A NEURAL NETWORK IMPLEMENTATION FOR INTEGRATING DISCONTINUITY AND DISPLACEMENT INFORMATION

By

Scott Ralph

B.Sc. Honours

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

COMPUTER SCIENCE

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

1991

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
Vancouver, Canada

Date Oct. 8, 1991

DE-6 (2/88)

Abstract

An artificial neural network is presented which performs the task of integrating discontinuity and displacement information to produce an estimate of displacement which is robust in portions of the image containing discontinuous motion. The behavior of the algorithm is *learned* from a series of randomly generated examples, and demonstrates that the necessary constraints required to perform fundamental visual computation can be extracted directly from a series of random images. By phrasing the problem as a machine-learning problem we are able to give explicitly the optimality criteria (given by the learning rule), without making explicit assumptions on the image features necessary to perform the computation.

The motion primitives consist of the sum-of-squared-differences (SSD) values over a set of oriented rectangular support regions. A set of related measures, called *shear values*, are used to detect the position and orientation of discontinuities in the displacement field. By using a set of support regions which vary in shape and size, the algorithm is able to exploit the discontinuity information and choose the support region which best captures the underlying motion of the region.

The resulting algorithm is compared to the traditional SSD algorithm with a single square support region, using both natural and synthetic images. Analysis of the algorithm indicates the neural network is able to reduce the distortion effects occurring near discontinuities and produces object boundaries which are significantly better representations of the object's true structure. The computed displacements show the neural network is able to interpolate over the SSD surface to produce displacements which are within sub-pixel accuracy. Additional confidence measures are given for the neural network and are compared to the traditional SSD algorithm.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my advisor James Little for his helpful suggestions and guidance throughout the development of this thesis. Thanks also to my second reader, David Lowe, for his additional comments and advice.

Thanks also to my many friends at the University of British Columbia who have made my experience here so enjoyable over the last two years. Special thanks to Nedenia Holm who helped keep my spirits up during the long hours of work, and Robert Scharein for his tireless proof-reading, discussion, and helpful comments.

Lastly, but most importantly, I would like to thank my parents, Earle and Judith, who have always offered support and encouragement in all that I undertake. It is to them that this thesis is dedicated.

Chapter 1

Introduction

Vision is our most important sense. It provides us with a remarkable amount of information enabling us to understand and react intelligently to our environment. It allows us to ascertain the position, orientation, and depth of objects, and does so passively. It is not surprising therefore that so much effort in artificial intelligence has been made to enable machines to see.

The lowest level of visual processing is the sampling of the visual image into a collection of discrete intensity values called *pixels*. Low-level visual tasks, such as edge detection, operate on these pixel values directly to create higher-level constructs such as edges and curves [Hil85, Can86, LB85]. Higher-level visual tasks operate on these low-level constructs to produce a description of the scene in terms of position, orientation, and depth of individual objects.

The majority of the successes in computational vision have been at the lower-level visual tasks such as edge detection. While debate continues as to the organization of the intermediate and higher-level visual tasks, one thing is clear; it is an extremely difficult and complicated task. This complexity is confirmed when one looks at the structure and size of the visual cortex of mammals. The total amount of neural hardware devoted to vision, as well as its intricate structure, affirms both its importance to survival, and the complexity of the process [MN87].

The relatively modest success of higher-level visual tasks is due in part to the imaging process. The projection of a three-dimensional scene of light intensities onto a two dimensional image destroys all depth information. In addition some structures in the image may occlude others leaving an image with many ambiguities which must be resolved.

1.1 Depth Cues

The reconstruction of the depth information from one or more images is a very difficult problem. It is not surprising therefore that a large number of visual "cues" must be used to achieve this goal. Two of these cues, binocular stereo and depth from motion, are described below. Binocular stereo and optical flow are similar in that they deal with a one dimensional and two dimensional apparent motion, respectively. Therefore much of the theory presented here will be equally applicable to both problems, however, to simplify the implementation, the algorithm presented here will concern the simpler problem of binocular stereo.

1.1.1 Binocular Stereo

The most intuitive of these cues is binocular stereo which uses two separate images taken from different perspectives. The difference in perspective induces a disparity between features of one image and the other. By knowing the geometry of the imaging system, one is able to solve for the depth of a feature given its relative displacement [Hor86].

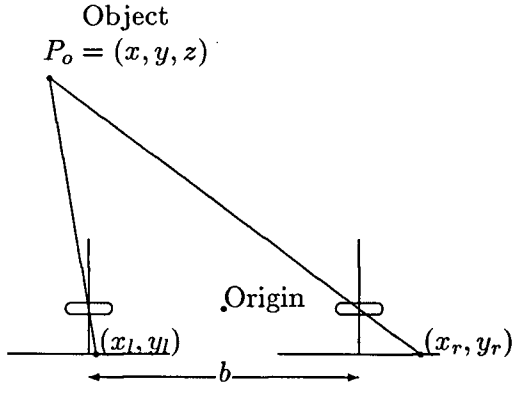


Figure 1.1: Geometry for the recovery of depth from stereopsis

To illustrate this process, consider a system of two rigidly mounted cameras whose optical axes are parallel, and are separated by a distance b (see Fig. 1.1). Assume also that the focal length of each camera is f. The line which passes through the center of each lens is called the *baseline*. In order to simplify the computation the baseline is taken to be perpendicular to both

optical axes, and parallel to the x-axis. Denote by $P_o = (x, y, z)$ the coördinates of the point in the scene relative to the origin placed on the baseline halfway between both lens centers. Provided the point P_o is visible to both cameras, two image points (x_l, y_l) and (x_r, y_r) will be produced in the left and right images, respectively. Such a pair of points is called a *conjugate pair.* From the geometry of the system

$$\frac{x_l}{f} = \frac{x + b/2}{z}, \qquad \frac{x_r}{f} = \frac{x - b/2}{z}, \qquad \text{and} \qquad \frac{y_l}{f} = \frac{y_r}{f} = \frac{y}{z}.$$

We may now compute the *disparity*, D, of the two points by taking the difference

$$D = x_l - x_r = \frac{bf}{z}.$$

Knowing both b and f allows us to solve for x, y, and z:

$$x = \frac{b(x_l + x_r)}{2D}, \qquad y = \frac{b(y_l + y_r)}{2D}, \qquad \text{and} \qquad z = \frac{D}{bf}.$$

This shows that the disparity of a conjugate pair is inversely proportional to the depth. This implies that the depth of nearby points may be measured accurately, while those further away are more prone to error.

The determination of depth from a two stereo images as described above presupposes the existence of a dense set of conjugate pairs. But how are these pairs obtained from the two images? This problem is named the *image correspondence problem.* If we assume that the brightness of a given point does not change from the left and right images, then we are faced with matching a point in one image to a corresponding point in the other which has the same (or nearly the same) brightness. The actual brightness of the point may differ in the right and left images due to noise in the sampling process, self-illumination, and other phenomena.

There are many problems with this approach. First, the brightness value may have many potential matches in the second image. The process of choosing the best match among these points cannot be computed from purely local brightness information. This is similar to the aperture problem in optical flow computation (discussed further in §1.1.2). Regions in the image with little brightness variation will also give rise to false matches in the image. Additionally there may be regions in the image (called occlusions) which are only visible from one of the two

cameras making the matching impossible. Because local brightness information is unable to sufficiently constrain the correspondence of points in the stereo images, additional constraints must be added to obtain a unique solution. These constraints include assumptions on the nature of the displacement, as well as the shape of the objects in the scene. The types of constraints which are often employed in binocular stereo and optical flow are discussed further in §1.3.

1.1.2 Optical Flow

In addition to the computation of the depth information directly from stereopsis, depth information may also be recovered from other less obvious cues, the most noteworthy being the use of optical flow.

When a point in the environment moves, or the observer is moving relative to a point, there is a corresponding motion induced in the image. If we define a motion field that assigns a three-dimensional velocity vector to each point in the scene, then the apparent motion induced in the image is called the *optical flow.* While there are pathological cases where the optical flow does not reflect the true motion of the scene, the optical flow generally corresponds closely to the true motion field [HS81].

If we are given two images taken over a relatively short interval of time, the optical flow field is a mapping in which each pixel is given a vector which indicates the apparent motion in the image. Suppose we are given two images, denoted $I_0(x, y)$ and $I_1(x, y)$, taken at two times t_0 and t_1, respectively, and we wish to find the optical flow for a point P at t_0. If t_0 and t_1 are sufficiently close together, we may assume that the brightness of a given point in the image remains constant. If we further assume that the magnitude of the optical flow is relatively small, and that the brightness values near the point P vary continuously, the problem reduces to finding a point in I_1 near P which has the same brightness value as P. Let C_0 and C_1 denote the iso-brightness contours of brightness values equal to that at point P (see Fig. 1.2).

We are faced with picking a point along C_1 near the point P. The problem of choosing an individual point along the contour C_1 is difficult since, in general, these contours will not even have the same shape [VP86].

This problem of choosing the correct point along the iso-brightness contour is called the

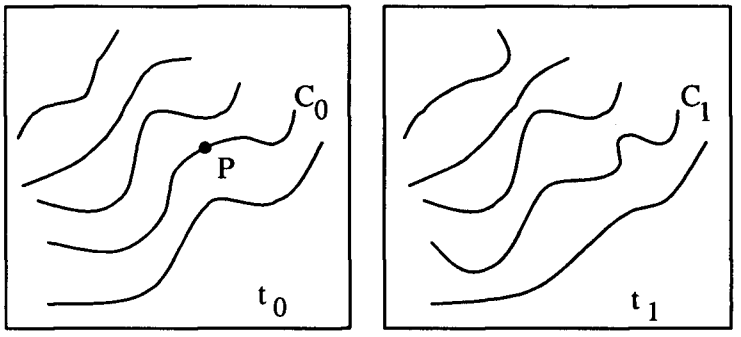


Figure 1.2: Iso-brightness contours of an image at times t_0 and t_1.

aperture problem [MU81]. This problem states that local variations in the brightness values does not uniquely determine the optical flow. Local brightness information is only able to determine the component of the optical flow which is perpendicular to the image gradient. The component of the optical flow which is parallel to the image gradient (*i.e.*, along the iso-brightness contour of that point) cannot be recovered locally [Hor86]. In some points in the image (such as the corner of an object) it may possible to recover both components of the motion, but for cases where the local image structure is one dimensional (such as an edge of an object) only one component can be recovered[1].

Additionally, relaxing the brightness constancy to allow variation due to changes in illumination, or dealing with occlusions will greatly increase the complexity of our task. More fundamentally, since local information is not able to determine a unique solution, additional constraints must be placed on the images. It is mostly the choice of these constraints which characterizes the performance of an optical flow algorithm. The issues related to the choice for constraints will be further discussed later in §1.3.

Once the optical flow is computed, it gives us a great deal of information about the scene. The process of determining depth from a motion field is called *motion parallax* and is perhaps the most fundamental of all monocular cues of depth. Motion fields contain the least assumption-laden information of the layout of the scene and allow the recovery of both the slant of surfaces [KD76], and the relative distances of features [NL74]. A detailed description of additional

[1] As we shall see in §1.3.1, the use of smoothness constraints such as in Horn-Schunck's optical flow algorithm *always* gives rise to the aperture problem since the gradients only provide one constraint for the two degree of freedom motion field.

information which can be obtained from the motion field is found in [Nak85].

1.1.3 Other Cues

Shading information of an object may also be used to recover the orientation of an object, provided knowledge of the light source and the surface characteristics of the object are known. The *image irradiance equation* of an object is a function which gives the relationship of surface brightness to surface orientation. The reflectance map depends on the properties of the surface material, as well as the distribution of light sources. This relationship is usually denoted as

$$E(x, y) = R(p, q)$$

where (x, y) is the position of the point, and p and q give the orientation of the surface at that point [Hor86]. p and q correspond to the gradients of the surface in the x and y directions, respectively.

For example, a Lambertian surface illuminated by a point source at the same place as the viewer is given by

$$E(x, y) = R(p, q) = (1 + p^2 + q^2)^{-1/2}.$$

From the reflectance map, a single image is only able to recover the surface orientation for a few points such as the points where the brightness is at a maximum or minimum. Because brightness has only one degree of freedom, and surface orientation has two, the reflectance map will not uniquely determine the surface orientation. If however one were to take two images at the same viewpoint under two different sources of illumination, a unique surface orientation may be obtained. This process is called *photometric stereo* [Woo78]. An excellent description of shape from shading and photometric stereo is found in [Hor75].

1.2 The Image Correspondence Problem

Of these cues for depth information, optical flow and binocular stereo are the most closely related. The similarity stems from the fact that both problems involve solving for the relative

displacement between features of two images. This problem is known as the *image correspondence problem.* Methods for solving the correspondence problem fall into two categories which will be referred to as *continuous methods* and *feature-based methods.*

Continuous methods are based on mechanisms which treat the images as a continuous three-dimensional spatiotemporal function. The motion of the scene corresponds to characteristic mathematical properties of the spatiotemporal image function which can be extracted by simple linear filters. While these properties can be described in a number of ways, they are often mathematically equivalent. One example of this type of approach is the use of Gabor functions which are sensitive to motion occurring at a particular direction and velocity [AB85]. It has been demonstrated that a set of six such filters are able estimate the local components of 2D translation, divergence, curl, and the shear/deformations which span the six degrees of freedom of 2D image motion resulting from 3D rigid-body motion [Eag91].

Feature-based methods differ in that they do not directly deal with the images at the level of individual intensity values, but rather first process the images and extract a set of *features.* These features are then tracked over time to obtain the estimate of the motion or disparity field. These features are usually edges or corners as they have a high likelihood of being relevent to objects in the scene. An example of this approach can be found in [LM87].

Continuous methods have the advantage of being able to use information from each point in the image, whereas feature-based approaches are limited to locations where a feature is present. The advantage of feature-based approaches is that these features are generally more resilient to changes in illumination, self-illumination and other phenomena which cause difficulties in continuous methods.

Regardless of whether continuous or feature-based methods are used to solve the correspondence problem, both are under-constrained. We will now turn to the subject of additional constraints which may be placed on the images to obtain a unique solution to the correspondence problem.

1.3 Constraints

The fact that the determination of optical flow and binocular stereo are under-constrained does not mean that all is lost. Ambiguities occur frequently in natural scenes, yet we are still able to draw meaning from them. This is because we are able to place further constraints on the image enabling sensible interpretations of the scene. The challenge faced by researchers in computer vision is to develop further constraints which allow unique and accurate solutions for the types of scenes the vision system is expected to solve.

While there are a large number of diverse algorithms for the computation of optical flow and binocular stereo, the most salient features of each are the choice of constraints used to further constrain the solutions.

1.3.1 Smoothness of the Flow Field

One intuitive constraint often employed in the computation of optical flow and stereo disparity is to enforce smoothness of the flow field [HS81, VP86]. This means that in addition to satisfying the constraints from the local brightness information, we should favor solutions in which the gradient of the flow is relatively small. This is certainly the case for rigid-body motion with the exception of regions near the boundary of an object. To illustrate this, let us denote the x and y components of the flow by $u(x,y)$ and $v(x,y)$, respectively. The process of enforcing locally smooth flow fields is often phrased as a regularization process of the Tikhonov type [PTK87]. The brightness constancy equation tells us that the total time derivative of the brightness is zero, *i.e.*, $\frac{dE}{dt} = 0$. If we use the following abbreviations

$$u = \frac{dx}{dt}, \qquad v = \frac{dy}{dt}, \qquad E_x = \frac{\partial E}{\partial t}, \qquad E_y = \frac{\partial E}{\partial y}, \qquad \text{and} \qquad E_t = \frac{\partial E}{\partial t},$$

then by the chain rule it follows that [FT79]:

$$E_x u + E_y v + E_t \;=\; 0. \tag{1.1}$$

The Tikhonov stabilizer introduced in [HS81] allows a unique solution to be obtained

$$\int \underbrace{(E_x u + E_y v + E_t)^2}_{e_c} + \lambda \underbrace{(u_x^2 + u_y^2 + v_x^2 + v_y^2)}_{e_s}\, dxdy. \tag{1.2}$$

We now seek the value of (u, v) which minimizes this expression. The first term of Eqn. (1.2), e_c, measures the residual of the optical flow constraint equation. This term gives us a measure of how well the choice of (u, v) corresponds to the local brightness information. The second term, e_s, measures the gradient of the optical flow and should be minimized to produce a smooth flow field. The parameter λ allows a potential solution to tradeoff the error from the brightness data with the smoothness of the flow field. The choice for an appropriate value for λ is crucial if the solution is to be useful. If λ is too large then the detail of the flow field will be smeared and reduced to a constant vector which is close to the mean velocity of the region. If, on the other hand, λ is too small, then the contribution of e_s to the regularization will be insufficient to adequately constrain the solution, and will only reflect the e_c term. Under these conditions the solution will contain large velocity gradients. Although λ is chosen empirically in practice, methods do exist to determine the optimal λ value [TA77], but require careful analysis of the nature of the ill-posed problem.

1.3.2 Correlation-Based Approaches

One intuitive and popular choice of constraints is embodied in a class of algorithms called *correlation-based* techniques [Ana89, BLP87]. When the projected motion of an object is small relative to the image, it is possible to limit the search for correspondence to small regions in the image. This allows us to simply search for discrete values of $(u(x, y), v(x, y)) \in (\pm\delta, \pm\delta)$ which minimizes Eqn. (1.2). For an $N = (n \times n)$ image we must choose a solution from a set of $(2\delta + 1)^{2N}$ possible fields [BLP87]. While this entails an enormous computational complexity, many of these fields will be far from smooth and can be eliminated from the search by using further constraints.

If we further assume that the objects are piecewise planar, and that the motion of the object is orthogonal to the viewing direction, then the image motion corresponding to a planar patch in the image will be constant over the projected area of the patch. We may now search for the correct displacement by choosing a small patch, P, centered at (x, y) and comparing it to the patches corresponding to each potential displacement. This reduces the number of potential vector fields which must be searched to $(2\delta + 1)^2$. The choice for the radius of each patch

(denoted ξ), as well as the magnitude of δ, depend on the scale of the objects in the scene as well as their expected velocities. Typically a patch of size (11×11) is sufficient for most images [BLP87].

By choosing an appropriate function ϕ to compare patches in the image, we may now construct an approximation to Eqn. (1.2)

$$\sum_{P} \phi(E_t(x,y), E_{t+\Delta t}(x + u\Delta t, y + v\Delta t)). \tag{1.3}$$

This search is often refered to as a *shift-and-compare* operation [BLP87].

It should be obvious that this approximation also minimizes the residual, e_e, but we no longer have a smoothness measure to minimize. This constraint is *implicitly* contained in our assumption of frontoparallel piecewise-planar motion and our choice for the size of the patch. If our choice for ξ is large, then we will impose a strong smoothness constraint on the flow field, while a smaller values enforces this constraint to a smaller degree. It should be noted that in regions where the motion is not frontoparallel translation, nor the surface planar, neither e_s nor e_e are minimized and the approximation breaks down, and the results may be poor. Little and Verri have shown that for motion corresponding to non-frontoparallel the shift-and-compare operator is often still able to recover a good approximation of the true motion [LV88].

The choice of the comparison function ϕ is often taken to be the sum-of-squared-differences (SSD), the sum of the absolute differences, or some similar measure. The total complexity of the shift-and-compare-operation now becomes $|P|(\delta + 1)^2$ multiplications, where $|P|$ is the the total size of each patch (in pixels). In later experiments we will use the SSD as our comparison operator, but may use the more general term of "correlation" since the SSD gives a reasonable approximation of the cross-correlation.

While this is a large solution space to search, the algorithm is easily parallelized [BLP87]. Consider a large number of fine-grain processors which are arranged in layers; one layer for each possible displacement. Each processor computes the comparison function, ϕ, for its assigned displacement which may differ in magnitude and direction. The second step involves the collection of "votes" from each layer in a small circular region around each point in the image. The best displacement is chosen from the set of votes using non-maximal suppression.

In regions where the motion is frontoparallel and coherent across the region, a large vote is expected. Displacement predictions are only obtained in regions where there is a clear winner; ties are discarded[2]. This approach to the computation of optical flow is similar to the Connection Machine™ implementation of the Marr-Poggio coöperative stereo algorithm [DP86].

There are a number of reasons why correlation-based techniques seem appropriate for the computation of optical flow and stereo. First, the shift-and-compare operation seems to capture the intuitive definition of what it means for two images to be in correspondence. There has also been a great deal of physiological and psychophysical evidence to indicate that correlation-based approaches are indeed biologically motivated. Reichardt used a scheme similar to the correlation-based approached described above, and was able to fully characterize the optomotor responses of the fly [Rei61]. Bülthoff *et al.* have given psychophysical evidence which demonstrates that this shift-and-compare strategy gives rise to behaviour similar to humans when confronted with illusions [BLP87], including the barber pole illusion, and the non-rigidity illusion [NS88]. There is also some evidence to indicate that other mechanisms such as fusing of texture boundaries may play an important rôle in recovering depth from a set of images [Cav87].

1.3.3 Stereo Correlation

While the above formulation is posed as an optical flow problem, the algorithm can be used to obtain disparity measures in binocular stereo with little modification [DP86]. This problem is easier since it reduces the search to a one dimensional space of potential displacements. This search space is one dimensional since an object which is imaged along an epipolar-line in the left image can only be imaged on the corresponding epipolar-line of the right image (if at all). *Epipolar lines* are the intersection of the image plane with a plane containing both lens centers [Hor86]. In practice the optics are usually arranged so that the epipolar lines are parallel to the x-axis, thus making the matching process easier. The conjugate-pairs are constructed by taking a planar region around each point, and matching it to a corresponding region in the alternate image along the corresponding epipolar line. An example of this type of approach can

[2]This will be refered to as the *winner-take-all* correlation-based algorithm.

be found in [Nis84].

1.3.4 Feature Based Correlation Methods

While the shift-and-compare algorithm described above uses a comparison operator on the raw image intensities, such a scheme can also be applied to other image features. Drumheller and Poggio used the correlation of zero-crossings of the image to produce disparity measures [DP86]. A similar use of correlation on zero-crossings is used for the computation of optical flow [BLP87, Nis84].

1.4 Problems at Discontinuities

The Horn-Schunck regularization, Eqn. (1.2), and the correlation-based assumptions of frontoparallel planar motion both exploit smoothness constraints to obtain a unique solution. While this is a reasonable assumption for most parts of real images, these assumptions clearly break down for points in the image which are close to the boundary of an object. In these regions the true motion field will contain discontinuities, which, when projected onto the imaging plane, induce a corresponding discontinuity in the flow field. Regions containing these discontinuities, because they violate the smoothness constraint, result in meaningless predictions for the apparent motion. Consider a point p in an image corresponding to the boundary of of an object which is moving in a static background. Solving for the apparent displacement of p using Eqn. (1.2) requires estimations for the derivatives u_x, u_y, v_x, and v_y, which determine the departure from smoothness term e_s. At these points in the image neighboring points will be uncorrelated and hence Eqn. (1.1) is violated. Since the estimations of the derivatives require finite extent these values will become meaningless (often becoming extremely large). This causes the solution of the minimization to produce unrealistically smooth flow fields which poorly satisfy the optical flow constraint equation, and thus is not characteristic of the true motion.

Similar problems at the discontinuities occur when using correlation-based approaches, albeit less severe. When attempting to match a small patch of an image located near a discontinuity, we are in effect attempting to match a region which contains features from both the

object itself, and of the background. This situation is depicted in Fig. 1.3. This may result in one of two phenomenon: the error surface becomes bimodal [SU87], or the minimum of the error surface shifts due to "smearing" effects of the two separate distributions [LG90]. In the bimodal case, two significant minima (local minima significantly smaller than the remaining minima) arise, one representing the displacement of the object, and the other representing the displacement of the background. These two minima must be disambiguated before the actual displacement can be obtained. While we might decide to hedge our bets and pick the displacement corresponding to the global minimum, there is no guarantee that this corresponds to the true motion. In fact, the relative error values for each minimum may depend more on the relative texture of the object and the background than the true displacement of the point. In the case where the error surface remains unimodal, the presence of the opposing motion may cause the minimum of the error surface to shift away from the true motion.

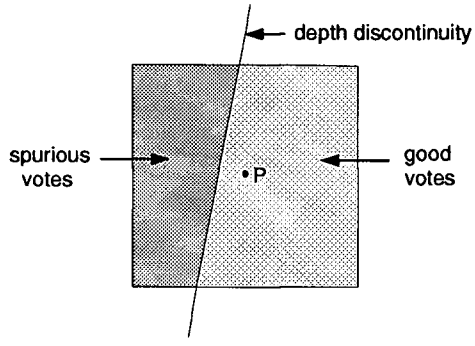


Figure 1.3: Support region near a discontinuity

The presence of the discontinuity partitions the support region into a region of "good votes" representing the object, and a region of spurious votes.

When either type of distortion of the error surface takes place, the apparent boundaries of the objects will become smeared. The flow for points on the objects near the boundaries may gravitate towards that of the boundary due to the averaging process of correlating over large regions. Correspondingly, the flow for points in the background near the object boundary

may gravitate towards that of the object. The magnitude of these distortion effects will be proportional to the radius of the patch used in the matching process.

Thus we are faced with a dilemma: how do we choose an appropriate size for our support region (radius of the patch)? Larger patches are better able to accurately resolve the true apparent motion of the region, but only near coherent motion. Smaller patches, on the other hand are less likely to be distorted by the discontinuities, but are more susceptible to noise.

In order to make reliable flow measurements in all portions of the image it will become necessary to first identify discontinuities in the flow field, and then to use this information to change dynamically both the size and the shape of the support region. This is a Catch-22 in that it is difficult to accurately locate discontinuities without reliable estimates of the flow. Likewise the flow estimates are difficult without localizing the discontinuities.

1.5 Improving Correlation Measures Near Discontinuities

Given knowledge of the discontinuities, one can improve the performance of the matching process by dynamically re-shaping the support region. Little and Gillett [LG90] propose using a set of five support regions which are obtained by bisecting the standard square region along the x and y axes (see Fig. 1.4). Denote by $\mathcal{S} = \{R, R_n, R_s, R_w, R_e\}$ this set of *sub-neighborhoods*, representing the north, south, west, and east orientations, respectively.

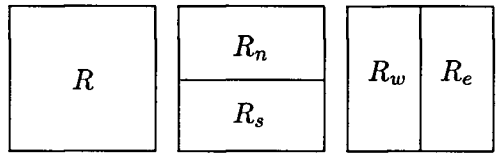


Figure 1.4: Set of 5 oriented rectilinear sub-neighborhoods

By choosing the appropriate sub-neighborhood we may throw away the spurious votes and keep only those votes which correspond to the displacement of the object point. For example, when trying to determine the displacement for point p in Fig. 1.3, the choice of R_e would omit all spurious votes and would enable a proper match.

This process, however, of choosing the appropriate sub-neighborhood presumes knowledge of both the location and orientation of the discontinuity. The process of locating the discontinuities can be achieved by comparing the motion predicted by each sub-neighborhood as suggested in [LG90]. In regions where the motion is coherent across the entire region, there will be a consensus among the sub-neighborhoods, while regions crossing a discontinuity will produce a conflict between at least two of the sub-neighborhoods. In addition to being able to detect the discontinuities, the orientation can also be recovered in most situations. This determination of the orientation of the discontinuity is discussed further in Chapter 3.

We could propose to extend our set of support regions to include other orientations, such as the diagonals, as well as including support regions taken at different scales (as found in [Ana89]). This would likely allow a more accurate estimation of motion in the presence of discontinuities, but, in an effort to minimize the overall complexity of our algorithm the minimal set of 5 sub-neighborhoods was chosen.

1.6 Integration of Discontinuity and Motion Information

Given a set of support regions (such as $\mathcal{S}$) which vary in size and orientation, as well as knowledge of the location and orientation of the discontinuities in the image, the task remains to integrate these two sources of knowledge into a common framework which is able to respond dynamically to the presence of discontinuities and thereby enable accurate motion determination for all points in the image.

The task of integrating visual knowledge from separate sources (also called visual modules) has been studied by a small number of vision researchers; the most notable being [Mar82, AS89, CY90]. Of special interest to our problem of integrating discontinuity and displacement information is the work of [PGL88] which used a stochastic process to integrate several early-vision cues. The collection of outputs from the early-vision cues were modeled by a series of coupled Markov random fields (MRFs). By explicitly representing the discontinuities of the separate fields (depth, brightness, motion, *etc.*) as a line process, the discontinuity information can be used to help fuse the information from the separate modules. This coupling of the

discontinuities across separate visual modules is motivated by the observation that because the cues share a common imaging geometry, discontinuities will tend to be preserved across multiple modalities.

The integration algorithm which is presented here differs from most traditional formalisms in that there is no explicit representation for the interaction of the separate modules, but instead this representation is "learned" from a sequence of examples. This model is an example of a *connectionist model* and is described in the following sections along with a description of the learning algorithm employed. The term *"connectionism"* refers to a set of computational models which are composed of a large number of relatively simple computing engines which are interconnected in a highly parallel manner. These computing elements are usually constructed so as to simulate actual neurons (at some level). While many researchers are interested in connectionist models from a purely computational viewpoint, the main motivation for the development and application of these models is their apparent ability to mimic human neuronal behaviour [MP43, Ros62]. An excellent introduction to connectionist models can be found in [HKP91]. A description of artificial neural networks (or connectionism) as well as a motivation for their use in computational vision is given in §1.7.

Chapter 4 describes an implementation of a correlation-based algorithm which is able to extract the location and orientation of discontinuities in an image from the set of oriented sub-neighborhoods described above. This discontinuity information is then used to select the size and shape of the support region. From this dynamic selection of support, a more accurate estimation of motion can be obtained in regions containing discontinuities. By reducing the smoothing effects near the discontinuities a more accurate representation of an object's shape can be obtained. The overall structure of the proposed algorithm is shown in Fig. 1.5.

1.7 Connectionist Models

People have long been interested in the mechanics of brain activity. The capability of the brain to perform tasks such as pattern recognition, visual processing, and speech recognition are far superior to that of contemporary computers. As serial computers are pushed closer and closer

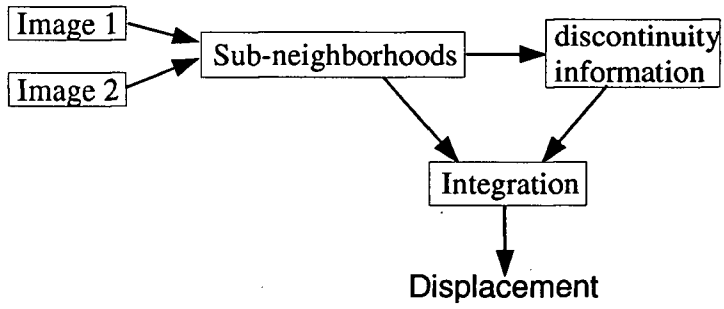


Figure 1.5: Architecture of the integration algorithm

to their theoretical limits, the search for a more powerful computational paradigm becomes more and more attractive.

The basic unit of computation in artificial neural networks is the individual neuron. By examining the behaviour of individual neurons several researchers have proposed mathematical models which describe their behaviour [Ros62]. A neural network (artificial or real) is a collection of interconnected neurons.

A neuron consists of three components: dendrite, cell body, and axon (see Fig. 1.6). The dendrites are the receptors of the neuron and collect nerve impulses from other neurons. The membrane of the neuron's cell membrane is capable of sustaining charge. When this charge reaches a certain threshold, the neuron will "fire". When this occurs a quick wave of depolarization spreads along the length of the outgoing axon where it will ultimately communicate to a neighboring neuron via the synapse which may be either excitatory or inhibitory. An important property of natural and artificial neural networks is the high degree of fan-out (called the *dendritic tree*) which allows a neuron to communicate to a large number of neighboring neurons.

One of the first group of researchers to propose a model of the neuron and its interconnections was McCullough and Pitts [MP43] who viewed the behaviour of neurons in a boolean logic framework and proved that any logical proposition could be realized using their model. One of the few qualities of their model which has been generally accepted is the "all-or-nothing" response of the neurons. The next influential model of neuronal behaviour was the *linear perceptron* [Ros62]. This model used a set of *weights* to represent the synaptic strengths (positive

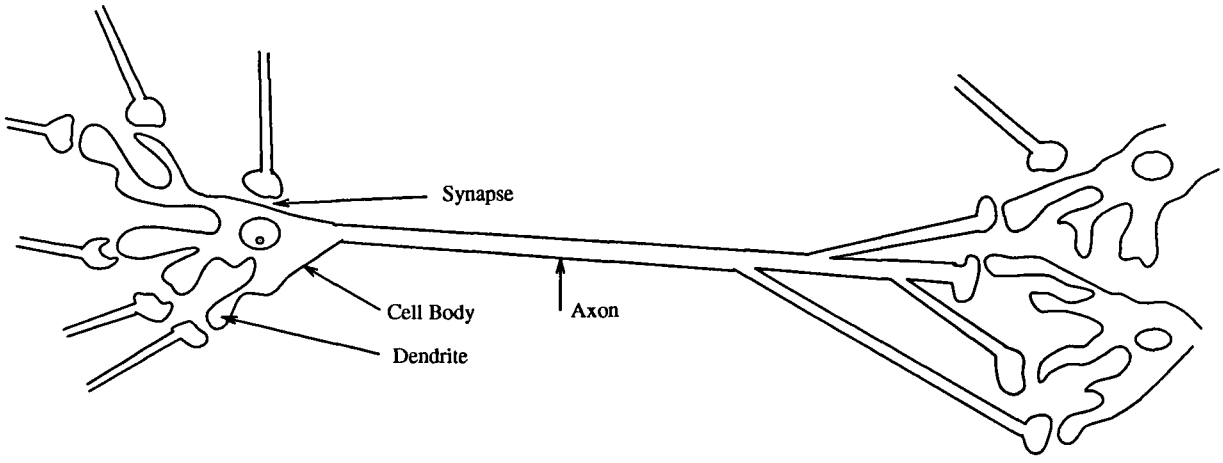


Figure 1.6: The anatomy of a neuron

for excitatory and negative for inhibitory) for each of the inputs (usually a real number). Each input is multiplied by its corresponding weight and is summed. This sum is then compared to a threshold to produce a single boolean value. Rosenblatt went on to prove that the perceptron was able to compute a great many interesting functions, and perhaps more importantly, the existence of a learning procedure whereby the set of weights could be iteratively modified to arrive at a correct solution [Ros62]. Despite many successes the linear perceptron was found to have many limitations due to its linear behaviour (see [MP88] for details).

The neuron model we shall adopt is a non-linear version of the perceptron introduced by Rosenblatt. This model does not have boolean valued outputs (like the linear perceptron), but instead has continuous real-valued output responses. Since neurons only permit two states, the outputs must now be interpreted as rates-of-fire over a small time interval. More will be said about this model in §1.8.2, but first let us outline the motivation for choosing a connectionist model for the problem at hand, as well as the advantages and limitations of such models.

The most obvious motivation for developing our integration algorithm within a connectionist framework is to gain a degree of biological plausibility. By performing our computation using primitives which attempt to simulate the behaviour of individual neurons, there is a greater likelihood that the resulting computation could be performed by the brain. Secondly, connectionist paradigms are interesting models of parallel computation. By examining the structure

and behaviour of these models we may gain insights into the parallel nature of the problem. The investigation of these models may also give insights for new heuristics which may be used to solve the problem using more traditional serial algorithms.

1.7.1 Biological Plausibility

The key justification for connectionist models stems from the observation that the basic brain functions can be described as computation [MP43, Mar82]. As an example, perception can be seen as the act of computing properties of the state of the world from the effects the state makes on sensory receptors. The nature of biological computation is very different from that of serial computation. While modern computers are able to perform individual calculations at a very fast rate, present von Neumann architectures perform serial computation. This serial computation means that at any given instant only a small fraction of the total available hardware is active. Neurons on the other hand are comparatively slow and are only able to transmit information at the rate of once every 5ms. The information the neuron is able to convey is limited to a few bits compared the to the intricate data structures which can be copied using pointer arithmetic in serial computers. In addition, the possible destinations a neuron is able to transmit is limited by its connections, which are fixed. The brain's ability to compute complex functions is achieved through massive parallelism which allows a much larger fraction of the total "hardware" of the brain to be active at a given time. The human brain contains approximately 10^{11} neurons, each of which is connected to as many as several thousand other neurons.

Marr has provided an influential analysis of the issue of *levels* in cognitive science and has outlined three such levels in his theory[3]:

Computational theory This includes the goal of the computation as well as the logic behind the strategy for carrying it out.

Representation and Algorithm This contains the representation of the input and output as well as an algorithm to carry out the computation.

Implementation How the algorithm will be physically realized.

[3]Note: Adapted from [Mar82].

It has been argued that since connectionist and conventional models of computation are Turing-equivalent, it does not matter whether we implement the algorithm using one model or the other. Comparing connectionist and conventional models of computation is similar to the comparison of an algorithm implemented using a high-level language specification such as Pascal, and a similar implementation in assembly language [RM86]. It is necessary to have the the assembly language program and the Pascal program map exactly only when the assembly language is obtained by compiling the Pascal source. If, however, the program is written in assembly language there is no guarantee that such a relationship will exist. In these cases it may be extremely difficult (or even impossible) to extract a higher-level interpretation of the underlying computation. Since there is a reason to believe that most cognitive processes (especially those relating to perception) are performed at the "lower-levels" rather than at the "higher-levels", it is unlikely that a particular high-level description of the computation will be identical to the low-level description. We may attempt to capture an approximation of the lower-level computation in our high-level description, and indeed it might prove useful to do so, but there is no guarantee that this approximation will be adequate for the problem at hand. The Turing-equivalence argument also ignores the empirical fact that the choice of the programming paradigm greatly influences the way in which the researcher thinks about the problem, as well as the types of solutions generated.

In some instances the interaction of a large number of computing elements may appear to coöperate to produce a computation which is greater than the sum of the elements [For90]. In such cases the overall behaviour of the system can only be viewed at the global rather than local level. This type of behaviour is called *emergent computation* and is eloquently discussed in [For90]. Lastly, this equivalence of connectionist and serial computation is a violation of the *hundred-step-rule* proposed in [FB82]. It has been determined by psychological experiments that humans take on the order of 500ms to perform complex visual tasks such as recognition. Knowing that a typical neuron takes approximately 5ms to fire and transmit its information to a neighboring neuron, we can establish an upper-bound for the total number of communication cycles to be about 100. If one is serious about developing computer vision algorithms which are biologically motivated, it is necessary to use a formalism which makes the number of iterations

explicit.

1.8 Choosing the Connectionist Model

There have been a number of connectionist models proposed since their revival in the early 1980's. Each of these models has their own particular strengths and weaknesses within different domains. These models fall into three classes: detailed neuronal models, unstructured models (sometimes refered to as PDP models [RM86]), and structured models.

The detailed neuronal model borrows heavily from experimental neuroscience to obtain a model which is as close to the true behaviour of the neuron as possible. For these models observed behaviours such as sensitivity to motion or orientation are mimicked. An excellent example of this style of work is found in [TP78].

PDP models have received the most attention of all the connectionist models to date. These models are often used for such tasks as pattern recognition and in other applications where the domain is poorly understood [RM86]. The work in these models concentrates strongly on mechanisms of learning rather than the exact modeling of individual neurons. Through the process of learning it is hoped that regularities of the input will be discovered and exploited to give rise to a robust solution.

Structured models such as [FB82] have been utilized in problems where the unstructured PDP models appear to be too simplistic. The focus of this work is on the design of systems, where the connectionist model is used to constrain the design considerations. Analysis of the visual cortex has provide much of the motivation for the work in this area.

1.8.1 Benefits of the PDP Model

There are many benefits we may hope to exploit by using the unstructured PDP model [RM86]:

The existence of a learning procedure. Since the nature of the problem is often not well defined, we are not able to exploit the detailed information of the biological systems and thus we are forced to use an unstructured PDP paradigm. By using the PDP model it is hoped that the structure and representation of the solution can be obtained from

the learning procedure. Supervised learning techniques find their solution by repeatedly modifying the connection strengths between each neuron so as to reduce some global error measurement [RH86].

Generalization of the input data. By discovering regularities in the input data, it is hoped that a solution which minimizes the global error measure of the learning examples will also perform well for inputs which have never been seen before.

Resilience to noise and incomplete data Because of the large fan-in and fan-out of the individual neuron it is unlikely that a single stimulus will give rise to an action potential. This suggests that human computation does not involve the kind of logic circuits used in conventional symbolic computation, but must instead involve some sort of stochastic process. A consequence of the stability achieved through this stochastic process is a better ability to deal with noise in the input data. Additionally, omissions in the input data may still provide enough information of the true data to allow the correct action potentials to occur.

Graceful degradation with damage. The effects of localized damage to artificial neural networks parallels that found in biological neural systems in that there is a gradual loss in ability due to damage [RM86]. In general, the degree of performance loss is roughly proportional to the total fraction of damage. In traditional paradigms of computation the loss in performance due to damage is very erratic. The negation of a single bit for example may cause no observable difference to the operation of the algorithm as long as the data is not used, or it may cause a complete failure.

While these benefits seem encouraging, research on connectionist models have uncovered limitations regarding what can be represented and computed. These limitations arise from the following constraints of our model:

- Once learning is completed the connection strengths are fixed and cannot be modified while the network is "running".

- Networks have to be extremely parallel to satisfy the one hundred step rule. This large degree of connectivity between the neurons gives rise to a correspondingly large computational complexity of the learning procedure.

- The brain has a finite number of neurons. This has repercussions to network implementations of recursive constructs. While it has been argued in [RM86] that these recursive constructs are not necessarily needed for cognitive tasks, it does pose a constraint on the types of algorithms realizable.

Now that the potential benefits and pitfalls of using the PDP model have been outlined, it is necessary to clearly define the model of computation. This model, *the multilayer feed-forward non-linear perceptron*, has many desirable properties which are described below.

1.8.2 The Multilayer Feed-Forward Non-linear Perceptron

The processing unit corresponding to the "idealized" neuron is depicted in Fig. 1.7 where the inputs are represented by x_j, and the output by y_i. The output of the unit is computed by the function:

$$y_i = f((\sum_{j=1}^{n} x_j w_{i,j}) + b_i).$$

The values of $w_{i,j}$ correspond to the strengths of the inhibitory/excitatory connections of the units synapses. b_i is a *bias* term which has the effect of thresholding the input and acts as an implicit connection to an input of unity[4]. A common choice for the function $f(\cdot)$ is given by

$$f(x) = \frac{1}{1+e^{-x}}$$

which gives us a non-linear output response to the input[5]. This non-linear property is important since it allows us to obtain additional computational power by adding additional layers of units. If the output response were linear then every multi-layer network could be realized by a corresponding single layer network[6].

[4] This is similar to the linear-perceptron of [Ros62] with the addition of a nonlinearity.

[5] This function is often called the *sigmoid* or *squashing* function.

[6] As noted in [Dru84] there are other reasons why a motion detection model *must* be non-linear. This is further discussed in §2.3.

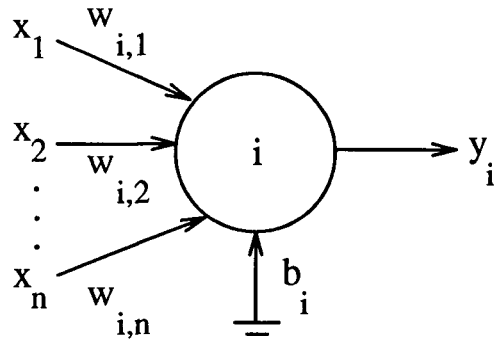


Figure 1.7: The idealized neuron

By organizing these processing units into layers as in Fig. 1.8 we obtain the multi-layer feed-forward perceptron which is an extension of Rosenblatt's perceptron model. This network is composed of three parts: the input layer, one or more hidden layers, and a single output layer. The units of one layer are connected only to units of the next layer, and no connections are permitted within the same layer, or to previous layers. This ensures that the network is acyclic and allows for easy evaluation and learning. To evaluate the network, the inputs are presented to the input layer, and the activation cascades forward until the entire output vector is computed.

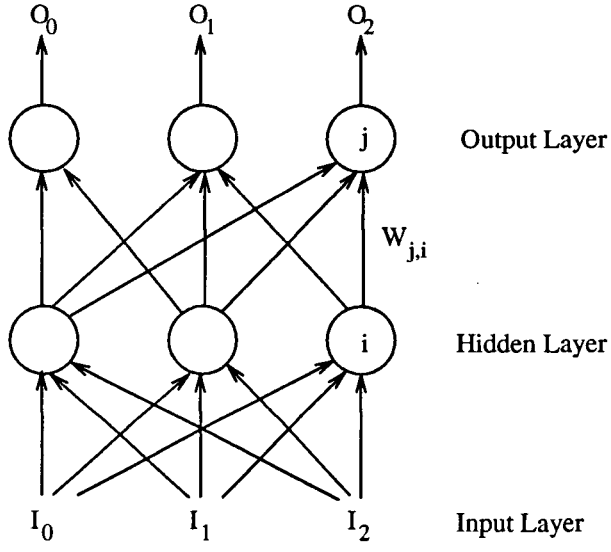


Figure 1.8: The multilayer feed-forward perceptron network

1.9 Learning Procedures

The goal of any connectionist learning procedure is to obtain values for the weight parameters, $w_{i,j}$, which allow for an acceptable set of responses from the inputs. The search of the weight space for appropriate values of the weight parameters is made based on a collection of examples called the *training set.* Connectionist learning procedures fall into one of two possible categories: supervised and unsupervised learning.

Supervised learning procedures act as an *active* "teacher" for the network. The training-set is composed of a set of input-output pairs corresponding to the desired outputs of the network. Given an initial set of weight values (usually random), the learning procedure systematically adjusts each weight in the network so as to obtain a better solution according to some function which characterizes the performance of the network over all training examples (such as the total squared error over all training sets). This type of learning procedure has been the most successful so far, but does require that the ideal outputs are known before training can take place. An example of this type of learning procedure is called *back-propagation* which uses gradient-descent to reduce the total squared error of the training examples [RH86]. This procedure (and variants thereof) is described in detail later in this section. This learning procedure is ideal for our application since the ideal outputs of the network (the true motion field) are known.

Unsupervised learning procedures learn their representations based purely on the input values and therefore the training set does not contain a set of desired outputs. In this sense the learning procedure is not an active teacher. The representation of the network is constructed purely on information of the input distribution. By exploiting regularities in the input distribution the network is able to classify the input vectors according to the most salient features of the input. This type of learning is appropriate when one does not have exact knowledge of the "correct" output of the network, or in cases where one wants to reduce the dimensionality of the input before further training is performed. One very powerful example of this type of learning is called competitive learning [Gro76] and has been shown to be a useful method of learning in many domains (especially pattern recognition problems). Other interesting unsupervised learning techniques include Becker and Hinton's work [BH89] using information theoretic

measures on the input distribution, and Sanger's work [San89] which is closely related to principle component analysis. An excellent introduction to unsupervised learning can be found in [RM86].

1.9.1 Back-propagation

Given a training set of desired outputs, we may define a function E which gives the total squared error over all training examples:

$$E = \frac{1}{2}\sum_{c}\sum_{j}(y_{j,c} - d_{j,c})^2$$

where j denotes an individual output of a single training example c, with $y_{j,c}$ and $d_{j,c}$ representing the computed and desired outputs, respectively [RH86].

To minimize this error measure, we may differentiate E and use gradient descent. The process of using Eqn. (1.4) to perform gradient-descent is called *back-propagation.* The expression which gives the appropriate gradient is:

$$\Delta w_{i,j} = -\epsilon\frac{\partial E}{\partial w_{i,j}}$$

where ϵ is a small scalar called the *learning rate* which scales the magnitude of the weight updates. By using the chain rule we may propagate these error derivatives from the output layer to each successive hidden layer to give the appropriate weight updates for each weight in the network.

In cases where the current solution is located in a tough, we may get oscillatory behaviour as the solution "sloshes" back-and-forth with little progress parallel to the trough. This type of behaviour can be reduced by introducing a *momentum term*, M, as suggested in [PH87] which dampens the oscillations:

$$\Delta w_{i,j}(t) = -\epsilon(\frac{\partial E}{\partial w_{i,j}} + M\ \Delta w_{i,j}(t-1)). \tag{1.4}$$

1.9.2 Convergence of Back-propagation

There are, however, two potential pitfalls of this learning algorithm. Unlike the perceptron learning procedure [Ros62], there is no guarantee of convergence to the optimal solution since

the error surface may be concave resulting in local-minima which are inescapable using gradient-based techniques. In practice such local minima rarely cause problems as the weight space is usually of sufficiently high dimensionality to allow escape in one of the dimensions. The second problem is that of computational complexity. Since the modification of a single weight in the network may change the entire error-surface, making it necessary to adjust all other weights in the network. This results in a potentially $\mathcal{O}(2^{|w|})$ learning process[7] where $|w|$ is the total number of weights in the network. In practice the worst-case $\mathcal{O}(2^{|w|})$ time complexity is rarely encountered. Plaut and Hinton have empirically characterized the typical time complexity of the back-propagation algorithm as $\mathcal{O}(|w|^3)$ [PH87], but this result should be taken with some caution since the time complexity is very dependent on the nature of the problem, and the representation of the inputs/outputs of the network.

1.10 Constructing the Network

Now that the details of the learning procedure have been outlined, there remain several other important issues that affect the network's ability to obtain a satisfactory solution. Some of these factors are described below:

Topology of the network The most important factor which determines the ability of a network to compute a particular function is the topology of the network. This includes the number of hidden units in the network, the placement of connections between the units, and the number of layers used. If insufficient numbers of hidden units are used the approximation to the ideal function will be poor. Adding more hidden units may increase the accuracy of the network but only if there are sufficient connections to the appropriate units. Care must be taken when choosing the number of hidden units as there is a point at which adding more hidden units results in a decrease in the network's ability to generalize. Therefore there is a tradeoff in the design of the network: too few hidden units results in a poor approximation of the ideal function, too many result in a solution which "memorizes" the entire training set rather than obtaining a useful generalization of the

[7] Judd [Jud87] has shown that the *general* learning problem is NP-complete, but he goes on to say that a polynomial learning time may be achievable by placing restrictions on the network or on the problems.

input data. Additionally, the computational complexity of the network increases rapidly as the number of hidden units and connections increases.

The representation of the input/output By choosing a connectionist model for our computation we have not avoided the problems of data representation which plague traditional sequential algorithms. Although the network will internally contain a distributed representation of the function, we still have a great number of options for the representation of the inputs and outputs of our network. Some of these schemes, while they may appear equivalent, may have a great effect on our ability to extract the relevent features of the input. If we choose to represent our ideal outputs in the training set in such a way that there is little apparent correlation to the input data, the learning procedure may take an unacceptably long time to terminate, or even worse, may fail altogether. For example, it is not obvious what representation would be best for texture segmentation. Do we use the raw image intensities as inputs, or do we use the fourier transform of the image or some other input? The answer to the question of how to represent the data requires that the designer be very aware of both the problem at hand, as well as the computational limitations of the network he/she is using.

The construction of the training set An important issue concerning the construction of the training set is the choice for the size of the training set. This is a difficult concept to quantify since it is dependent on both the inherent complexity of the function (which is often not known), as well as the amount of information each training sample gives about the input distribution. A theoretical upper-bound on the number of examples needed for classification problems was given by [Hau90] as $N \geq \frac{|w|}{\epsilon}$, where ϵ is the desired upper-bound on the number of errors due to mis-classification. While this seems like a pessimistic upper-bound, most problems will require substantially fewer examples due to regularities of the input distribution.

1.11 Thesis Layout

Chapter 2 describes previous work in the areas of motion detection and the detection of discontinuities in images, and is included to give a historical perspective of the problem, as well as giving a framework for which the results of this thesis are to be compared. The important and very influential work of Reichardt [Rei61] is first presented in §2.1 and provides much of the motivation for the correlation-based approaches which will be investigated. The Horn-Schunck algorithm given in [HS81] is described in §2.2 and gives a good introduction to gradient-based techniques for the computation of optical flow, as well as giving a good introduction to regularization theory. The work of Drumheller's synthesized one-dimensional motion detector [Dru84] is presented in §2.3. This work is similar to that of this thesis in that the solution is "learned" from a set of examples, but differs in the computational model and learning procedure employed. Lastly, the work of Spoerri and Ulmann [SU87] is given in §2.4. This work provides an alternative approach to the use of sub-neighborhoods to identify motion discontinuities in an image. This work also gives excellent background in the problem of discontinuity detection and discusses the relative merits of several approaches.

Chapter 3 outlines the difficulties encountered at discontinuities when using a correlation-based scheme, and illustrates a set of measures which may be used to extract useful discontinuity information. The effects of texture in an image on the performance of correlation-based techniques is also demonstrated using synthetic images.

Chapter 4 gives a description of the integration algorithm including a description of the structure of the neural network, the representation of the inputs and outputs of the network, and the generation of the training set. The training of a neural network for computing binocular stereo is presented in Chapter 5. The performance of the neural network is then analyzed, using both synthetic and natural images, and compared to the traditional winner-take-all algorithm.

Chapters 6 and 7 give some concluding remarks on the analysis of the neural network, and suggest possible improvements and future directions of this research.

Chapter 2

Previous Work

In this chapter we will review some of the models proposed for motion detection and the detection of discontinuities in an image. This will provide a historical perspective for the problem and will provide the tools necessary to understand and interpret the models presented in this thesis.

2.1 Werner Reichardt's Minimal Model for Motion Detection

One of the first and most influential models proposed for the detection of motion is described in [Rei61]. This model was used to characterize the optomotor responses of the *Chloropahnus viridis* beetle, and was later used to describe flies.

This symmetric model (pictured in Fig. 2.9) operates on the principle that each of the two units (the left and right sides of the figure) computes (approximately) the auto-correlation of the brightness function, $\xi(x)$, from their receptors[1]. The auto-correlation of the brightness functions, $\Phi_{\xi\xi}(\tau)$[2], is dependent on τ, which is chosen depending on the velocity of the pattern, the distance Δs between the receptors, and the delay of the filters H_A and H_B. In cases where there is a significant peak in $\Phi_{\xi\xi}(\tau)$ there will be a correspondingly large peak in the response R. Thus R is said to be tuned for the velocity v. The sign of R will be the same as the velocity, v, but the magnitude of R may be disturbed by changes in contrast of the input pattern.

[1] In the model, D, F, and H are linear filters, and M_A and M_B are multipliers. S is a low-pass filter which is used to approximate an infinite time averager. All functions L represent timing functions and can be ignored. In addition D is used to *differentiate* the brightness values and is included here for biological accuracy since most photo-receptors are sensitive to *changes* in illumination. This distinction is unimportant for our purposes.

[2] For a stationary signal $\xi(x)$, $\Phi_{\xi\xi}(\tau) = E[\xi(x) - \xi(x-\tau)]$ where $E[\,]$ is the expectation operator.

For example, it may be possible that a low-contrast pattern which is moving at the "right" velocity (*i.e.*, one for which $\tau = 0$) results in a response which is smaller in magnitude than a high-contrast pattern moving at the "wrong" velocity (*i.e.*, a velocity for which $\tau \neq 0$). This problem, as noted in [HK86], is common to many biological systems. The actual determination of the correct velocity, v, is performed at a higher-level by comparing the responses from a set of motion receptors R, all tuned for different velocities.

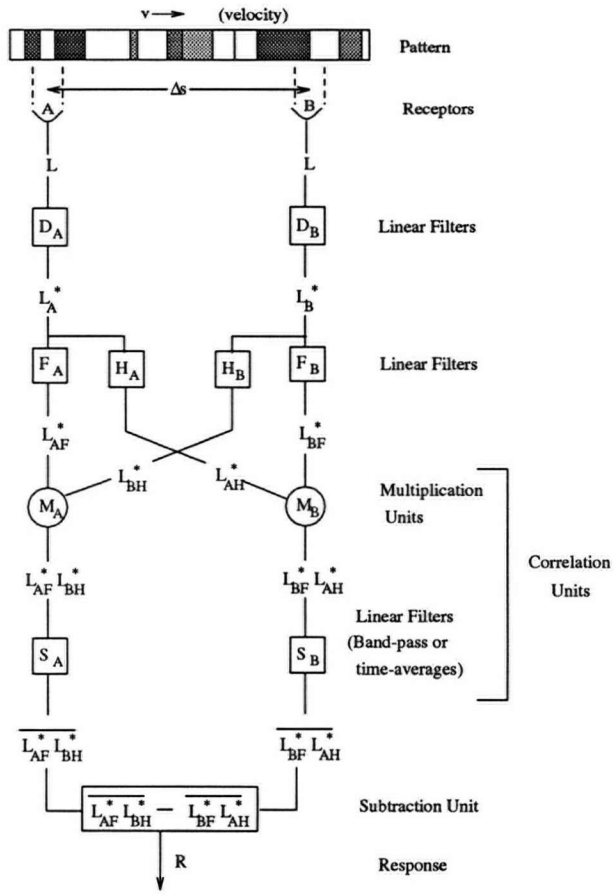


Figure 2.9: Reichardt's "minimal model" used to describe the optomotor response of the *Chlorophanus viridis* beetle. Redrawn from [Rei61] with minor notational modifications.

An important feature of this work is that it goes further than describing the behaviour of a single cell, but instead is able to characterize the optomotor response for the *entire* insect. Reichardt and his colleagues, through elaborate apparatuses, measured the tendency of the beetle to turn when presented with a moving stimulus. In these experiments the beetle was

suspended but free to rotate by flapping its wings. When presented with a moving pattern, the beetle tends to turn so that it follows the pattern and thus reduces the displacement between itself and its surroundings.

There are a number of close similarities between Reichardt's formulation for the detection of motion and those found in the shift-and-compare operation. If we assume that the individual brightness patterns do not change over a relatively short period of time (*i.e.*, the brightness constancy constraint), then the cross-correlation values will closely approximate the auto-correlation of the brightness function $\xi(x)$, but rather than using a time averaging (filters S_A and S_B), there is a spatial averaging over the support region. Unlike Reichardt's model which is able to recover the *sign* of the velocity from the response R (the magnitude is potentially contrast-dependent), the shift-and-compare obtains the sign of the velocity by using two copies of the comparison function: one for the positive velocity, and the other for the negative.

van Santen and Sperling [vSS84] have proposed a modified version of Reichardt's motion detector, called the *elaborated Reichardt motion detector*. The main difference in this model is the addition of spatial filters to prevent spatial aliasing. Since Reichardt's original model has photoreceptors which are point shaped, their spatial impulse responses are simple delta-functions and hence pass all spatial frequencies. van Santen and Sperling also discuss an arrangement of symmetric and anti-symmetric filters which prevent spatial aliasing.

2.2 Horn-Schunck

A contrast to the correlation-based approaches of Reichardt and Drumheller is the optical flow algorithm proposed in [HS81]. This formulation, which has received much attention from researchers in optical flow, assumes that the optical flow field is a differentiable vector field, and obtains a solution by explicitly extracting approximations of the image gradients.

If we use the following abbreviations for the various derivatives of the image brightness and the motion field:

$$u = \frac{dx}{dt}, \qquad v = \frac{dy}{dt}, \qquad E_x = \frac{\partial E}{\partial x}, \qquad E_y = \frac{\partial E}{\partial y}, \qquad \text{and} \qquad E_t = \frac{\partial E}{\partial t}$$

then, using the chain rule, the brightness constancy constraint, $\frac{dE}{dt} = 0$, becomes:

$$E_x u + E_y v + E_t = 0. \tag{2.5}$$

This is called the *optical flow constraint equation* [FT79]. Since we have one equation and two unknowns, (u and v), we must introduce further constraints to obtain a unique solution. If we assume that the flow field is locally smooth then we may use a regularization process of the Tikhonov type [PTK87].

This results in two terms: e_s the departure from smoothness, and e_c the error in the optical flow constraint equation. These are given by:

$$e_s = \int\int ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))\, dx\, dy, \tag{2.6}$$

$$e_c = \int\int (E_x u + E_y v + E_t)^2\, dx\, dy. \tag{2.7}$$

We now seek values for u and v which minimizes the regularization expression

$$R = e_c + \lambda e_s \tag{2.8}$$

where λ is the regularization parameter which weighs the relative merits of smoothness and departure from the optical flow constraint equation.

In the discrete case we may approximate e_s and e_c at a point (i, j) in the image by $s_{i,j}$ and $c_{i,j}$, respectively:

$$\begin{aligned} s_{i,j} &= \frac{1}{4}((u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2) \\ &\quad + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2), \\ c_{i,j} &= (E_x u_{i,j} + E_y v_{i,j} + E_t)^2, \end{aligned}$$

where E_x, E_y, and E_t are estimates of the derivatives at the point (i, j).

The discrete version of the regularization process now becomes[3]:

$$\hat{R} = \sum_i \sum_j (s_{i,j} + \lambda c_{i,j}),$$

[3]This formulation is from [Hor86] and differs from that of §1.3 in that the λ term multiplies the error term rather than the smoothness term. These two formulations are equivalent with proper transformations of λ.

so we seek values $u_{i,j}$ and $v_{i,j}$ which minimizes $\hat{R}$. By differentiating $\hat{R}$ with respect to $u_{i,j}$ and $v_{i,j}$ and solving for the extremum, we have:

$$\begin{aligned}(1+\lambda(E_x^2+E_y^2))u_{i,j} &= +(1+\lambda E_y^2)\ \overline{u}_{i,j} - \lambda E_x E_y \overline{v}_{i,j} - \ \lambda E_x E_t, \\ (1+\lambda(E_x^2+E_y^2))u_{i,j} &= -\lambda E_y E_x\ \overline{u}_{i,j} + (1+\lambda E_x^2)\overline{v}_{i,j} - \ \lambda E_y E_t\end{aligned}$$

where $\overline{(\cdot)}$ represents a local average. This suggests an iterative scheme as in:

$$\begin{aligned}u_{i,j}^{n+1} &= \overline{u}_{i,j}^{n} - \lambda \frac{E_x \overline{u}_{i,j}^{n}\ + E_y \overline{v}_{i,j} + E_t}{1+\lambda(E_x^2+E_y^2)} E_x, \\ v_{i,j}^{n+1} &= \overline{v}_{i,j}^{n} - \lambda \frac{E_x \overline{u}_{i,j}^{n}\ + E_y \overline{v}_{i,j} + E_t}{1+\lambda(E_x^2+E_y^2)} E_y.\end{aligned}$$

To obtain estimates of the spatial and temporal derivatives we may consider a three-dimensional cube around the point (i,j) and use a set of difference equations. If we let i, j, and k be indices for x, y, and t, respectively, then:

$$\begin{aligned}E_x &\simeq \frac{1}{4\delta x}(E_{i+1,j,k} + E_{i+1,j,k+1} + E_{i+1,j+1,k}\ + E_{i+1,j+1,k+1}) \\ &\quad -\frac{1}{4\delta x}(E_{i,j,k} + E_{i,j,k+1} + E_{i,j+1,k}\ + E_{i,j+1,k+1}) \\ E_y &\simeq \frac{1}{4\delta y}(E_{i,j+1,k} + E_{i+1,j+1,k} + E_{i,j+1,k+1}\ E_{i+1,j+1,k+1} \\ &\quad -\frac{1}{4\delta y}(E_{i,j,k} + E_{i+1,j,k} + E_{i,j,k+1}\ E_{i+1,j,k+1}) \\ E_t &\simeq \frac{1}{4\delta t}(E_{i,j,k+1} + E_{i+1,j,k+1} + E_{i,j+1,k+1}\ + E_{i+1,j+1,k+1}) \\ &\quad -\frac{1}{4\delta t}(E_{i,j,k} + E_{i+1,j,k} + E_{i,j+1,k} + \ E_{i+1,j+1,k})\end{aligned}$$

An interesting property of this algorithm is the propagation of information from a single point in the image to its neighbours. For the initial few iterations the motion field reflects a local minimum of $\hat{R}$ based on information from a small neighbourhood around each point. After several iterations the size of the neighborhood increases radially until the solution reflects a more global minimum.

Since the Horn-Schunck algorithm requires accurate gradient information in order to obtain a good approximation of the true motion field, the performance of the algorithm will suffer greatly at points where there is a motion discontinuity. At these points, neighboring points

will be uncorrelated and hence Eq. (2.5) is violated. Since the estimations of u_x, u_y, v_x, and v_y require finite extent these values will become meaningless (often becoming extremely large). This causes the solution of the minimization to have unrealistically smooth flow fields which poorly satisfy the optical flow constraint equation. To avoid the problems at the boundary of objects it is necessary to mark points in the image where the gradients become large. Once these points are marked we can ensure that the regularization process does not smooth over object boundaries.

2.3 Drumheller's Learned Motion Receptor

Of particular interest to the work of this thesis is the work of [Dru84] in which a one-dimensional motion detector is "learned" from a series of examples. In this model the algorithm can be thought of as a "black-box", where the inputs are a series of buffers containing small regions of a sampled translating signal, and a series of outputs which are to be tuned to a set of velocities.

If we denote the number of buffers by P, and the length of each buffer by S, then we may define our output by the function $Ax = B$, where A represents a matrix whose rows consist of a series of examples (each row is PS columns), x is the set of stimuli, and B is a set of desired outputs.

The simplest form of A would be a linear function. This would allow us to simply place a large number of example rows in our linear system, and to solve for A by finding the *Moore-Penrose Pseudoinverse* of A [Alb72], which would be optimal (in a least-squared sense). As illustrated in [Dru84] the simple-minded linear approach is doomed to fail. For any linear system H it can be shown that

$$\overline{H(x_1, \ldots, x_n)} = H(\overline{x_1}, \ldots, \overline{x_n})$$

where $\overline{(\cdot)}$ represents the infinite time-average. If we now let $p_1, \ldots p_n$ represent the inputs from the photoreceptors for a particular velocity, and $n_1, \ldots n_n$ the corresponding inputs for the same pattern moving in the opposite direction, then since $\overline{p_1}, \ldots, \overline{p_n} = \overline{n_1}, \ldots, \overline{n_n}$, the time average of the output to the two patterns will be the same for both directions. In addition, if H is

linear, then negating the values of x_i will produce an output which is in the opposite direction of the original pattern.

To remedy these problems, Drumheller chose A to instead be a non-linear polynomial function[4], so that the output now corresponds to $Ax = V$ where,

$$\begin{aligned} v_i &= x_0 \\ &\quad + \sum_{\alpha_1=1}^{n} a_{i\alpha_1} x_{\alpha_1} \\ &\quad + \sum_{\alpha_1=1}^{n} \sum_{\alpha_2=1}^{n} a_{i\alpha_1}\ a_{i\alpha_2} x_{\alpha_1\alpha_1} \\ &\quad + \cdots \\ &\quad + \sum_{\alpha_1=1}^{n} \cdots \sum_{\alpha_k=1}^{n} a_{i\alpha_1\cdots\alpha_k} x_{\alpha_1\cdots\alpha_k} \end{aligned} \tag{2.9}$$

with $n = PS$. Here a_{ij} denotes the j-th element of the i-th row of A, and v_i is the i-th output. The expression $x_{\alpha_1\cdots\alpha_m}$ denotes an m-way matrix of coefficients which multiplies the products of elements of rows of A; x_0 is a constant offset factor. From this form it is easy to identify the linear component (the first summation), and the non-linear component (the remaining summations). This expression is closely related to the *Volterra series expansion* for continuous non-linear systems:

$$\begin{aligned} y(t) &= f_0 \\ &\quad + \int_{-\infty}^{+\infty} f_1(\beta) x(t-\beta)\ d\beta \\ &\quad + \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f_2(\beta_1,\beta_2) x(t-\beta_1)\ x(t-\beta_2)\ d\beta_1\ d\beta_2 \\ &\quad + \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} f_k(\beta_1,\cdots,\beta_k)\ x(t-\beta_1)\cdots x(t-\beta_k)\ d\beta_1\ \cdots\ d\beta_k \\ &\quad + \cdots \end{aligned} \tag{2.10}$$

The Volterra series expansion, as well as the related *Weiner series expansion* are standard tools for the identification of non-linear systems. Poggio pointed out the similarity of *system identification* and *learning* in [Pog75], and it is this similarity which Drumheller uses to formulate his learning process.

[4]Hence A must be thought of as an operator.

A common method for measuring the impulse response of a linear system is to excite the system with a Gaussian white noise signal, $x(t)$, and then cross-correlate the input with the response, $y(t)$, The impulse response, $h_1(\sigma)$ is proportional to this cross-correlation[5],

$$h_1(\sigma) = \frac{1}{C}\overline{y(t)x(t-\tau)}. \tag{2.11}$$

From this result, the k-th order Volterra kernel, $f_k(\sigma_1,\ldots,\sigma_k)$, can now be determined from the same cross correlation of the output $y(t)$ of the system with a k-th order delay of the input:

$$f_k(\sigma_1,\ldots,\sigma_k) = \frac{1}{2A^2}\overline{y(t)x(t-\sigma_1)\cdots x(x-\sigma_k)}. \tag{2.12}$$

The learning is now phrased as finding the values of $x_{\alpha_1,\ldots,\alpha_k}$ which minimizes the least squared error of both sides of Eq. (2.12).

The examples were constructed using random images (intensities in the range of [0,1]) and consisted of the superposition of 10 random sinusoids. Once the random images were created, the images were sampled into the set of of buffers corresponding to the $n = PS$ columns of A. These buffers contained more or less the same signal, but were shifted with respect to each other. To ensure that the resulting coefficients could be interpreted, the shift was limited to an integer number of pixels between the buffers.

The result of this work showed that the optimal solution was indeed obtainable from the set of random images. In addition the interpretation of these coefficients was easily possible if presented in a graphical form. As a result of interpretation of the coefficients it was discovered that the solution obtained implemented directly a correlation algorithm. Drumheller does not address the problem of estimating motion from images which contain discontinuities and hence is susceptible to the same kinds of systematic errors found in all correlation-based approaches with fixed support.

[5]The autocorrelation of $x(t)$ is $\Phi_{xx}(\tau) = Cu_0(t)$, where $u_0(t)$ is the unit response function.

2.4 The Detection of Discontinuities

2.4.1 Introduction

As was discussed in Chapter 1, segmentation of the image into regions which likely correspond to different objects is a necessary pre-condition for the accurate determination of most vision modules including motion, stereopsis, shape from shading, and others. Without prior knowledge of the object boundaries these algorithms will often impose a smoothness constraint across the object boundary causing a degradation in the computed property. In the absence of cues from intensity edges, the depth/motion field may be our only cue for the determination of the object boundaries.

Unfortunately, we are faced with a cyclic dependency: the computation of the motion discontinuities is difficult without an accurate flow field, and *vise versa.* As a result, there are two basic styles of discontinuity detection: those that detect the discontinuities *before* the computation of the full flow field, and those that do so *afterward.*

Algorithms which compute the discontinuities after the computation of the flow field include: image growing techniques which attempt to group regions according to similar velocity [Pot75], center-surround operators which identify regions where there is a significant velocity difference between the center and surround [NL74], iterative techniques such as interleaving edge-detection with the smoothing process [Sch84a], and others.

An excellent example of an algorithm which computes the discontinuities prior to the computation of the entire flow field is the work of Spoerri and Ullman and is discussed in [SU87]. By computing the discontinuities before the flow field is computed we may achieve a decoupling of the computation of visual modules. In addition [SU87] uses only local computations and allowed for an implementation on a fine grained SIMD[6] multi-processor architecture. This algorithm will now be discussed in detail.

[6] Single instruction multiple data.

2.4.2 Spoerri and Ullman

The input to the algorithm consists of a set of motion primitives which lie in a circular neighborhood of a point in the image. The radius of this neighborhood is typically 5 to 8 pixels. Possible primitives which may be used include: image intensities, zero-crossings, and other edge features. Intensity values have the benefit that they can be computed at each pixel, whereas zero-crossings may be sparse or non-uniform across the image. The disadvantage of using intensity values is that they are more sensitive to noise and changes in illumination, while the zero-crossings are far more stable since they often correspond to a physical phenomenon in the scene.

From these primitives a histogram is constructed which corresponds to the number of motion primitives which support a given motion[7]. From these histograms a number of measures are introduced such as the *signal-to-noise ratio, local support*, and *peak-ratio*, from which the discontinuity information is inferred.

The operation of the algorithm is based on the observation that for pixels which are near a discontinuity there will be two peaks in the velocity distribution of the histograms, one for each side of the object boundary. The following measures are used to detect the bimodality of the histograms:

Peak-ratio The ratio of the height of the largest and second largest peak in the histogram. This value will approach unity at the boundary of the object.

Signal-to-noise-ratio The signal to noise ratio is the number of votes for the largest peak to the number of votes for all other velocities. This value should be close to unity in regions of coherent motion (assuming noise-free motion primitives) and drop to 0.5 or lower at points near a discontinuity.

Local-support This is the ratio of votes for the largest peak to the total number of votes. This is close to unity near coherent motion and becomes smaller as the distance to the discontinuity decreases.

[7]The motion is assumed to be one of a discrete number of possible motions.

The detection algorithm attempts to measure the likelihood that the measures result from a depth-discontinuity in the image. Two statistical methods are used to measure this likelihood: a Chi-squared test, and the Kolmogorov-Smirnov test.

The Chi-squared test measures how well the local histogram can be fitted to a Gaussian distribution. The parameters of the Gaussian are obtained by assuming that it is centered, and passes through the maximum. The Kolmogorov-Smirnov test is a non-parametric measure which measures the absolute difference between the cumulative density functions of the two histograms. This method has the advantage that fewer assumptions are made on the form of the histograms that are compared.

Both of these measures have been shown to work well at determining the boundaries of random-dot stereograms and natural images. The value of these measures is generally monotonic reaching a maximum at pixels on the boundary. By marking the global maximum (non-maximal suppression) a reliable boundary can be obtained. These results are pleasing since they are determined from statistical properties of the two motion fields rather than ill-defined heuristics on the motion field (such as region growing [Pot75]).

Chapter 3

Discontinuity Information

Section 1.4 introduced the difficulties facing correlation-based algorithms near a depth-discontinuity. I will refer to this problem as the *figure-background problem* since solving for the correct displacement requires segmenting the support region into two regions: one corresponding to the motion in the foreground, and the other to the motion of the background. As we shall see, the degree to which a discontinuity in the support region may affect the ability to resolve the apparent motion is, in part, related to the texture of the image. This can be understood if one compares the motion fields of two types of images: natural band-limited images, and random-dot stereograms.

Since we are matching regions of the image at time t_0 to regions at time t_1 there is an asymmetry introduced, namely the difference in behaviour of ϕ at occlusions and dis-occlusions. *Occlusions* correspond to portions of the image which are present at t_0 but are not present at t_1, while *dis-occlusions* occur in the converse situation. In the t_0 frame of reference occlusions and dis-occlusions correspond to *coverings* and *uncoverings*, respectively. Portions of the image which become covered result in poor displacement predictions since the best match will correspond to random correlations in the image. Conversely, for portions of the image which become uncovered, the displacement predictions are only effected from the effects of averaging over an incoherent support region. It is worth noting that it is the disparities of points of dis-occlusions which we are interested in improving since there is insufficient information at occlusions to allow the recovery of displacement using ϕ.

3.1 The Figure-Background Problem

Let ξ denote the radius of a square support region used for matching points in one image to points in the other. If we constrain the displacement field so that $(u(x,y), v(x,y)) \in (-\delta, \delta)$, then we may compute the displacements of a point (x,y) by choosing a vector (u,v) which minimizes

$$\phi_\xi(E_t(x,y), E_{t+\Delta t}(x+u, y+v)), \qquad \text{where}$$

$$\phi_\xi(x,y;u,v) = \sum_{i,j \in (-\xi,\xi)} (E_t(x+i, y+j) - E_{t+\Delta t}(x+u+i, x+v+j))^2.$$

This corresponds to locating the minimum of the error surface $\phi(u,v)$ as shown in Fig. 3.10. In regions of coherent motion, this error surface will contain a single global minimum which corresponds to the true motion (see Fig. 3.10(a)). If, however, there is a discontinuity which crosses the support region, the corresponding error surface may contain more than one minimum (see Fig. 3.10(b)), one corresponding to the motion of the background, and one to the foreground.

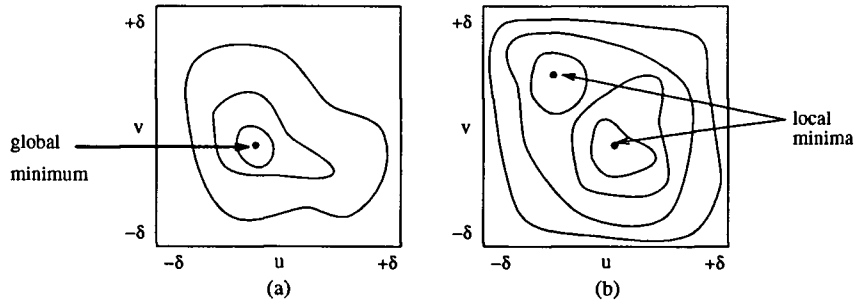


Figure 3.10: (a) Unimodal and (b) bimodal error-surface contours.

The task of choosing which of these two minima correspond to the true motion of the point in the image based purely on the error values themselves is very sensitive to the texture characteristics of the objects in the scene. If there is sufficient texture in the image, the auto-correlation of the image will contain a significant peak, even in portions of the image which are near a discontinuity. This significant peak allows the correct recovery of the true motion. This is not the case if there is insufficient texture in the image as is often the case in low-frequency "natural images". This sensitivity to texture can be demonstrated by considering two types of images which contain very different textures: "natural" band-limited images, and highly

textured random-dot stereograms.

3.1.1 Random-dot Stereograms

Consider a random-dot stereogram where the image intensities are randomly chosen to be 0 or 1 with equal probability. To simplify matters, consider the problem of determining the stereo disparity along the x-axis (it is assumed that $v(x,y) = 0,\ \forall\, x, y$).

In the case where there is coherent motion across the entire support region, then we can expect an error value of zero at the true displacement (assuming a discrete translation), and a mean error value of $M = \frac{1}{2}(2\xi+1)^2$ for all other displacements (see Fig. 3.11(a)). Additionally, as we increase the size of the support region, ξ, the error values will asymptotically approach M with decreasing variance. This distribution of error values is similar to a negated delta function and makes localization easy.

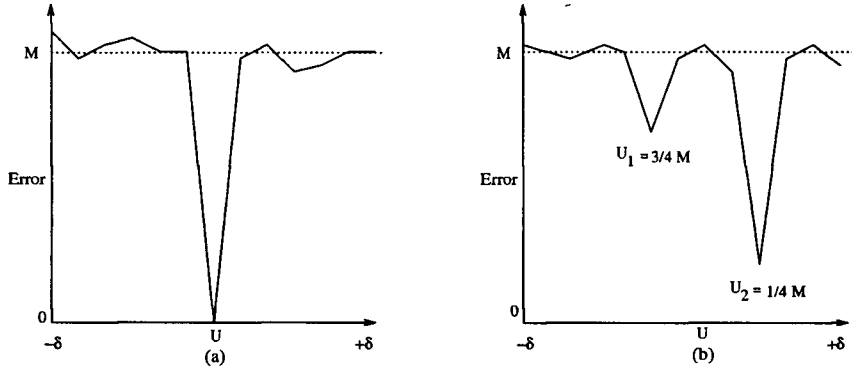


Figure 3.11: Error values for random-dot stereogram: (a) Coherent depth field, and (b) Discontinuous depth field.

If the support region contains a discontinuity, then we expect the error distribution to contain two local minima which are significantly less than M (see Fig. 3.11(b)). If we denote the fraction of the support region which votes for U_2 by α, then the expected value for the error at U_2 is

$$\frac{1}{2}\alpha(2\xi+1)^2.$$

Since the error value at U_2 is $\frac{1}{4}M$, then we can estimate the fraction of the support region

which supports the $u(x) = U_1$ hypothesis by:

$$\alpha = \frac{\phi(U_1)}{\frac{1}{2}(2\xi + 1)^2}.$$

For example, solving for α at U_2 in Fig. 3.11(b) would give $\alpha = 0.75$. This tells us that approximately 75% of the support region supports the hypothesis that $u(x, y) = U_2$. The worst-case arises when the discontinuity bisects the support region giving equal error values at the minima. It is worth noting that since the size of the support region is odd, the fraction of the support region which corresponds to the same motion as the center pixel will always be greater than 0.5. This implies that provided the boundaries of the objects are relatively smooth, a simple winner-take-all strategy of choosing the displacement corresponding to the smallest error value will statistically perform very well on highly textured images such as random-dot stereograms. Such a scheme for determining the displacements in random-dot stereograms has been shown in [LG90]. The displacement fields obtained from this scheme has been shown to work very well, even at points near object boundaries.

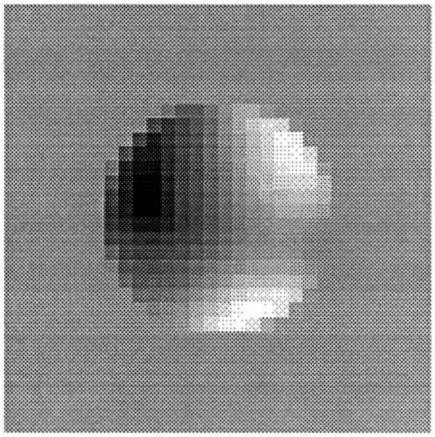
This good performance is partially due to the large amount of information available from the random-dot images. Since a region in the left image is perfectly correlated with its corresponding region in the right image, and are statistically very unlikely to match an incorrect region, there is a very large signal for the correct velocity. In fact the autocorrelation of a random-dot stereogram will be a delta-function, and is therefore easy to identify. In fact this signal will usually be sufficiently strong, even in regions near discontinuities, to allow good disparity measurements.

The fact that random-dot stereograms are statistically unlikely to give rise to ambiguous motion fields makes them an unreasonable model for for testing real vision algorithms. In addition, natural scenes contain a great deal of low frequency components. This means that neighboring pixels will not be independent as they are in random-dot stereograms. For these reasons it is necessary to explore the behaviour of correlation-based schemes with images which are more representative of those occurring naturally.

3.1.2 Natural Band-limited Images

To examine the behaviour of our correlation algorithm near discontinuities, it was necessary to construct synthetic images which were more representative of the types of images found naturally. While it would have been possible to use natural images, synthetic images allowed for easy modification of the texture parameters.

The texture found in the synthetic images was constructed by taking the superposition of a number of randomly chosen sinusoids[1]. The wavelength of each sinusoid was chosen randomly in a specified interval. The angle of propagation, as well as the phase shift of each sinusoid was chosen randomly from the interval $[0, 2\pi)$. To reduce the effects of the higher frequency sinusoids in the image, the amplitude of each sinusoid was proportional to its wavelength. By changing either the number or the wavelengths of these sinusoids, one is able to control the amount of texture information in the images. An example of this type of image is found in Fig. 3.1.2. To simulate the motion of an object, regions of the textured image were "cut out" and repositioned according to the desired displacement.



This band-limited image was generated by superposition of 20 sinusoids; the wavelengths randomly chosen from $[10, 30]$ pixels.

Figure 3.12: Randomly textured disc on a uniform background.

Figure 3.13 shows the flow field obtained from a band-limited image. The texture for the center disc and the background were constructed from two randomly generated sets of 20 sinusoids with wavelengths from 10 to 30 pixels. The flow was computed using a (13×13) support region. The true-displacement of the disc is two pixels to the right in a stationary textured background. From this needle-diagram of the flow field the over-generalization of the

[1] This technique was originally introduced in [Dru84].

object boundary to pixels in the nearby background is quite apparent. The size of this *swelling* of the object boundaries is proportional to the size of the support region. This phenomenon can be understood if one looks at the behaviour of ϕ across the boundary of the object.

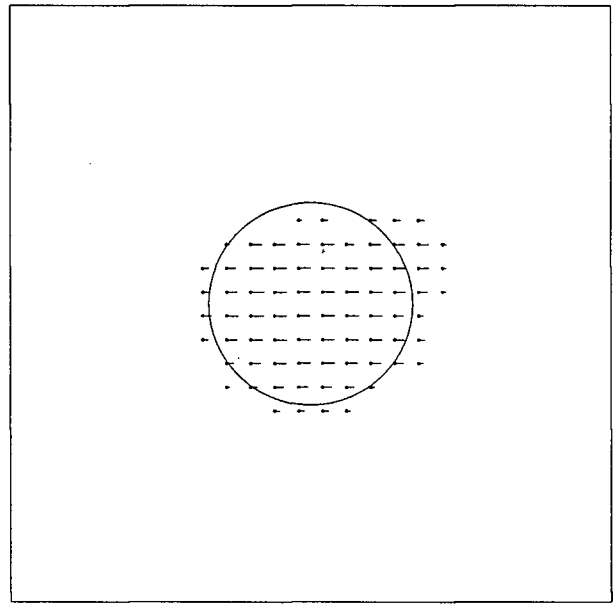


Figure 3.13: Needle diagram for band-limited (100×100) image. Disc motion is $(u, v) = (+2, 0)$. Flow markers are placed every 4 pixels with $\xi = 13$ and displayed in the t_0 frame of reference.

In random-dot stereograms, each object has the same image characteristics: a mean brightness value of 0.5 with each pixel uncorrelated to all others. Natural images, on the other hand, generally contain many low-frequency components which give rise to regions of similar brightness. The assumption of our model is that by ignoring the DC component of the brightness in a small region, we can use the local variation in brightness to correctly match the region. This assumption may have difficulties near an object boundary since the support region may contain two regions of different brightness values separated by an intensity step-edge. This situation is depicted in Fig. 3.14 in which a dark circular object is moving in a light, static background. Displacements which attempt to match pixels of the light background to dark object pixels will result in a large error value, so the displacement corresponding to the smallest error for the

point P_0 will attempt to match the contours of the step-edges in the two images[2]. This seems reasonable since the step-edge is the most salient feature to match across the two images, and is likely to correspond to a meaningful feature in the image. This behaviour of "tracking" the intensity edge is similar in spirit to the optical flow algorithm proposed by Hildreth [Hil84].

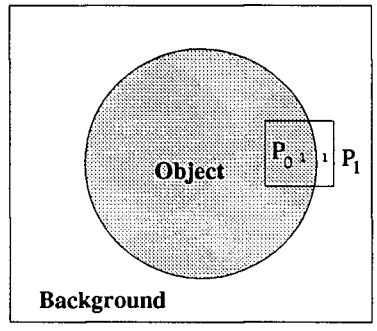


Figure 3.14: Behaviour of ϕ across an intensity-edge of an image.

To understand why the swelling phenomenon takes place, consider the point P_1 in the background, close to the boundary of the object. Since this point is close to P_0, the support regions will be very similar to each other (especially if the region size is large compared to the distance separating the points), the same edge-tracking behaviour will occur. Since the motion of the intensity-edge has no relationship to the displacement of the background, the resulting motion field will reflect the motion of the edge. This effect can be seen in the right-hand portion of the background in Fig. 3.13. While it might seem reasonable to simply locate the step-edges and avoid the matching of features on different sides of the edge, this process would be unable to differentiate between edges arising from a depth discontinuity and those arising from the texture of the object.

In cases where the brightness of the foreground and background are similar, the effect of this edge tracking is less pronounced since the error values better reflect the brightness information of the background. Even in these instances the flow field may be degraded by smoothing due

[2]This is similar to the behaviour exhibited in Reichardt's motion detection model [Rei61] in which a low-contrast pattern moving in the "prefered" velocity may produce a significantly smaller response than a high-contrast pattern moving in the opposite direction (see §2.1 for details). Unlike Reichardt's model which is able to recover the correct sign of the velocity, the cross-correlation matching functions are not able to recover the sign of the velocity. Instead we replace the single motion receptor for a given velocity with two correlation values: one the positive direction, and another for the negative.

to the fact that we are averaging over a large region. This can be seen in the top portion of the disc in Fig. 3.13 where the predicted flow lies between the ideal motion of the foreground and the ideal motion of the background.

A common technique used to improve the performance of correlation-based motion algorithms is to convolve the images with a band-pass filter such as a difference-of-Gaussian filter (DOG). This has the effect removing the low frequency components of the image and emphasizing the high-frequency components such as the step-edges. This emphasizing of the step-edge may help the proper tracking of points of the object near the boundary, but may worsen the displacement estimates for points in the background near the discontinuity. In addition, there is still a sensitivity to the nature of the texture of the images even after the DOG is applied.

To avoid these effects on the comparison operator, ϕ, near a discontinuity, the size and shape of the support region must be changed dynamically. For example, by choosing the sub-neighborhood R_w to estimate the motion of P_0 and R_e for P_1, only features corresponding to the correct side of the discontinuity are used for the matching and thus an accurate estimation of motion can be obtained. Before one can choose the correct size and shape for the support region, an accurate means of measuring the position and orientation of the discontinuity must be developed. This process is described next.

3.2 Discontinuity Information

As was mentioned in §1.4 the detection of discontinuities of the motion field is a difficult problem. This problem would become trivial provided accurate flow estimations were available, but since this is impossible to obtain without already locating the discontinuities we are faced with a dilemma. In order to overcome this interdependency of discontinuity and flow information, an iterative approach will be used. The first step of this process uses the information obtained from the set of oriented sub-neighborhood and compares their output to detect the discontinuity. This discontinuity information can then be used to guide the choice for our support region and thus obtain a more accurate estimation of the displacement.

Suppose we have a region of an image which contains a discontinuity in the displacement

field. The process of detecting the discontinuity involves estimating the first derivative of the flow field and locating positions where the magnitude becomes large. To estimate the derivatives $u_x(x,y), u_y(x,y), v_x(x,y)$, and $v_y(x,y)$, we may use four difference equations given by

$$u_x(x,y) = \lim_{\epsilon \to 0} \frac{u(x+\epsilon,y) - u(x,y)}{\epsilon} \quad , \quad u_y(x,y) = \lim_{\epsilon \to 0} \frac{u(x,y+\epsilon) - u(x,y)}{\epsilon},$$
$$v_x(x,y) = \lim_{\epsilon \to 0} \frac{v(x+\epsilon,y) - v(x,y)}{\epsilon} \quad , \quad v_y(x,y) = \lim_{\epsilon \to 0} \frac{v(x,y+\epsilon) - v(x,y)}{\epsilon}.$$

Since we are not interested in accurate estimations of the derivatives, but rather only identifying points where they become large, we may omit the denominator of these expressions. To obtain estimations of $u(x,y), v(x,y)$, and values of nearby points, we may use the values obtained from the overlapping sub-neighborhoods. More formally, our comparison function for the overlapping sub-neighborhoods is

$$\phi_N(x,y;u,v) = \sum_{i,j \in N} (E(x+i,y+j) - E(x+u+i,x+v+j))^2$$

where N is one of:

$$\begin{aligned} R &= \{(i,j) | \ -\xi \le i \le \xi, -\xi \le j \le \xi\} \\ R_n &= \{(i,j) | \ -\xi \le i \le \xi, -\xi \le j \le 0\} \\ R_s &= \{(i,j) | \ -\xi \le i \le \xi, 0 \le j \le \xi\} \\ R_w &= \{(i,j) | \ -\xi \le i \le 0, -\xi \le j \le \xi\} \\ R_e &= \{(i,j) | \ 0 \le i \le \xi, -\xi \le j \le \xi\} \end{aligned}$$

If we now denote the motion obtained by minimizing the error function ϕ_i as $u_i(x,y)$ and $v_i(x,y)$, then our estimations for the derivatives of the motion field become:

$$\hat{u}_x(x,y) = u_{R_w}(x,y) - u_{R_e}(x,y), \qquad \hat{v}_x(x,y) = v_{R_w}(x,y) - v_{R_e}(x,y)$$
$$\hat{u}_y(x,y) = u_{R_s}(x,y) - u_{R_n}(x,y), \qquad \hat{v}_y(x,y) = v_{R_s}(x,y) - v_{R_n}(x,y)$$

Additionally, define two quantities, hereafter called *shear measures*, denoted by S_{NS} and S_{EW}. These values measure magnitude of the motion difference between the north-south and east-west pairs of sub-neighborhoods, and thus approximate the magnitude of the partial derivatives in the x and y directions.

$$S_{NS}(x,y) = || \ (u_{R_n}(x,y), v_{R_n}(x,y)) - (u_{R_s}(x,y), v_{R_s}(x,y)) \ ||_2$$

$$S_{EW}(x,y) = || (u_{R_e}(x,y), v_{R_e}(x,y)) - (u_{R_w}(x,y), v_{R_w}(x,y)) ||_2$$

Provided the sub-neighborhoods are able to accurately recover the motion of their respective region, the estimations will be accurate. However, like the square support regions discussed earlier, the presence of discontinuities in the support region may cause one or more of the sub-neighborhoods to produce an incorrect displacement, resulting in poor derivative estimates. For example, if the discontinuity contour is oriented vertically so that it bisects the square support region into two roughly equal regions, then the values of $u_x(x,y)$ and $v_x(x,y)$ are likely to be accurate since R_w and R_e contain votes from a single motion source. The values of $u_y(x,y)$ and $v_y(x,y)$, however, which should ideally be zero, are likely to be inaccurate since R_s and R_n both contain equal support from both sides of the discontinuity. A similar phenomenon occurs for a discontinuity oriented horizontally.

To compute the location and orientation of the discontinuity contour using the above measures, it is necessary to impose restrictions on the types of contours we wish to detect. As will be demonstrated, if we model the discontinuity locally by a straight line, with either a vertical or a horizontal orientation, the location and orientation of the discontinuity can be computed using only the four sub-neighborhoods.

3.2.1 Detecting Discontinuities

In order to detect discontinuities, we are only interested in identifying positions where the magnitude of the partial derivative becomes large. Since the shear values S_{NS} and S_{EW} approximate the partial derivative in the x and y directions, a discontinuity will give rise a large value for one or more of these values. By choosing an appropriate threshold, θ, for the shear values, we can define a boolean function $I(x,y)$ which is true for all positions where a discontinuity is believed to occur, and false otherwise, as:

$$D(x,y) = \begin{cases} \text{true} & \text{if } \max(S_{NS}(x,y), S_{EW}(x,y)) > \theta \\ \text{false} & \text{otherwise} \end{cases} \tag{3.13}$$

To test the reliability of the above detection scheme, a synthetic image of a translating disc was generated and $D(x,y)$ computed for each point. The results of this test are depicted in

Fig. 3.15, where filled dots represent the locations where $D(x, y) =$ true. The true boundary of the disc, which was moving in a static background with $(u, v) = (3, 2)$, is marked by the circle. The size of the support region was chosen to be $\xi = 9$, and the value of the threshold, θ, was 1.0. This demonstrates that the detection scheme is able to locate the discontinuity boundary quite well within three to four pixels of the true boundary. This localization could be improved by using a smaller support region, but we may experience a decreased ability to detect the discontinuity. Our ability to localize the true contour is reduced in the right-hand portion of the disc, because this region corresponds the "leading-edge" of the disc, where, due to the translation, portions of the first image are covered in the second image. Because the features of this portion of the image are not present in the second, the resulting motion will correspond to random correspondences of the two images. This is not discouraging since the SSD surface is incapable of conveying the true displacement at these points.

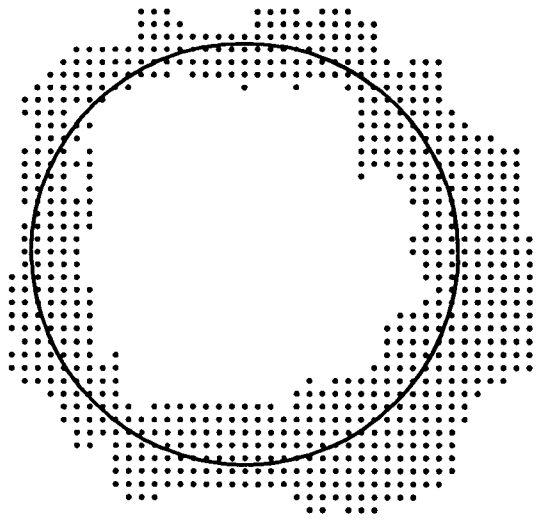


Figure 3.15: Detection of discontinuities using $D(x, y)$: Solid dots represent points in the image where a discontinuity is predicted ($\xi = 9, (u, v) = (3, 2)$ in the disc's interior, $\theta = 1$, points taken in t_0 frame of reference)

In using this scheme there is a tradeoff between detection and localization of the true discontinuity boundary and care must be taken to choose an appropriate support size. Using a

large support region increases our ability to detect the discontinuity, but is able to localize the boundary as accurately as a smaller support region.

3.2.2 Determination of the Orientation of a Discontinuity Contour

Once a position in the image has been identified by $I(x,y)$ as a discontinuity, the task remains to find its orientation. The orientation of the discontinuity is important since it provides us useful information to guide our choice for the shape of the support region. The orientation of the discontinuity can be determined by comparing the error values, $\phi()$, of the best matches for the four sub-neighborhoods, and determining which of the two pairs (north/south or east/west) best describe the motion of the scene. For example, if we are near a discontinuity which is best approximated by a vertical line, then we would expect the total error of the east/west neighborhood pair to be smaller than the total of the north/south pair. The converse is true for a horizontal discontinuity. Formally, define $O(x,y)$ which gives the best approximation of the discontinuity orientation as

$$O(x,y) = \begin{cases} \text{horiz.} & \text{if } (\phi_{R_n}(x,y) + \phi_{R_s}(x,y)) \leq (\phi_{R_w}(x,y) + \phi_{R_e}(x,y)) \\ \text{vert.} & \text{otherwise} \end{cases} \tag{3.14}$$

An example of the computation of $O(x,y)$ for positions where $I(x,y) =$ true is given in Fig. 3.16 where solid dots represent a horizontal hypothesis, and hollow represent vertical. From this example it is quite clear that this scheme for approximating the orientation of the discontinuity contour by horizontal and vertical line segments corresponds closely to the optimal fit. While there are some spurious orientation predictions in this example, $O(x,y)$ is able to segment the discontinuous regions into four parts, corresponding to the two horizontal and two vertical regions. In addition, the boundaries of these regions occur very near the point where the true orientation lies halfway between the vertical and horizontal hypothesis. By comparing the values of $O(x,y)$ in the example to the optimal values at the same locations, it was found that 491 of the total 632 points were correctly classified (approximately 78%).

In addition to using the error values of these pairs of sub-neighborhoods, clues for the determination of the correct orientation of the boundary may be found by examining the shape

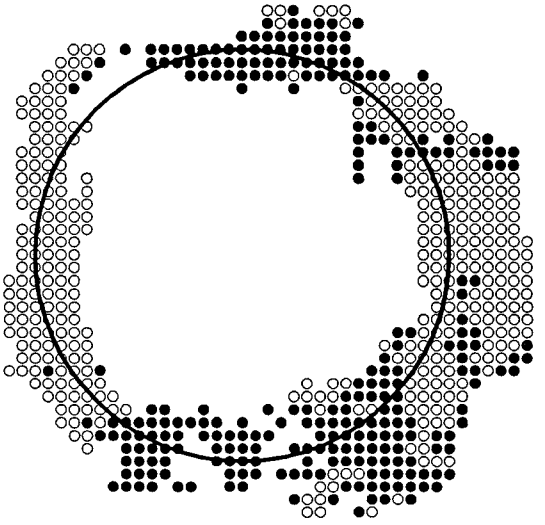


Figure 3.16: Orientation of discontinuity contour: solid dots represent a horizontal hypothesis, hollow dots represent a vertical hypothesis. True discontinuity contour is given by the circle. Positions are given in a t_0 frame of reference.

of the error surfaces. Such features of the error surface may be discovered in the learning process of the network.

Once the correct orientation of the discontinuity boundary has been established we still must determine which of the two opposing sub-neighborhoods correctly capture the underlying displacement of the object point. There are a number of cues we may exploit to obtain this goal.

Perhaps the most obvious cues are the error surfaces themselves. Since only one of the sub-neighborhoods will contain a completely coherent motion field, we can expect the minimum error value to occur in the support region corresponding to the true motion. Additionally we may also exploit the fact that a support region observing the correct motion will likely contain a more prominent valley as compared to the support regions containing incoherent motion [BA90]. Thus it is likely that a sub-neighborhood observing the correct displacement will contain a more prominent valley as compared to the other sub-neighborhoods. It is hoped that both of these cues may be discovered in the input distribution of the training set during the learning process.

Chapter 4

The Implementation of the Neural Network

The integration module of the algorithm illustrated in Fig. 1.5 consists of a single feed-forward neural network, with inputs corresponding to the cross-correlation matching functions for each of the five sub-neighborhoods, as well as the related shear values. The goal of the learning process is to obtain a set of weights which will effectively integrate these two sources of knowledge and produce reliable motion predictions as output. The structure of the network, the representations of the inputs and outputs, as well as the construction of the training sets will now be described in detail.

4.1 The Input/Output Representations

As was mentioned in §1.10 the choice of how to represent the image information may greatly influence the network's ability to discover salient features of the input distribution, and thereby learn a meaningful approximation of the objective function. There are a large number of potential representations, varying from the low-level description of the "raw" brightness values, to higher-level information such as the error-values of the cross-correlation comparison functions. Drumheller developed an algorithm which, using the brightness values of a one-dimensional image, successfully "learned" a set of filters which were sensitive to a given orientation and velocity [Dru84].

While this result is pleasing in that it uses primitives which closely correspond to that

of the human visual system, there are a number of reasons which the choice of a higher-level representation of the image information, namely the cross-correlation error values, seem appropriate:

- The cross-correlation seems to "capture" the intuitive meaning of image correspondence and also corresponds closely to features found in biological visual systems. If indeed these primitives **do** capture the essence of image correspondence, then the use of lower-level primitives such as the brightness values would require the learning process re-discover functions which are similar to the cross-correlation function.

- The sum-of-squared-differences operator has been shown to perform well in regions containing coherent motion [LG90].

- Since the cross-correlation is composed of simple arithmetic functions (addition, multiplication, and subtraction), it could easily be realized with a collection of non-linear perceptrons. By adding an additional layer these functions could be computed by our model.

- By providing these higher-level primitives the complexity of the network, and therefore the complexity of the learning process, is reduced significantly, allowing for a more manageable implementation of the algorithm.

A similar argument can be made regarding the shear values since we could require that the network extract discontinuity information directly from the brightness values. Again the multi-layer perceptron certainly possesses the computational power to compute these measures directly from the brightness values since they are simple characteristics of the error surfaces, but may require the addition of an unacceptably large number of hidden units. Additionally, if these shear values do capture significant information about the discontinuities of the image data (as seems likely from the experiments in §3.2.1 and §3.2.2) then similar representations of this information would be constructed in the learning process anyway. Again it is our hope that by adding the shear values as inputs to the network, the learning process can be simplified.

By examining the weight-space of the trained network we may gain insights as to the usefulness of the cross-correlation and shear values for the discrimination of disparity. If, for example, the weights values corresponding to the shear inputs are relatively large compared to the weights of the correlation values, then we may suspect that they provide useful information about the input distribution. On the other hand, if the magnitude of these weights are relatively small, then we may suspect that they are not useful, and the discontinuity information is extracted directly from the correlation values.

While the standard sum-of-squared-differences (SSD) could be used directly as the inputs to the network, there are a number of reasons why it is necessary to modify our correlation measures before they can be used as inputs to the network:

- Since the magnitudes of the cross-correlation values will depend on both the quantization range of the pixels, as well as the area of the support regions, it will be necessary to scale the values so that the resulting network is not tied to any individual scale or imaging process.

- Ideally we will also want to scale the values according to the shape of the support region as well since the expected SSD value for the larger square support region are larger than those of the half-regions. This will allow for easier interpretation of the resulting weight parameters after learning has taken place.

- Normalizing the input is crucial for the learning process since we must know the expected range of the inputs before we can initialize the weights prior to learning. If the range of the random initial weights is not chosen carefully we may get saturation of the hidden units early in the learning process causing the learning to stall. This is because the derivative of the sigmoid function is small for both large and small total input, causing the magnitude of the weight updates to be very small, making the learning of the hidden unit extremely slow.

With this in mind, our scaled version of the SSD values become:

$$\hat{\phi}_i(x,y;u,v) = \frac{\phi_i(x,y;u,v)}{\alpha^2|i|}, \qquad i \in \mathcal{S} \tag{4.15}$$

where the brightness values are assumed to be in the range $[0,\alpha]$ and $|i|$ is the area of the support region (in pixels) of the support region i.

Thus, since we have five support-regions to consider, and two shear values, the total number of inputs needed is given by:

$$5(2\delta+1)+2 \quad \text{for the 1-d case, and}$$
$$5(2\delta+1)^2+2 \quad \text{for the 2-d case}$$

The size of our network depends on two parameters: the size of the set of potential displacements (the value of δ) as well as the motion problem we are interested in solving (binocular stereo vs. optical flow). Unfortunately, since the learning time of the network is likely to be $\mathcal{O}(|w|^3)$ [PH87], and $|w|$ grows linearly with the number of inputs, we will have to make restrictions on the motion which is to be investigated. It was decided that it was more reasonable to restrict our networks to the one-dimensional motion problem (corresponding to binocular stereo correspondence), rather than making restrictions on δ.

Now we must consider the representation for the outputs of the neural network. The most obvious choice would be to assign an individual output unit for each of the potential displacements we wish to consider. This would allow the determination of the displacement to the nearest full pixel. However, it should be possible to obtain better accuracy if we construct the outputs in such a way that they are able to convey sub-pixel displacements. This seems like a reasonable expectation since the network will likely be able to interpolate the error surfaces to get a more accurate displacement prediction.

To accomplish this we have to change our output scheme so that real-valued displacements can be represented. This is achieved by adopting a distributed output representation which is similar to that found in [HMR86][1]. In this scheme we arrange a set of output units $\mathcal{O}_i(x)$, $i \in [-\delta,\delta]$, where the response of each unit to the true displacement x is given by the equation:

$$\mathcal{O}_i(x) = e^{\frac{|x-i|^2}{2\sigma^2}}, \qquad i \in \{-\delta,\delta\}, \tag{4.16}$$

[1]This type of representation is often referred to as "coarse-coding". The individual outputs respond over a range of disparities rather than a single disparity value.

$$\sigma = (4 \ln 4)^{-\frac{1}{2}} \simeq 0.4247$$

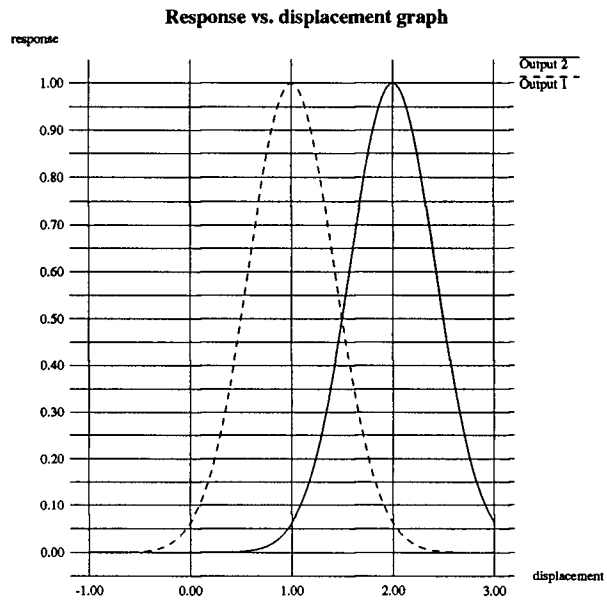


Figure 4.17: Ideal output unit response for output units O_1 and O_2 for a given displacement x.

This is a scaled Gaussian response with σ chosen such that the response of a unit 0.5 pixels away from the true output would give an output of 0.5 (see Fig. 4.17). This ensures that the largest output value of the network is unity, and the total output over all units should ideally be very close to one. Given a set of outputs corresponding to a displacement that lies between two integer quantities, we may now compute the predicted displacement to a finer level of precision than could be obtained from a "winner-take-all" strategy.

If we choose such an output representation we must be sure that we are able to accurately reconstruct the disparity from the set of output responses of the neural network. If we were optimistic about the network's ability to **exactly** reproduce the ideal output, then we could use Eqn. (4.16) and simply solve for the displacement x. In reality the output of the neural network will never correspond exactly with the optimal values, so we will have to relax our constraints on the outputs to obtain a solution which is reasonable. Suppose that we are given a set of outputs, O_i, and that the maximum value occurs at O_{max}. To obtain a solution for x we may take the value of the maximum, O_{max}, as well as the points adjacent to the maximum

(O_{max-1} and O_{max+1}), and locally approximate the ideal Gaussian response by a quadratic. From this quadratic we may solve for the point $\hat{x}$ which is the maximum which should closely correspond to the true displacement x.

The economy of such a representation is apparent; it is possible to form fine discrimination with only a small number of coarsely-tuned overlapping units. A familiar example of this type of distributed representation is found in color vision [Bon88]. The response of any one of the three broadly tuned color receptors is ambiguous, but the relative activities of all three allow the discrimination of a large number of colors.

4.2 The Network Topology

The network topology which we will adopt will be a fully-connected single hidden layer feed-forward network. Since the ideal number of hidden units is not known *a priori* this will have to be established empirically. A frequently used rule-of-thumb has been to use a number of hidden units which is proportional to the number of input units squared, but since there is reason to believe that our classification problem is fairly well behaved we will likely require fewer hidden units.

4.3 Creation of the Training Set

To obtain the training samples it was necessary to create them from synthetic images. While it would be more satisfactory to obtain these samples from sequences of natural images, the painstaking process of calibration and setting up the scene for the large number of images required would be too forbidding.

To generate these synthetic images an algorithm for generating an object with arbitrary shape and undergoing a defined motion was developed. A large number of parameters were specifiable by the user including the position, texture, and the direction and magnitude of the object's motion. The motion sequence generated from this algorithm consists of a pair of images corresponding to the time steps.

Once the pair of images were created, points were selected at random and the SSD and

shear values were computed. Since the ideal motion of the point is known the ideal output vector can be computed according to Eqn. (4.16), and these with the correlation and shear values comprise a single training pattern. This process is continued for each pattern generated until the entire training set is generated. The size of this training set is dependent on the range of displacements to be considered, as well as the total number of weight parameters in the network. This process of generating the training patterns will now be described in greater detail.

4.3.1 Generation of the Synthetic Images

To generate the synthetic images, some means of generating a random texture is needed. As was described in §3.1.2 this is implemented by superimposing a series of randomly chosen sinusoids. This is similar to the process described in [Dru84] but generalized to two dimensions. For each sinusoid a number of parameters must be supplied including:

$\lambda_i \in \Re$ The wavelength (in pixels) of the sinusoid.

$a_i \in \Re$ The amplitude.

$\gamma_i \in [0, 2\pi)$ The angle of propagation of the sinusoid taken in a counter-clockwise direction from the $x-$axis.

$\nu_i \in [0, 2\pi)$ The phase-shift of the wave relative to the origin.

Thus, for a given position (x, y) in the image, the amplitude of a given sinusoid i is

$$\psi_i(x, y) = a_i \sin(\frac{2\pi}{\lambda_i}(x \cos\gamma_i + y \sin\gamma_i + \nu_i))$$

We may now define a *texture description* τ which is a set of n 4-tuples containing the values of the sinusoid parameters.

$$\tau = \{< \lambda_1, a_1, \gamma_1, \nu_1 >, \ldots, < \lambda_n, a_n, \gamma_n, \nu_n >\}$$

We may now determine the brightness value B_τ for any point in the image by summing over each component in the texture description.

$$B_\tau(x,y) = \sum_{i=1}^{i \leq n} \psi_i(x,y)$$

While this gives the brightness of a texture at a given point, this is not sufficient for constructing the image $E(x,y)$. To obtain $E(x,y)$ we must do two things: (1) integrate over the area of each pixel to avoid aliasing, and (2) quantize this real-valued brightness value into an integer pixel value. Thus the equation for the image texture now becomes:

$$E(x,y) \quad = \quad \lfloor k_1 + k_2 \int_0^1 \int_0^1 B_\tau(x+x', y+y')\, dx'\, dy' \rfloor \tag{4.17}$$

where k_1 and k_2 are appropriately chosen so that the values of $E(x,y)$ fall into the desired range (for the experiments in Chapter 5 a range of [0,255] was used for 8-bit pixel intensities).

Now that we have a means of obtaining brightness values from a texture description, all that remains is to choose two such texture descriptions, one for the object and one for the background, and the desired velocities for the background and the foreground. From this description is it easy to create the set of images corresponding with the desired motion. This formulation has several advantages:

- Since B_τ is defined $\forall x, y \in \Re$, we may place the object in any position, not just at discrete pixel positions. This avoids the problems of "cut-and-paste" methods of creating synthetic images and allows us to simulate the apparent motion more accurately.

- By choosing the number of sinusoids in τ, and the distribution from which the λ_i's are chosen, there is a great deal of control over the texture in the image. For example if we choose a large number of sinusoids and/or choose small values for λ_i we obtain a highly textured image.

For the experiments in Chapter 5 a fixed number of sinusoids were used (20), and the wavelengths λ_i were chosen randomly from an interval $[\lambda_{min}, \lambda_{max}]$, with a uniform distribution. The amplitudes a_i were chosen such that $a_i \propto \lambda$. This ensures that the high-frequency components of the images will not dominate the textures, and parallels the $1/F$ noise models which are often used to characterize random processes found in nature [Sch84b].

4.3.2 The Synthetic Object

To avoid biases for discontinuities oriented along a particular direction, the shape of the object was chosen to be a circular disc so as to introduce samples with discontinuities oriented in large number of orientations into the training set. Ideally one would want to also include objects of other shapes such as rectangles and irregular objects. This would further reduce the possibility of introducing a bias for objects of a particular shape. Only circular objects were considered for the training set so as to reduce the number of training patterns. It was hoped that the generalization produced for circular objects would also perform well for general images. The radius of the circular disc was varied so that there was no bias for features of a particular scale. The disparity of the object, as well as the motion of the background, were both chosen to lie within $[-\delta, \delta]$ and the two images corresponding to the left and right viewpoints were created. An example of such a pair of stereo images is given in Fig. 4.18. It is from these sets of images that the cross-correlation and shear values are to be taken.

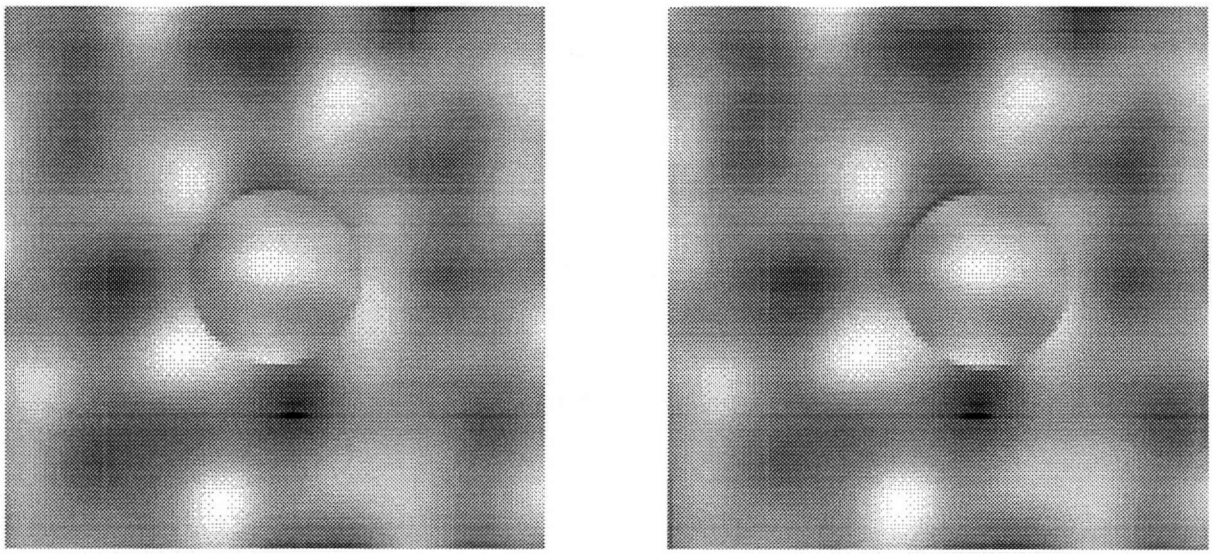


Figure 4.18: Right and left synthetic stereo images

The central disc of the above (100 × 100) image has a radius of 20 pixels and is undergoing a motion of 3.5 pixels to the right; the background is moving 8.5 pixels to the right. The texture of both foreground and background contained 20 sinusoids with amplitudes randomly chosen from [10, 40] pixels.

4.3.3 Sampling the Images

The training patterns were constructed from a set of images described above. From each image a small number of random points (20 or more) were chosen, and the cross correlation measures for the 5 support regions, as well as the shear values were computed. This process was repeated until there were sufficient samples to characterize all possible combinations of the motions of the object and the background.

4.4 Learning with Back-propagation

Once the training set had been created we are now faced with with the task of learning our motion predictions from back-propagation. Since we do not know how many hidden units are necessary for our classification problem, we must discover this empirically.

The learning was implemented using **BP**, a commercially available software package from the PDP research group [RM90]. This version of back-propagation includes the momentum term (see Eqn. (1.4)), as well as providing for easy modification of the network topology.

Chapter 5

Experiments

In order to discover the number of hidden-units required to obtain a solution which allows accurate displacement predictions, three networks were trained containing 15, 30 and 60 hidden-units, respectively. The networks contained 11 output units corresponding to the disparities $\{0,\ldots,10\}$, 57 input units corresponding to the 5 groups of 11 inputs for the SSD values, and 2 additional units for the shear measures. This topology is depicted in Fig. 5.19.

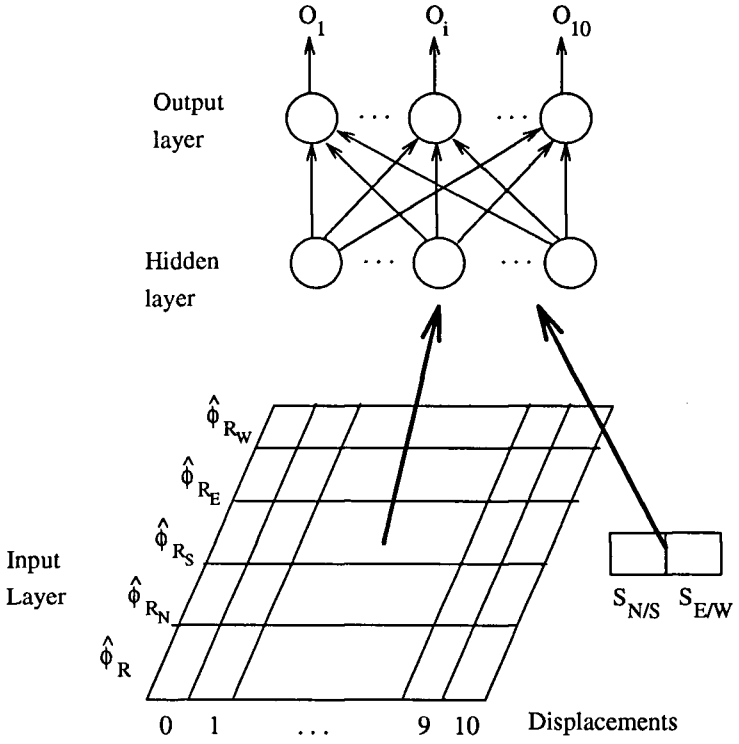


Figure 5.19: The network topology for a displacement range of [0,10] pixels.

Each image was generated by randomly[1] choosing a velocity in the range $[0, 10]$ for both the disc and the background displacements. The radii of the discs were randomly chosen from 5 to 30 pixels. The texture of each image was varied in each image by randomly choosing the values of λ_{min} and λ_{max} from which λ_i was chosen. 20 sinusoids we used for the texture description of both the object and the background. The width of this interval was chosen to be 30 pixels with λ_{min} taken randomly from $[5, 20]$ pixels. 500 images were generated with disc and background motion randomly chosen, and 50 sample points were taken from each image for a total of 25,000 total samples. The SSD values corresponded to the sub-neighborhoods of a (13×13) support region as this was the smallest region size which was able accurately to resolve the image disparities in regions of coherent displacements.

The network weights $(w_{i,j})$ and the bias values (b_i) were initially set to random values in the range $[0, 1]$ before the training was invoked. The learning rate ϵ was initially set to 0.01, and the momentum term M was kept at a constant value of 0.9 throughout the training. The error derivatives were accumulated for each pattern in the training set, and the weight updates were applied at the end of each epoch[2]. The learning rate was increased after the first several hundred epochs, and was again decreased once the error gradients became small (several orders of magnitude smaller than the initial gradients).

The training of the networks was performed three times for each topology, each with a different set of initial weights. The results of the learning process is depicted in Fig. 5.20 which gives the total squared error for each epoch. For each topology the results of all three trials were effectively identical and hence only one of the trials is presented in the error-graph.

The results of the training show that of the three networks (hereafter called HU-15, HU-30, and HU-60 for the 15, 30, and 60 hidden-unit networks, respectively) only two, HU-30 and HU-60, were able to approximate adequately the optimal disparity function. This demonstrates that more than 15 hidden-units are required to compute the disparity from the input data. When we compare the solutions obtained from HU-30 and HU-60 we see that the total squared error is apparently converging to 5×10^3 for both networks. This implies that the 30 additional hidden-units present in HU-60 do not allow for a noticeably better generalization of the input

[1] Unless otherwise specified, a random assignment will refer to a uniformly random distribution.

[2] The term epoch is generally used to signify a single pass through the entire training set.

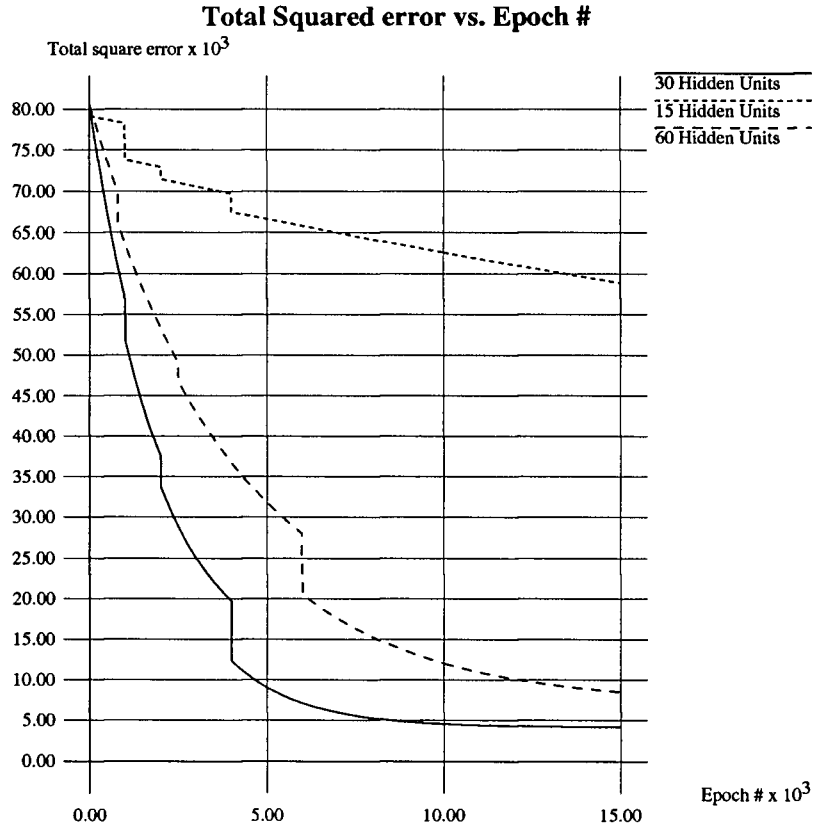


Figure 5.20: Total-squared-error (over all patterns) vs. # epochs for the 15, 30, and 60 hidden unit networks.

data, and that 30 hidden-units are sufficient to capture the relevant features of the input.

While it is possible that, given sufficient training, the HU-60 network would discover additional properties of the input and thus obtain a better solution, it seems more likely that both networks have discovered the global minimum of their respective error-surfaces, and further training is unnecessary. Because the solutions obtained by the HU-30 and HU-60 networks seem to be perform equally well, HU-30 was chosen for further investigation since it requires a factor of two less computation for the evaluation of a set of inputs.

One might ask why the solutions obtained have a residual error of 5×10^3 and why we were not able to reduce the error still further. There are a number of potential answers to this question:

- There may be insufficient numbers of training patterns to allow for a better solution.
- Some training patterns may contain ambiguities due to regions of near-uniform brightness or spatial aliasing.
- The size of the support regions may limit the resolution of the motion and discontinuity primitives.
- Some of the patterns correspond to points near a discontinuity with a diagonal orientation. The motion of these points can not be recovered accurately from the set of rectilinear sub-neighborhoods used.

5.1 Performance Analysis

While the total-squared error (TSE) does give a general measure of the networks performance, since we are using a distributed output representation this measure does not give a good indication of the response to individual input patterns. For example consider the sets of outputs for a given set of target outputs depicted in Table 5.1.

We can see that the outputs for case 2 have a smaller squared error than case 1, but correspond to a much poorer disparity output. For case 2 the disparity-error is slightly larger than 0.5 pixels while the outputs for case 1 do not even correspond to a unique disparity value

Case	Output											TSE
	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	
Optimal	0	0	0	0	0	1	0	0	0	0	0	-
Case 1	0	0	0	0	0.3	0.3	0.3	0.3	0	0	0	0.71
Case 2	0	0	0	0	0.6	0.4	0	0	0	0	0	0.72

Table 5.1: Example error values for output response

but rather a range of disparities. Therefore we must seek a better measure of performance of the distributed representation.

To get a better idea of the performance of the neural network as compared to the "winner-take-all" strategy[3], the disparity predictions of both methods were computed for each of the 25,000 training patterns. These disparity values were then compared to the known disparities as depicted in Fig. 5.21. The $x-$axis gives the radius of an interval, and the $y-$axis gives the fraction of the 25,000 samples whose disparity was contained by an interval centered at the true disparity of each point. Strictly speaking the performance of the neural network should be tested on a set of novel patterns since the solution obtained may be partially the result of exploiting random correlations of the input data. Since the inputs were generated randomly there is little likelihood that such random-occurrence correlations among the inputs will significantly effect the behaviour of the network.

From this graph it is apparent that the HU-30 network is able to determine the disparity of the training samples much more accurately than the single square support-region of the WTA algorithm. If we assume that the samples of the training set are representative of "natural" images, then we may construct a set of confidence intervals as given in Table 5.2.

This table of confidence measures gives us the radius, ρ, such that with confidence C

$$|x - \hat{x}| < \rho$$

where x is the true disparity; $\hat{x}$ is the computed disparity.

[3]To ensure a more fair comparison of the HU-30 and WTA algorithms, the WTA algorithm was modified to allow displacements within sub-pixel accuracy. This was achieved by interpolating over the SSD surface; more specifically by locally fitting the SSD to a quadratic and solving for the minimum.

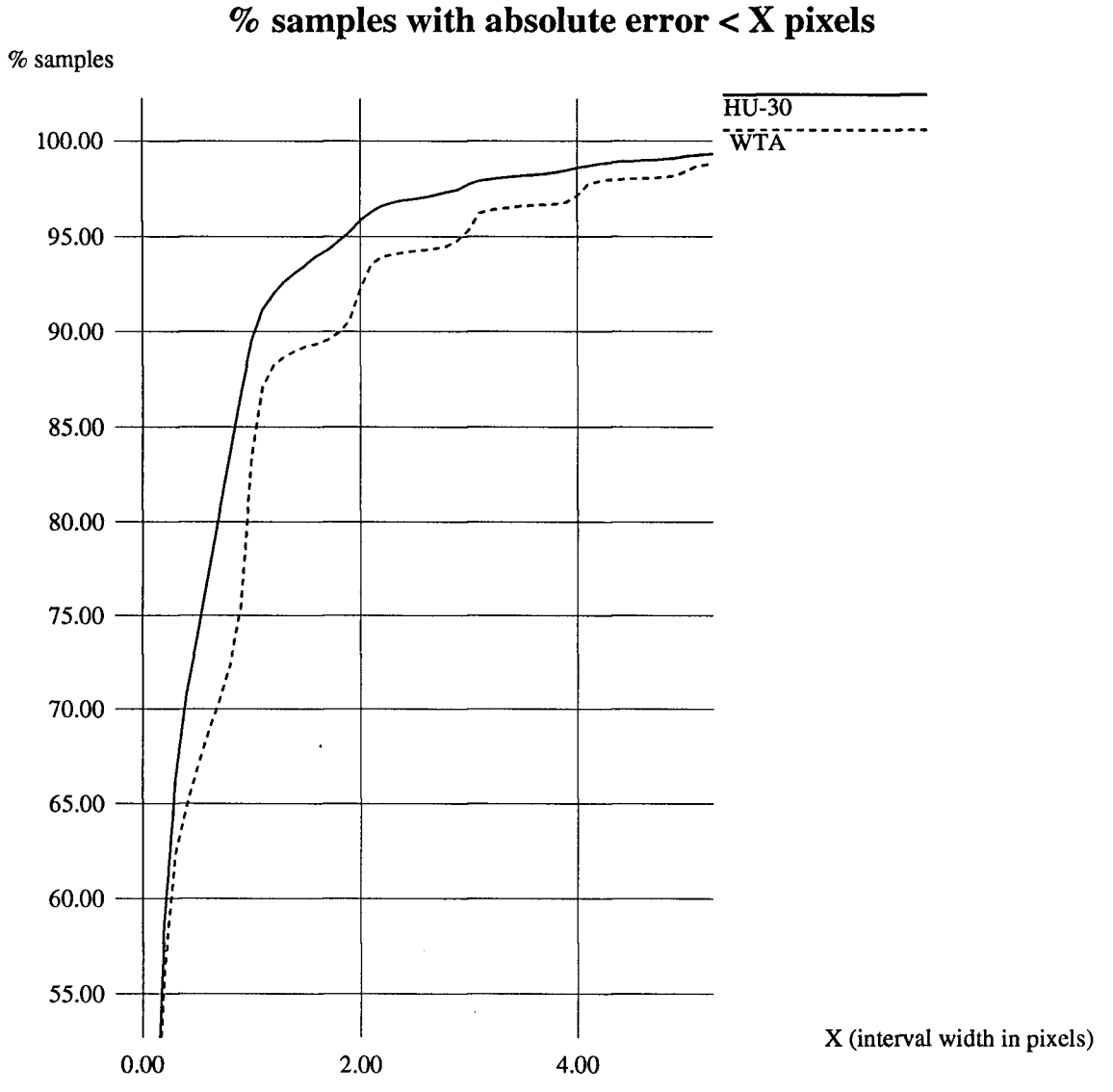


Figure 5.21: Accuracy of the motion predictions for the 25,000 training-set samples using (1) the traditional "winner-take-all" (WTA) algorithm, and (2) the neural network implementation (HU-30).

C = % Confidence	ρ = Interval Width (Pixels)		
	Winner-take-all	HU-30	improvement
50	0.141	0.129	0.012
60	0.243	0.213	0.030
70	0.662	0.383	0.279
80	0.978	0.688	0.290
90	1.830	1.021	0.809
95	2.964	1.853	1.111
97	3.981	2.549	1.432

Table 5.2: Widths of confidence intervals for winner-take-all and HU-30

This result show that for relatively low confidence levels (50-60%), both WTA and HU-30 are able to resolve the apparent disparity to less that 0.2 pixels from the true disparity. For more reasonable confidence levels (85-90%), however, we find that the resolving power of HU-30 to be far superior to that of WTA. For example, if our goal were to resolve the apparent disparity to within a pixel of the true disparity, then we are able to classify each pixel with a 90% success-rate as compared to 80% using WTA.

The smooth shape of the HU-30 confidence function gives us reason to believe that network is indeed performing an interpolation of the comparison functions, $\phi_i(\cdot)$ to obtain sub-pixel accuracy. This hypothesis is further corroborated by the 10% improvement in the confidence for disparities with less than one pixel deviation from the true disparity.

5.1.1 Image Segmentation and Object Boundaries

Often in computer vision applications it is necessary to segment a scene into a set of objects. Depth and motion cues allow us to identify the object boundaries which give the underlying geometry of the objects. As was noted in Chapter 3, traditional correlation-based stereo and optical flow algorithms (WTA) are prone to error at the boundaries of objects. This may cause the shape of the boundaries to become degraded.

To test the degree to which HU-30 is able to make use of discontinuity information, and thus obtain a better object contour, a series of test images were created (see Fig. 4.18).

The disparities of the image were 8.5 and 3.5 pixels to the right for the central disc (radius

of 20 pixels) and background, respectively. To segment the scene, pixels were classified into two groups: pixels with disparities greater than 6 pixels corresponding to the disc, and pixels with disparities less than 6 pixels corresponding to the background. Once the pixels were classified, the contour was constructed. Fig. 5.22 and 5.23 correspond to the contours generated by the disparities from the WTA and HU-30 algorithms, respectively.

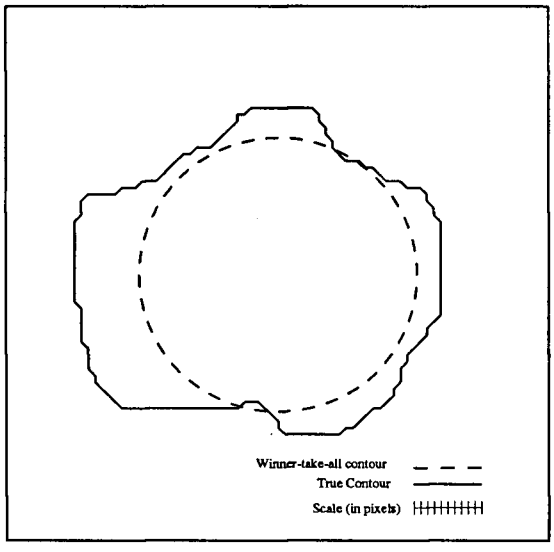


Figure 5.22: Disc contour of WTA

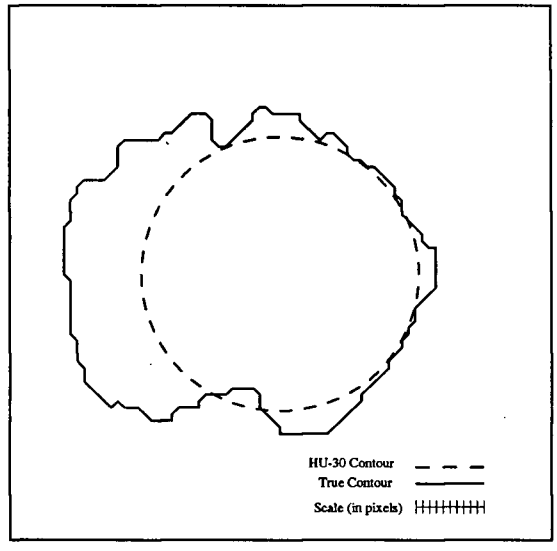


Figure 5.23: Disc contour of HU-30

The center disc has a radius of 20 pixels and is undergoing a motion of 3.5 pixels to the right; the background is moving 8.5 pixels to the right. The object contours were generated by marking the 6-pixel disparity crossings. The texture of both foreground and background contained 20 sinusoids with amplitudes randomly chosen from $[10, 40]$ pixels. Disparity measures taken from the (13×13) for Fig. 5.22 and the corresponding sub-neighborhoods in Fig. 5.23.

Close examination of the object boundaries shows that for portions of the image which are visible from both the left and right viewpoints (*i.e.*, the right portions of the disc) the localization of the object boundary is greatly improved. The location of the HU-30 boundary is predicted to within zero to 2 pixels of the true contour (on average), while the boundary predicted by WTA is significantly poorer, occurring up to 5 pixels from the true boundary. In portions of the image containing occlusions the localization of the object boundary is much poorer for both algorithms. This demonstrates that for dis-occluding boundaries the neural network is able to accurately recover the object contour while occlusions provide too little

information to recover the disparity when using either algorithm.

To further characterize the performance of the stereo algorithms, it is necessary to analyze the performance at occluding and dis-occluding boundaries separately. To accomplish this consider an object placed at the center of the left image which is moving to the right relative to the background. We may now partition the image into two equally-sized regions: The left side which will contain occlusions, and the right side which will contain only dis-occluding boundaries.

To study the performances of the WTA and HU-30 algorithms for these two separate regions a set of 100 randomly generated images were created and the total disparity error for each pixel in the two regions was recorded. The average total disparity error for the 100 images according to region is given in Table 5.3.

Algorithm	Mean total absolute displacement error		
	Dis-occluding region	Occluding region	Total
HU-30	1054.4	2856.1	3910.5
WTA	1595.0	3170.4	4765.4
% improvement	33.9	9.9	17.9

Table 5.3: Total absolute disparity error statistics for 100 randomly generated images

These results seem to indicate that for regions in the image which containing dis-occlusions there is a 34% improvement in the total disparity error. The gains made by HU-30 over WTA are lessened significantly in regions of the image where occlusions occur and gives a mean total error which is only a 10% improvement. This improvement is most likely a result of the sub-pixel interpolation of the disparities in regions which are unaffected by the boundary.

5.1.2 Natural Images

To get an idea of how well HU-30 performs on natural images, two aerial pictures (see Fig. 5.24) corresponding to the left and right viewpoints of a set of buildings were used. The disparities (which were between 1 and 8 pixels) were obtained from both the WTA and the HU-30 algorithms. The support regions were (13 × 13) pixels.

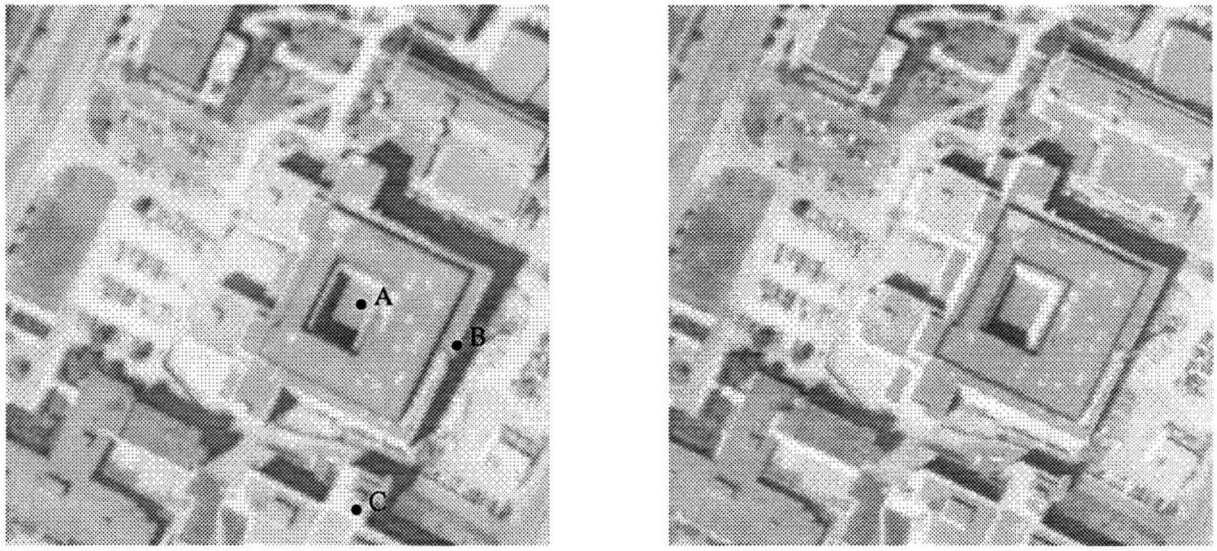


Figure 5.24: Left and right stereo images of UBC image

UBC Acute Care Hospital scanned from BC Air Photos 79046 37-38. Both images are (256×256) 8-bit pixels. The displacements varied from approximately 2 to 8 pixels.

From these disparity maps it is apparent that the disparity measures (and hence depth) of the HU-30 algorithm are qualitatively superior to that of WTA. This improvement in the disparity values is especially apparent near the depth discontinuities in the image. For example, the contour of the "hole" of the central building (point A of Fig. 5.24) is much closer to the true contour, and contains less smearing effects. This can also be seen by examining the overall shape of the building in the lower portion of the image (point C in the image) which is much better preserved in the WTA disparity map. In addition, smoothing effects are apparent at point B in the image, where, instead of containing two separate disparity values (*i.e.*, the building and the ground), the disparities predicted from WTA are a smoothened version of the true step-edge.

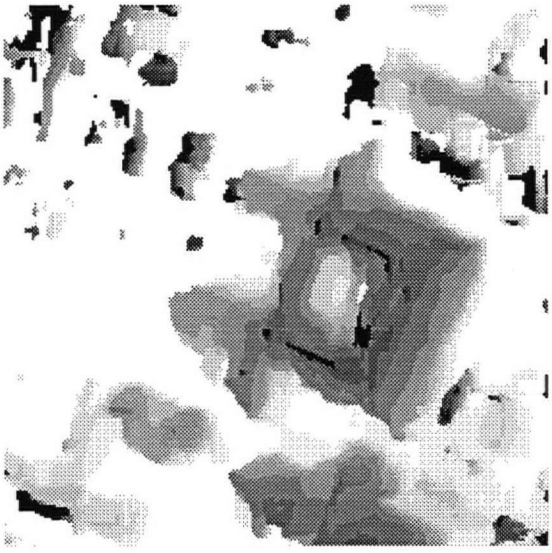


Figure 5.25: Winner-take-all disparities for UBC image

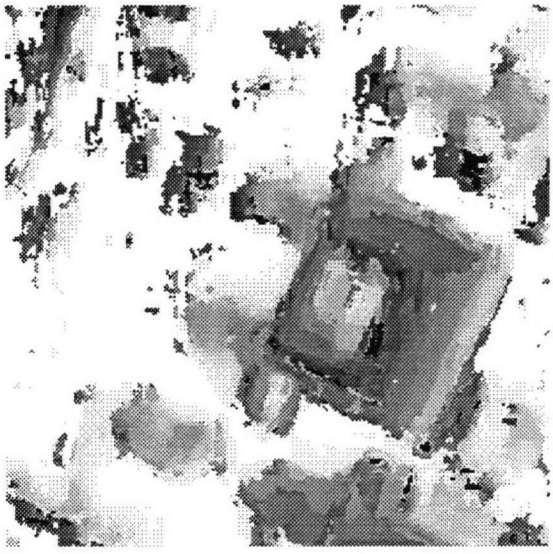


Figure 5.26: Neural network disparities for UBC image

Figure 5.25 and 5.26 give the results of the disparity predictions for both the WTA and HU-30 algorithms, respectively, for the stereo image pair of Fig. 5.24. The disparities range from zero pixels (denoted by white) to 8 pixels (denoted by black). (Note: disparity map is in the left image frame-of-reference).

Chapter 6

Conclusions

Using the shear values defined from the sum-of-squared-differences of a set of oriented rectilinear sub-neighborhoods, the presence and orientation of discontinuities of a flow field can be inferred. The orientations were restricted to two possible hypotheses: horizontal and vertical. Tests on randomly generated band-limited images indicated that approximately 80% of the points at, or near a discontinuity were classified to the correct orientation.

An artificial neural network was used to integrate the two sources of knowledge, namely the cross-correlation values from the different support regions and the discontinuity information from the shear values, to produce a single prediction for disparity of the point (binocular stereo). The correct representation for the integration task was *learned* from a set of 25,000 training examples which were obtained from randomly generated band-limited synthetic images.

The resulting algorithm compared to the traditional winner-take-all correlation algorithm performed significantly better for both synthetic and natural images. Analysis of the confidence intervals showed that the neural network implementation correctly classified ~90% of the training patterns to within 1.0 pixels of the true disparity compared to only ~80% using the winner-take-all algorithm.

On synthetic images, constructed by the superposition of randomly generated sinusoids, and natural images, the neural network reduced the distortion effects occurring near discontinuities and produced object boundaries which were significantly better representations of the object's true structure. Smoothing effects near the discontinuities, which smear step-edges in the disparity map, were reduced using the neural network as compared to the winner-take-all strategy. Analysis of the outputs of the neural network indicated that the neural network interpolated

over the SSD surface and thus produce disparities to sub-pixel accuracy.

By phrasing the problem of integrating discontinuity and motion information as a learning problem, a network is able to capture sufficient information from the input distribution to allow reasonable disparity predictions. It has demonstrated that the necessary information needed to perform the integration task can be extracted directly from a suitable set of randomly generated examples. The performance of the neural network on natural images corroborates the hypothesis that sufficient information is available from the synthetic images to allow the network to characterize the behaviour of natural images. This indicates that the synthetic image model employed is a reasonable approximation to natural images for the purposes of learning in low-level visual tasks.

Chapter 7

Future Work

An obvious modification to the present model would be to extend the model to allow the determination of optical flow. In theory, since the proposed techniques are equally applicable to the problem of optical flow, the only modification which would be necessary is is to change of the topology of the neural network. This would involve replacing the one-dimensional array of cross-correlation inputs with a two-dimensional matrix of inputs. A similar modification would also be necessary for the output units so as to allow for the encoding of a two-dimensional output. As was discussed in Chapter 4, this would drastically increase the computational complexity of the learning procedure, as well as the number of training examples necessary to obtain a good generalization of the inputs. This does not seem feasible using the present network model and learning procedure.

There are a number of possible ways in which we might choose to reduce the complexity described above. One possibility might be to attempt to reduce the dimensionality of the data by using unsupervised learning techniques such as [BH89, Gro76, San89]. Such techniques could be applied in the first layer of the network, and could potentially reduce the number of hidden units needed in subsequent layers. In addition to reducing the complexity, there is reason to believe that the resulting representation obtained from unsupervised learning procedures is potentially easier to interpret [San89]. Further gains in the learning time might also be achievable through the use of more sophisticated gradient-descent methods such as *conjugate-gradient descent* (see Chapter 5 of [Str86] for details).

While the present implementation is limited to predicting one-dimensional motion, this model be easily extended to identify depth-discontinuities in the image. While the naïve strategy

for determining the presence and orientation of discontinuities given by Eq. (3.13) and Eq. (3.14) have been shown to work reasonably well, there is reason to believe that the neural network should be able to infer this information more accurately. This could be realized by exploiting further information about the discontinuities from the error surfaces themselves (for example the valleys may become more rounded [BA90]). Much of this information is probably already represented in the present neural network since this information is needed for accurate disparity predictions. Additional information from the error-surface may also be exploited by the neural network to allow the identification of occlusions in the scene (perhaps in a similar manner to [LG90]).

In the future one might want to investigate a number of alternative sets of inputs to the network. This would provide additional insights as to the kinds of visual primitives which are best suited for the extraction of motion. These might include:

- The use of Blake and Zisserman's *weak continuity constraint.* This constraint has the effect of weighting heavily the differences which lie within an expected range and remains uncommitted about data outside this range [BZ87].
- The use of confidence measures on a set of features, such as zero-crossings, rather than the raw intensity values [DP86, BLP87].
- The use of motion-primitives at multiple scales [Ana89], or the use of more than two consecutive images.
- The use of additional inputs to denote the presence of intensity discontinuities. This may greatly improve the localization of disparity discontinuities.

There are many potential improvements related to the generation of the training patterns. These may include the use of polygonal objects containing corners and straight edges. Such objects are more likely to correspond to the types of objects which are found in computer vision applications. Additionally the use of textures obtained from natural images may greatly improve the performance of the the neural network on real vision problems.

At present there is an asymmetry in the ability to predict displacements near occluding and disc-occluding boundaries. This is due to the fact that we are only matching points in

the t_0 frame of reference to points in the t_1 frame of reference. If additional inputs were added to the network which encoded the SSD for ϕ in the t_1 frame of reference, then the resulting generalization could potentially eliminate the asymmetry, and thus provide accurate displacements for both occluding and dis-occluding boundaries. Examples of algorithms which use a correlation-based approach in which matching is performed on both time frames can be found in [MP76, Fua91, HLLJ91].

As was mentioned in Chapter 1, part of the appeal of neural networks is the study of the weight-space representation of the trained network. This analysis potentially gives one a better understanding of the operation of the neural network, as well as giving new insights to the problem at hand. This is potentially very difficult since each hidden unit is combining features of several error surfaces, and thus the weight-space is difficult to interpret. This work is ongoing and is not presented in this thesis.

The neural network implementation for the determination of stereo disparity is presently not very efficient. The large number of multiplications and the evaluation of the sigmoid function make the evaluation of the network unacceptably slow. In addition the computation proceeds serially, computing each disparity on a pixel-by-pixel basis. Future work on this project will involve the implementation of the neural network on a parallel architecture; specifically a set of transputers. This would allow the computation of the cross-correlation measures to be computed in parallel, along with the evaluation of the network. Such a parallel implementation of the neural network disparity algorithm is crucial to its practical use on real vision problems.

Bibliography

[AB85] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2:284–299, 1985.

[Alb72] Arthur Albert. *Regression and the Moore-Penrose Pseudoinverse.* Academic Press Inc., New York, 1972.

[Ana89] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[AS89] John Y. Aloimonos and David Shulman. *Integration of Visual Modules – An Extension of the Marr Paradigm.* Academic Press Inc., San Diego, Ca., 1989.

[BA90] M. J. Black and P. Anandan. Constraints for the determination of discontinuity from motion. In *Proceedings of Eighth International Conference on Artificial Intelligence*, pages 1060–1066, Cambridge, Ma., 1990. MIT Press/AAAI Press.

[BH89] Suzanna Becker and Geoffrey Hinton. Spatial coherence as an internal teacher for a neural network. Technical Report CRG-TR-89-7, The University of Toronto, Toronto, Ont., 1989.

[BLP87] Heinrich H. Bülthoff, James J. Little, and Tomaso Poggio. Parallel motion algorithm explains barber pole and motion capture illusion without "tricks". *Journal of the Optical Society of America*, 4:34, 1987.

[Bon88] Robert M. Bonton. Color vision. *Annual Review of Psychology*, 38:69–100, 1988.

[BZ87] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, Cambridge, Ma., 1987.

[Can86] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[Cav87] Patrick Cavanagh. Reconstructing the third dimension: Interactions between color, texture, motion, binocular disparity, and shape. *Computer Vision Graphics and Image Processing*, 37:171–195, 1987.

[CY90] James Clark and Alan Yuille. *Data Fusion for Sensory Information Processing Systems.* Kluwer Academic Publishers, Norwell, Ma., 1990.

[DP86] Michael Drumheller and Tomaso Poggio. On parallel stereo. In *Proceedings of IEEE Conference on Robotics and Automation*, pages 1439–1448, Washington, DC, 1986. IEEE.

[Dru84] Michael Drumheller. Synthesizing a motion detector from examples. Master's thesis, Massachusetts Institute of Technology, 1984.

[Eag91] Roy Eagleson. Measurement of visual motion: non-commutative harmonic analysis on the 2-D affine lie group. *The Journal of Spatial Vision*, 1991. In press.

[FB82] Jerome A. Feldman and Dana Ballard. Connectionist models and their properties. *Cognitive Science*, 6, 1982.

[For90] Stephanie Forrest. Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks. *Physica D*, 42(1–3):1–11, 1990.

[FT79] C. L. Fennema and W. B. Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9(4):301–315, 1979.

[Fua91] Pascal Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. Rapports de Recherche No. 1369, INRIA-Sophia Antipolis, 1991.

[Gro76] Stephen Grossberg. Adaptive pattern classification and universal recoding: Part I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[Hau90] David Haussler. Probably almost correct learning. In *Proceedings of Eighth International Conference on Artificial Intelligence*, pages 1101–1108, Cambridge, Ma., 1990. MIT Press/AAAI Press.

[Hil84] Ellen C. Hildreth. Computations underlying the measurement of visual motion. *Artificial Intelligence*, 23(3):309–354, August 1984.

[Hil85] Ellen C. Hildreth. Edge detection. A.I. Memo No. 858, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Ma., 1985.

[HK86] E. Hildreth and C. Koch. The analysis of visual motion: from computational theory to neuronal mechanisms. AI-Memo-919, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1986.

[HKP91] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation: Lecture Notes Volume I, Santa Fe Institute Studies in the Sciences of Complexity.* Addison Wesley, The Advanced Book Program, Redwood, Ca., 1991.

[HLLJ91] J. M. Hakkarainen, J. J. Little, H. S. Lee, and J. L. Wyatt Jr. Interaction of algorithm and implementation for analog vlsi stereo vision. In *Proceedings of 1991 SPIE Conference on Visual Information Processing: From Neurons to Chips*, April 1991.

[HMR86] G. Hinton, J. McClelland, and D. Rumelhart. *Distributed representations*, chapter 3, pages 77–109. MIT Press, Cambridge, MA., 1986.

[Hor75] B. K. P. Horn. Obtaining shape from shading information. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 115–155. McGraw-Hill Book Co., New York, 1975.

[Hor86] B. K. P. Horn. *Robot Vision.* MIT Press/McGraw-Hill, Cambridge, MA, 1986.

[HS81] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[Jud87] J. Stephen Judd. Complexity of connectionist learning with various node functions. COINS TR 87-60, Department of Computer and Information Sciences, The University of Massachusetts, Amherst, MA, 1987.

[KD76] J. J. Koenderink and A. J. Van Doorn. Local structure of movement parallax of the plane. *Journal of the Optical Society of America*, 66:717–723, 1976.

[LB85] D. G. Lowe and T. O. Binford. The recovery of three-dimensional structure from image curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:320–326, 1985.

[LG90] James Little and Walter Gillett. Direct evidence for occlusion in stereo and motion. *Image and Vision Computing*, 8(4):328–340, 1990.

[LM87] D. T. Lawton and C. C. McConnell. Perceptual organization using interestingness. In *Proc. AAAI Workshop Spatial Reasoning and Multi-Sensor Fusion, 1987*, 1987.

[LV88] James J. Little and Alessandro Verri. Analysis of differential and matching techniques for optical flow. In *IEEE Workshop on Visual Motion*, pages 173–180, Irvine, Cal., April 1988.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman and Company, San Francisco, 1982.

[MN87] John H. R. Maunsell and William T. Newsome. Visual processing in monkey extrastriate cortex. *Annual Review of Neuroscience*, 10:363–401, 1987.

[MP43] Warren McCulloch and Walter Pitts. A logical calculus of the immanent in nervous activity. *Bulletin of mathematical biophysics*, 5:115–133, 1943.

[MP76] David Marr and Tomaso Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 15 October 1976.

[MP88] Marvin Minsky and Seymour Papert. *Perceptrons.* MIT Press, Cambridge, Ma., second edition, 1988.

[MU81] David Marr and Shimon Ullman. Directional selectivity and its use in early visual processing. *Proceedings of the Royal Society of London B.*, 211:151–180, 1981.

[Nak85] K. Nakayama. Biological image motion processing: A review. *Vision Research*, 25(5):625–660, 1985.

[Nis84] H.K. Nishihara. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.

[NL74] K. Nakayama and J. M. Loomis. Optical velocity patterns, velocity sensitive neurons, and space perception: a hypothesis. *Perception*, 3:63–80, 1974.

[NS88] K. Nakayama and Gerald Silverman. The aperture problem I: The perception of non-rigidity and motion direction in translating sinusoidal lines. *Vision Research*, 28:739–746, 1988.

[PGL88] Tomaso Poggio, Edward B. Gamble, and James J. Little. Parallel integration of vision modules. *Science*, 242(4877):436–440, October 21 1988.

[PH87] D. C. Plaut and G. E. Hinton. Learning sets of filters using back-propagation. *Computer Speech and Language*, 2:35–61, 1987.

[Pog75] Tomaso Poggio. On optimal nonlinear associative recall. *Biological Cybernetics*, 19:201–209, 1975.

[Pot75] J. L. Potter. Velocity as a cue to segmentation using motion information. *IEEE Transactions on Systems, Man, Cybernetics SMC-5*, pages 390–394, 1975.

[PTK87] Tomaso Poggio, Vincent Torre, and Christof Koch. Computational vision and regularization theory. In Martin Fischler and Oscar Firschein, editors, *Readings in Computer Vision*, pages 638–643. Morgan Kaufmann Publishers, Inc., Los Altos, CA., 1987.

[Rei61] Werner Reichardt. Autocorrelation, a principle for the evaluation of sensory information. In W. A. Rosenblith, editor, *Sensory Communication.* Wiley, New York, 1961.

[RH86] David Rumelhart and Geoffrey Hinton. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[RM86] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.* MIT Press, Cambridge, MA., 1986.

[RM90] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 3: Methods, Programs, and Excercises.* MIT Press, Cambridge, MA., 1990.

[Ros62] F. Rosenblatt. *Principles of Neurodynamics.* Spartin, New York, 1962.

[San89] Terrence Sanger. Optimal unsupervised learning in a single-layer linear feed-forward neural network. *Neural Networks*, 2:459–473, 1989.

[Sch84a] B. G. Schunck. The motion constraint equation for optical flow. In *Proc. 7th Int. Conf. Pattern Recognition*, pages 20–22, Montreal, PQ, 1984.

[Sch84b] Heinz Georg Schuster. *Deterministic Chaos.* Physik-Verlag, Weinheim, FRG., 1984.

[Str86] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley Ma., 1986.

[SU87] Anselm Spoerri and Shimon Ullman. The early detction of motion boundaries. In *Proceedings of the first international conference on computer vision*, pages 209–218, Washington, DC, 1987. Computer Society Press for the IEEE.

[TA77] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems.* Winston, Washington, D.C., 1977.

[TP78] Vincent Torre and Tomaso Poggio. A synaptic mechanism possibly underlying directional selectivity to motion. *Proceedings of the Royal Society London B*, 202:409–416, 1978.

[VP86] A. Verri and Tomaso Poggio. Regularization theory and shape constraints. AI-Memo-916, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1986.

[vSS84] Jan P. H. van Santen and George Sperling. Temporal covarience model of human motion perception. *Journal of the Optical Society of America A*, 1(5), 1984.

[Woo78] R. J. Woodham. Photometric stereo. AI-Memo-479, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, 1978.