Connectionist Models Summer School
Morgan Kaufmann Publishers, San Mateo, California 198

Learning to Solve Random-Dot Stereograms of Dense and Transparent Surfaces with Recurrent Backpropagation

Ning Qian and Terrence J. Sejnowski

Department of Biophysics
The Johns Hopkins University
Baltimore, MD 21218, USA

ABSTRACT

Binocular depth perception, or stereopsis, depends on matching corresponding points in two images taken from two vantage points. In random-dot stereograms the features to be matched are individual pixels. We have used the recurrent backpropagation learning algorithm of Pineda (1987) to construct network models with lateral and feedback connections that can solve the correspondence problem for random-dot stereograms. The network learned the uniqueness and continuity constraints originally proposed by Marr and Poggio (1976) from a training set of dense random-dot stereograms. We also constructed networks that can solve sparse random-dot stereograms of transparent surfaces. The success of the learning algorithm depended on taking advantage of translation invariance and restrictions on the range of interactions.

1. Introduction

There are many visual cues for depth, but only one of them requires two eyes. Binocular depth perception is based on the difference between corresponding positions on the two retinal images, called the image disparity. Random-dot stereograms demonstrate that perception of depth does not depend on monocular cues such as the recognition and identification of shapes and objects (Julesz, 1960). A stereogram is a uniform field of random dots, but some of the dots in the left image are shifted relative to the corresponding dots in the right image. When fused binocularly, the central shifted area is perceived as floating above the background. An example of such a stereogram is shown in Fig. 2a.

The recovery of depth from a pair of images requires a solution to the correspondence problem (Poggio and Poggio, 1984): Proper matches must be made between features (such as pixels, edges, etc.) found in the right and left images. For a given feature in the left image there may exist more than one feature in the right image that can potentially match it. In this paper, we will only deal with random-dot stereograms for which the natural features are the black dots (one may also use both black and white dots as matching features or use black-white and/or white-black transitions). The matching is a nontrivial problem because each dot in one image can potentially match all the other dots within a certain range along the same horizontal line in the other image.

Many solutions to the correspondence problem for random-dot stereograms have been proposed. The present work comes closest to the method of Dev (1975) and Marr and Poggio (1976), who used a neural network model with binary units and local interactions between them. Marr and Poggio determined the connections in their network based on two physical constraints: continuity of surfaces and uniqueness of depth along lines of sight. In this paper we show that these two constraints can be learned automatically using recurrent backpropagation (Pineda, 1987). Further, we report that the learning algorithm can construct networks that solve transparent random-dot stereograms, which the network model of Marr and Poggio (1976) was not able to handle.

Other algorithms have been devised that can solve transparent random-dot stereograms, such as that of Prazdny (1985), but these methods cannot be implemented as neural network models. Szeliski and Hinton (1985) replaced Prazdny's support function with a network implementation, but they also selected the match at the end with a *max* function. Hebbian learning has been applied to the correspondence problem for stereograms by Sun et. al. (1987). Their method depended on the use of dense disparity maps and is probably not applicable to transparent stereograms.

2. Learning Algorithm

The standard backpropagation learning algorithm (Rumelhart et al., 1986) is unlikely to solve the stereo correspondence problem for random-dot stereograms. False matches must be resolved by interactions between the units in the network that represent depth, which cannot be easily represented in a strictly feedforward network. Pineda (1987) has recently generalized the stan-

dard backpropagation to networks with arbitrary connectivity. We briefly describe this formalism and then modify it for the stereo problem.

The dynamics of a recurrent network is determined by:

$$dx_i/dt = -x_i + f_i(u_i) \qquad (1)$$
$$u_i = \sum_j w_{ij}x_j + I_i$$

where x_i represents the activity of ith unit, w_{ij} is the weight from jth unit to ith unit, and I_i represents the external input to the ith unit. A commonly used form for the nonlinear function $f_i(u)$ is the logistic function $(1+e^{-u})^{-1}$. A subset of units receive inputs, another subset are considered outputs, and units that are neither input nor output are called hidden units. Within the formalism of recurrent backpropagation the same unit can both be an input and an output unit. In most of our simulations, the input and output units were the same, and there were no hidden units. Given initial conditions and an input, the network usually evolved toward a fixed point by Eq. (1), although this is not guaranteed for an arbitrary set of weights. These fixed points can be used to store information.

The differences between the actual and desired fixed points, J_i, are propagated to all the other units by computing the error signals y_i obtained as the fixed point of a related dynamical system:

$$dy_k/dt = -y_k + f_k'(u_k^\infty)(\sum_r w_{rk}y_r + J_k) \qquad (2)$$

where u_k^∞ is the fixed point value of u_k. The weights are changed according to:

$$dw_{rs}/dt = \varepsilon y_r^\infty x_s^\infty \qquad (3)$$

where ε is the leaning rate. In the actual simulation the difference equation corresponding to Eq. 3 was modified by the inclusion of a momentum term α. We used the following parameters in our simulations unless otherwise stated: $\Delta t = 0.9$, $\varepsilon = 1.0$, and $\alpha = 0.9$.

In the present network we wanted the output values to be 0 or 1, so that a continuous-valued output unit was considered to be on if its output exceeded 0.5, the midpoint of the output range, and off if the value was less than 0.5. The correct output values in our training examples were mainly off when sparse disparity maps (see Section 4) were used. This typically caused the learning algorithm to make all of the weights in the network inhibitory which would turn all the units off. We modified the error backpropagation procedure slightly to reduce this problem: the error J_i of a unit was set to zero if it was classified correctly (above or below 0.5) even though it was not perfect (zero or one). This is equivalent to using a margin of 0.5. After the success rate of the network leveled off, we sometimes used the unmodified learning algorithm with a margin of 0.45 to further improve the performance. The activation of a unit in the resulting network was on average above or below the midpoint by 0.2. This modification also made the weights grow more slowly and the learning was considerably faster. With the unmodified learning algorithm, the weights in a network tended to grow fast and the convergence of Eqs. 1 and 2 become slower and slower.

3. Network Architecture

The neural network model was similar to that of Marr and Poggio (1976), except that our units had continuous rather than binary values. The network consisted of several disparity levels, which can also be considered as depth levels because depth and disparity are monotonically related. Processing units represented possible matches between the two images and were located at the intersections of the depth planes with lines of sight from the observer, as shown in Fig 1. The zero disparity layer is the plane of fixation.

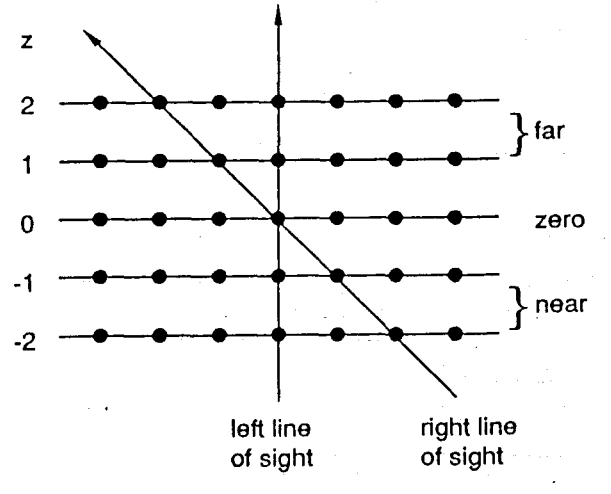

Fig. 1: Network architecture. Each horizontal line actually represents a 2-D surface. Although the two lines of sight should be symmetric, it is more convinient to "tilt" the network to the left a little so that the units are lined up along the z-axis.

Each unit in the network was labeled by its integer coordinates (x, y, z) and each weight can be specified by:

$$(x_1, y_1, z_1;\ x_2, y_2, z_2) \tag{4}$$

which represents a connection from a unit at location (x_1, y_1) in layer z_1 to a unit at location (x_2, y_2) in layer z_2. Since the problem is translationally invariant, this symmetry was designed into the network explicitly by setting:

$$(x_1, y_1, z_1;\ x_2, y_2, z_2) =$$
$$(x_1 + M, y_1 + N, z_1;\ x_2 + M, y_2 + N, z_2) \tag{5}$$

where M and N are arbitrary integers (Giles, et al., 1988). This greatly reduced the number of independent weights in the network as well as the amount of training time and the number of training examples needed. Thus, each independent weight in the network was denoted by a quadruple:

$$(\Delta x, \Delta y, z_1, z_2) \tag{6}$$

which represents the weight connecting a unit in layer z_1 to a unit in layer z_2 offset in x and y directions by Δx and Δy.

Each unit is connected only to those nearby units that are relevant to the solution of the problem. For solving random-dot stereograms with fronto-parallel planes, the networks were connected in the following way: Each unit made connections, A, with set of units in the same layer and also a set of connections, B, to units in other layers that were along the two lines of sight of the right and left eyes. Each unit also had a bias that was treated as a weight from a unit that was always on. Formally,

$$A = \left\{(\Delta x, \Delta y, z, z) \mid (\Delta x)^2 + (\Delta y)^2 \le R^2\right\} \tag{7}$$

where R is an integer which determines the size of the neighborhood and

$$B = \left\{(0, 0, z_1, z_2) \mid z_1 \neq z_2\right\} \cup$$
$$\left\{(\Delta x, 0, z, z + \Delta x) \mid \Delta x \neq 0\right\} \tag{8}$$

The number of connections in A depends on the value of R, the range of interaction in Eq. 6. For example, if $R = 1$, there will be four connections on each disparity level, so the total number of connections in A is $4 \times D$ where D is the total number of disparity levels considered. Similarly, the number of connections in B is $2D(D-1)$. Variations on the architecture described above will be mentioned later.

4. Input and Output Representations

The compatibility map for an input to a network is defined as all the possible matches between dots along the same horizontal lines in two images within a certain range (corresponding to all the disparity levels represented in the network). One may consider matches between two points of the same intensity (black-black or white-white) or only between two black points. The resulting data in the former case will be called the *dense* compatibility map, C_d, while the later will be called the *sparse* compatibility map, C_s (Szeliski, 1986). More precisely:

$$C_d[x, y, z] = \begin{cases} 1 & \text{if } L[x, y] = R[x + z, y] \\ 0 & \textit{otherwise} \end{cases} \tag{9}$$

and

$$C_s[x, y, z] = \begin{cases} 1 & \text{if } L[x, y] = R[x + z, y] = 1 \\ 0 & \textit{otherwise} \end{cases} \tag{10}$$

where $L[x, y]$ and $R[x, y]$ are the binary values of the left and right image intensities at (x, y). Of all possible matches in a compatibility map, the sublet that are correct is called the disparity map. In our models, only sparse compatibility maps were used as inputs, but both sparse and dense disparity maps were used as outputs.

In our first attempt to train a network to solve the correspondence problem, we set the initial states of the units to the compatibility map, let the network settle down into a final state, and used the correct disparity map as the target to compute the errors. This procedure failed. An explanation, suggested in Pineda (1988), is that a network with an initial set of random weights may not have enough attractors to store all the different disparity maps. Two or more desired attractors may be located within the basin of one attractor in the initial network and the learning algorithm can only drive this single attractor to the average position of the desired attractors. However, the attractors and the basin boundaries depend on the input I_i as well as the weight matrix, so clamping the inputs could help to break the degeneracy. This was an effective solution and, as shown below, clamping I_i also helped to solve the correspondence problem for transparent random-dot stereogram.

5. Results

All the network performances reported below were measured on testing patterns not used for training. The performance figures are the percentages of the output units with correct activation values.

Dense disparity. In most of the simulations, we used only three disparity planes and 30×30 random-dot stereograms, so there were a total of 2,700 units in the network. However, the resulting set of weights can be used to solve stereograms of any size because of translational invariance. For $R = 1$ in Eq. 4 there were 12 weights in A, 12 weights in B and 3 biases, for a total of 27 independent weights. These were assigned random values at the beginning of learning. The 6 training examples were balanced so that there were about an equal number of 1's in each of the three disparity planes. A dense disparity map was used as the desired output so that our results could be compared with those of Marr and Poggio (1976). The performance of the network on new stereograms was 99% after 250 presentations of random-dot stereograms. An example is shown in Fig. 2. Although the inputs were clamped to the compatibility map during the learning, they do not have to be clamped during testing after training.

The learning was much faster when further constraints were placed on the weights. All the weights in group A have a similar role, and following learning they were all within 10% of the average. Learning is faster when all the weights in A were made identical, all the weights in B were identical, and all the biases were the same. Table 1 shows the three independent weights that were found. Additional ways to speed up the learning were also examined. Networks with units having values in the interval (-1, 1) by using $f(u) = tanh(u/2)$ learned faster, as reported earlier for standard back- propagation (Stornetta & Huberman, 1987). Only 7 presentations were required to achieve a performance of 99% for three independent weights and values of (-1, 1).

Table1: Weights for Dense and Sparse Disparity Map

weights	dense	sparse
set A	1.386	0.544
set B	-1.717	-0.590
bias	-1.292	-0.872

All weights within sets A, B and biases were constrained to be the same (see text).

The addition of irrelevant weights had the opposite effect and slowed down the learning. In one experiment one connection was added, (2, 0, 0, -1), and in a second experiment two weights were added, (-1, 0, 0, -1) and (1, 0, 0, 1). In both cases, these extra weights gradually decayed to zero, ending with a good solution, but requiring more presentations were required to reach the solution. If there were too many extra weights, however, the learning failed to find a solution, and instead made all the weights negative. This is a local minimum with all the units turned off.

Sparse disparity. The network model of Marr and Poggio (1976) used a dense disparity map (see Fig. 2b) for the output, which prevented the network from solving transparent random-dot stereograms. The stereograms in Fig. 2a is, in fact, perceived as a square with black dots on it floating above the background, and not as a dense square. We show here that recurrent back-propagation can construct a network that computes a sparse disparity map as the output and that the same set of weights also solves transparent random-dot stereograms.

Assume that the density of black dots in stereogram is p. From the definition of compatibility and the construction of stereogram, the compatibility map will have regions of correct matches where the density, p, of black dots is high, and regions of false matches where the density, p^2, is low, as shown in Fig. 2b. Units in a high density area are more likely to be turned on than units in the low density area because the initial state of the network is set to the compatibility map and the connections within the same disparity level are excitatory. For the stereogram in Fig. 2a there is almost no overlapping high density areas along lines of sight so that, with the help of the inhibitory connections, all units in the high density areas are turned on while all the units in low density areas are turned off. This gives a dense disparity map as output.

For a sparse disparity map, only those units that represent an actual match should be turned on in the high density regions. This cannot be accomplished simply by reducing the excitatory connection strengths because the units, including those that were on initially, do not receive enough positive input and will gradually decay to zero. One way to solve the problem is to bias each unit with its compatibility value to prevent decay. This bias, by itself, tends to keep the network in its initial compatibility state.

Recurrent backpropagation was used to train a network on sparse random-dot stereograms. The inputs I_i in Eq. 1 were clamped to the compatibility map (something introduced earlier simply to help the leaning algorithm converge). Letting every unit have an excitatory connection to itself also helped somewhat. Fig. 3 shows that the performance of the trained network on sparse disparity was almost 100% for a testing stereogram not included in the training set. The weights, shown in Table 1, are

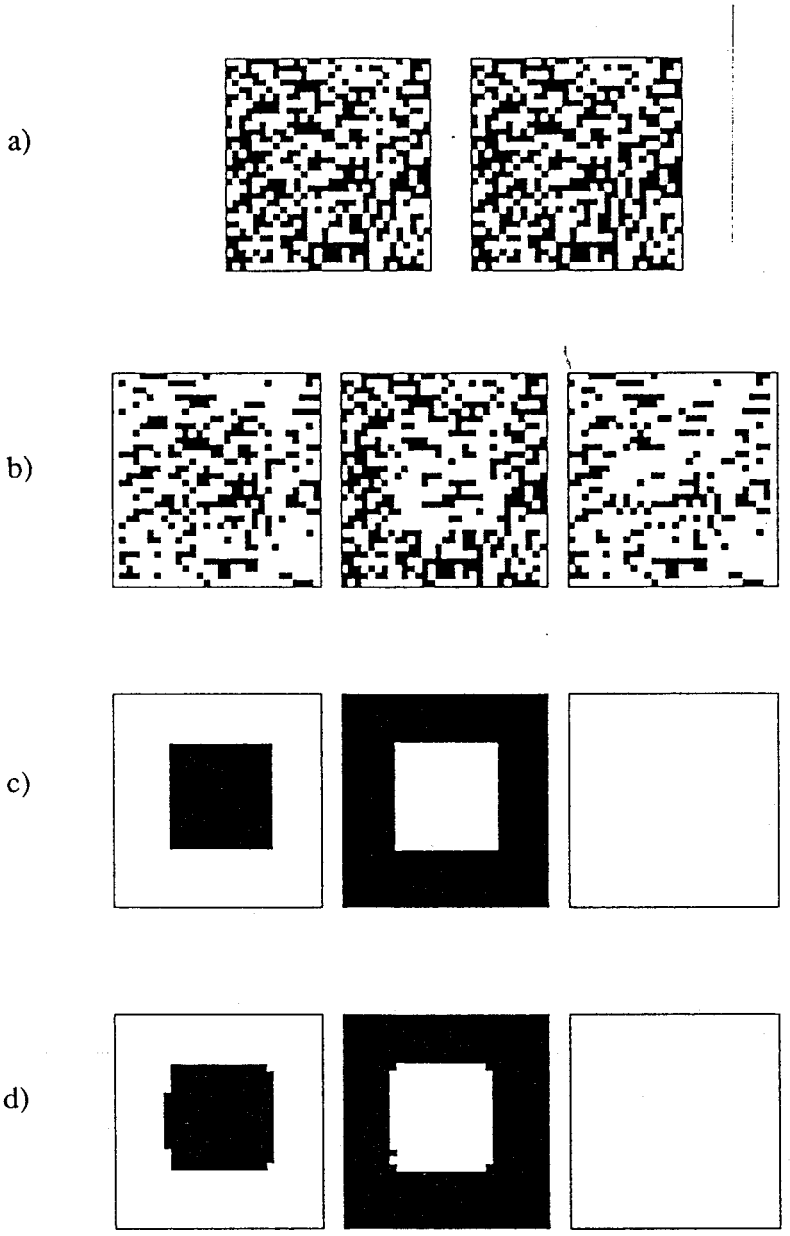

Fig. 2: (a) A random dot stereogram with black dot density 0.5, (b) Compatibility map, (c) Disparity map, (d) Performance of network. For the display of the compatibility and the disparity maps in Figs. 2 to 5, the depth "stack" shown in Fig. 1 has been unfolded by laying the depth layers side by side (the leftmost plane is the nearest).

a)

b)

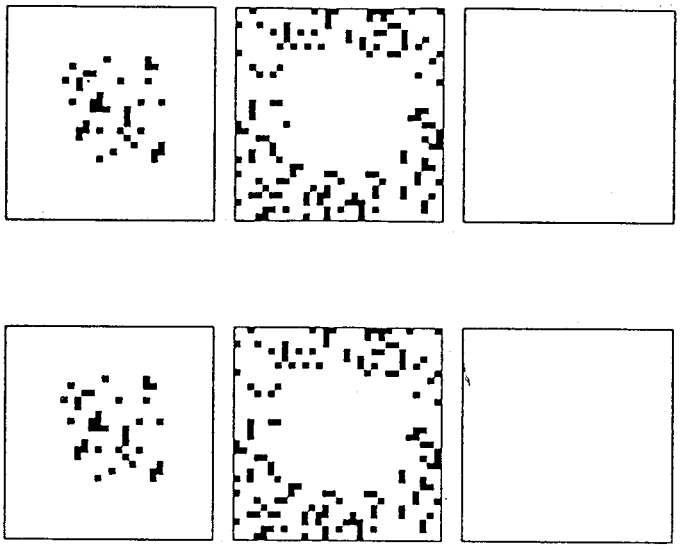

Fig. 3: (a) A sparse disparity map for a stereogram with black dot density 0.2, (b) Performance of network.

a)

b)

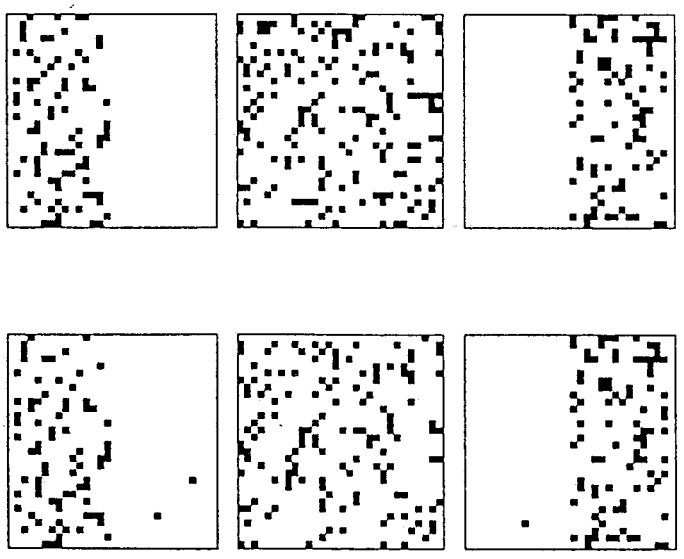

Fig. 4: (a) Target disparity map with black dot density 0.2 in the solution areas, (b) Performance of network.

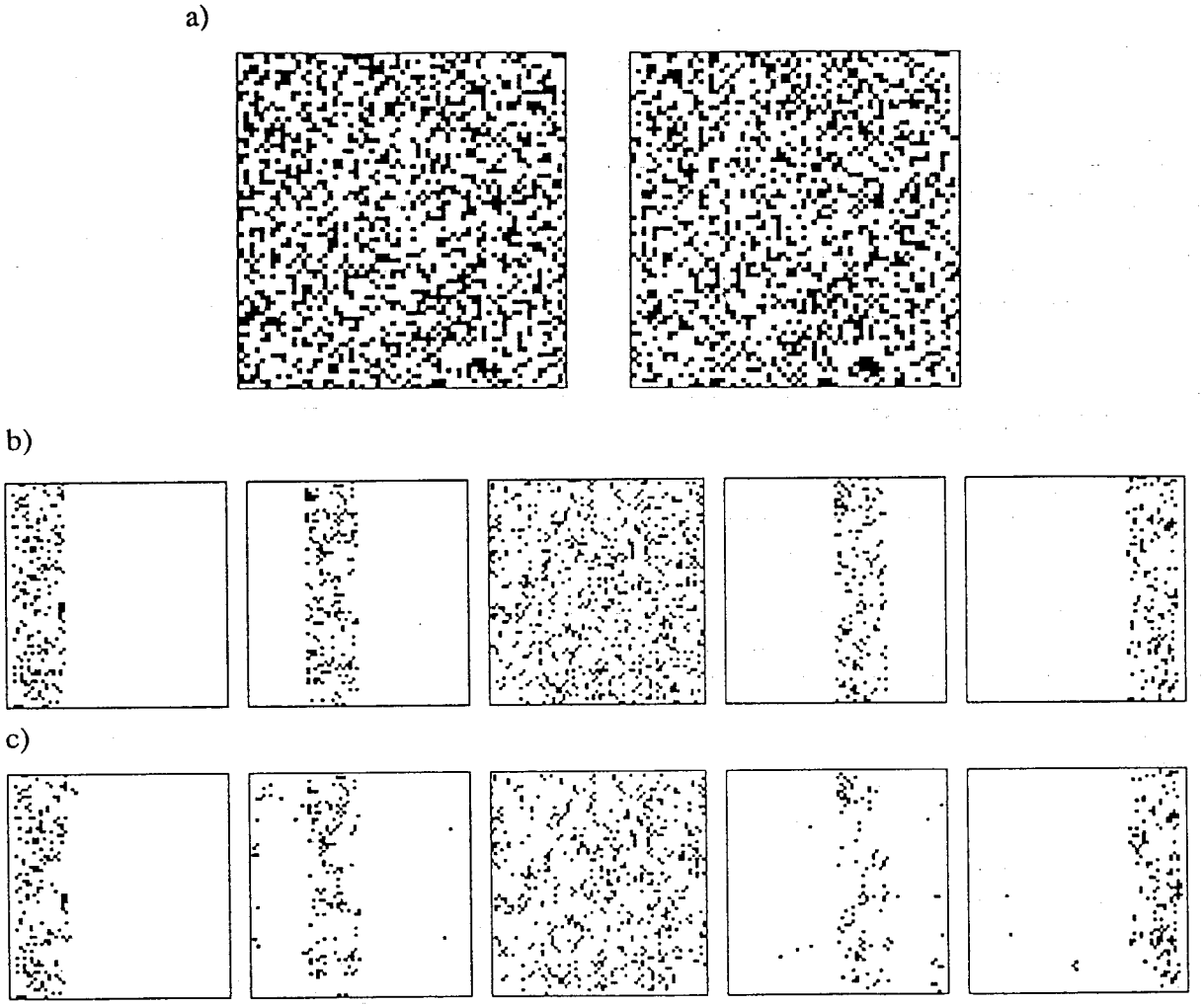

Fig. 5: (a) A 68×68 transparent random-dot stereogram, (b) Target disparity map with black dot density 0.2 in the solution areas, (c) Performance of network.

weaker that those for dense disparity maps, as expected.

Without any modification, the sparse-disparity network gives a performance of 97% on transparent random-dot stereograms. After being trained on transparent stereograms, the network performance on a testing set of stereograms is improved slightly to 97.7%, but never reached the nearly perfect performance achieved for non-transparent stereograms. An example on a testing pattern is shown in Fig. 4. It is worth noting that humans also mislocalize isolated dots in transparent stereograms.

There are two reasons why transparent random-dot stereogram are harder to solve: 1) the difference in the density of black dots between solution and non-solution areas of the compatibility map is smaller than that for the non-transparent case, and 2) according the construction method for a transparent random dot stereogram (Szeliski, 1986), only a proportion, $1/(1+2p-3p^2+p^3)$, of the black dots in the solution area of the compatibility map are correct matches assuming that the black dot density on the solution areas of the sparse disparity maps is p. There is no way to tell the false matches from correct ones based on compatibility map only. Some information was lost when a transparent stereograms was converted into a compatibility map.

We also trained a number of networks with five disparity planes and with stereograms as large as 100×100. The learning algorithm successfully solved the problem but much more computer time was needed. Fig. 5 shows the result on a transparent stereogram of a network trained on sparse disparity maps.

6. Discussion

In all networks that we successfully trained, the weights between units in the same disparity plane were excitatory and the weights between units along the two lines of sight were inhibitory. Thus, the recurrent back-propagation was able to learn from examples the continuity and uniqueness constraints proposed by Marr and Poggio (1976). This success should be tempered with the finding that a successful solution and the rate of learning were closely related to the constraints that were imposed on the connectivity and the number of independent weights in the network. For a problem where intuition is not available for a good initial architecture, the recurrent back-propagation learning may not find a good solution, or may find one only after an excessive number of training examples.

There are many problems for which lateral and feedback connections may be essential, such as the problem that we have studied, so that recurrent back-propagation should have wide applicability. However, randomly wiring up a network or simply using a fully-connected network with the hope that the learning algorithm will find a solution automatically is unlikely to succeed for a large-scale problem.

Even though the network was initialized with asymmetric connection strengths, the learning procedure developed weights that were approximately symmetrical. Thus, the final network is similar to a continuous Hopfield network (Hopfield, 1984) and the process of settling can be considered a form of content-addressable memory, with a compatibility map as a recall cue and the corresponding disparity map as the closest local minimum. For networks that can store sparse disparity maps, the total number of stored local minima is very large, much larger than the number of neurons in the network. This is possible partly because the stored vectors are quite sparse (many fewer 1's than 0's), but more importantly, all the local minima follow certain inherent rules and the information is redundant. However, the redundancy is not at all apparent by casual inspection of the stored vectors. This suggests that the capacity of a given network for correlated patterns can be enormous.

All the network models proposed so far for solving the correspondence problem for random-dot stereograms, including the ones in this paper, have used a local encoding of disparity; that is, each unit in the network corresponded to a possible match of a particular point in the left image and a particular point in the right image. The "tuning curve" (response verses disparity) for such a unit had a width that was approximately equal to the visual angle spanned by one dot in a random-dot stereogram. However, disparity sensitive neurons in the visual system are broadly tuned (Poggio & Fischer, 1977; LeVay & Voigt, 1988). Psychophysical measurements of stereo hyperacuity and stereo interpolation are consistent with these coarsely-tuned mechanisms (Lehky and Sejnowski, 1988). Coarse coding is more efficient and noise resistant than local representations (Sejnowski, 1988). Attempts to use coarse coding for disparity in our networks have not yet been successful. There may be something missing in our understanding of biological circuits that must be added to our models before we can solve the stereo problem with coarsely-tuned units like those found in the visual system.

References

Dev, P. (1975) Perception of depth surfaces in random-dot stereograms: A neural model. *Int. J. Man Machine Studies* **7**, 511-528.

Giles, C. L., Griffin, R. D., Maxwell, T. (1988) Encoding geometric invariances in higher-order neural networks. In: *Neural Information Processing Systems*, D. Z. Anderson (Ed.), American Institute of Physics: New York. pp. 301-309.

Julesz, B. (1960) Binocular Depth Perception of Computer Generated Patterns. *Bell Sys. Tech. J.* **38**, 1001-1020.

Hopfield, J. J. (1984) Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. *Proc. Nat. Acad. Sci. USA* **81**, 3088-3092.

Lehky, S. R. & Sejnowski, T. J. (1988) Model of Depth Interpolation Using a Distributed Representation of Disparity. *Investigative Ophthalmology and Visual Science Supplement* **29**, 398

LeVay, S. & Voigt, T. (1988) Ocular Dominance and Disparity Coding in Cat Visual Cortex. *Visual Neuroscience* (in press)

Marr, D. & Poggio, T. (1976) Cooperative Computation of Stereo Disparity. *Science* **194**, 283-287.

Pineda, F. J. (1987) Generalization of Back-Propagation to Recurrent Neural Networks. *Phys. Rev. Lett.* **59**, 2229-2232.

Pineda, F. J. (1988) Dynamics and Architecture in Neural Computation. *Journal of Complexity* (in press).

Poggio, G. F. & Fischer, B. (1977) Binocular Interaction and Depth Sensitivity in Striate and Prestriate Cortex of Behaving Rhesus Monkey. *J. Neurophysiol.* **40**, 1392-1405.

Poggio, G. F. & Poggio, T. (1984) The analysis of stereopsis, *Annual Reviews of Neuroscience* **7**, 379-412.

Prazdny, K. (1985) Detection of Binocular Disparities. *Biol. Cybernetics* **52**, 93-99.

Sejnowski, T. J. (1988) Neural Populations Revealed. *Nature* **332**, 308.

Stornetta, W. S. & Huberman, B. A. (1987) An Improved Three-Layer Backpropagation Algorithm. *IEEE International Conference on Neural Networks Proceedings* , II-637.

Sun, G. Z., Chen, H. H., & Lee, Y. C. (1987) Learning Stereopsis with Neural Networks. *Technical Report UMIACS-TR-87-26, CS-TR-1872, University of Maryland.*

Szeliski, R. (1986) Cooperative Algorithms for Solving Random-dot stereograms. *Technical Report CMU-CS-86-133, Carnegie-Mellon University.*

Szeliski, R. & Hinton, G. (1985) Solving Random-Dot Stereograms Using the Heat Equation. *Proceedings of the IEEE computer Society Conference on Computer Vision and Pattern Recognition, (IEEE Computer Society: Piscataway), NJ pp. 284-288.*