Interactive Shape from Shading*

Gang Zeng Yasuyuki Matsushita† Long Quan Heung-Yeung Shum†

Hong Kong University of Science and Technology
Clear Water Bay, Kowloon,
Hong Kong, P. R. China
{zenggang,quan}@cs.ust.hk

Microsoft Research Asia†
Beijing Sigma Center, No.49, Zhichun Road,
Beijing, P. R. China
{yasumat,hshum}@microsoft.com

Abstract

Shape from shading (SfS) has always been difficult for real applications due to its intrinsic ill-posedness. In this paper, we propose an interactive SfS method which efficiently uses human knowledge in order to resolve ambiguity. We propose a global solution of continuous surfaces with a few constraints of surface normals that are interactively imposed to regularize the problem. A surface is divided into local patches, and each local solution is estimated with a fast marching SfS. It is shown that the boundaries of local solutions constitute a weighted Voronoi diagram, which allows for the formation of a global solution from the local ones. Finally, we optimize this global estimation by minimizing an energy functional based on shading and smoothness priors. Reconstruction results from both synthetic and real images demonstrate the usability of the new approach for various modeling applications.

Keywords: Interaction, Shape from Shading, Fast Marching, Shortest Path, Poisson Refinement.

1. Introduction

Shape-from-Shading (SfS) has long been a classic and fundamental problem in computer vision. The problem is essentially ill-posed even given the assumptions of Lambertian reflection and a known directional light source. In order to resolve ambiguities, various kinds of additional assumptions are used in the literature. Most of these assumptions are not general, so it has been difficult to solve the problem over various types of scenes.

In this paper, we propose an interactive approach to the SfS problem which efficiently resolves its inherent ambiguity using human knowledge.

*This work was done while the first author was visiting Microsoft Research Asia.

1.1. Prior Work

In the context of shape recovery from a single image, there are two main streams of research in the literature; SfS and interactive modeling.

SfS has long been studied as a central problem in computer vision. Here we only review some of the work that relate closely to our work. Readers are referred to recent surveys [12, 19, 4]. Among various types of approaches, Fast Marching SfS (FM-SfS) [10] is found to be computationally efficient as its complexity is $O(N \log N)$ where N is the number of pixels. Aside from its numerical efficiency, this method is known to give a viscosity solution to an *Eikonal equation*. Recently, Tankus *et al.* [16] extended this work to handle perspective lighting conditions. One problem of FM-SfS is that it is limited to computing a surface with only one maxima (or minima) point, hence it is a local solution and has difficulties in constructing complex surfaces. A method is proposed in [9] to automatically estimate a global solution. However, this method is limited to a regular surface; more precisely, a Morse surface [5] that may not be found commonly in real scenes.

In interactive modeling, many different approaches have been used to realize 2.5D modeling from a 2D input image, e.g., methods which directly assign depth values to pixels in an image [14, 8], a method which uses surface normals as input [18] and sketch-based modeling [20, 6] which facilitates shape modeling using 2D strokes. While these methods are suitable in modeling simple shapes; however, they are not effective to model complex surfaces because of the amount of user interaction required.

1.2. Proposed Approach

The limitations of previous approaches have motivated us to propose a new approach, an *interactive SfS* method, for the purpose of practical shape modeling from a single shading image. Our method uses human knowledge as the prior to make the problem tractable. Given a small number of surface normals, our method automatically determines

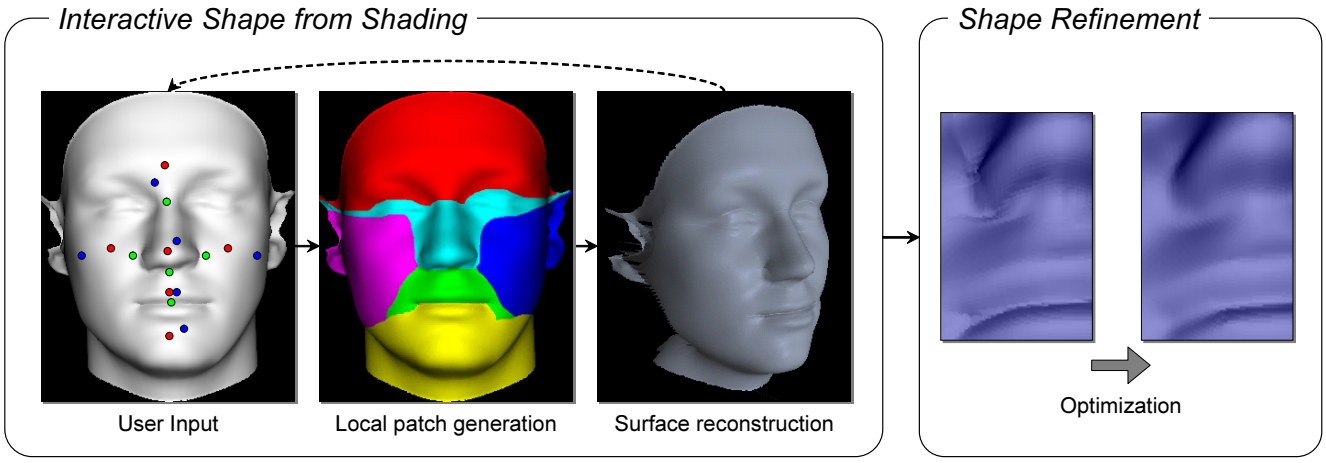


Figure 1. An overview of the interactive SfS and shape refinement. Interactive SfS receives user input (blue points in the left figure), and automatically segments an image into a set of local patches. Local surfaces are then reconstructed individually while maintaining the depth continuity among patches. Shape refinement is applied after interactive surface construction.

the peaks of the surface and localizes the problem by segmenting the input image into a set of *local patches* as illustrated in Fig. 1. SfS is then locally applied to reconstruct each local surface patch, and the local solutions are then combined to form a smooth global surface. This approach is motivated by the fact that SfS works well for solving local problems [10, 1, 3]. For satisfactory interaction speed, we use FM-SfS [10] which is considered to be the fastest solver for a local SfS problem. The limitations of FM-SfS are 1) an accurate peak position is required, and 2) it only solves local problems. In order to combine FM-SfS in our interactive method, we have developed algorithms which enable peak detection, segmentation, and automatic merging of local solutions. In this way, we derive a global solution over piecewise C^1 continuous surfaces without using boundary assumptions, which has been a difficulty in prior works.

The reason that we chose surface normals as the user inputs is that it is easier for a human to recognize a surface orientation over its depth [11]. This fact is also used in [18] for interactive modeling. The problem of peak detection is formulated as the shortest path problem on a network of a 2D image grid in Sec. 2. This formulation also allows us to use the solutions of classic graph theory such as minimum spanning tree and Voronoi diagram, to determine the boundaries of local patches. The details of peak detection, segmentation and surface reconstruction are found in Sec. 3.

The shape estimated with interactive SfS is visually plausible with the help of user interaction; however, it still has small discontinuities in surface normal, especially around the boundaries of the local patches. For shape refinement, we minimize an energy functional which is defined by a few assumptions based on smoothness and shading priors. This optimization is applied only once after interactive SfS as illustrated in Fig. 1. The details are found in Sec. 4.

In Sec. 5, results from both synthetic and real images are shown to demonstrate the usability and accuracy. Finally, we conclude the paper with future work in Sec. 6.

2. Formulation

Given an input shading image $I(x, y)$ and a set of user-defined surface normals $\mathcal{N} = \{\mathbf{n}_i\}$ at positions $\mathcal{P} = \{\mathbf{p}_i\}$, the goal is to reconstruct a surface $z(x, y)$ of the scene using its shading image based on the following assumptions.

Assumptions: Throughout this paper, we assume that the shading image is photographed orthographically, and the scene is composed of Lambertian surfaces which exhibit single-bounce reflections and are illuminated from a known direction by a point light source at infinity. The surface is assumed to be piecewise C^1 continuous, and cast shadows will not be considered.

2.1. Shading image formation model

Let us first review the shading image formation model for a 3D Lambertian object. Let the light source direction be given by $(\mathbf{l}, -1), \mathbf{l} \in \mathbb{R}^2$ and the surface normal by $(\nabla z(x, y), -1)$, where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is the gradient operator. The shading image is formed as their inner product:

$$I(x, y) = \frac{\mathbf{l} \cdot \nabla z(x, y) + 1}{\sqrt{||\mathbf{l}||^2 + 1}\sqrt{||\nabla z(x, y)||^2 + 1}}. \quad (1)$$

For the simple vertical light source case whose direction is the same as the viewing direction $\mathbf{l} = (0, 0)$, the shading image is given by $I(x, y) = \frac{1}{\sqrt{||\nabla z(x,y)||^2+1}}$. The problem becomes the reconstruction of $z(x, y)$ from its gradient magnitude that is given by following Eikonal equation,

$$||\nabla z(x, y)|| = \sqrt{I^{-2}(x, y) - 1}. \quad (2)$$

For the oblique light source case where the light source direction is different from that of the viewer $\mathbf{l} = (l_x, l_y), ||\mathbf{l}|| \neq 0$, we transform the problem from the image coordinate system (x, y, z) to the light source coordinate $(\hat{x}, \hat{y}, \hat{z})$. Thus, the problem becomes almost the same as the vertical light source case, but different in that the right hand side of Eq. (3) depends on the surface itself:

$$||\nabla \hat{z}(\hat{x}, \hat{y})|| = \sqrt{\hat{I}^{-2}(\hat{x}, \hat{y}) - 1}. \quad (3)$$

This dependency can be written as

$$\hat{I}(\hat{x}, \hat{y}) = I(l_x\hat{x} - l_y\hat{y} + l_x\hat{z}, l_y\hat{x} + l_x\hat{y} + l_y\hat{z}), \quad (4)$$

where $\hat{I}$ is a transformed shading image.

2.2. Fast Marching formulation

As described by Kimmel *et al.* [10], FM-SfS is able to compute a consistent solution in $O(N \log N)$ time for both the simple vertical light source and the oblique light source cases where N is the total number of pixels (grid points). Their algorithm actually performs a plane sweep along the lighting direction $(\mathbf{l}, -1)$. At any time, this plane intersects the surface of the scene $\hat{z}(\hat{x}, \hat{y})$ with a 2D closed curve. Seen from the shading image in the light source coordinates $\hat{I}(\hat{x}, \hat{y})$, the 2D projection of this curve moves as the plane sweeps over the surface. Supposing the plane moves at a constant speed, a moving speed $|\nabla T(\hat{x}, \hat{y})|$ of the 2D curve can be defined at each point by

$$|\nabla T(\hat{x}, \hat{y})| = |\nabla \hat{z}(\hat{x}, \hat{y})| = \sqrt{\hat{I}^{-2}(\hat{x}, \hat{y}) - 1}. \quad (5)$$

In this way, the time when the 2D curve visits a pixel is related to the depth value of this pixel, and the 3D surface reconstruction problem can be converted into the 2D curve propagation problem. Using Fast Marching Method (FMM), this 2D curve propagation problem $T(\hat{x}, \hat{y})$ is solved efficiently, and as a result, the surface shape $\hat{z}(\hat{x}, \hat{y})$ is estimated. For the oblique light source case, the dependency is overcome by adopting the smallest depth value from all the neighbors of the updated point [10]. Based on this method, in the following part of the paper, we use the light source coordinate system and denote it by (x, y, z) (in the previous section, it was $(\hat{x}, \hat{y}, \hat{z})$), and thus we have $\mathbf{l} = (0, 0)$.

At its core, FMM is closely related to Dijkstra's method [2] which is a classic algorithm for computing the shortest path on a network. At the same time, it relies on a heap to find the smallest element of the expanding front, and can be considered as a Huyghen's principle algorithm where the expanding wave is confined to the network paths. For example, let us consider a particle that is always on the expanding front. The trail of this particle is actually the shortest path from its starting position to its current position. In fact, the expanding front and the shortest path are two different views of the same problem. Therefore, we use FMM to solve the shortest path problem. Later we will see that the shortest path allows us to back-track information, thus helps us to determine the peaks and the boundaries of local surface patches.

2.3. Shortest path formulation

Now we formalize the global SfS problem as a set of the shortest path problems. Given the position of a peak $\mathbf{o}$, which has a surface normal toward the lighting direction, we aim to find the distance $D_{\mathbf{o}}(x, y)$ to other points (x, y) on the network. The network here is considered as the 2D grid on the image plane and is used to solve the 2D curve propagation problem by computing the shortest path. A weight value $W(x, y)$ is defined on each vertex of the network as follows.

$$W(x, y) \equiv |\nabla T(x, y)| = |\nabla z(x, y)| = \sqrt{I^{-2}(x, y) - 1}. \quad (6)$$

The distance $D_{\mathbf{o}}(x, y)$ is related to the sum of the vertex weight W, and therefore it equals to the relative depth value of a point (x, y) on the local patch. Moreover, the local tangent direction of the shortest path is the direction along which the distance increases most significantly, and therefore this direction is considered as the 2D projection of the surface normal.

In a general case, given a set of peaks $\mathcal{O} = \{\mathbf{o}_j\}$, we aim at finding a weighted distance $D(x, y)$ from each point (x, y) to its nearest reference point (peak), such that this distance is equal to the depth $z(x, y)$. In other words, we have

$$z(x, y) = D(x, y) = \min\{D_{\mathbf{o}_j}(x, y) + z(\mathbf{o}_j) : \mathbf{o}_j \in \mathcal{O}\}, \quad (7)$$

where $z(\mathbf{o}_j)$ is an additional shifting term of the peak $\mathbf{o}_j$. Again, the local tangent line of the shortest path is the 2D projection of the surface normal.

In this way, the global SfS problem is converted into the weighted shortest path problem, which is related to a weighted Voronoi diagram. Note that it cannot be solved directly without knowing $z(\mathbf{o}_j)$, which is actually the altitude of the peak. The detection of $z(\mathbf{o}_j)$ is converted into the problem of finding the minimum spanning tree, and is further discussed in Sec. 3.

3. Interactive SfS

In this section, we introduce interactive SfS. The idea of our algorithm is locally computing surface patches $D_{\mathbf{o}_j}(x, y)$ from each peak $\mathbf{o}_j \in \mathcal{O}$ and connecting them together to form a whole surface $z(x, y)$. To distinguish peaks from other singular points, a small number of surface normals are given in a shading image. This section describes our method of detecting peaks using these surface normals

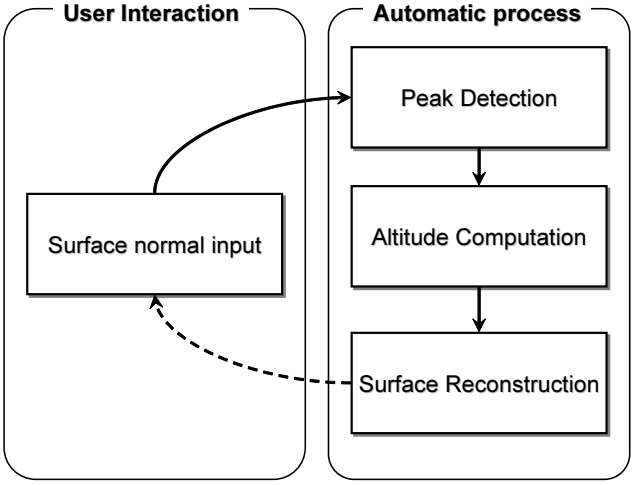


Figure 2. A flow chart of the interactive SfS, which includes four parts: user input of surface normals, peak detection, altitude computation and surface reconstruction. The whole process is repeated until a satisfactory result is found.

and combining local solutions to generate a global solution with simultaneously determine the boundaries of local solutions. The method is composed of the following four steps as shown in Fig. 2.

In the first step, a user inputs a few surface normals $\mathcal{N} = \{\mathbf{n}_i\}$ on points $\mathcal{P} = \{\mathbf{p}_i\}$. The normal information is then used to detect peaks $\mathcal{O} = \{\mathbf{o}_j\}$ that are consistent with the given surface normals in the second step. In the third step, relative altitudes of peaks $z(\mathcal{O}) = \{z(\mathbf{o}_j)\}$ are estimated using saddle points which can be automatically detected. This estimation relies on the geometric relationship among the peaks in the network, and the problem is solved using the minimum spanning tree algorithm. Finally, in the fourth step, local surface patches $D_{\mathbf{o}_j}$ are computed from each peak $\mathbf{o}_j \in \mathcal{O}$. A global surface $z(x, y)$ is then estimated by connecting local patches $D_{\mathbf{o}_j}(x, y)$ at boundaries which are simultaneously detected using a weighted Voronoi diagram. The above steps are repeated until the user finishes interaction.

3.1. Peak Detection

We aim at detecting the peaks $\mathcal{O}$ that are consistent with the given surface normals $\mathcal{N}$ at positions $\mathcal{P}$. To achieve this, we first compute the shortest paths from $\mathbf{p}_i$ to the other points. Among these paths, the correct path from $\mathbf{p}_i$ to $\mathbf{o}_i$ is determined, since the 2D projection of $\mathbf{n}_i$ is equivalent to the local tangent direction of this path. Since the peak is one type of singular point whose surface normal direction is the same as the lighting direction, it always has the brightest intensity. We use this as an additional constraint to determine the peak along the path. In practice, in order to handle a reasonable degree of error on the input normal, we search the paths within a small angle, and detect the peak $\mathbf{o}_i$ as the nearest brightest point from $\mathbf{p}_i$. Other singular points such as valleys and saddle points will never be found in this way, not only because the algorithm always back-tracks the shortest path toward the altitude-ascending direction, but also because the user can always revise the input based on the results.

In the more general cases where multiple normals are given, it may happen that several normals share a common peak (i.e., their shortest paths intersect at this peak point.) Finally, we need to merge the detected peaks to have a minimum set of $\mathcal{O}$ that are consistent with the input normals $\mathcal{N}$. This also indicates an efficient input scheme, where the number of inputs $|\mathcal{N}|$ equals the number of peaks $|\mathcal{O}|$ on the surface. In summary, the peak detection proceeds as follows.

For each $\mathbf{n}_i \in \mathcal{N}$,
1. Compute the shortest path from $\mathbf{p}_i$ using $\mathbf{n}_i$, which contains the peak $\mathbf{o}_i$.
2. Determine $\mathbf{o}_i$ along this path by taking the nearest brightest point.

Merge the peaks to obtain the minimum set of $\mathcal{O}$.

3.2. Altitude Computation

Once peaks $\mathcal{O}$ are determined from the given normals $\mathcal{N}$, the algorithm steps forward to the altitude computation phase. Peak altitudes are closely related to patch boundaries, which are essential for the global surface reconstruction. This section describes the relationship between peak and patch boundary, and introduces the method to compute them.

[Relationship between peak and patch boundary]
Let us first consider two adjacent peaks $\mathbf{o}_1$ and $\mathbf{o}_2$ as illustrated in Fig. 3. We aim at spreading two local patches $D_{\mathbf{o}_1}(x, y)$ and $D_{\mathbf{o}_2}(x, y)$ from these two peaks and then combining them. The depth value of a point (x, y) is the sum of two parts, the relative depth value of this point on the patch and the altitude of this patch, or

$$z(x, y) = D_{\mathbf{o}_j}(x, y) + z(\mathbf{o}_j). \tag{8}$$

In general, given a continuous surface, points on the patch boundary have the depth value

$$D_{\mathbf{o}_1}(x, y) + z(\mathbf{o}_1) = D_{\mathbf{o}_2}(x, y) + z(\mathbf{o}_2); \tag{9a}$$
$$z(\mathbf{o}_1) - z(\mathbf{o}_2) = D_{\mathbf{o}_2}(x, y) - D_{\mathbf{o}_1}(x, y). \tag{9b}$$

Thus, the patch boundary has a close relationship with peak altitudes $z(\mathcal{O})$. In fact, we can obtain a patch boundary given peak altitudes, and vice versa. Our strategy is to first find a point on the patch boundary in order to determine the peak altitudes, and then use these peak altitudes to estimate the whole patch boundary. For convenience, we call a set of points on the patch boundary as "boundary points".

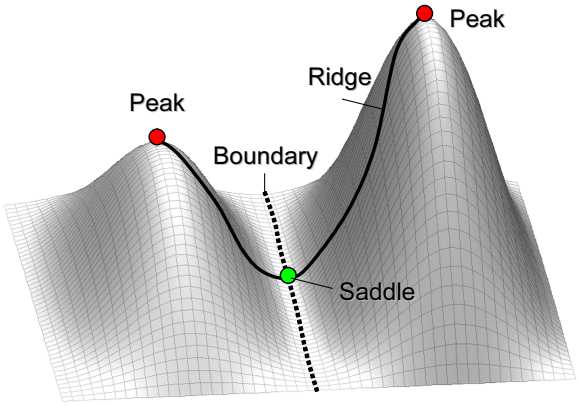


Figure 3. Illustration of peak, saddle, ridge and boundary.

[Altitude computation between two adjacent peaks]
In order to detect a boundary point, we first define the *ridge* as the shortest path that links two peaks on the network which is defined in Sec. 2.3, i.e., a set of points (x, y) which minimize the sum of distances $D_{\mathbf{o}_1}(x, y) + D_{\mathbf{o}_2}(x, y)$. We also define a *saddle point* $\mathbf{s}$ as the intersection of the ridge and the patch boundary as illustrated in Fig. 3. The saddle point divides the ridge into two monotonic parts since $\frac{\partial \mathbf{s}}{\partial \mathbf{r}} = 0$, where $\mathbf{r}$ is the local direction along the ridge on the saddle point $\mathbf{s}$. Thus, this saddle point is a singular point, which is the brightest point on the ridge that can be easily detected.

[Altitude computation among multiple peaks]
We now extend the altitude computation to the case of multiple peaks. n peaks produce $C_2^n = \frac{1}{2}n(n-1)$ pairs of peaks in a brute-force manner. However, since a ridge is only meaningful for adjacent peaks, the computation can be reduced to a subset of pairs. This motivates us to first compute the minimum spanning tree, whose vertices are the peaks, and edges are defined by the distances among these peaks. In our method, the minimum spanning tree is computed by Prim's algorithm [15], and after that, the number of pairs is reduced to $O(n)$. The relative altitudes of those pairs are computed by Eq. (9). Finally, since all peaks are directly/indirectly connected to each other by the tree, we can obtain the altitudes of all peaks. In summary, the altitude computation is described as follows.

1. Find the minimum spanning tree of the peaks.
2. For each branch $(\mathbf{o}_j, \mathbf{o}_k)$ of the tree,
   a. Compute the ridge between them.
   b. Search along the ridge to detect the saddle point $\mathbf{s}$ as the brightest point.
   c. Obtain the relative altitude $z(\mathbf{o}_j) - z(\mathbf{o}_k)$ by computing $D_{\mathbf{o}_k}(\mathbf{s}) - D_{\mathbf{o}_j}(\mathbf{s})$.
3. Assign altitudes $z(\mathcal{O})$ for all peaks based on the relative altitudes.

3.3. Surface Reconstruction

This step derives local solutions $\{D_{\mathbf{o}_j}(x, y)\}$ using obtained peaks $\mathcal{O}$ and their altitudes $z(\mathcal{O})$, and combines them together to obtain a global solution $z(x, y)$.

According to Eq. (9), for a simple case that all peaks have the same altitude, the patch areas constitute a Voronoi diagram of the peaks. In general, if the peaks have different altitudes, the patch areas become bigger or smaller and constitutes a weighted Voronoi diagram.

[Weighted Voronoi diagram for boundary detection]
The patch area $\mathcal{D}_j$ of a peak $\mathbf{o}_j$ is a set of points where the sum of the distance from the point (x, y) to $\mathbf{o}_j$ and the peak altitude $z(\mathbf{o}_j)$ is smaller than that of any other peaks:

$$\mathcal{D}_j = \{(x, y) : D_{\mathbf{o}_j}(x, y) + z(\mathbf{o}_j) \leq D_{\mathbf{o}_k}(x, y) + z(\mathbf{o}_k) \quad \forall \mathbf{o}_k \in \mathcal{O}, k \neq j\}. \tag{10}$$

In our method, the patch boundary estimation and local surface reconstruction are performed simultaneously. First, the altitude of each peak $z(\mathbf{o}_j)$ is used to determine which level-set the peak $\mathbf{o}_j$ belongs to. The peak is put into the level-set to indicate the reference altitude of the local patch. By running the FMM, the boundaries are detected, and local surface patches are computed and merged into a whole surface $z(x, y)$ as

$$\begin{aligned} z(x, y) &= \bigcup_j \{D_{\mathbf{o}_j}(x, y) + z(\mathbf{o}_j) : (x, y) \in \mathcal{D}_j\} \\ &= \min_j \{D_{\mathbf{o}_j}(x, y) + z(\mathbf{o}_j) : \mathbf{o}_j \in \mathcal{O}\}. \end{aligned} \tag{11}$$

In summary, the surface reconstruction is described as follows.

1. For each peak $\mathbf{o}_j \in \mathcal{O}$,
   a. Compute which level-set $\mathbf{o}_j$ belongs to based on its altitude $z(\mathbf{o}_j)$.
   b. Put the peak $\mathbf{o}_j$ into this level-set.
2. Run the FMM to simultaneously detect the boundaries, construct the local surface patches and connect them into the whole surface $z(x, y)$.

4. Shape Refinement

Our interactive SfS guarantees a continuous connection of the local patches. However, surface normals are not enforced to be smooth at the patch boundaries, because the computation is local and directional. In order to refine this surface estimation to obtain an optimal surface $Z(x, y)$, we minimize an energy functional of the form:

$$E_{total} = E_{data} + \alpha E_{smooth} + \beta E_{prior}, \tag{12}$$

where the first term enforces the similarity between the observed image and the synthesized view. The second term imposes smoothness as

$$E_{smooth} = \iint ||\nabla Z(x,y)||^2 dxdy. \quad (13)$$

We also add the third term to account for a user prior E_{prior} using the initial surface estimate as

$$E_{prior} = \iint (Z(x,y) - z(x,y))^2 dxdy. \quad (14)$$

4.1. Soft Poisson process

Jin *et al.* [7] observed that a direct approach of solving the data term is unstable due to the coupling between surface appearance and its normal. In the presence of measurement noise, the surface will bend and ripple to fit the data. This behavior can be suppressed by increasing the factor of the smoothness term, but this results in oversmoothed reconstructions. To circumvent this instability, we use a relaxed cost functional as done in [7] in which the normal is decoupled from the surface via an auxiliary vector field $\mathbf{v}(x,y)$. Now the problem is minimizing this new energy function jointly with respect to both $Z(x,y)$ and $\mathbf{v}(x,y)$:

$$E_{data} = \iint ||\nabla Z(x,y) - \mathbf{v}(x,y)||^2 dxdy, \quad (15)$$

whose solution can be obtained by solving a Poisson equation, $\nabla^2 Z(x,y) = \nabla \cdot \mathbf{v}(x,y)$, where $\nabla^2 = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})$ and $\nabla\cdot$ are Laplacian and divergence operators respectively.

The overall cost functional is formed by a weighted sum of the three costs:

$$\begin{aligned} E_{total} &= \iint ||\nabla Z(x,y) - \mathbf{v}(x,y)||^2 + \alpha ||\nabla Z(x,y)||^2 \\ &\quad +\beta (Z(x,y) - z(x,y))^2 dxdy. \end{aligned} \quad (16)$$

It can be shown that the gradient descent flow minimizing the total energy is given by

$$\begin{aligned} Z_t(x,y) &= (\nabla^2 Z(x,y) - \nabla \cdot \mathbf{v}(x,y)) + \alpha \nabla^2 Z(x,y) \\ &\quad +\beta (z(x,y) - Z(x,y)). \end{aligned} \quad (17)$$

4.2. Updating the auxiliary field

The auxiliary field is updated by the cost functional

$$\begin{aligned} E_{\mathbf{v}} &= \iint \left(I(x,y) - \frac{\mathbf{l}\cdot\mathbf{v}(x,y) + 1}{\sqrt{||\mathbf{l}||^2 + 1}\sqrt{||\mathbf{v}(x,y)||^2 + 1}} \right)^2 \\ &\quad +\gamma ||\nabla Z(x,y) - \mathbf{v}(x,y)||^2. \end{aligned} \quad (18)$$

To minimize this energy functional, we first find a robust surface normal $\mathbf{n}(x,y)$, whose orientation is computed by the Eigensystem of the neighborhood of $Z(x,y)$. Its magnitude can be computed from the shading constraint:

$$I(x,y) = \frac{\mathbf{l}\cdot\mathbf{n}(x,y) + 1}{\sqrt{||\mathbf{l}||^2 + 1}\sqrt{||\mathbf{n}(x,y)||^2 + 1}}. \quad (19)$$

With this robust normal, the negative energy gradient minimizing the cost functional Eq. (18) is given by

$$\mathbf{v}_t(x,y) = \mathbf{n}(x,y) - \mathbf{v}(x,y) + \gamma\{\nabla Z(x,y) - \mathbf{v}(x,y)\}. \quad (20)$$

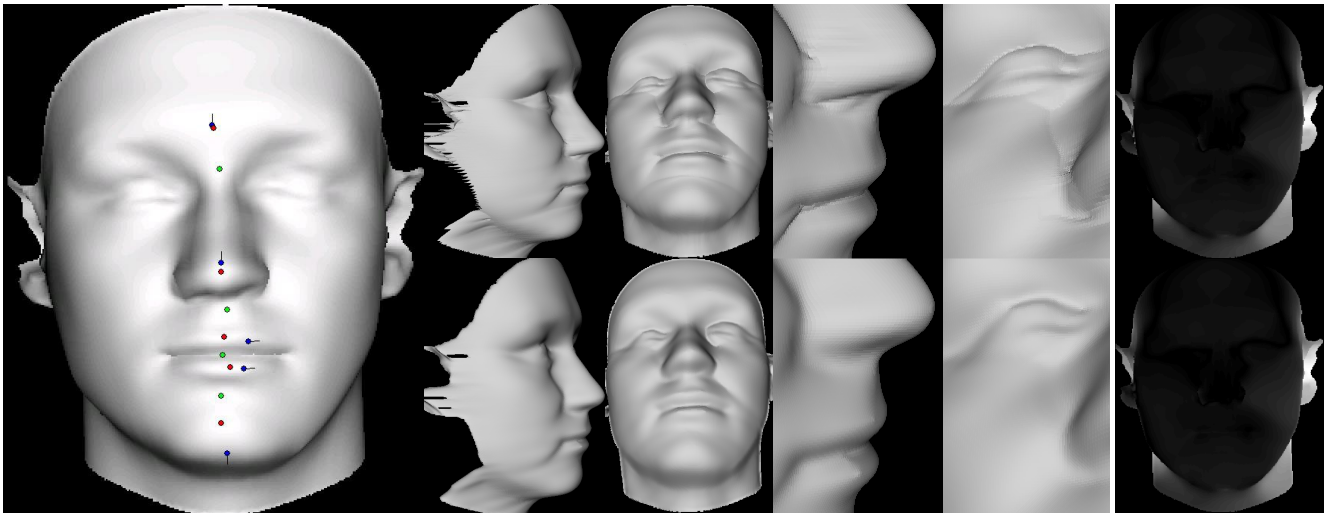
The numerical implementation is carried out by the steepest descent method.

5. Results

The proposed method is tested on both synthetic and real-world data. For the synthetic scenes, two typical head examples are constructed with our method as shown in Fig. 4 and Fig.5. In Fig. 4(a) and Fig. 5(a), using a set of user input surface normals (blue points), peaks and saddle points (red and green points respectively) are first detected. The surface is then estimated as shown in Fig. 4(b)-(c) and Fig. 5(b)-(e) in each top row. This has been achieved interactively with user prior information about the objects, and users can change their inputs until satisfactory results are obtained. In practice, we also allow the user to directly refine the peaks and saddles to handle the sharp boundary on the lips. We can see particularly accurate recovery of details where the surface geometry is complicated in Fig. 4(d)-(e) and Fig. 5(f)-(h) in the top row. As mentioned, a small number of surface normals are needed (5 or 6 for these two examples). The surface estimation of the first step is done in about $0.15\sim0.20$ *sec* with Pentium4 2.8 GHz CPU for the image of Fig. 4 of 300×300 resolution. This high speed enables a user to interactively provide the surface normals while viewing the resulting surface.

The shape refinement stage optimizes the surface estimation by a shading constraint and a smoothness prior. We have simply set α=0.1, β=γ=0.5 in Eq. (16) for the experiments. The results of refinement are shown in Fig. 4(b)-(c) and Fig. 5(b)-(e) in the bottom row. Compared to the result of interactive SfS, the resulting optimal surface is more smooth and natural as seen in close-up views (Fig. 4(d)-(e), Fig. 5(f)-(h) below). The surface refinement is done in about 20 *sec* for the image of Fig. 4.

Fig. 4(f) shows the error between the ground truth and surface estimations in both steps. The error is scaled for the visualization purpose. Please note that the biggest error (at the brightest points) is roughly 50 units (the unit is set to the same as the pixel size). As shown in the figure, the error is quite small near peaks, while it slightly increases as it goes away from the peaks. In this example, the large



(a) (b) (c) (d) (e) (f)

Figure 4. Head reconstruction from a synthetic image. The top row shows the results of the interactive SfS; the bottom row shows the results with the refinement stage. (a) Input image with the input normal (blue points), peaks detected (red points) and saddle points (green points); (b)-(c) Different views; (d)-(e) Close-up views; (f) Error between surface estimates and the ground truth.

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5. Head reconstruction from a synthetic image. The top row shows the estimations of the interactive SfS stage; the bottom row shows the surfaces with the refinement stage. (a) Input image with the user input surface normal (blue points), peaks detected (red points) and saddle points (green points); (b)-(e) Surface estimations from different views; (f)-(h) Close-up views.

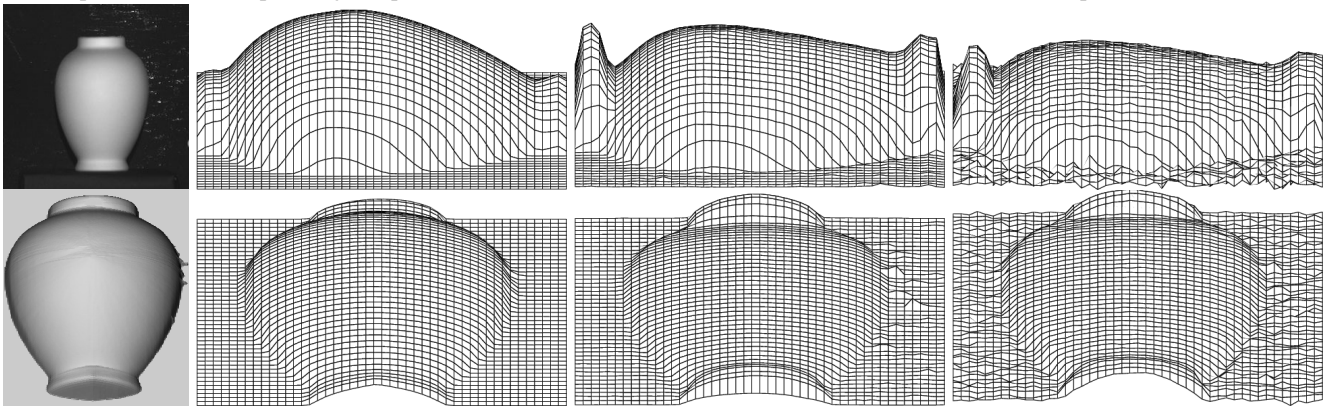


Figure 6. Comparison using a real-world scene. The leftmost column shows the input image and a view of the reconstructed shape. The other columns are the comparison among our method (2nd), Lee and Rosenfield [13] (3rd), and Tsai and Shah [17] (4th).

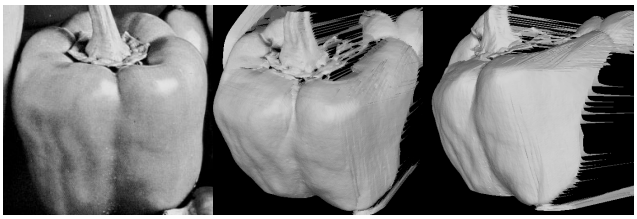


Figure 7. Result of a complex real-world scene. From left to right, input image, two different views of the reconstructed shape are shown.

error is caused by the depth discontinuity between the face and neck (or ears).

We also show two real examples in Fig. 6 and Fig. 7. A comparison is made with Tsai *et al.*'s method [17], and Lee *et al.*'s method [13] in Fig. 6, which proves that our method can generate a more reasonable result. In Fig. 7, we show our results of a more complex shape, a pepper. It is important to notice that the existing SfS methods fail to get a reasonable shape. For their results, we refer the reader to [19]. With the interactive SfS method, a quite accurate shape is reconstructed.

6. Conclusions

In this paper, we have proposed an interactive SfS method, which uses human knowledge as a prior in real-time interactions. This approach makes the SfS problem well-posed over the class of piecewise C^1 continuous surfaces where no automatic method with realistic assumptions exists. The interactive SfS consists of alternating steps of user input and estimation of the surface. To achieve our goal, we have developed an algorithm to determine peaks from the given surface normals and to estimate the boundaries of local solutions. Our method first computes local solutions from each peak, and merges them to form a global solution while maintaining the depth continuity among the individual segments. The problem is formalized by the classic graph problems such as the shortest path, minimum spanning tree and Voronoi diagram, and efficiently solved using FMM. In the shape refinement stage, the surface estimate is optimized by minimizing a regularized form of the energy functional which is defined by a few prior assumptions based on shading and smoothness priors.

Finally, we show the results of surface reconstruction from both synthetic and real images to demonstrate the applicability and accuracy of the approach. We believe that SfS can be applied to a wide variety of real-world scenes by adding a small amount of user interaction. In future work, we intend to use the proposed approach to achieve practical applications such as single view relighting.

Acknowledgements

This work is supported by the Hong Kong RGC grant HKUST 6182/04E and 6188/02E.

References

[1] M. Bichsel and A. Pentland. A simple algorithm for shape from shading. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 459–469, 1992.

[2] E. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematic*, 1:269–271, 1959.

[3] P. Dupuis and J. Oliensis. Direct method for reconstructing shape from shading. In *Proc. of IEEE Computer Vision and Pattern Recognition*, pages 453–458, 1992.

[4] J. Durou, M. Falcone, and M. Sagona. A Survey of Numerical Methods for Shape from Shading . Rapport de recherche 2004-2-R, IRIT, janvier 2004.

[5] H. Griffiths. *Surfaces*. Cambridge University Press, 1981.

[6] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *Proc. of ACM SIGGRAPH*, pages 409–416, 1999.

[7] H. Jin, D. Cremers, A. Yezzi, and S. Soatto. Shedding light on stereoscopic segmentation. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages 36–42, June 2004.

[8] S. Kang. Depth painting for image-based rendering applications. Technical report, Compaq Computer Corporation, Cambridge Research Lab., December 1998.

[9] R. Kimmel and A. Bruckstein. Global shape from shading. *Computer Vision and Image Understanding*, 62:360–369, 1995.

[10] R. Kimmel and J. Sethian. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision*, 14(3):237–244, 2001.

[11] J. Koenderink. Pictorial relief. *Phil. Trans. of the Roy. Soc.: Math., Phys, and Engineering Sciences*, 356(1740):1071–1086, 1998.

[12] R. Kozera. An overview of the shape from shading problem. *Machine Graphics and Vision*, 7(1):291–312, 1998.

[13] C.-H. Lee and A. Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *Artificial Intelligence*, 26(2):125–143, 1985.

[14] B. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *ACM SIGGRAPH Proceedings*, pages 433–442, 2001.

[15] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[16] A. Tankus, N. Sochen, and Y. Yeshurun. Perspective shape-from-shading by fast marching. In *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages 43–49, Jun. 2004.

[17] P. Tsai and M. Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12(8):487–498, 1994.

[18] L. Zhang, G. Dugas-Phocion, J. Samson, and S. Seitz. Single view modeling of free-form scenes. *Journal of Visualization and Computer Animation*, 13(4):225–235, 2002.

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

[20] D. Zorin and A. Hertzmann. Illustrating smooth surfaces. In *Proc. of ACM SIGGRAPH*, pages 517–526, 2000.