θ_0 with the x-axis, where

$$(C - A)\sin 2\theta_0 = 2B\cos 2\theta_0.$$

The second term drives the state radially towards the unit circle.

We can easily generalize this example to finding the smallest eigenvalue of a symmetric matrix M (of arbitrary size). Here we have to minimize

$$f(\mathbf{x}) = \mathbf{x}^T M\mathbf{x} \qquad \text{subject to} \quad g(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x} - 1 = 0.$$

We see that

$$f_{\mathbf{x}} = 2\,M\mathbf{x} \qquad \text{and} \qquad g_{\mathbf{x}} = 2\,\mathbf{x},$$

and so

$$f_{\mathbf{x}} \cdot g_{\mathbf{x}} = 4\,\mathbf{x}^T M\mathbf{x} \qquad \text{and} \qquad g_{\mathbf{x}} \cdot g_{\mathbf{x}} = 4\,\mathbf{x} \cdot \mathbf{x}.$$

As a result we can use the feedback scheme

$$\frac{d\mathbf{x}}{dt} = -\alpha\big((\mathbf{x} \cdot \mathbf{x})\,M\mathbf{x} - (\mathbf{x}^T M\mathbf{x})\mathbf{x}\big) - \beta(\mathbf{x} \cdot \mathbf{x} - 1)\mathbf{x}.$$

The first term, orthogonal to $\mathbf{x}$, drives the state along the surface of a sphere towards one of the two intersections of the sphere with a line through the origin in the direction of the sought after eigenvector. The second term drives the state radially towards the unit sphere.

The generalization to more than one constraint should be obvious. We start by removing from the gradient of $f(\mathbf{x})$ components in the directions of

the gradients of each of the constraint functions $g(\mathbf{x})_i$. Then additional terms are added, proportional to each of the functions $g(\mathbf{x})_i$ in the direction of their gradients.

Reversal of Gradient Component and Addition of Penalty Terms

Platt and Barr have suggested that a modification of gradient descent can be applied to the unconstrained problem obtained by introducing a Langrangian multiplier [Platt & Barr 88]. To mininize $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$, they suggest first constructing the function

$$F(\mathbf{x}, \lambda; \mu) = f(\mathbf{x}) + \lambda g(\mathbf{x}) + \mu g^2(\mathbf{x}),$$

where λ is a Lagrangian multiplier (to be found), while μ is a parameter on the penalty term selected to assure global convergence. Typically, the method converges when started near the solution even when the penalty term is not added. This term is usually needed, however, to prevent divergence when at some distance from the solution. This suggests varying the parameter, according to a predetermined schedule, as the solution is approached. Now

$$F_{\mathbf{x}} = f_{\mathbf{x}} + \lambda g_{\mathbf{x}} + 2\mu g(\mathbf{x}) g_{\mathbf{x}},$$
$$F_{\lambda} = g(\mathbf{x}).$$

In many cases the feedback scheme

$$\frac{d\mathbf{x}}{dt} = -\alpha (f_{\mathbf{x}} + \lambda g_{\mathbf{x}}) - \beta g(\mathbf{x}) g_{\mathbf{x}},$$
$$\frac{d\lambda}{dt} = +\alpha g(\mathbf{x}),$$

leads to the solution (note the positive sign in the second equation).

Applied to the example above, we have

$$F(x, y, \lambda; \mu) = (A\,x^2 + 2B\,xy + C\,y^2) + \lambda(x^2 + y^2 - 1) + \mu(x^2 + y^2 - 1)^2,$$

so we obtain the set of equations

$$\frac{dx}{dt} = -\alpha\big((A + \lambda)x + B\,y\big) - \beta(x^2 + y^2 - 1)x,$$
$$\frac{dy}{dt} = -\alpha\big(A\,x + (B + \lambda)y\big) - \beta(x^2 + y^2 - 1)y,$$
$$\frac{d\lambda}{dt} = +\alpha(x^2 + y^2 - 1).$$

5. 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 where shown to be considerably simpler to implement than circuits based on the closed form solutions usually sought for in digital implementations. It was noted that simple feedback networks with local connections can invert local operations. This is of interest since 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. The use of gradient projection was explored 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.

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.

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 sub-sampling after low-passs filtering. Each layer contains half the number of nodes in its predecessor.

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, since 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 probably started more than twenty years ago, but was never very visible. 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 Meads 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!

6. Acknowledgements

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.

6.1. References

Abdou, I.E.& 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, November (see Appendix).

Ahuja, N. & B.J. Schachter (1983) *Pattern Models*, John Wiley, New York, NY.

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

Anderson, B.O. & 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, June.

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

Berzins, V. (1984) "Accuracy of Laplacian Edge Detectors," *Computer Vision, Graphics and Image Processing*, Vol. 27, No. 2, pp. 195–210, April.

Blake, A. & A. Zisserman (1988) *Visual Reconstruction*, MIT Press, Cambridge, MA.

Brooks, M.J. & B.K.P. Horn (1985) "Shape and Source from Shading," *Proceedings of the International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 18–23, pp. 932–936. Also MIT AI Laboratory Memo 820, January.

Bruss, A.R. & B.K.P. Horn (1983) "Passive Navigation," *Computer Vision, Graphics, and Image Processing*, Vol. 21, No. 1, pp. 3–20, January.

Cagney, F. & 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," MIT AI Laboratory Technical Report 720, July.

Courant, R. & D. Hilbert (1953) *Methods of Mathematical Physics*, Vol. I, John Wiley & Sons, New York, NY.

Courant, R. & D. Hilbert (1962) *Methods of Mathematical Physics*, Vol. II, John Wiley & Sons, New York, NY.

DeWeerth, S.P. & 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.

Frankot, R.T. & R. Chellappa (1988) "A Method for Enforcing Integrability in Shape from Shading Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 4, pp. 439–451, July.. Also Report USC-IPI-105, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, 1986.

Gamble, E. & T.A. Poggio (1987) "Visual Integration and Detection of Discontinuities: The Key Role of Intensity Edges," MIT AI Laboratory Memo 970, October.

Garabedian, P.R. (1964) *Partial Differential Equations*, Wiley.

Geman, S. & 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, November.

Gennert, M. & S. Negahdaripour (1987) "Relaxing the Constant Brightness Assumption in Computing Optical Flow," MIT AI Laboratory Memo 975, June.

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, & Y.H. U (1980) *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 2, No. 1, January, 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, MIT, May.

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, No. ', pp. 39–69, April.

Grzywacs, N. & A. Yuille (1987) "Massively Parallel Implementations of Theories for Apparent Motion," MIT AI Memo 888.

Hartley, R. (1985) "A Gaussian-Weighted Multi-Resolution Edge Detector," *Computer Vision, Graphics and Image Processing*, Vol. 30, No. 1, pp. 70–83, April.

Haralick, R.M. (1980) "Edge and Region Analysis for Digital Image Data," *Computer Graphics and Image Processing*, Vol. 12, No. 1, pp. 60–73, January.

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, February.

Harris, J.G. (1986) "The Coupled Depth/Slope Approach to Surface Reconstruction," MIT AI Laboratory Technical Report 908, June. Also (1987) *Proceedings of the IEEE International Conference on Computer Vision*, London, England, June 8–11, pp. 277–283.

Harris, J.G. (1989) "An Analog VLSI Chip for Thin-Plate Surface Interpolation," *Proceedings of IEEE Neural Information Processing Systems Conference*, November 28–December 1, 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, June.

Hatamian, M. (1987) "A Fast Moment Generating Chip," *Proceedings of the International Conference on Digital Signal Processing*, Florence, Italy, 7–10 September, pp. 230–234.

Hildreth, E. (1980) "Implementation of A Theory of Edge Detection," MIT AI Laboratory Technical Report 579, April.

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, April.

Horn, B.K.P. (1970) "Shape from Shading: a Method for Obtaining the Shape of a Smooth Opaque Object from One View," MIT Project Mac Technical Report TR-79. Also MIT AI Laboratory Technical Report 232.

Horn, B.K.P. (1971) "The Binford-Horn Linefinder," MIT AI Laboratory Memo 285, July.

Horn, B.K.P. (1972) "VISMEM: A bag of 'robotics' formulae," MIT AI Laboratory Working Paper 34, December.

Horn, B.K.P. (1974) "Determining Lightness from an Image," in *Computer Graphics and Image Processing*, Vol. 3, No. 1, December, pp. 277–299.

Horn, B.K.P. (1979) "Automatic Hill-Shading and the Reflectance Map," *Proceedings of the Image Understanding Workshop*, Palo Alto, CA, April 1979, 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, January. Also, same title (1982) *Geo-Processing*, Vol. 2, 1982, pp. 65-146.

Horn, B.K.P. (1983) "The Least Energy Curve," *ACM Transactions on Mathematical Software*, Vol. 9, No. 4, pp. 441–460, December.

Horn, B.K.P. (1986) *Robot Vision*, MIT Press, Cambridge, MA & McGraw-Hill, New York, NY.

Horn, B.K.P. & 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, November.

Horn, B.K.P. & M.J. Brooks (1986) "The Variational Approach to Shape from Shading," *Computer Vision, Graphics and Image Processing*, Vol. 33, No. 2, pp. 174–208, February. Also (1985) MIT AI Memo 813, March.

Horn, B.K.P. & 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, January.

Horn, B.K.P. & B.G. Schunck (1981) "Determining Optical Flow," *Artificial Intelligence*, Vol. 16, No. 1–3, pp. 185–203, August.

Horn, B.K.P. & E.J. Weldon Jr. (1988) "Direct Methods for Recovering Motion," *International Journal of Computer Vision*, Vol. 2, No. 1, pp. 51–76, June.

Hutchinson, J., Koch, C., Luo, J. & C.A. Mead (1988) "Computing Motion using Analog and Binary Resistive Networks," *IEEE Computers*, Vol. 21, pp. 52-63, March.

Ikeuchi, K. (1984) "Reconstructing a Depth Map from Intensity Maps," *International Conference on Pattern Recognition*, Montreal, Canada, July 30–August 2, pp. 736–738. Also "Constructing a Depth Map from Images," MIT AI Laboratory Memo 744, August 1983. Also AD-A135679 available from National Technical Information Service.

Ikeuchi, K. & B.K.P. Horn (1981) "Numerical Shape from Shading and Occluding Boundaries," *Artificial Intelligence*, Vol. 17, No. 1–3, pp. 141–184, August. Also in *Computer Vision*, Brady, J.M. (ed.), North-Holland Publishers.

Knight, T. (1983) "Design of an Integrated Optical Sensor with On-Chip Pre-Processing," Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT.

Koch, C., Marroquin, J. & A. Yuille (1986) "Analog 'Neuronal' Networks in Early Vision," *Proceedings National Acadamy of Siences, USA* (Biophysics), Vol. 83, pp. 4263–4267, June. Also (1985) MIT AI Laboratory Memo 751, June.

Korn, G.A., & T.M. Korn (1968) *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill.

Larson, N.G., K. Nishihara & B.K.P. Horn (1981) "Digital Gaussian Convolver," Patent Application, Registry No. 26192, April 22.

Longuett-Higgins, H.C. & 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 & C. Mead (1988) "An Experimental Subthreshold, Analog CMOS two-dimensional Surface Interpolation Circuit," *Proceedings of IEEE Neural Information Processing Systems Conference*, Denver, November.

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

MacLeod, I.D.G. (1970b) "On Finding Structure in Pictures," in *Picture Language Machines*, S. Kaneff (ed.), 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, MIT.

Marr. D. (1976) "Early Processing of Visual Information," *Philosophical Transactions of the Royal Society B*, Vol. 275, pp. 1377–1388.

Marr, D. & 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. & 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, March.

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, January.

Platt, J.C. & 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. & V. Torre (1984) "Ill-Posed Problems and Regularization Analysis in Early Vision," MIT AI Laboratory Memo 773, October.

Poggio, T.A., H. Voorhess & A. Yuille (1985) "A Regularized Solution to Edge Detection," MIT AI Laboratory Memo 833, May.

Rifman, S.S. & 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*, J.T. Tippet *et al.* (eds.), MIT Press, Cambridge, MA, pp. 159–197.

Rosenfeld, A. & M. Thurston (1971) "Edge and Curve Detection for Visual Scene Analysis," *IEEE Transactions on Computers*, Vol. 20, No. 5, p. 562–569, May.

Rosenfeld, A., M. Thurston & Y.H. Lee (1972) "Edge and Curve Detection: Further Experiments," *IEEE Transactions on Computers*, Vol. 21, No. 7, p. 677–715, July.

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. & 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. & B.K.P. Horn (1983) "Atmospheric Effects in Satellite Imaging of Mountainous Terrain," *Applied Optics*, Vol. 22, No. 11, pp. 1702–1716, June.

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

Strat, T.M. (1979) "A Numerical Method for Shape from Shading for a Single Image," S.M. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge MA.

Tanner, J.E. & C. Mead (1984) "A Correlating Optical Motion Detector," *MIT Conference on Very Large Scale Integration*, pp. 57–64.