On Learning Shapes from Shades

Subhajit Sanyal   Mayank Bansal   Subhashis Banerjee   Prem Kalra
Department of Computer Sc. & Engg.
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, INDIA
{subhajit,mayban,suban,pkalra}@cse.iitd.ernet.in

Abstract

Shape from Shading (SFS) is one of the most extensively studied problems in Computer Vision. However, most of the approaches only deal with Lambertian or other specific shading models, and are hence limited in their applicability to real images. In this contribution we propose a general unified framework in which it is possible to incorporate different illumination and shading models. We effectively deal with the usual multiple local minima problem of the SFS domain through some minimal user interactions. Features such as pyramidal refinement, parallelizability of solution evolution, global smoothness of solution give our framework a definite edge over most other existing SFS schemes. Results on real images demonstrate the efficacy of our approach.

1. Introduction

Shape from shading, or the recovery of 3D shape from the shading information in an image, has been an area of active interest in the computer vision community. Starting from the initial works of Horn [5] to the recent works of *Prados et al.* [9], there has been a whole gamut of techniques and algorithms. The survey [16] contains a detailed study of several techniques that have been suggested in the last three decades up to 1999.

Although there is an abundance of methods and algorithms, majority of them restrict themselves with strong assumptions on surface reflectance properties or on the imaging geometry. Most of the existing methods assume the surface to be *Lambertian* and assume orthographic projection as the imaging model. Such simplistic assumptions for the surface reflectance model fail to explain satisfactorily the shading in the images of real world surfaces. Effects such as surface specularities are either by-passed by some preprocessing of the image or require multiple images to be handled effectively. *Prados et al.,* in their recent works[9], allow for perspective imaging models. However, the shading model assumes *Lambertian* surfaces which restrict the domain of applicability of such techniques.

In this paper we present a general framework for shape from shading which supports different shading and illumination models under various imaging constructions. Going beyond the *Lambertian* assumption, the complete generality of our shading and illumination model allows us to capture specularities, shadows, etc.

For surface reconstruction we adopt a global minimization approach, where we minimize a variational error metric. A major objection to a global minimization approach is that the recovered surface tends to disregard local features in an attempt to impart a global smoothness. In contrast local approaches [8, 7] aimed to capture local features closely often require *a priori* height information. Moreover, the results in such approaches are often fraught with considerable noise. In our scheme local features can be captured in two ways. One can either adopt a pyramidal refinement scheme where one first obtains a smoother general surface, and then proceeds to work on the details. The other option is to directly specify more control points in regions with more detail, as described later on. Other existing methods which follow propagation approaches [1] require specification of height values from singular points of the surface etc., in the initialization phase. Our method needs no such specific height values which are usually difficult to obtain.

To minimize the error we employ a stochastic gradient descent algorithm which utilizes the error as a reinforcement to *learn* the solution. In our method the complete generality of the shading model makes such a stochastic algorithm favorable over common gradient descent algorithms as the gradient information, required in such algorithms, has to be estimated non-analytically and would be highly noisy and erroneous. The algorithm provably converges to a local minimum. Our method fits an interpolating surface through a sparse set of sample points in every iteration of the algorithm. To ensure that the attained minimum is the one corresponding to the correct shape user intervention is used (if required and infrequent in practice) to modify the shape and pull it out of an incorrect local minima. Another attractive feature of our method is the enormous parallelizability of the optimization algorithm, which makes the algorithm

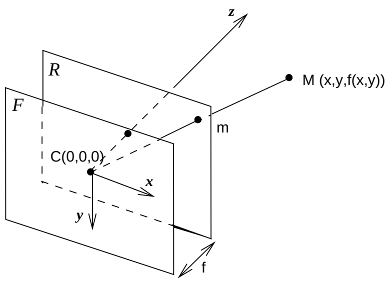


Figure 1: The Camera Coordinate System

feasible for addressing real world situations.

The reuslts obtained, both for synthetic and real images, under different illumination models clearly illustrate the robustness of our algorithm.

The remainder of the paper is organized as follows. Section 2 poses the SFS problem in a variational setting which naturally incorporates regularization in the error metric. Section 3 discusses some of the illumination models that can be handled in the generic framework of Section 2. The basic notions of the stochastic reinforcement learning scheme, utilized to minimize the error, is described in Section 4. The algorithm is outlined in Section 5. The scheme of hierarchical reconstruction for dealing with complex surfaces and other finer aspects is presented in Section 6. Section 7 presents the results and Section 8 concludes the contribution.

2. Problem Formulation

We represent the object to be modeled as a piecewise continuous function, $f(x,y)$ [15] which represents the *depth map*. The objective is to determine the value of the function f at some fixed (chosen by the user) locations (x,y) such that a continuous smooth surface fitted through these set of 3D points represents the actual surface as closely as possible. We utilize the shading information from the given image as a measure of the degree of match between the actual and the reconstructed surfaces. The existence and uniqueness of solution being an important aspect in the SFS problem, we propose an interactive setting (with learning automata as the underlying basis) where the user can pull the system out of local minima and guide it to an acceptable solution.

In what follows, we describe a *variational* formulation (similar to the formulation for stereo in [2]) for the shape from shading problem.

We will assume here that the camera performs a perspective projection of the 3D world on the retinal plane as shown in Fig.1. The optical center, noted C in the figure, is the center of projection and also the origin of our world

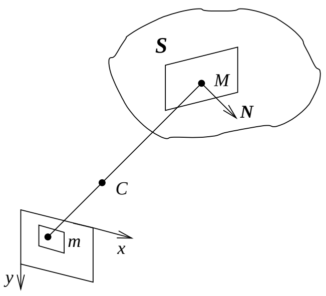


Figure 2: In the image, a square window around the projected pixel m is considered

coordinate system. The z-axis is directed towards the object and is perpendicular to the retinal plane. The image of the 3D point M is the pixel m on the retinal plane $\mathcal{R}$. Such a transformation can be described by a perspective projection matrix $\mathbf{P}$. Assuming that the focal length f of the camera is known, $\mathbf{P}$ takes the simple form:

$$\mathbf{P} = \mathbf{K}\left[\begin{array}{cc} \mathbf{I_3} & \mathbf{0} \end{array}\right]$$

where $\mathbf{K}$ is the camera internal matrix [4].

We denote the 3D surface by $\mathcal{S}$ which can be conveniently represented as $(x,y,f(x,y))$ in the coordinate system of the camera focal plane, described above. Let M be a point on this surface and $\mathbf{N}$ be the surface normal at M as depicted in Fig.2. Let $\mathbf{l}$ denote the (known) light source location in the coordinate system described above. The intensity value in the image at pixel m is denoted by $I(m)$ or $I(u,v)$. Let $\mathcal{L}(\cdot)$ be the *illumination kernel* corresponding to the chosen shading model (discussed in detail in the next section).

We now define our *variational error metric*. The surface $\mathcal{S}$ represents a photometrically consistent solution [6] when the following mean square error is minimized for each point $M = (x,y,f(x,y))^T$ on the surface:

$$E(\mathbf{S},\mathbf{N},x,y) = \frac{1}{4pq}\int_{-p}^{p}\int_{-q}^{q}(I(m+m') - \mathcal{L}(\mathbf{S},\mathbf{N},\mathbf{L},x,y))^2 \,\mathrm{d}m' \tag{1}$$

where,

$$\mathbf{L} = \mathbf{l} - (x,y,f(x,y))^T$$

is the local light source direction at the point M

$$\begin{array}{rrcl} & \mathbf{N} & = & (f_x, f_y, -1)^T \\ \text{and,} & m & \equiv & \mathbf{P}.(x,y,f(x,y))^T \end{array}$$

The total error can then be written as:

$$C(\mathbf{S},\mathbf{N}) = \iint E(\mathbf{S},\mathbf{N},x,y)\,\mathrm{d}x\,\mathrm{d}y \qquad (2)$$

The integration in (1) is carried over a small window of size $4pq$ in the image assuming that the corresponding (small) patch on the surface $\mathcal{S}$ has the same normal **N** and the same local light source direction **L**. This approach automatically imposes a smoothness constraint on the surface and regularizes the variational problem. The global smoothness of the solution can be further enforced by a suitable choice of the surface fit, as indicated in Section 5. The integral in (2) is carried over the entire surface $\mathcal{S}$ to obtain the total cost metric.

The imaging model described in (1) is a general one and can accommodate both perspective and orthographic projection models. Moreover, there is no restriction on the location of the light source which can be finite or infinite, on-axis or off-axis.

3. Illumination Models

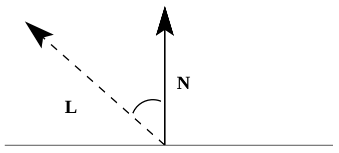
The *illumination kernel* $\mathcal{L}(\cdot)$ in the above formulation allows for different illumination models in the shading computation. Depending on the kind of image, the user may judge for instance, whether the object is specular or the *Lambertian* model is sufficient. A suitable model may then be chosen and substituted for the kernel $\mathcal{L}(\cdot)$. This general framework extends the applicability of the SFS paradigm to a larger set of *real images*.

Depending on their physical properties, surfaces can be categorized as pure *Lambertian*, pure specular, hybrid, or more sophisticated surfaces. Next, we describe the reflectance models and discuss their properties related to shape from shading.

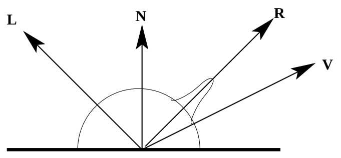
3.1 *Lambertian* and Specular Reflectance Models

Lambertian surfaces are surfaces having only diffuse reflectance, i.e., surfaces which reflect light in all directions. The brightness of a *Lambertian* surface is proportional to the energy of the incident light. The amount of light energy falling on a surface element is proportional to the area of the surface element as seen from the light source position (the foreshortened area). The foreshortened area is a cosine function of the angle between the surface orientation and the light source direction (see Fig.3(a)). Assuming that the light source has a unit strength, the *Lambertian* surface can be modeled as the product of the diffuse reflectance of the surface K_d, and the foreshortened area $\cos\theta_i$ as follows:

$$\mathcal{L}(\mathbf{S},\mathbf{N},\mathbf{L},x,y) = K_d \cos\theta_i = K_d \frac{\mathbf{N}.\mathbf{L}}{|\,\mathbf{N}\,||\,\mathbf{L}\,|} \qquad (3)$$



(a) *Lambertian Reflectance Model*



(b) *Specular Reflectance Model*

Figure 3: Reflectance Models

where θ_i is the angle between the surface normal **N** and the local light source direction **L**.

Specularity only occurs when the incident angle of the light source is equal to the reflected angle. It is formed by two components: the specular spike and the specular lobe. The specular spike is zero in all directions except for a very narrow range around the direction of specular reflection. The specular lobe spreads around the direction of specular reflection. The model developed by Phong[3] represents the specular component of reflection as powers of the cosine of the angle between the perfect specular direction **R** and the viewing direction **V** (see Fig.3(b)). This model is capable of predicting specularities which extend beyond a single point; however, the parameters have no physical meaning. In this case the *illumination kernel* becomes

$$\mathcal{L}(\mathbf{S},\mathbf{N},\mathbf{L},x,y) = K_s \cos^n\theta_r = K_s \left(\frac{\mathbf{R}.\mathbf{V}}{|\,\mathbf{R}\,||\,\mathbf{V}\,|}\right)^n \qquad (4)$$

where,

$$\begin{aligned} \mathbf{R} &= 2(\mathbf{L}.\mathbf{N})\mathbf{N} - \mathbf{L} \\ \text{and,}\quad \mathbf{V} &= (-x,-y,-f(x,y))^T \end{aligned}$$

K_s represents the specular reflectance of the surface. The specular exponent is denoted by n and measures the spread of the specular lobe around **R**.

Most surfaces in the real world are neither purely *Lambertian* nor purely specular, they are a combination of both. That is, they are hybrid surfaces and can be represented by:

$$\mathcal{L}(\mathbf{S},\mathbf{N},\mathbf{L},x,y) = K_d \frac{\mathbf{N}.\mathbf{L}}{|\,\mathbf{N}\,||\,\mathbf{L}\,|} + K_s \left(\frac{\mathbf{R}.\mathbf{V}}{|\,\mathbf{R}\,||\,\mathbf{V}\,|}\right)^n \qquad (5)$$

3.2 Handling Shadows

The proposed algorithm allows effective handling of *shadows* by borrowing the scheme from a basic *Raytracer*[3] kernel. Consider a 3D surface $\mathcal{S}$ and a point $M = (x, y, z)$ on it. This point is considered to be shadowed if the ray joining the point to the light source (assumed point source) is obstructed by another primitive. In our algorithm, we test every point on the surface for shadows. In case the point is under shadow, then the value of the *illumination kernel* $\mathcal{L}(\cdot)$ at that point is assumed to be zero (and must match the intensity in the image if the point is really under shadow). Otherwise, the value is computed in the usual manner.

Next, we give details of the important aspects of the learning automata framework.

4. Continuous Action Learning Automata (CALA)

Learning automata are adaptive decision making units that learn to choose an optimal action from a set actions by interacting with a random environment. A continuous action learning automaton is an automaton with a continuous space of actions, i.e. the set of actions of a CALA is the real line $\mathcal{R}$. Networks of CALA has been effectively used for optimization of multivariable objective functions. In the optimization of multivariable functions the network of CALA can be identified to be in a *game* setting where each CALA tries to maximize its profit. Details of Continuous Action Learning Automaton schemes can be found in [10],[12]. For completeness we present a brief overview.

To describe the operations of a network of CALA, we consider a situation where we want to maximize a function $f : \mathcal{R}^N \to \mathcal{R}$, where we call the arguments of f as parameters. To achieve this objective we consider a stochastic game of N Continuous Action Learning Automata [12], [11] corresponding to the N parameters of the function f. Thus each parameter of the optimal set for the function f is to be decided from the action sets of the N CALA in the network. Each round of the game, corresponding to each iteration of the algorithm, consists of each of the players choosing a specific action from their respective action sets. The CALA then receive payoffs for their selected actions from the environment. The payoff is the function value evaluated at the specific values of the actions selected by the N CALA. In game playing terminology such a situation is called a *common payoff* game.

Each CALA C_j in the network is associated with a normal action probability distribution $N(\mu_j(k), \sigma_j(k))$, where k is the time-step corresponding to the k^{th} iteration. This probability distribution, usually referred to as the action probability distribution, dictates the action chosen by the CALA at time-step k. The CALA updates the mean $\mu(k)$ and standard deviation $\sigma(k)$ based on the feedback it receives from the environment.

Let $\langle \mathbf{x}(k) \rangle$ denote the tuple of actions chosen by the N CALA at time-step k. In response to the tuple of actions chosen, the environment provides the stochastic reinforcement β to each of the CALA. We assume that β takes values in $[0, 1]$. Let

$$F(\mathbf{x}) = E[\beta | j^{th} \text{ CALA chooses } x_j \in \mathcal{R}] \tag{6}$$

An action tuple $\langle \mathbf{x}^* \rangle$ is said to be *optimal* if

$$F(\mathbf{x}^*) \geq F(\mathbf{x}) \ \forall \mathbf{x} \in \mathcal{B}^N(\mathbf{x}^*, \varepsilon) \tag{7}$$

Here $\mathcal{B}^N(\mathbf{x}^*, \varepsilon)$ is an ε-ball in $\mathcal{R}^N$ centered at $\mathbf{x}^*$. Thus if any one of the players chooses a different strategy while the others keep their strategy unchanged the payoff decreases and the optimal action set is a Nash equilibrium. In other words $\mathbf{x}^*$ is a local maximum of $F(\mathbf{x})$.

The algorithm for a network of CALA in a game playing situation proceeds as follows. At the k^{th} time-step, the environment is presented with the action tuples $\langle x_1(k), x_2(k), x_3(k), \ldots, x_j(k), \ldots, x_N(k) \rangle$ and $\langle \mu_1(k), \mu_2(k), \mu_3(k), \ldots, \mu_j(k), \ldots, \mu_N(k) \rangle$. The environment computes the responses $\beta_{\langle \mathbf{x}(k) \rangle}$ and $\beta_{\langle \mu(k) \rangle}$ for the two tuples respectively. Then all the automata update their action probability distributions in the following fashion

$$\begin{aligned} \mu_j(k+1) &= \mu_j(k) \\ &+ \lambda \mathcal{F}_1(\mu_j(k), \sigma_j(k), x_j(k), \beta_{\langle \mathbf{x}(k) \rangle}, \beta_{\langle \mu(k) \rangle}) \end{aligned} \tag{8}$$

$$\begin{aligned} \sigma_j(k+1) &= \sigma_j(k) \\ &+ \lambda \mathcal{F}_2(\mu_j(k), \sigma_j(k), x_j(k), \beta_{\langle \mathbf{x}(k) \rangle}, \beta_{\langle \mu(k) \rangle}) \\ &- \lambda K[\sigma_j(k) - \sigma_L] \end{aligned} \tag{9}$$

Here λ is the learning rate parameter controlling the step size $(0 < \lambda < 1)$. K is a large positive number controlling the shrinking of σ and σ_L gives a lower bound for σ. The algorithm is said to have converged when the $\mu_j(k)$s stop changing *appreciably* and $\sigma_j(k)$s are close to σ_L. The functions $\mathcal{F}_1$ and $\mathcal{F}_2$ are defined as follows.

$$\mathcal{F}_1(\mu, \sigma, x, \beta_{\langle \mathbf{x} \rangle}, \beta_{\langle \mu \rangle}) = \left(\frac{\beta_{\langle \mathbf{x} \rangle} - \beta_{\langle \mu \rangle}}{\phi(\sigma)} \right) \left(\frac{x - \mu}{\phi(\sigma)} \right) \tag{10}$$

$$\mathcal{F}_1(\mu, \sigma, x, \beta_{\langle \mathbf{x} \rangle}, \beta_{\langle \mu \rangle}) = \left(\frac{\beta_{\langle \mathbf{x} \rangle} - \beta_{\langle \mu \rangle}}{\phi(\sigma)} \right) \left[\left(\frac{x - \mu}{\phi(\sigma)} \right)^2 - 1 \right] \tag{11}$$

Here $\phi(\sigma)$ is

$$\begin{aligned}\phi(\sigma) &= \sigma_L \text{ for } \quad \sigma \leq \sigma_L \\ &= \sigma \quad \text{for} \quad \sigma > \sigma_L > 0 \end{aligned} \tag{12}$$

The rationale behind the updation schemes presented above is the following. If $x(k)$ receives a better response compared to $\mu(k)$, then $\mu(k)$ is shifted towards $x(k)$ otherwise it is moved away. The standard deviation $\sigma(k)$ is shrunk normally, apart from situations when an action choice $x(k)$ *far away* from $\mu(k)$ receives a better response or when a $x(k)$ *very close* to $\mu(k)$ receives a worse response.

In [12], [11], one can find a detailed study of the asymptotic behavior of the algorithm. For games with *common payoff*, the algorithm converges to one of the maximal points of the function f, if certain assumptions, essentially regarding the smoothness of the response function, hold true. One can argue that the response function in our scheme as described in (14) satisfies these assumptions.

We note that the extremely useful feature of the above algorithm is that it does not require any gradient information. The algorithm stochastically seeks the optimum and follows the gradient in an expected sense and converges to the optimal points. However, such an algorithm might turn out to be slow in practice. This shortcoming is suitably answered in [13] where the authors show that the algorithm has immense potential for parallelizability. In fact utilizing n computing resources, one can obtain n-fold speed-up.

5. The Basic Surface Reconstruction Algorithm

In the image space, a set of points $\mathcal{B}_I$, on the boundary (visible silhouette) of the object is specified. Subsequently, some locations inside this boundary $\mathcal{C}_I$ (whose purpose will become clear in the ensuing discussion) are also indicated. The region inside the boundary is specified as a mask and is used to determine the pixels in the image which will be used in the *shading* computation. As an example, in Fig.6, the points in $\mathcal{B}_I$ are shown as empty circles while the points in $\mathcal{C}_I$ are shown as filled circles. The white region in Fig.6 is the primary region of interest (mask) where the depth computation is performed. The object is assumed to be at some *average depth* Z_{avg} from the camera. Let, $\mathcal{B}_W$ and $\mathcal{C}_W$ denote sets of world points (on the plane $z = Z_{avg}$) corresponding to those in the sets $\mathcal{B}_I$ and $\mathcal{C}_I$ respectively, obtained from a knowledge of the camera. All the points are thus associated with this depth i.e. $f(x,y) = Z_{avg}, \forall (x,y) \in \mathcal{B}_W \cup \mathcal{C}_W$. While the boundary points are kept fixed, the depths of the interior points keep changing as the algorithm proceeds. In Fig.4 we see a snap-shot of a surface at a certain time step of the algorithm where the positions of the points in $\mathcal{C}_W$ are indicated by black dots.

Now, the objective of the algorithm is to determine the values $f(x,y)$ for all points $(x,y) \in \mathcal{C}_W$ such that the piece-wise continuous function fitted through these 3D points represents the photometrically consistent shape of the object. As introduced in section 4, this problem can be formulated as a stochastic game between CALA. We associate a CALA $\mathcal{C}_i$ with each of the locations (x_i, y_i) in $\mathcal{C}_W$. The action variable for each CALA $\mathcal{C}_i$ is the depth $f(x_i, y_i)$ of the surface at the point (x_i, y_i). We propose a game with *common payoff* with the variational error metric introduced in section 2 acting as the common reinforcement.

At any stage of the algorithm, the reinforcement β_z provided to the CALA for the chosen action set $\langle z_i \rangle$ is computed as follows (it is to be noted that the reinforcement β_μ for the mean set $\langle \mu_i \rangle$ is also computed in an identical fashion and henceforth we refer to the reinforcement simply as β unless specified otherwise). First, a (continuous) smooth surface $\mathcal{S}$ is fitted through the points $\{(x_i, y_i, z_i) : (x_i, y_i) \in \mathcal{C}_W\} \cup \{(x, y, Z_{avg}) : (x,y) \in \mathcal{B}_W\}$ by minimizing a *thin plate functional* similar to that used in [15]. Note that the choice of a *thin-plate* surface imparts a global smoothness to the solution. Now, for evaluation of the variational error metric, we discretize the integrals in (2) to obtain:

$$C(\mathbf{S}, \mathbf{N}) = \sum_{x_g} \sum_{y_g} E(\mathbf{S}, \mathbf{N}(x_g, y_g, z_g), x_g, y_g) \tag{13}$$

The region within the boundary, $\mathcal{B}_W$ is sampled finely on a discrete grid $\mathcal{G}$, where for each grid location $(x_g, y_g) \in \mathcal{G}$, the depth z_g and the surface normal $\mathbf{N}(x_g, y_g, z_g)$ are computed from the fitted surface. The integral in (1) is discretized in a similar fashion by considering a window of size $4pq$ around the projected pixel m and averaging out the mean square error. Note that handling the variational problem in a quantitative domain allows us to effectively handle the *finite light source* scenario without getting into complex analytical expressions. The reinforcement β thus turns out to be:

$$\beta = 1.0 - \frac{C(\mathbf{S}, \mathbf{N})}{\mid \mathcal{G} \mid} \tag{14}$$

The algorithm is run until the convergence conditions listed in section 4 are met. This concludes the description of the basic algorithm.

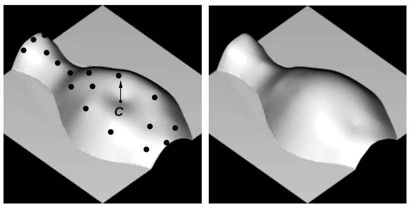
6. Finer Aspects

In this section, we present some enhancements to our basic algorithm that allow for better control over the reconstructed surface and help in speeding up the algorithm.

6.1. User Interaction

After the initial user interaction/input, the algorithm proceeds to converge to a *local minimum* as noted in Section

4. At this stage, the user requirements govern whether the result obtained is acceptable as is or a better result is expected. Fig.4 illustrates this idea at an intermediate stage of reconstruction of the surface of a vase, where the algorithm showed no significant change in the *payoff* value β, indicating a *local minimum*. A slight undesirable trough is visible on the surface at the location marked C in Fig.4(a). The system allows the user to intervene to pull the system out of this local minimum. The arrow depicts the direction and magnitude of change in the depth value at the said location initiated by the user to pull the system out of the local minimum. Fig.4(b) depicts the surface after the algorithm has adjusted to the interaction. The user can also add to the set of points $\mathcal{C}_I$ (in case the details in a particular region need to be enhanced) or delete from the existing set (to impart smoothness to a local region around the deleted location). For the newly added points, the μ values are assigned by interpolating from the current surface $\mathcal{S}$ while the σ values are specified by the user.



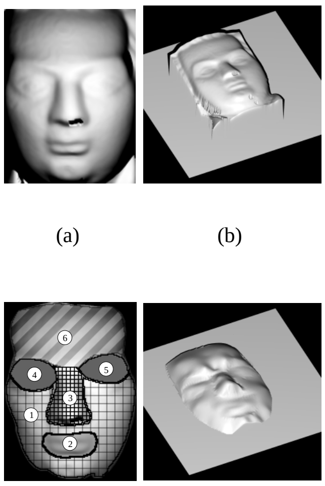
(a) (b)

Figure 4: Illustration of User Interaction: (a) Before Interaction and (b) surface after pulling on CALA C. Black dots indicate locations in $\mathcal{C}_W$.

Note that a change δ in the mean value μ_i of a CALA C_i s.t. $\delta \in (\mu_i - \sigma_i, \mu_i + \sigma_i)$ describes a change in the playing strategy chosen by player i (CALA C_i). From the definition of *Nash Equilibrium* in (7), it is clear that the system cannot be pulled out of a local minimum, if player i is the only CALA whose μ is updated by the user at the minimum. Thus, the change initiated by the user must be large enough (outside the probability hill) to ensure that the chosen strategy does not belong to the current strategy set of C_i. The system would then be able to overcome the minimum.

6.2. Hierarchical Reconstruction

We propose a scheme of *hierarchical reconstruction* for complex surfaces which cannot be suitably represented as an interpolating *thin plate functional* through a set of 3D points. One such example is a face, where features like nose, lips etc., preclude dealing with the whole face as a single thin plate functional. We illustrate the procedure on the classical Mozart face in Fig.5. Instead of dealing with



(a) (b)

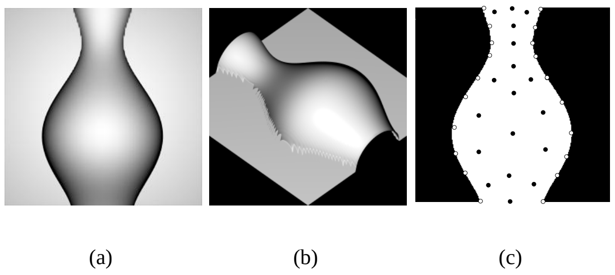
(c) (d)

Figure 5: Hierarchical Reconstruction of Mozart's face: (a)&(b) Original image and surface, (c) levels and (d) the reconstructed surface.

the whole face as one instance, the user segments out each sub-region of the face and specifies the sets $\mathcal{B}_I$ and $\mathcal{C}_I$ and the mask, separately for each of the regions as shown in Fig.5(c) where the number of sub-regions is 6. The algorithm is then run sequentially on each sub-region, by including only the image pixels within the mask of that sub-region, for the payoff computation. The reconstructed surface for each sub-region acts as a seed for initializing the μ values of the boundary points of the next (adjoining) sub-region under consideration. In this manner, the method allows dealing with complex surfaces while maintaining C^0 continuity along creases. In the first stage, reconstruction of sub-region 1 shown in Fig.5(c) is carried out first. The adjoining sub-region (nose) 3 is considered next. The boundary height values for sub-region 3 are initialized from the surface obtained for sub-region 1 and the reconstruction algorithm run for this sub-region. The other subregions are considered subsequently and finally, Fig.5(d) shows the complete reconstructed surface.

6.3. Placement of CALA

As one can expect, the convergence rate of the CALA algorithm is directly governed by the number of locations in $\mathcal{C}_I$. It is therefore expected that these locations be marked out in a topology aware manner, as far as possible. Consider the lateral view of a *surface of revolution* as an example. After the boundary of the SOR is marked out and held fixed, populating $\mathcal{C}_I$ with more number of points on the axis of revolution and fewer around the axis serves the same purpose



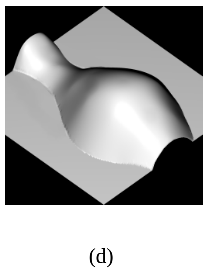


Figure 6: Vase: (a) Original image, (b) original surface, (c) points in the sets $\mathcal{B}_I$ (white circles) and $\mathcal{C}_I$ (black circles) and (d) the reconstructed surface.

as having a uniform density of points all around, though the former takes much less time to converge. Also, since the surface has a degree of freedom at the marked locations only, more locations need to be marked in regions where finer detail is required.

6.4. Parallelizability

As noted in Section 4, the CALA algorithm scales linearly with the size of the available resources. This makes it a particularly favorable choice among other similar (slow) schemes which are not generally scalable. As indicated in Section 4, the learning rate is controlled by the parameter λ, see equation(8). Setting a high value for λ for a single threaded run of the algorithm might lead to faster convergence but the results will be poor. On the availability of more computing resources higher values of λ give equally good results and in this sense speed-up is achieved. For details on parallelizing CALA, the reader is referred to [13].

7. Results

The scheme proposed has been extensively tested for a number of cases. Fig.6 depicts the result for the reconstruction of a synthetic *Lambertian* Vase for which the illumination kernel in (3) was appropriate.

Then the algorithm was tested for a synthetic Cone (Fig.7) satisfying the hybrid reflectance model. If the *Lambertian* kernel (3) is used, the algorithm converges to the reconstruction in Fig.7(b) which shows significant errors in regions of highlight. However, when the more appropriate hybrid kernel (5) is used, the results (see Fig.7(c)) improve significantly.

The algorithm works well for images with black shadows as is shown in Fig.9.

For testing the algorithm in real situations, a *painted-clay Vase* (shown in Fig.8(a)) was pictured under illumination from an incandescent lamp (10W) placed near the

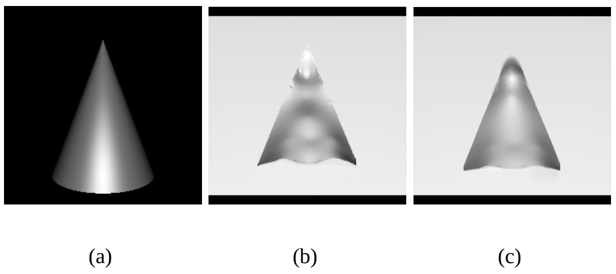


Figure 7: Synthetic Cone ($l = (0,0,0)$, $K_s = 0.5$, $K_d = 0.5$, $n = 20$): (a) Original image, (b) reconstructed surface - Diffuse only, (c) reconstructed surface - both Diffuse and Specular.

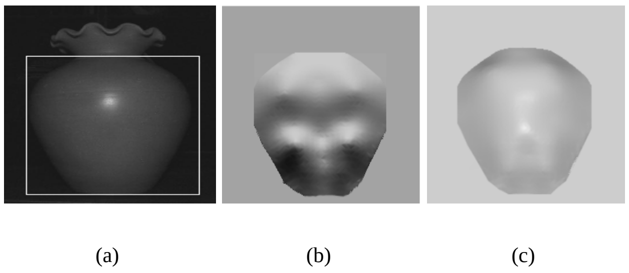


Figure 8: Real Vase ($l = (0,0,0), f = 6.8mm$): (a) Original image, (b) reconstructed surface - Diffuse only, (c) reconstructed surface - both Diffuse and Specular.

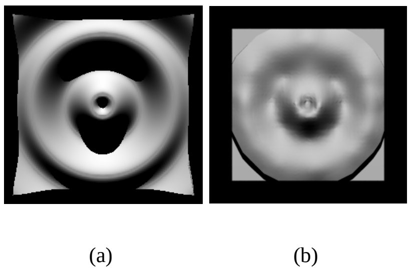


Figure 9: Presence of Shadow ($l = (0,-0.5,-1.0)$, $f = 1.0$): (a) Original image, (b) reconstructed surface

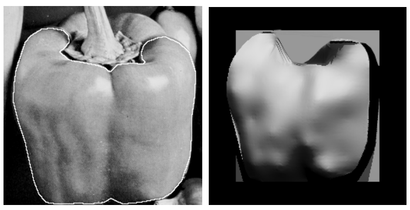

(a) (b)

Figure 10: Pepper Image ($l = (0.766, -0.642, -1.0)$): (a) Original image, (b) reconstructed surface

camera center. An `Olympus C4100Z` camera was used for the imaging purposes. As in the synthetic case, the use of *Lambertian* model fails to give good results (see Fig.8(b)). The presence of a specular highlight necessitates the use of the hybrid kernel in (5).The reconstruction (for the region within the white rectangle) under this model is shown in Fig.8(c)[1].

Another real image tried was the *pepper* image from the UCF[2] repository. The light source location for this image is known but the camera unknown. Hence a set of plausible camera parameters were used to reconstruct the surface (for the region within the dotted white boundary) shown in Fig.10.

It is to be noted that in all these examples no user intervention was necessary (during the solution evolution process) to arrive at the solution.

In all these cases the fixed points on the boundary were identified from the silhouette in the image and were associated with the same average depth. This is an approximation in the context of perspective imaging as the points on the silhouette should be associated with the occluding contour [14]. Although this approximation does affect the quality of results, the solutions obtained are acceptable.

8. Conclusion

In this contribution, we have proposed a novel generic scheme for the shape from shading problem. Our approach can effectively handle different illumination and imaging models. This flexibility allows one to handle real images more effectively. Being essentially a non-analytical framework, the method handles light sources placed at finite distances from the object as well as placed at infinity. Features such as progressive reconstruction and pyramidal refinement make the task of modeling easier. Convergence issues such as those of local minima can also be suitably dealt with by minimal user intervention in the solution evolution process.

A definite direction for extension of this contribution is the learning of the light source location. Moreover, the surface illumination parameters such as the albedo etc. could also be learned along with the shape.

[1]To see an animation of the solution evolution process please visit http://vglab.cse.iitd.ernet.in/research/research_group1/2/sfs/index.shtml

[2]University of Central Florida: by anonymous ftp under the pub/tech_paper/survey directory at eustis.cs.ucf.edu

References

[1] M. Bichsel and A. Pentland. A simple algorithm for shape from shading. *IEEE Proc. Computer Vision and Pattern Recognition*, pages 459–465, 1992.

[2] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. *Technical Report 3021, INRIA*, November 1996.

[3] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley, August 4 1995.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[5] B. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. *MIT AI-TR*, 1970.

[6] B. K. P. Horn. *Robot Vision*. MIT Press, 1986.

[7] C. Lee and A. Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *Artificial Intelligence*, 26:125–143, 1985.

[8] A. Pentland. Local shading analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:170–187, 1984.

[9] E. Prados and O. Faugeras. Perspective shape from shading and viscosity solutions. *IEEE Proceedings of ICCV'03*, 2:826–831, October 2003.

[10] G. Santharam, P. Sastry, and M. Thathachar. Continuous action set learning automata for stochastic optimization. *Journal of the Franklin Institute*, 5(331B):607–628, 1994.

[11] P. S. Sastry, V. V. Phansalkar, and M. A. L. Thathachar. Decentralised learning of nash equilibria in multi-person stochastic games with incomplete information. *IEEE Transactions on Systems, Man and Cybernetics*, 24:769–777, May 1994.

[12] P. S. Sastry, K. Rajaraman, and S. R. Ranjan. Learning optimal conjunctive concepts through a team of stochastic automata. *IEEE Transactions on Systems, Man and Cybernetics*, (23):1175–1184, 1993.

[13] M. A. L. Thathachar and M. T. Arvind. Parallel algorithms for modules of learning automata. *IEEE Transactions on Systems, Man and Cybernetics*, Part B, 28:24–33, 1998.

[14] K.-Y. K. Wong, P. R. S. Mendonça, and R. Cipolla. Reconstruction of surfaces of revolution from single uncalibrated views. *BMVC*, 2002.

[15] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. Single view modeling of free-form scenes. *Proc. Computer Vision and Pattern Recognition*, 2001.

[16] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 21(8):690–706, August 1999.