Learning Shape from Shading by a Multilayer Network

Guo-Qing Wei, *Member, IEEE,* and Gerd Hirzinger, *Senior Member, IEEE*

Abstract—**The multilayer feedforward network has been usually used for learning a nonlinear mapping based on a set of examples of the input–output data. In this paper, we present a novel use of the network, in which the example data are not explicitly given. We consider the problem of shape from shading in computer vision, where the input (image coordinates) and the output (surface depth) satisfy only a known differential equation. We use the feedforward network as a parametric representation of the object surface and reformulate the shape from shading problem as the minimization of an error function over the network weights. The stochastic gradient and conjugate gradient methods are used for the minimization. Boundary conditions for either surface depth or surface normals (or both) can be imposed by adjusting the same network at different levels. It is further shown that the light source direction can be estimated, based on an initial guess, by integrating the source estimation with the surface estimation. Extensions of the method to a wider class of problems are discussed. The efficiency of the method is verified by examples of both synthetic and real images.**

I. Introduction

THE problem of shape from shading is to infer the surface shape of an object based on a single intensity image. Under some simplified assumptions, the surface depth function satisfies a partial differential equation [9]. Most existing solutions, except for the local methods [6], [15], [25], are based on a variational reformulation of the problem [1], [2], [7], [10], [11], [13], [33], which usually involves a regularization term of smoothness due to the ill-posedness in the discrete solution of the problem. Horn [10] noted, however, that during the iteration the weighting factor of the smoothness term should be gradually reduced to zero, to avoid the algorithm from walking-away from the correct solution. Too rapid a reduction would cause instabilities (raggedness) in the surface. There is also a lower limit, in the presence of noise, on the value of the smoothness weight, beyond which the solution becomes again unstable [10, p. 65]. Since surface depth and surface gradients were treated as separate variables [1], [2], [10], [13], [33], the integrability constraint has to be imposed, by either using subspace projection [7] or adding a penalty term in the variational integrand [1], [10], [33]. In the latter case, the surface depth and surface gradients can be adjusted simultaneously [10], [33]. The weighting factor of the integrability term, however, should also be reduced [10] to an appropriate value during the iteration. It remains an open problem to determine the optimal values of both the smoothness weight and the integrability weight [31].

In this paper, we present a new solution of the shape from shading problem based on using a multilayer feedforward network as the parameterization of the object surface. Due to this parametric and continuous (smooth) representation, both the height (depth) and the gradients can be described by the same set of variables: the network weights. This avoids, in a natural way, the explicit use of both the smoothness and the integrability terms. The determination of shape from shading can then be converted to an ordinary extremum problem in which a cost function is to be minimized with respect to the network weights. This conversion also represents a new way of applying the feedforward network: Usually, a set of example data of the network input–output should be given for the learning of a nonlinear mapping [28], [30]; while in our case, the example data are not available; we only know that the network output satisfies a given differential equation. Thus, we are treating a more general problem than the learning of a nonlinear mapping from examples. Regarding the use of neural networks in shape from shading, we are aware of only one previous work (Lehky and Sejnowski [17]): Under the framework of learning from examples, a multilayer feedforward network was used to learn from thousands of (small) images the curvature magnitude and the curvature orientation at one surface point in the image center. With our method, we are able to learn from one image the complete surface. The learning is undertaken in the pixel level, instead of the picture level [17]. Here, learning also means parameter identification (of the object surface). It is further shown in this paper that boundary conditions about surface depth and surface gradients can be learned by the same network, too. We demonstrate further how *a priori* knowledge about the surface can be utilized to estimate the illuminant direction by integrating the source estimation with the surface estimation, resulting in improvements of both the estimates.

The paper is organized as follows. In Section II, we describe the shape estimation method, assuming a known illuminant direction. In Section III, we show how to estimate the illuminant direction by integrating with the surface estimation. We report some experimental results in Section IV. In Section V, we discuss the possible extensions of our methods to a wider class of problems like the calculus of variations and the solution of any differential equations. In Section VI we draw some conclusions.

Manuscript received October 29, 1994; revised July 19, 1995. A primary version of this paper was presented at the German-Austrian Joint Conference on Pattern Recognition, *Mustererkennung-1994,* Vienna, Austria, and received the Best Paper Award.

The authors are with the Institute of Robotics and System Dynamics, German Aerospace Research Establishment, FF-DR/RS, DLR, 82234 Oberpfaffenhofen, Germany.

Publisher Item Identifier S 1045-9227(96)04399-8.

II. Height and Shape Estimation

A. The Problem

Suppose the surface of an object is represented by $z = z(x, y)$ in a camera coordinate system $x - y - z$, with the x–y plane coinciding with the image plane, and the z axis coinciding with the optical axis of the camera. Assuming orthographic projection and the Lambertian reflectance model, the image intensity at position (x, y) of the image plane can then be computed as [9]

$$\begin{aligned} I(x, y) &= \eta \mathbf{n} \cdot \vec{L} \\ &= R(p, q) \\ &= \eta \frac{-pl_1 - ql_2 + l_3}{\sqrt{p^2 + q^2 + 1}} \end{aligned} \tag{1}$$

where η is the composite albedo, $\mathbf{n}$ is the surface normal at $[x, y, z(x, y)]$

$$\begin{aligned} \mathbf{n} &= (n_1, n_2, n_3)^{\mathrm{T}} \\ &= \left(\frac{-p}{\sqrt{p^2 + q^2 + 1}}, \frac{-q}{\sqrt{p^2 + q^2 + 1}}, \frac{1}{\sqrt{p^2 + q^2 + 1}} \right)^{\mathrm{T}} \end{aligned} \tag{2}$$

$$\begin{aligned} p &= \frac{\partial z}{\partial x} \\ q &= \frac{\partial z}{\partial y} \end{aligned} \tag{3}$$

p and q are called the surface gradients at (x, y); $\vec{L} = (l_1, l_2, l_3)$ is the illuminant direction. Equation (1) is called image irradiance equation, which is a nonlinear partial differential equation of the first order on z; $R(p, q)$ is called the reflectance map.

Suppose in the following that the parameters η and $\vec{L}$ are known. The shape from shading problem can thus be stated as the determination of $p(x, y)$, $q(x, y)$, and $z(x, y)$ from a given image $I(x, y)$. Horn [10] reformulated the problem as the minimization of

$$\begin{aligned} \int \int_D \{ & \underbrace{[(I(x, y) - R(p, q)]^2}_{\text{intensity error}} + \lambda \underbrace{(p_x^2 + p_y^2 + q_x^2 + q_y^2)}_{\text{smoothness}} \\ & + \mu \underbrace{[(z_x - p)^2 + (z_y - q)^2)]}_{\text{integrability}} \} \, dx \, dy \end{aligned} \tag{4}$$

with respect to $p(x, y)$, $q(x, y)$ and $z(x, y)$. In (4), the first term is the intensity error, the second term a smoothness measure of the surface, and the third term a penalty term of nonintegrability; the parameters λ and μ are the relative weights of the smoothness term and the integrability term, respectively. The above minimization can be performed by solving the corresponding Euler equations [3]. The finite difference method was used by Horn [10] to iteratively adjust both the height z and the gradients (p, q) on a discrete grid of points.

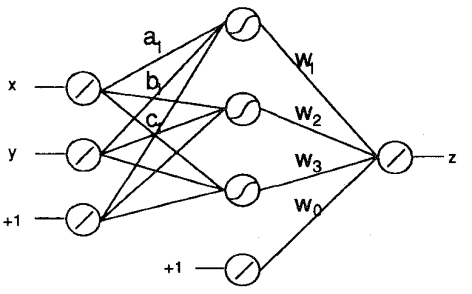

Fig. 1. A network structure of size 2×3×1.

B. The Forward Network Representation of a Surface

Instead of solving for p, q, and z only at a finite number of points, we shall recover them on the continuous domain. The first step to achieve this is to represent z, and thus p and q, parametrically. Since no *a priori* knowledge about the shape of the surface is assumed available, the representation should be flexible enough to approximate any surfaces. With this criterion, we resort to the recent discovery about the multilayer feedforward network that a three-layer (one hidden layer) forward network whose input and output units are linear and whose hidden units are semilinear is able to approximate any real-valued continuous function over a compact set (bounded closed subset) up to any accuracy [5], [8], [12]. This justifies us to model the object surface $z = z(x, y)$ in terms of a feedforward network as

$$z = \sum_{i=1}^{N} w_i \phi(a_i x + b_i y + c_i) + w_0 \tag{5}$$

where N is the number of units in the hidden layer, $\{w_i, a_i, b_i, c_i\}$ are the network weights, $\phi()$ is the semilinear function, which is the sigmoid function of the form $\phi(s) = 1/(1+e^{-s})$ in our case, with s being a dummy variable. Fig. 1 shows the structure of a network representation of a surface by using three hidden units (here the bias unit, i.e., the unit with a constant input of one, is not counted in this number.)

Although a three-layer network is theoretically able to approximate any surface, it may require an impractically large number of hidden units in the case of complex surfaces [21]. It has been found, however, that an adequate solution can be obtained with a tractable network size by using more than one hidden layers [21, p. 133]. Thus, in the following we shall not limit ourselves to the case of only one hidden layer.

C. Solving the Image Irradiance Equation

If we rewrite (5) as

$$z = z(x, y, \vec{W}) \tag{6}$$

where $\vec{W}$ represents the vector of all the weights, we can analytically compute the surface gradients through (3) as $p = z_x(x, y, \vec{W})$ and $q = z_y(x, y, \vec{W})$, here the subscripts represent the partial derivatives. Based on this parametric

form, the image irradiance equation can be rewritten as

$$I(x, y) = \eta \frac{-z_x(x, y, \vec{W})l_1 - z_y(x, y, \vec{W})l_2 + l_3}{\sqrt{z_x(x, y, \vec{W})^2 + z_y(x, y, \vec{W})^2 + 1}} \quad (7)$$

which turns out to be an equation on the network weights $\vec{W}$. Since the equation is satisfied at each image position (x, y), we get a system of equations. Due to the nonlinearity of these equations, we cannot expect the solution for $\vec{W}$ to be unique. Thus, we shall just seek one of the solutions satisfying the image irradiance equations. But as long as the surface is intrinsically unique for the given problem, the different solutions for $\vec{W}$ should result in the same surface z. The uniqueness issue was discussed by Oliensis in [23], [24].

A least squares solution for $\vec{W}$ is obtained by minimizing the total intensity error $E^{(I)}$

$$E^{(I)} = \sum_{i \in D_I} E_i^{(I)} = \sum_{i \in D_I} \{I_i - R[z_x(x_i, y_i, \vec{W}), z_y(x_i, y_i, \vec{W})]\}^2 \quad (8)$$

where D_I is the index set of all image points; (x_i, y_i) are the image coordinates at pixel i; I_i is the corresponding image intensity, and $E_i^{(I)}$ is the intensity error there. According to (8), the estimation of shape from shading has been converted to an ordinary extremum problem in which only a finite number of parameters $\vec{W}$ are to be determined. Note that it is not a functional minimization problem any more [compare with (4)], and there is no smoothness term nor integrability term in it.

To minimize (8), we recall the process in learning an input-output mapping using a multilayer network [30]. The analogy motivates us to use the stochastic gradient method [19] to learn the weights. Notice that in learning an input–output mapping by neural networks [28], [30], the surface values z_i's at coordinates $\{(x_i, y_i)\}$ should be assumed known. The purpose there is to build a network which maps the coordinates $\{(x_i, y_i)\}$ to the given values $\{z_i\} = \{z(x_i, y_i)\}$. In our problem, however, we are required to determine the z_i's, knowing only that they satisfy a differential equation. If still viewed in the mapping framework, our network is to map the (x_i, y_i)'s onto a surface which generates the given image intensities I_i's. Notice also that an explicit one-to-one correspondence between a surface value z_i and the intensity value I_i does not exist physically. In terms of the neural networks terminology, we shall call the pairs $\{(x_i, y_i; I_i), i \in D_I\} = \{\mathcal{T}_i^{(I)}\} = \mathcal{T}^{(I)}$ the intensity training patterns. According to the stochastic gradient method, we repeatedly present the training patterns to the network and modify the network weights after each presentation of a training pattern $\mathcal{T}_i^{(I)}$, in the direction opposite to the gradient of the corresponding error $E_i^{(I)}$. That is, the change of the weights after the presentation of $\mathcal{T}_i^{(I)}$ is made by

$$\Delta\vec{W} = -\beta \frac{\partial E_i^{(I)}}{\partial \vec{W}} = \beta(I_i - R_i)(R_p z_{x\vec{W}} + R_q z_{y\vec{W}}) \quad (9)$$

where β is the learning rate, R_i is the computed intensity at position i using the current weights; $R_p = \partial R(p, q)/\partial p$, $R_q = \partial R(p, q)/\partial q$, $z_{x\vec{W}} = \partial^2 z/\partial x \partial \vec{W}$, and $z_{y\vec{W}} = \partial^2 z/\partial y \partial \vec{W}$ are quantities all evaluated at position i using the current weights. Similar to [30], we can also introduce a momentum term α to avoid possible oscillations. That is, the weights will be adjusted according to

$$\vec{W}(n+1) = \vec{W}(n) - \beta \frac{\partial E_i^{(I)}}{\partial \vec{W}(n)} + \alpha \Delta \vec{W}(n-1) \quad (10)$$

where n indexes the number of presentations. Typical values for β and α in our experiment are $\beta = 0.3$ and $\alpha = 0.6$, respectively. When the iteration is getting closer to the solution, however, the value of β should be reduced. We automatically reduce it by 5% of the current value whenever the total error $E^{(I)}$ computed by accumulating the individual errors $E_i^{(I)}$'s shows any oscillations.

We shall, in the following, also call the image irradiance equation, when it is to be satisfied at a particular point, the intensity constraint at that point, for the convenience of coupling it with boundary conditions.

D. Boundary Conditions

We know that a partial differential equation possesses multiple solutions. To single out a specific individual solution, additional conditions have to be added to the differential equation. Prescribing the values of the solution on the boundary of the region of interest (the image boundary) is one way to constrain the solution. In the variational formulation of the shape from shading problem, one specifies the boundary condition by giving gradient values p and q on the image boundary [10] (more exactly, on the boundary of the region of integration in the functional). When nothing is known about the boundary gradients, the natural boundary condition has to be assumed, which, together with the Euler equations, serves as the necessary condition for the functional minimization [3, p. 208]. The natural boundary condition, however, appears as an artifact of the variational reformulation of the original shape from shading problem (which is a first-order partial differential equation), since different variational formulations of the same shape from shading problem may require different natural boundary conditions[1] [3]. In our formulation of the shape from shading problem, the imposition of a natural boundary condition is not necessary. Besides, any other *a priori* knowledge about the surface can be used to constrain the solution. These constraints can be either known depth values or known gradient values anywhere in the image (not necessarily on the image boundary). Compared with the initial values of the characteristic curves used in the solution of a first-order partial differential equation [4], [9], [10], our constraints are more flexible, since we do not require both the depth and the gradients to be given at the same image points. This property also facilitates the integration of

[1] In Zheng and Chellapa's formulation [33], a different natural boundary condition from that in Horn [10] should be derived. They simply used the same natural boundary condition as Horn [10] did.

shape from shading with stereo vision where usually only sparse depth, not necessarily also the surface normals, is available.

Despite the above discrepancies, we shall, for convenience, still call our constraints boundary conditions. In the following, we shall also specialize them by using either the term gradient constraints or the term depth constraints, according to whether the gradients or depth is given.

In the case of gradient constraints, suppose we are given the surface gradients $\{(p_j, q_j)\}$ at a set of points $\{(x_j, y_j)\}$ indexed by the set D_g. We form the total gradient error $E^{(g)}$ as

$$\begin{aligned} E^{(g)} &= \sum_{j \in D_g} E_j^{(g)} \\ &= \sum_{j \in D_g} \tfrac{1}{3}[(n_{1,j} - \hat{n}_{1,j})^2 \\ &\quad + (n_{2,j} - \hat{n}_{2,j})^2 \\ &\quad + (n_{3,j} - \hat{n}_{3,j})^2] \end{aligned} \tag{11}$$

where $E_j^{(g)}$ is the surface gradient error at position j. The vectors $\mathbf{n}_j = (n_{1,j}, n_{2,j}, n_{3,j})^T$ and $\hat{\mathbf{n}}_j = (\hat{n}_{1,j}, \hat{n}_{2,j}, \hat{n}_{3,j})^T$ are the given and the computed surface normals at position j, respectively; the latter is a function of the network weights [refer to (2), (3), and (6)].

In the case of depth constraints, suppose we are given depth values $\{z_k\}$ at a set of points $\{(x_k, y_k), k \in D_d\}$ with D_d as the index set. The total depth error is defined as

$$\begin{aligned} E^{(d)} &= \sum_{k \in D_d} E_k^{(d)} \\ &= \sum_{k \in D_d} (z_k - \hat{z}_k)^2 \end{aligned} \tag{12}$$

where $E_k^{(d)}$ is the depth error at position k, $\hat{z}_k$ is the computed depth at position k by (6).

The exertion of boundary conditions is then equivalent to minimizing (8), (11) and (12) simultaneously with respect to the network weights. If we treat both the gradient constraints and the depth constraints as a new kind of training patterns, we can then adjust the network weights, based on $\{\partial E_j^{(g)}/\partial \vec{W}\}$ and $\{\partial E_k^{(d)}/\partial \vec{W}\}$, in a similar way to that in (10) of the last section.

Due to the unknown range of the depth error, which is usually much larger than that of the intensity error (we normalize the intensity values to the range [0, 1]), we have to introduce a weighting factor w_d to the depth error, so that the weight adjustment due to the depth constraints would not overwhelm the contributions from the intensity constraints. We determine the weight by

$$w_d = \frac{\sum_{i \in D_I} E_i^{(I)}}{\sum_{k \in D_I} E_k^{(d)}}. \tag{13}$$

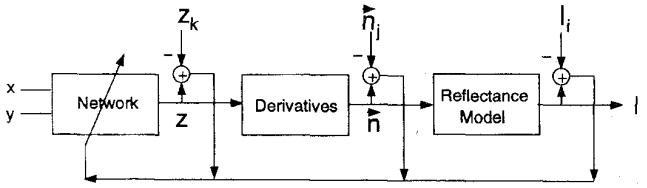

Fig. 2. A unified training framework.

The value of w_d thus determined is usually underestimated. Thus, during the training we increase w_d by 10% of its current value, whenever the total depth error shows a tendency to increase. The same weighting process can be used for the gradient constraints. But since we have used the surface normals (which are of the range [0, 1]) instead of the original gradients for the training, a constant weighting factor of two has been found to work well.

E. Computational Steps

We have seen that the network weights can be learned from three kinds of constraints: intensity constraints, gradient constraints, and depth constraints. Fig. 2 shows the complete training diagram. Thanks to the feedforward structure of the surface representation, we are able to compute the gradients $\partial E_i^{(I)}/\partial \vec{W}$, $\partial E_j^{(g)}/\partial \vec{W}$, and $\partial E_k^{(d)}/\partial \vec{W}$ in a similar (but not the same) way as in the backpropagation algorithm [30] (see Appendix A).

The following summarizes the steps in the learning of network weights in the presence of all the introduced constraints.

1) Collect all the intensity constraints, the gradient constraints and the depth constraints to form the intensity training patterns $\mathcal{T}^{(I)} = \{(x_i, y_i; I_i)\}$, the gradient training patterns $\mathcal{T}^{(g)} = \{(x_j, y_j; p_j, q_j)\}$, and the depth training patterns $\mathcal{T}^{(d)} = \{(x_k, y_k; z_k)\}$. The entire training patterns are $\mathcal{T} = \mathcal{T}^{(I)} \cup \mathcal{T}^{(g)} \cup \mathcal{T}^{(d)}$.
2) For all the intensity patterns in $\mathcal{T}^{(I)}$, repeat the network training using the stochastic gradient method, until the total intensity error drops below a prescribed value.
3) For all the training patterns in $\mathcal{T}$, repeatedly do the network training, until the total error, as the sum of the intensity error, the weighted gradient error and the weighted depth error, ceases to decrease.

The initial network weights are set by random values in $[-0.5, 0.5]$. Step 2 is to provide a good initial surface for imposing boundary conditions. For images with singular points (see the definition in the next section), most part of the surface can be established before adding boundary conditions [23], [24]. If only boundary gradient constraints are present, i.e., no boundary depth constraints, we can ignore Step 2, since we do not have to determine the weighting factor of the gradient constraints dynamically.

When the iteration gets closer to the solution, the conjugate gradient method [29] can be switched on to speed up convergence and to improve accuracy. But it is not advisable to use the conjugate gradient method at the very beginning of Steps 2 or 3, since the process may get into local minima. The

stochastic gradient method, however, has been found to rarely get stuck in local minima [30].

F. Miscellaneous

In this section, we address some issues related to input–output normalization, the multiresolution implementation, and the choice of network size.

Normalization: To make the range of the network weights independent of image size, we normalize the network input (image coordinates) to the range [0, 1] through

$$\tilde{x} = \frac{x}{N_x}$$
$$\tilde{y} = \frac{y}{N_y} \tag{14}$$

where N_x and N_y are the image size in the x and y directions, respectively. Using the normalized coordinates $\tilde{x}$ and $\tilde{y}$, the quantities $z_{\tilde{x}}$, $z_{\tilde{y}}$, $z_{\tilde{x}\vec{W}}$, and $z_{\tilde{y}\vec{W}}$ computed by (22)–(31) in Appendix A for weight adjustment should be denormalized to give

$$\begin{aligned} z_x &= \frac{\partial z}{\partial \tilde{x}} \cdot \frac{\partial \tilde{x}}{\partial x} \\ &= \frac{z_{\tilde{x}}}{N_x} \\ z_y &= \frac{\partial z}{\partial \tilde{y}} \cdot \frac{\partial \tilde{y}}{\partial y} \\ &= \frac{z_{\tilde{y}}}{N_y} \end{aligned} \tag{15}$$

$$z_{x\vec{W}} = \frac{z_{\tilde{x}\vec{W}}}{N_x}$$
$$z_{y\vec{W}} = \frac{z_{\tilde{y}\vec{W}}}{N_y}. \tag{16}$$

Notice that if we use the sigmoid instead of the linear function in the output layer, as is usual in the learning of an input-output mapping [30], there should be a normalization and denormalization of the network output z, and accordingly, a modification of (15) and (16). In that case, however, we should also know *a priori* the depth range of the true surface, so that the normalized depth matches the output range [0, 1] of the sigmoid function.

Multiresolution: The multiresolution scheme has been used in the solution of many computer vision problems for the purpose of reducing computational cost. Through subsampling of the original image, the calculation is first done in the lowest level of resolution. Then one propagates the solution to a higher level as an initial state of the computation at that level. For the shape from shading problem, one propagates the depth and gradients by an appropriate interpolation scheme [33]. With our neural network method, the interpolation is automatic due to the continuous solution. We subsample the original image as usual, but keep the values of the sampled image coordinates the same as in the original image. Then we normalize the sampled coordinates by the original image size. The training is done as usual. When we transfer from a lower resolution to a higher one, we simply add new training patterns. No change has to be made to the existing network in the transition process.

Network Size: Due to the reason stated in Section II-B, we choose to use two hidden layers instead of one. As for the size of each hidden layer, we have no analytical methods for its determination. As an alternative, we employ a heuristic approach. First, an initial size for the two hidden layers is estimated as $N_1 = N_x'/3$ and $N_2 = N_y'/3$, where N_x' and N_y' are, respectively, the subsampled (reduced) image size in the x- and y-directions, under which the image details do not lose. This estimation is empirical and subjective. We denote this network by $\mathcal{N}_0$ and will later refer to it as the basis network. After the training of $\mathcal{N}_0$, we obtain a surface $\mathcal{S}_0$ represented by $z_0(x, y, \vec{W}_0^*)$, where $\vec{W}_0^*$ is the converged weights of $\mathcal{N}_0$. If the rms residual error $e_0^{(I)}$ of the image intensity of $\mathcal{S}_0$ is less than a predefined value, $\mathcal{S}_0$ is regarded as the solution $z(x, y) = z_0(x, y, \vec{W}_0^*)$. Otherwise, we build another refining network $\mathcal{N}_1$ to improve the surface estimate as

$$z(x, y) = z_0(x, y, \vec{W}_0^*) + z_1(x, y, \vec{W}_1) \tag{17}$$

where $z_1(x, y, \vec{W}_1)$ is the surface represented by $\mathcal{N}_1$, and $\vec{W}_1$ is the weight vector of $\mathcal{N}_1$. To determine $\vec{W}_1$, we use the surface representation (17) to solve the image irradiance equation again by the stochastic gradient method. The size of $\mathcal{N}_1$ should be larger than that of $\mathcal{N}_0$ to ensure the increased approximating ability of (17). We set the size of $\mathcal{N}_1$ as being, say, 20% larger than that of $\mathcal{N}_0$. The role of $\mathcal{N}_0$ is to reduce the computational cost in the training of $\mathcal{N}_1$. Since $z_0(x, y, \vec{W}_0^*)$, and thus $z_{0x}(x, y, \vec{W}_0^*)$ and $z_{0y}(x, y, \vec{W}_0^*)$ are fixed quantities for each pixel, they can be prestored in a table. By setting the initial output-layer weights of $\mathcal{N}_1$ to zero, the starting surface in the adjustment of $\vec{W}_1$ is exactly $\mathcal{S}_0$ [refer to (5) and (17)]. During the training of $\mathcal{N}_1$, we adjust the weights only for the pixels whose intensity errors are larger than the rms intensity error of the current surface, which is $e_0^{(I)}$ for the first iteration. This kind of filtering keeps only around 20% of all the pixels in consideration. Although the sigmoid function is global, its derivatives are relatively local, which are the quantities involved in the weight adjustment for the intensity patterns. This ensures the stability of the above training scheme for $\mathcal{N}_1$. This surface refinement can be continued until the desired accuracy is reached.

III. Estimating the Illuminant Direction

In the last section, we assume the direction of the light source to be known prior to the surface estimation. In practice, this illuminant direction can only be roughly estimated before the surface estimation, since most source estimation algorithms [15], [26], [33] are based on assumptions about statistical properties of the surface. Without the use of additional constraints, we could not expect the image alone to allow for an accurate estimation of the illuminant direction. This can be understood by noting that even when the exact illuminant direction is given, we are not guaranteed to arrive at a unique surface estimate, not to say to let both the surface and the source be unknown. Notice also that there is an inherent two-way ambiguity (concave/convex) in the combination of the surface shape and the illuminant direction [2].

In this section, we shall show how to use some *a priori* knowledge about the surface, e.g., boundary conditions, together with a rough guess of the illuminant direction to arrive at an accurate estimate of the illuminant direction. We consider especially the case in which the *a priori* knowledge alone, e.g., boundary depth, is unable to provide such an accurate estimate of the illuminant direction, but could be integrated with our surface estimation procedure to achieve this. We present two methods of this kind.

The first method is based on representing the illuminant direction by the surface normal at a singular point (the brightest point). At singular points, the surface normals have the same orientation as the illuminant direction. Suppose the surface gradient at a singular image point (x_s, y_s) is (p_s, q_s). The illuminant direction can then be expressed as

$$\begin{aligned}\vec{L} &= (l_1, l_2, l_3)^T \\ &= \frac{1}{\sqrt{p_s^2 + q_s^2 + 1}} (-p_s, -q_s, 1)^T \end{aligned} \tag{18}$$

where

$$\begin{aligned} p_s &= \frac{\partial z(x_s, y_s, \vec{W})}{\partial x} \\ q_s &= \frac{\partial z(x_s, y_s, \vec{W})}{\partial y}. \end{aligned} \tag{19}$$

Inserting (18) and (19) into (7), we get an equation with the network weights as the only unknowns[2]

$$\begin{aligned} I(x, y) = \eta[&z_x(x, y, \vec{W}) z_x(x_s, y_s, \vec{W}) \\ &+ z_y(x, y, \vec{W}) z_y(x_s, y_s, \vec{W}) + 1] \Big/ \\ &[\sqrt{z_x^2(x, y, \vec{W}) + z_y^2(x, y, \vec{W}) + 1} \\ &\cdot \sqrt{z_x^2(x_s, y_s, \vec{W}) + z_y^2(x_s, y_s, \vec{W}) + 1}]. \end{aligned} \tag{20}$$

Equation (20) allows us to use the same learning method as in the last section to determine the network weights, and thus, both the object surface and the illuminant direction. We first use an initial guess of the illuminant direction to recover only the surface through minimizing the intensity error (8), in the same manner as in Step 2 of Section II-E. This establishes an initial surface. Then, a new intensity error is formed, based on (20), as the sum of the squared residuals of (20) evaluated at each image position. This new intensity error together with the errors of the boundary conditions (11) and/or (12) are then minimized with respect to the network weights, in a similar way to Step 3 of Section II-E. We call this method of source estimation the singular point method.

The second method is based on the alternative adjustment of the surface normals and the illuminant direction [2]. First, we establish an initial surface in the same way as in the singular point method. Then, the obtained surface is used to solve for the illuminant direction $\vec{L}$ in terms of (7), which can be done in linear computation. With the obtained illuminant direction, we adjust the network weights (the surface) for one cycle in terms of both the residual error of (7) and the errors of the boundary conditions. The obtained surface is then used to re-estimate the illuminant direction. This process repeats until the convergence is reached. We call this estimation process the iterative least squares (LS) method.

The two methods have been found to perform complementarily. The singular point method has a wider range and a quicker rate of convergence, while the iterative LS method is less sensitive to noise and is more general than the singular point method, since it does not depend on the identification of a singular point. Thus we propose to combine the two methods if a singular point is available. We can first use the singular point method to quickly find a rough solution and then use the iterative LS method to refine the estimate.

IV. Experiments

A. Examples

Example 1—Test of Accuracy: The first example is a synthetic sine surface [Fig. 3(a)], $z = -2 \cos(\pi\sqrt{x^2 + y^2}/32)$, with the origin at the image center. To specify the illuminant direction $\vec{L}$, we employ the two parameter description: the tilt angle τ and the slant angle γ [33]. Fig. 3(b) is a synthetic image of size 32×32 with the illuminant direction being at $\tau = 30°$ and $\gamma = 10°$. The image was generated by orthographic projection, and no noise was added. In this and the next example, we used a network of size $2 \times 10 \times 10 \times 1$. We compared our method with Horn's [10] and Leclerc and Bobick's [14] algorithms under different boundary conditions. Fig. 3(c) shows the recovered surface by our algorithm without any boundary conditions, after 1500 cycles of gradient searches plus 100 cycles of conjugate gradient searches. Fig. 3(d) shows the improved surface by adding boundary depth to Fig. 3(c), after 800 more cycles of gradient searches and further 200 cycles of conjugate gradient searches. Here one cycle means a complete pass of the training patterns. Fig. 3(e) and (f) show, respectively, the recovered surfaces, under the same boundary condition, by Horn's method after 10 380 cycles of iterations (with λ being reduced from 1.0 to 0.0001 and μ set to 0.1), and by Leclerc and Bovick's method after 1547 cycles of conjugate gradient searches. The average error, deviation and maximum error of the surface depth in pixel units, and of the surface normals in degrees for the three methods are shown in Table I. Fig. 3(g), (h), and (i) shows the respective error surfaces, magnified by a factor of four. Table I also lists the depth errors and normal errors for our method and for Horn's method under boundary gradient constraints. (The method of Leclerc and Bovick cannot consider boundary gradients.) Under boundary gradient constraints, the absolute depth error has no meaning, since the surface can only be determined up to a shift. To investigate the noise sensitivities of the methods, we added $\pm 5\%$ random noise to the image intensities. The results are also shown in Table I. From the table,[3] we can see that our method is able to reconstruct

[2] Here we assume that the albedo η has been estimated by using either the singular point or existing methods [33].

[3] For Horn's and Leclerc and Bovick's algorithms in the presence of image noise, we have picked up the best results from those with different stopping values of the smoothness weight λ, i.e., different lower bounds of λ.

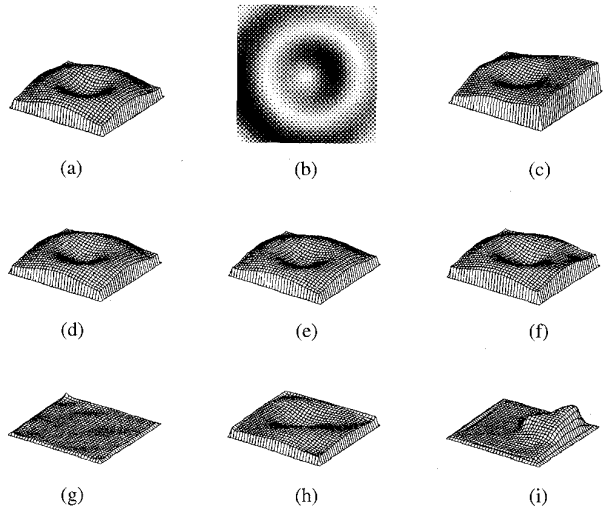

Fig. 3. Recovery of a rotated sine surface: (a) true three-dimensional (3-D) surface; (b) the input image; (c) recovered surface by our algorithm without boundary conditions; (d) recovered surface by our algorithm with boundary depth; (e) recovered surface by Horn's algorithm with boundary depth; (f) recovered surface by Leclerc and Bobick's algorithm with boundary depth; (g) our error surface; (h) Horn's error surface; and (i) Leclerc and Bobick's error surface.

TABLE I
THE ACCURACY COMPARISON OF DIFFERENT METHODS UNDER DIFFERENT BOUNDARY CONDITIONS

Methods		Boundary depth		Boundary gradient	
		depth error	norm. error (°)	depth error	norm. error (°)
		av.,dev.,max.	av.,dev.,max.	av.,dev.,max.	av.,dev.,max.
Ours	0% noise	0.038,0.032,0.272	1.2,1.3,17.3	–,0.064,–	1.3,0.9,5.9
	5% noise	0.093,0.081,0.357	3.1,1.9,15.8	–,0.125,–	2.6,1.4,6.7
Horn's	0% noise	0.196,0.264,0.700	3.5,4.5,26.2	–,0.278,–	1.1,1.3,17.1
	5% nosie	0.482,0.476,2.163	5.2,3.4,29.6	–,0.481,–	3.1,1.72,14.2
L.& B.'s	0% noise	0.249,0.372,1.472	6.5,5.9,31.7	—	—
	5% noise	0.472,0.545,2.240	8.1,7.2,35.6	—	—

a more accurate and more stable surface than the others are.

Example 2—Normals Discontinuities and the Integrability Weight: The second example is a concave spherical cap behind a plane [Fig. 4(a)]. The sphere radius is 15 pixels, and the circle radius 14 pixels at the plane of intersection. This surface contains discontinuous surface normals. Fig. 4(b) is a synthetic image of size 32 × 32 with the illuminant direction at $\tau = 150°$ and $\gamma = 10°$ (no image noise). Fig. 4(c) and (d) show, respectively, the recovered surfaces of Horn's and Leclerc and Bovick's algorithms without boundary conditions. Fig. 4(e)–(h) show, respectively, the sequence of surfaces reconstructed by our algorithm (also without boundary conditions) after 150, 300, and 800 cycles of gradient searches and 600 more cycles of conjugate gradient searches. We can see from this sequence how the surface builds itself. It begins from the area with the highest contrast and then propagates toward areas of weaker contrast. The same has been observed with Horn's and Leclerc and Bovick's algorithms. This experiment demonstrates that the surface propagation in our algorithms in the presence of normal discontinuities is more efficient than that in the other algorithms.

From Fig. 4, it can be seen that our method recovers the correct surface even without boundary conditions (for this example). With Horn's method (and also with Leclerc and Bovick's one), only when boundary conditions (either boundary normals or boundary depth) are added, can the correct solution can be recovered. The value of the integrability weight μ in Horn's method, however, should be chosen carefully. In this example, we set $\mu = 1$ and keep it unchanged during the iteration. To see the influence of the integrability weights on the solution in the presence of boundary conditions, we tried to set $\mu = 0.1$. In the case of boundary depth, we still get the correct solution; whereas in the case of boundary gradients, we get a solution which is similar to that without boundary conditions. This means that without an appropriate choice of the integrability weight (and an appropriate way of reducing it, as Horn [10] proposed to do), the solution may deviate from the correct one.

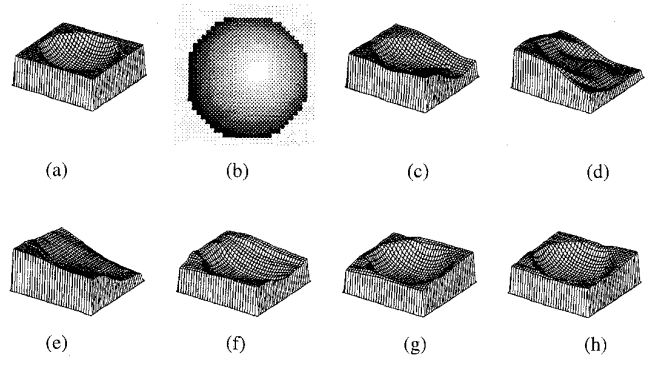

Fig. 4. Recovery of a concave spherical cap behind a plane: (a) the true 3-D surface; (b) the image; (c) recovered surface by Horn's algorithm without boundary conditions; (d) recovered surface by Leclec and Bobick's algorithm without boundary conditions; (e)–(h) sequences of surfaces recovered by our algorithm without boundary conditions.

Example 3—Real Images: Here we present results of our algorithm on two real images. The first image is the Agrippa statue, see Fig. 5(a). The image size is 128 × 128. The illuminant direction is manually estimated to be $\tau = 135°$ and $\gamma = 50°$. The network size is chosen as 2×20 × 20×1. We use three levels in the multiresolution training. No boundary conditions are assumed to be known. Shown in Fig. 5(b) is the learned image by the network. It can be seen that near the lip the learned image (and therefore the corresponding 3-D structure) is too smooth.[4] This is because of both the inappropriate choice of the network size and the relatively small size of the lip in the whole image: A small network can only learn the rough structure. To remedy this, a refining network of size 2×25 × 25×1 is added to the basis network in terms of the strategy described in Section II-F. The refining network successfully learned the fine structure. The learned image and the recovered 3-D surface are depicted in Fig. 5(c) and (d), respectively. Fig. 5(a) is another real image, the David statue. The illuminant direction and the initial network size are the same as that for the Agrippa statue. Similarly, the basis network fails to recover the details of the eye, see

[4]It is important to point out that with Horn's algorithm [10], this correspondence of smoothness between image and 3-D structure is not true if λ and μ were not appropriately chosen. This means that it is not always possible to evaluate the reconstructed 3-D shape by merely checking the reconstructed 2D image.

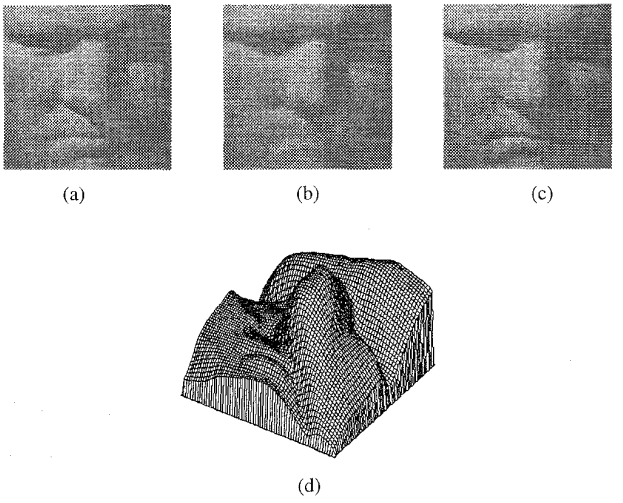

Fig. 5. Surface recovery of the Agrippa statue: (a) the image; (b) the learned image by the basis network; (c) the learned image by adding a refining network; and (d) the recovered 3-D surface.

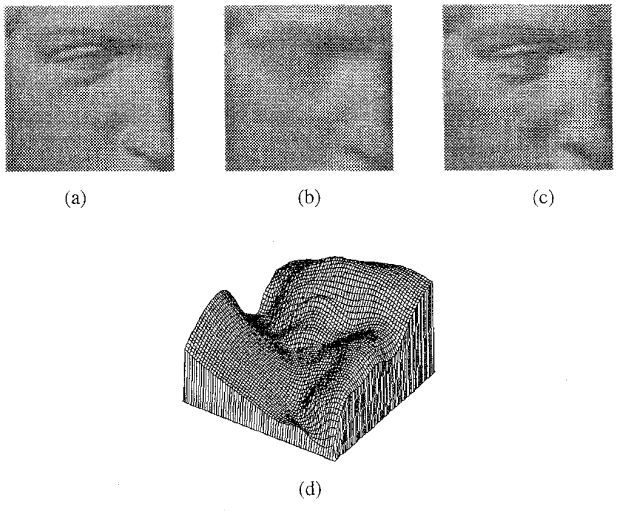

Fig. 6. Surface recovery of the David statue: (a) the image; (b) the learned image by the basis network; (c) the learned image by adding a refining network; and (d) the recovered 3-D surface.

the learned image in Fig. 6(b). Adding a refining network of size $2\times25 \times 25\times1$ successfully recovers the fine structure. Fig. 6(c) and (d) show, respectively, the learned image and the recovered 3-D shape of the David statue. It was also found that in the initialization of the refining network, setting the initial weights of each layer (except for the output layer) to random values in the same range as the corresponding layer of the basis network can speed up convergence.

Example 4—Source Estimation: We tested our source estimation methods in Section III on the synthetic image of the sine surface in Fig. 3(b) with various initial guesses of the illuminant direction. We intensively investigated the case of boundary depth. The case of boundary gradients is trivial, since the boundary normals together with the corresponding image intensities are sufficient to determine the illuminant direction in a closed form [see (1)]. We found that the singular point

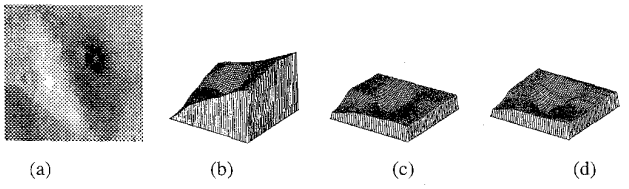

Fig. 7. Surface recovery on a moon image: (a) the image; (b) the recovered surface by our algorithm without boundary conditions, under Zheng and Chellapa's illuminant direction; (c) the improved surface by our algorithm in terms of partial boundary normals and a readjustment of the illuminant direction; and (d) the recovered surface by our algorithm without boundary conditions, under our estimate of the illuminant direction.

method converges within a deviation range of 40° (even 50°) from the correct illuminant direction, whereas the iterative LS method converges within a range of 30°. For both methods, the illuminant direction can be estimated within 0.5°.

Next, we show an example on a real image. Fig. 7(a) is a subimage of the moon. The image size is 128 × 128. Since there is no *a priori* knowledge about the illuminant direction, we use the method of Zheng and Chellapa [33] to obtain an estimate. This gives $\tau = 26°$ and $\gamma = 27°$. Fig. 7(b) shows the recovered surface by our algorithm without using any boundary conditions under this estimated illuminant direction. The recovered surface is a little bit oblique. This is due to the errors in the estimate of the illuminant direction (see [32] for an analysis). Nevertheless, we can improve both the estimates of the surface and the illuminant direction by using the knowledge that the normals at the upper and right boundaries of the image are toward the viewer. Shown in Fig. 7(c) is the improved surface by our singular point method. The surface is more consistent with the human perception. The improved estimate of the illuminant direction is at $\tau = 14.10°$ and $\gamma = 51.20°$. To see whether the errors in Fig. 7(b) are really due to the erroneous illuminant direction, we use our estimate of the illuminant direction to re-estimate the surface by dropping the boundary conditions. The resulting surface is shown in Fig. 7(d). The improvement over Fig. 7(b) can be clearly seen.

B. Several Comments

1) Multiple Solutions: In the absence of any boundary conditions, there will be at least two solutions to a shading cue, which correspond to concave/convex surfaces with opposite illuminant directions (in the image plane); for example, Fig. 4(b) may also be perceived as being convex with the illuminant direction at $\tau = -30°$. When the illuminant direction is known (or approximately known, as is in our source estimation), however, this two-way ambiguity does not exist, except that the illuminant direction coincides exactly with the viewing direction. (Almost all shape-from-shading computations assume the known illuminant direction.) Other solution-ambiguities, which are intrinsic to differential equations, should be resolved by boundary conditions; for example, the cheek of David could appear either as being toward or as being away from the viewer without violating the intensity

constraint if no boundary conditions were applied (refer to Fig. 6).

2) Local Minima: This deals with the ability of the gradient search algorithms to find the correct solution of the problem, i.e., the global minima of the cost function. When the learning rate β and the weighting factor w_d of the depth constraint are chosen appropriately (e.g., $\beta \leq 0.3$), our gradient algorithm has not been found to get stuck into local minima. Local minima might occur if β and w_d were set too large. With the conservative estimate of w_d in Section II-E, we get lower convergence rate, but not local minima. Concerning the conjugate gradient algorithm in the aspect of local minima, see Section II-E.

3) Computational Complexity: This depends on the image size and the network size. A complex surface may require a larger network than a simple surface does. As an example, the learning of the sine surface of Example 1 in the case of gradient constraints took 15 min in a SGI work station. (Horn's method took 5 min, but with reduced accuracy.) For complex surfaces like the Aggipa statue, our algorithm takes longer time in finding the fine structure, due to the globalness of the sigmoid function. This problem can be dealt with by using locally-tuned radial basis function networks [20], [28]. Drastic reduction in the computational time has been achieved in our recent experiment.

V. Discussion

In the last section, we have seen how our method successfully solves the shape from shading problem. In this section, we shall discuss its extensions to the solution of a wider class of problems. We shall also relate our solutions to existing ones, when possible.

We first consider the general problem of calculus of variations. A variational problem in one function z of two independent variables x and y can be stated as the minimization of

$$D[z] = \iint_G f(x, y, z, z_x, z_y, \cdots)\, dx\, dy \qquad (21)$$

where G is the domain of integration; $f()$ is a function of x, y, z, and the partial derivatives of z up to the kth order. By inserting the analytic expression (6) of z into (21), we go over into an ordinary (instead of functional) minimum problem for $D[z]$ as a function of the network weights $\vec{W}$. Due to the nonlinearities in the parameterization of z, we could not expect to integrate out (21) in a closed form even for a simple $f()$. Instead, we can find an approximate solution by discretizing (21) and applying any optimization techniques [29]. The accuracy of the solution depends on the size of the discretization mesh. With this approach, a class of vision problems appearing in variational forms [27] could be dealt with.

The above solution of the variational problem can be related to the direct method of Ritz [3]. The Ritz method tries to construct a sequence of minimizing functions $\{z_n\}$, each of which is a linear combination $z_n = c_1\omega_1 + c_2\omega_2 + \cdots + c_n\omega_n$ of a complete set of basis functions $\omega_1, \omega_2, \cdots$. The minimizing sequences are determined by substituting z_n for z in (21) and minimizing the obtained expression with respect to $c_1, \cdots, c_n$. The exact solution for z is obtained by the limit $z = \lim_{n\to\infty} z_n$. An approximate solution is obtained as z_n, with n being sufficiently large. Comparing our solution with the Ritz method, we find one significant difference: The basis functions in Ritz's method are fixed, and thus, "the success of the method in any particular case will depend on the proper choice of the basis function ω_i;" [3, p. 176] while in our approach, the basis functions are also to be determined, refer to (5) for the composition of ϕ; that is, the basis functions are learned from the data.

The extension of our approach to the solution of any differential equations is quite direct. It can be done similarly to the solution of the image irradiance (1).

For the differences between the neural network solution and the finite element solution [18], [22] of differential equations (see [16] for an application of the finite element method in the shape from shading problem), we point out the following: 1) The finite element method requires different programming considerations for different choices of shape functions [22] and for different boundary conditions as well as for different boundary shapes. Our neural network method, on the contrary, uses the same program for different choices of network size and for different boundary conditions and boundary shapes. 2) No conformity [22] requirement, which is concerned with the continuity of the solution between elements, is to be considered by our neural network approach.

VI. Conclusions

In this paper we have presented a new way of applying the multilayer feedforward network. For the first time, we use this network as a general parameterization tool to solve the partial differential equation in shape from shading. The method, in contrary to the conventional solution by regularization techniques, employs no explicit smoothness term nor integrability term. An attractive feature of such a solution is that *a priori* knowledge of different kinds (e.g., depth and normals) can be integrated in an unified computational scheme. The use of the *a priori* knowledge also aids the estimation of the illuminant direction. Experiments have verified the efficiency of our method in both the surface and the source estimations. Extensions of our approach to a wider class of problems are promising.

Appendix A
Calculating the Stochastic Gradient

To compute $\partial E_i^{(I)}/\partial \vec{W}$, $\partial E_j^{(g)}/\partial \vec{W}$, and $\partial E_k^{(d)}/\partial \vec{W}$, we need to compute $\partial z/\partial \vec{W}$, $\partial z_x/\partial \vec{W}$, and $\partial z_y/\partial \vec{W}$. Similar to the backpropagation algorithm [30], these quantities can be computed recursively. For the convenience of formulations, we shall make some uniform treatment of the variables as follows. First, we let $x_1 = x$ and $x_2 = y$. Suppose we have a network of L layers. We call the input layer 1, the first hidden layer 2, and so forth. The output layer is then layer L. Suppose the lth layer has n_l units, not including the bias unit. We denote

the input of the nth unit in layer l by $S_n^{(l)}$, the output of that unit by $O_n^{(l)}$. According to Fig. 1, we have $O_1^{(1)} = S_1^{(1)} = x_1$, $O_2^{(1)} = S_2^{(1)} = x_2$, and $O_1^{(L)} = z$. If we let the bias unit in layer l be the $(n_l + 1)$th unit, then $O_{n_l+1}^{(l)} = 1$.

Denote by $w_{ji}^{(l)}$ the weight connecting the ith unit in layer l and the jth unit in layer $(l+1)$. We can compute z, z_x, z_y, $z_{x\vec{W}}$, and $z_{x\vec{W}}$ using the following formulas.

a) Forward propagation to compute z, z_x, z_y: Initialize

$$\frac{\partial O_i^{(1)}}{\partial x_k} = \delta_{ij}, \quad i = 1, 2; k = 1, 2.$$

where δ_{ij} is the Kronecker symbol.

For $l = 2, 3, \cdots, L$

$j = 1, 2, \cdots, n_l$

{

$$S_j^{(l)} = \sum_{i=1}^{n_{l-1}+1} w_{ji}^{(l-1)} O_i^{(l-1)} \tag{22}$$

$$O_j^{(l)} = \phi[S_j^{(l)}] \tag{23}$$

$$\frac{S_j^{(l)}}{\partial x_k} = \sum_{i=1}^{n_l} w_{ji}^{(l-1)} \frac{\partial O_i^{(l-1)}}{\partial x_k}; \quad k = 1, 2. \tag{24}$$

$$\frac{\partial O_j^{(l)}}{\partial x_k} = \phi'[S_j^{(l)}] \frac{\partial S_j^{(l)}}{\partial x_k}; \quad k = 1, 2. \tag{25}$$

}

Then

$$\begin{aligned} z &= O_1^{(L)} \\ z_x &= \frac{\partial O_1^{(L)}}{\partial x_1} \\ z_y &= \frac{\partial O_1^{(L)}}{\partial y_2}. \end{aligned} \tag{26}$$

b) Backward propagation to compute $z_{x\vec{W}}$ and $z_{y\vec{W}}$: Initialize

$$\begin{aligned} \frac{\partial z}{\partial S_1^{(L)}} &= 1 \\ \frac{\partial^2 z}{\partial x_k \partial S_1^{(L)}} &= 0; \quad k = 1, 2. \end{aligned} \tag{27}$$

For $l = L, L-1, \cdots, 2$

$1, 2, j = \cdots, n_l$

$i = 1, 2, \cdots, n_{l-1} + 1$

{

$$\begin{aligned} \frac{\partial z}{\partial w_{ji}^{(l-1)}} &= \frac{\partial z}{\partial S_j^{(l)}} \cdot O_i^{(l-1)} \\ \frac{\partial^2 z}{\partial x_k \partial w_{ji}^{(l-1)}} &= \frac{\partial^2 z}{\partial x_k \partial S_j^{(l)}} \cdot O_i^{(l-1)} \end{aligned} \tag{28}$$

$$+ \frac{\partial z}{\partial S_j^{(l)}} \frac{\partial O_i^{(l-1)}}{\partial x_k} \tag{29}$$

$$\frac{\partial z}{\partial S_i^{(l-1)}} = \left[\sum_{n=1}^{n_l} \frac{\partial z}{\partial S_n^{(l)}} w_{ni}^{(l-1)}\right] \phi'[S_i^{(l-1)}] \tag{30}$$

$$\begin{aligned} \frac{\partial^2 z}{\partial x_k \partial S_i^{(l-1)}} &= \left[\sum_{n=1}^{n_l} \frac{\partial^2 z}{\partial x_k \partial S_n^{(l)}} w_{ni}^{(l-1)}\right] \phi'[S_i^{(l-1)}] \\ &\quad + \left[\sum_{n=1}^{n_l} \frac{\partial z}{\partial S_n^{(l)}} w_{ni}^{(l-1)}\right] \phi''[S_i^{(l-1)}] \\ &\quad \cdot \frac{\partial S_i^{(l-1)}}{\partial x_k}; \quad k = 1, 2 \end{aligned} \tag{31}$$

}

where ϕ' and ϕ'' are the first- and second-order partial derivatives of $\phi(S)$ with respect to S, respectively. Equations (30) and (31) backpropagate the first- and second-order differential properties of z.

ACKNOWLEDGMENT

The authors are very thankful to Dr. Q. Zheng at the University of Maryland, P. S. Tsai at the University of Central Florida, and Dr. K. M. Lee, previously at the University of Southern California, for providing test images. The comments of the anonymous reviewers are gratefully acknowledged.

REFERENCES

[1] M. J. Brooks and B. K. P. Horn, "The variational approach to shape from shading," *Comp. Vis., Graph. Image Processing,* vol. 33, pp. 174–208, 1986.

[2] ——, "Shape and source from shading," in *Proc. Int. Joint Conf. Artifcial Intell.,* Los Angeles, CA, 1985, pp. 932–936.

[3] R. Courant and D. Hilbert, *Methods of Mathematical Physics,* vol. 1. New York: Interscience, 1953.

[4] ——, *Methods of Mathematical Physics,* vol. II. New York: Interscience, 1962.

[5] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Cont., Sig., Syst.,* vol. 2, pp. 303–314, 1989.

[6] F. P. Ferrie and M. D. Levine, "Where and why local shading analysis works," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 11, pp. 198–206, 1989.

[7] R. T. Frankot and R. Chellapa, "A method for enforcing integrability in shape from shading algorithms," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 10, pp. 439–451, 1988.

[8] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks,* vol. 2, pp. 183–192, 1989.

[9] B. K. P. Horn, "Obtaining shape from shading information," in *The Psychology of Computer Vision,* P. H. Winston, Ed. New York: McGraw-Hill, 1975, pp. 115–155.

[10] ——, "Height and gradient from shading," *Int. J. Comput. Vision,* vol. 5, no. 1, pp. 37–75, 1990.

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

[12] F. Hornik, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, pp. 359–363, 1989.

[13] K. Ikeuchi and B. K. P. Horn, "Numerical shape from shading and occluding boundaries," *Artificial Intell.,* vol. 17, pp. 141–184, 1981.

[14] Y. G. Leclerc and A. F. Bobick, "The direct computation of height from shading," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition* Honolulu, HI, 1991, pp. 552–558.

[15] C.-H. Lee and A. Rosenfeld, "Improved methods of estimating shape from shading using the light source coordinate system," *Artificial Intell.,* vol. 26, pp. 125–143, 1985.
[16] K. M. Lee and C. C. J. Kuo, "Shape from shading with a linear triangular element surface model," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 15, no. 8, pp. 815–822, 1993.
[17] Lehky and Sejnowski, "Network model of shape-from-shading: Neural function arises from both receptive and projective fields," *Nature,* vol. 333, pp. 452–454, June 1988.
[18] R. K. Livesley, *Finite Elements: An Introduction for Engineers.* Cambridge, U.K.: Cambridge Univ. Press, 1983.
[19] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification.* Cambridge, MA: MIT Press, 1983.
[20] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computa.,* vol. 1, pp. 281–294, 1989.
[21] R. Hecht-Nielsen, *Neurocomputing.* Reading. MA: Addison-Wesley, 1990.
[22] D. H. Norrie and G. de Vries, *An Introduction to Finite Element Analysis.* New York: Academic, 1978.
[23] J. Oliensis, "Uniqueness in shape from shading," *Int. J. Comp. Vision,* vol. 6, no. 2, pp. 75–104, 1991.
[24] ——, "Shape from shading as a partially well-constrained problem," *Comput. Vision, Graph., Image Processing,* vol. 54, no. 2, pp. 163–183, 1991.
[25] A. P. Pentland, "Local shading analysis," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. PAMI-6, pp. 170–187, 1984.
[26] ——, "Finding the illuminant direction," *J. Opt. Soc. Amer. A,* vol. 72, no. 4, pp. 448–455, 1982.
[27] T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature,* vol. 317, pp. 314–319, 1985.
[28] T. Poggio and F. Girosi, "Networks for approximation and learning," in *Proc. IEEE,* vol. 78, no. 9, 1990, pp. 1481–1497.
[29] W. H. Press *et al., Numerical Recipes in C: The Art of Scientific Computing,* 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.
[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* D. E. Rumelhart and J. L. McClelland, Eds., vol. I. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[31] R. Szeliski, "Fast shape from shading," *Comput Vision, Graph., Image Processing: Image Understanding,* vol. 53, no. 2, pp. 125–153, 1991.
[32] G. Q. Wei and G. Hirzinger, "Learning shape from shading by neural networks," Tech. Rep., Feb. 1994.
[33] Q. Zheng and R. Chellapa, "Estimation of illuminant direction, albedo, and shape from shading," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 13, pp. 680–702, 1991.

Guo-Qing Wei (S'88–M'90) was born in October 1962 in Jiang-su Province, P.R. China. He received the B.S., M.S., and Ph.D degrees in electrical engineering in 1983, 1986, and 1989, respectively, all from the Southeast University (formerly the Nanjing Institute of Technology), P.R. China.

From 1989 to 1991, he was an Assistant Professor at the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. Since 1991, he has been a Senior Research Scientist at the Institute of Robotics and System Dynamics, German Aerospace Research Establishment, Oberphaffenhofen. His areas of interest include computer vision and neural networks.

Dr. Wei received the Pattern Recognition Prize (a Best Paper Award) from the German Pattern Recognition Committee (DAGM) and the Austrian Pattern Recognition Committee (ÖAGM) in 1994. He was a member of the New York Academy of Sciences, and is a member of the IEEE Computer Society.

Gerd Hirzinger (M'89–SM'90) received the Dipl.-Ing. degree and the doctor's degree from the Technical University of Munich, in 1969 and 1974, respectively.

In 1969, he joined DLR (the German Aerospace research establishment) where he first worked on fast digital control systems. In 1976, he became head of the Automation and Robotics Laboratory of DLR and got several awards for innovative technology transfer from robotics research to applications. In 1991, hereceived a Professorship from the Technical University of Munich. Since 1992, he has been Director of DLR's Institute for "robotics and system dynamics." He has published approximately 100 papers in robotics, mainly on robot sensing, sensory feedback, mechatronics, man-machine interfaces, telerobotics, and space robotics. He was Prime Investigator of the space robot technology experiment ROTEX, the first real robot in space, which flew onboard shuttle Columbia in April 1993.

Dr. Hirzinger is Vice-program Chairman of the IEEE Conference on Robotics and Automation 1994 and 1995 and program chairman of IROS (Intelligent Robot Systems Conference) 1994. In a large number of other international robot conferences he was program committee member or invited plenary speaker. For many years he has been chairman of the German council on robot control and administrative committee member of the IEEE Society on Robotics and Automation. In 1994, he received the Joseph–Engelberger-Award for his achievements in the robotic science and the Leibniz-Award, the highest scientific award in Germany.