A SPATIALLY MULTIRESOLUTION, MRF OPTIMIZATION BASED APPROACH TO THE SHAPE FROM SHADING PROBLEM.

Markus Louw Fred Nicolls
Department of Electrical Engineering
University of Cape Town
South AFrica
email: markus.louw@gmail.com

ABSTRACT
This paper extends the MRF formulation approach [6] for solving the shape from shading problem. Our method uses simulated annealing Gibbs sampling to minimize a lattice energy for an MRF formulation which characterizes the Shape from Shading (SFS) problem under Lambertian reflectance conditions. We explore a spatial multi-resolution approach which refines the digital elevation map of the surface in a coarse to fine manner. This framework allows us to deal with images as projections rather than affine mappings, and allows us to incorporate a local occlusion constraint to cater for more general scene/lighting configurations.

KEY WORDS
Shape from Shading, MRF, Lambertian illumination.

1 Introduction and Literature Review

The interested reader is referred to two surveys, [11] (1999), and [5] (2004). In the former, SFS approaches are classified into minimization (e.g. [9]), propagation (e.g. [7]), local (e.g. [8]), or linear (e.g. [10]) approaches. In the latter, they are classified into methods based on partial differential equations (characteristic strips [1], power series expansion [2], and viscosity solutions (e.g.[3])), minimization methods [4], and methods which approximate the image irradiance equation, which contain the local and linear methods surveyed in [11].

This work builds on that previously done in [6], in which a Loopy Belief Propagation (LBP) formulation is used to solve a Markov Random Field which minimizes a set of energy terms which correspond to differences in the synthetic reflectance map vs. observed data, with additional possibility for putting smoothness constraints on that surface.

The algorithm here developed is called Gibbs Projective Multi-Res SFS (GPM SFS). This work improves on [6] (LBP-SFS) in that here we use a wavelet style spatial multiresolution to approximate the correct surface elevation map in a coarse to fine manner, whereas [6] runs on the entire image at once. We also use Gibbs sampling rather than belief propagation to minimize the lattice energy, which converges more reliably, and has running time which scales linearly with the size of the state vectors. Also, we use a projective camera model rather than an affine camera model. Finally we show how to include support for local self-occlusion of the surface from the light source(s).

2 Lambertian Reflectance Model

This algorithm calculates a surface on the Lambertian assumption that the intensity of a pixel is proportional to the inner product of the direction vector of the incident light and the surface normal at the point of intersection. We follow the notation of [5], to formulate this. The image irradiance equation is

$$R(\overrightarrow{n}(x)) = I(x) \tag{1}$$

where $I(x)$ is the image irradiance (usually the intensity) measured at location x, and $R(\overrightarrow{n}(x))$ is the reflectance function on the surface which takes the normal at point x as an argument. The surface normal may be calculated as

$$\frac{1}{\sqrt{(1+p(x)^2+q(x)^2)}}(-p(x), -q(x), 1) \tag{2}$$

in which

$$p = \partial u/\partial x_1 \text{ and } q = \partial u/\partial x_2 \tag{3}$$

where u is the height of the surface. If there is a unique light source at infinity, and shining in direction $\overrightarrow{w} = (w1, w2, w3)$, the pixel intensity is the inner product

$$R(\overrightarrow{n}(x)) = w \cdot \overrightarrow{n}(x). \tag{4}$$

3 MRF formulation to solve SFS

This algorithm calculates an optimal set of labels for the height at each corner vertex on a surface. We assert that a corner vertex occurs at the corner of a pixel (as is usual in SFS an image is used as the basis for the surface); at the intersection of four pixels, one corner vertex represents the height of the surface at that location. Each triplet of vertices describes a unique plane, and the orientation of that plane relative to the direction of the light source allows a probability to be assigned to that configuration for that triplet,

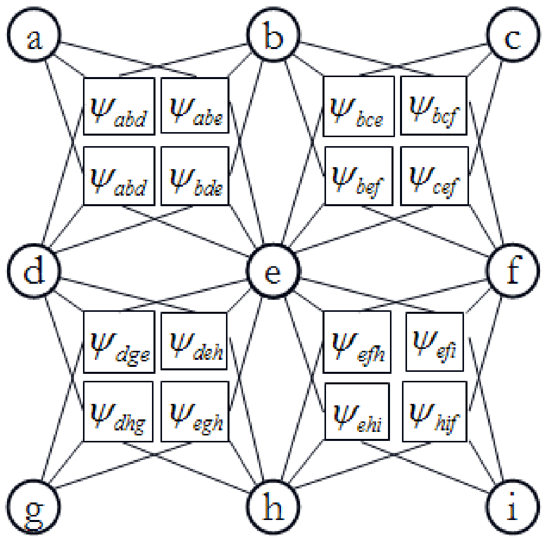

Figure 1. Depiction of the locations of 9 corner vertex nodes (round) over 4 pixels. The energy terms associated with each triplet are represented by squares. They are connected by lines to their corresponding vertex nodes.

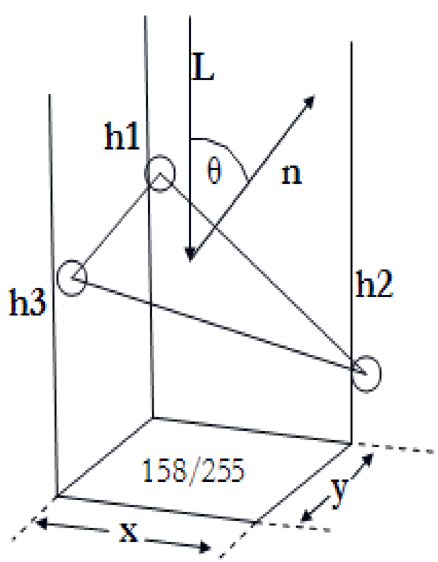

Figure 2. Depiction of the plane generated by corner vertex nodes each at a particular height. The inner product of the plane's normal and the light source's direction gives the pixel intensity at the pixel corresponding to those three corner vertex nodes (Lambertian reflectance model), which in this diagram is 158/255 .

given the observed reflectance for that image region. A diagrams for the topology for this scheme with pixels, corner vertex nodes, and the corresponding energy terms for each triplet, is shown in Fig.1. The plane generated by each triplet of pixels forms an angle against the incident light, giving an illumination for that pixel. This is shown in Fig. 2. Next we define a Markov Random Field (MRF) on this set (lattice) of vertex nodes X, given the image data Y and explicit range data (which gives a prior probability for the height of the surface at a particular location on the surface):

$$P(X|Y,Z) \propto \prod_{\substack{i,j,k \\ i<j<k}} \exp(-\psi^t_{ijk}(x_i,x_j,x_k,y_i)) \prod_i \exp(-\psi_i(x_i,z_i)) \quad (5)$$

As far as possible we follow the notation of [6]. The energy of a particular corner vertex node taking on a particular value is:

$$\psi^t_{ijk}(x_i,x_j,x_k,y_i,L) = |y_i - |\overrightarrow{n} \cdot \overrightarrow{L}|| \quad (6)$$

where y_i is the pixel intensity of the pixel contained by the three vertex nodes, x_i is the height of the vertex at right angles to the other two, x_j is the height of the vertex horizontal to that vertex, and x_k is the height of the remaining vertex. We thus have the following equations for the partial derivatives in the height (with respect to change in position in the horizontal and vertical directions on the image) in terms of the three heights of the surface at the points where the three corner vertex nodes lie:

$$p = \partial u/\partial x_1 = (x_j - x_i)/\partial x_1 \quad (7)$$

and

$$q = \partial u/\partial x_2 = (x_j - x_i)/\partial x_2. \quad (8)$$

Assuming square pixels and an overall scale of one unit per pixel width, we set ∂x_1 and ∂x_2 to 1. We can then use Eqn. 2 to calculate the plane.

As in [6] we can extend the energy function to include static scene/moving lightsource information (on the assumption that all points on the surface are always visible to both the camera and to all light sources). We adjust Eqn. 6 above to be:

$$\psi^t_{ijk}(x_i,x_j,x_k,y_i,\overrightarrow{L}) = \sum_{n=1}^{N} |y_{ni} - |\overrightarrow{n} \cdot \overrightarrow{L_n}||, \quad (9)$$

where N is the number of images (one for each lightsource), and n iterates over each of the images, so y_{ni} is the pixel intensity of the pixel contained by the three vertex nodes, in the n_{th} image. $\overrightarrow{L_n}$ is the lightsource direction in the n_{th} image.

3.1 Boundary conditions and range data

In Shape from Shading algorithms, it is usually necessary to establish some boundary conditions, since all surface heights calculated (if only shape from shading be used) are relative to each other, but not against any fixed frame of reference. In addition, the specification of boundary conditions may solve some of the ambiguities, since there are usually a number of surfaces which may generate a particular intensity map under particular lighting conditions. The MRF formulation allows such boundary conditions and range data to take on the form of either hard or soft constraints. Each corner vertex node x_i may be given a prior probability on the heights of its state vector, such that

$$p(X_i = l) \propto \exp(-(h(X_i, l) - u(X_i))), \tag{10}$$

where $u(X_i)$ is the specified range or depth at X_i, and $h(X_i, l)$ gives the height corresponding to label l for X_i. Whether the point is given a value because it lies on a known boundary, or because we have range data about the point, the point is treated the same way. The potential $\psi_i(x_i, z_i)$ in Eqn. 5 incorporates such a constraint.

4 Minimizing lattice energy using Gibbs sampling

The labels of nodes in a Markov Random field may be estimated using Gibbs sampling. It is important for this algorithm to randomly initialize the states of all corner vertex nodes: if they are given similar values (e.g. all zeros) this can lead to bad convergence. The overall Gibbs sampling algorithm is:

1. Randomly initalize state vector labels for all corner vertex nodes
2. for $t \leftarrow 1..N$
3. for $i \leftarrow 1..M$
4. $j \leftarrow$perm(i)
5. Collect energies $E(X_j = k|N(X_j))$ of each possible label k of current node X_j according to Eqn. 13.
6. Calculate the probabilities of each state given the energy for each state:
   $p(X_j = k|N(X_j)) =$
   $f(k, E(X_j|N(X_j)), kT(t))$
7. Choose a label $L(X_j)$ for this node by randomly sampling from the pdf for the state probabilities of this node.
8. end
9. end

In the above algorithm, N is the number of times we traverse the lattice, M is the number of nodes, perm(i) denotes the index into the set of corner vertex nodes, randomly permuted, so the nodes are visited in a random order (we can also adjust the order so the samples begin in areas of high certainty). $T(t)$ denotes the temperature at a particular iteration (given a temperature schedule for simulated annealing). Function $f(\cdot)$ is usually of the form:

$$f(k, E(X|N(X)), kT(t)) = \frac{\exp(-E(X = k|N(X))/kT(t))}{\sum_{n=1}^{S} \exp(-E(X = n|N(X))/kT(t))}, \tag{11}$$

where S is the number of label values which X could take on. The temperature schedule we used was

$$kT(t) = \text{maxTemp} - \frac{t}{M}(\text{maxTemp} - \text{minTemp}), \tag{12}$$

with maxTemp $= 50$ and minTemp $= 0.001$, i.e. the temperature decreases linearly per iteration down to a value of almost zero.

The following equation describes the calculation of the local clique energy of a vertex node given its neighbours. All energies of all cliques in which this node appears are summed, with all nodes given particular labels. We calculate the energy of a node's state as:

$$\begin{aligned} E(X = k|N(X)) = & \textstyle\sum_{i=1}^{n(X)} \sum_{j=1}^{n(X)} \cdots \\ & \psi_{X,N(X,i),N(X,j)}(k, L(N(X,i)), L(N(X,j))) \cdots \\ & +\psi_{N(X,i),X,N(X,j)}(L(N(X,i)), k, L(N(X,j))) \cdots \\ & +\psi_{N(X,i),N(X,j),X}(L(N(X,i)), L(N(X,j)), k), \end{aligned} \tag{13}$$

where $N(X)$ denotes the neighbours of node X, (we overload the notation so that $N(X, i)$ denotes the i_{th} neighbour of node X), $n(X)$ denotes the number of neighbours for node X, and $L(X)$ denotes the current label of node X. The function $\psi_{XYZ}(\cdot)$ is so specified that its value is zero if given nodes XYZ there is no triangular energy term over nodes X, Y, Z (i.e. if connected they are not such a triangular plane or smoothing triplet as we have specified).

5 Multi-Resolution in state vectors for corner vertex node elevations

This MRF formulation allows us to use a coarse-to-fine multi-resolution manner (as in [6]): for each of R resolutions, after N Gibbs sampling iterations (in one such iteration we sample each of the vertex nodes once), we may iterate through each of the M corner vertex nodes and adjust the heights which each element in the vertex node's state vector corresponds to, so that we may home in on a closer approximation of the correct value. The multi-resolution algorithm is described as follows:

1. for $r \leftarrow 1..R$
2. for $i \leftarrow 1..N$
3. Decrease temperature: $T \leftarrow T - c$
4. for $j \leftarrow 1..M$
6. Sample state estimate for j_{th} corner vertex node using Gibbs sampling at temperature T
7. end
8. end
9. for $j \leftarrow 1..M$

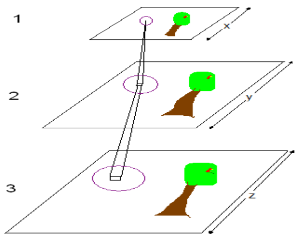

Figure 3. An image (bottom) is compressed twice, according to some scale factor. The probabilistic height map calculated at resolution 3 is then expanded and used as an initial value for calculating the height map at resolution 2, and so on.

10. Increase height resolution of j_{th} corner vertex node
11. end
12. end

In step 10. above, the new set of heights for node j are then calculated using a reduced height resolution about the Gibbs sampled value.

6 Spatial Multi-Resolution

We have implemented this algorithm to work on a spatially multi-resolution framework. The steps for this are as follows:

1. for $i \leftarrow 1..R$
2. for $n \leftarrow 1..P$
3. Compress n_{th} image at resolution i, Y_i^n to Y_{i+1}^n
4. end
5. end
6. for $i \leftarrow R..1$
7. Run Gibbs sampling algorithm on $\overrightarrow{Y}_i$ initialized about expanded values in H_{i+1} (if $i+1 < R$) to form height map H_i (using a scaling width appropriate forthe current spatial resolution).
8. end
9. Run Gibbs sampling algorithm on height map H_1

In the above scheme P is the number of images of the surface. It is necessary to remember the scaling width used at each resolution, since the distances x and y between corner vertex nodes as shown in Fig. 2 will depend on the current spatial resolution. At the lowest resolution (the original image), x and y are 1, but at a scale factor of 2, x and y are 2, and so on, depending on the spatial decomposition scheme used. The scaling value affects Eqn. 7 and Eqn. 8, and all corresponding energy terms. In Fig. 3 we can see a depiction of the wavelet-style multi-resolution image decomposition scheme. The square region in the circle represents the same region on all images, i.e. on the object surface.

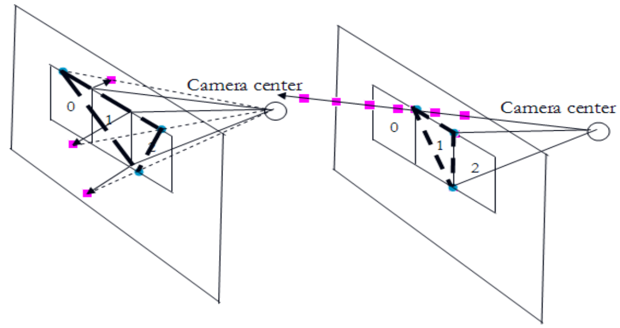

Figure 4. On the left, we see that if the labels parameterize the perpendicular distance from the image plane, different values for any connected triplet may cause the interior triangle to span many image pixels. On the right, if the ray-style parameterization is chosen, the interior triangle is always within a single pixel.

7 Image projections

If we wish to deal in projective coordinates we could either assert that the state on a corner vertex node corresponds to its depth behind the camera plane (Fig. 4, left), or that its elevation refers to its distance along the ray from the camera center to the corner of the pixel (Fig. 4, right). The former case has the advantage of allowing a constant density of corner points along the surface of the object, but corner vertex nodes may be associated with different image pixels at different elevations (Fig. 4, left). The latter alternative allows corner vertex nodes to refer to the same image pixel regardless of their elevation (Fig. 4, right). However the calculation of the energy term changes from a scaled variant of Eqn. 5, and if the camera moves with respect to the surface, we lose any advantage offered by this parameterization. We chose the first option in our implementation. In Fig. 4, the squares represent the discretized locations corresponding to particular labels on the state vector for any corner vertex nodes. The intersections with the image plane on projection are shown as circles. The large numbered squares inside the image plane are pixels. The energy function for corner vertex node triangles (6) now becomes:

$$\psi^t_{ijk}(x_i, x_j, x_k, P, \mathbf{Y}, \overrightarrow{L}) = \cdots$$
$$|a(Y, P \cdot D(x_i), P \cdot D(x_j), P \cdot D(x_k)) - |\overrightarrow{n} \cdot \overrightarrow{L}||, \quad (14)$$

where P is the camera projection matrix, Y represents the image data, $D(\cdot)$ is a function which returns the 3D homogeneous coordinate of the corner vertex node in its argument, and $a(\cdot)$ is a function which averages the intensities of pixels interior to the three given 2D image coordinates, given an image Y. Similarly, Eqn. 9 for multiple light-sources becomes

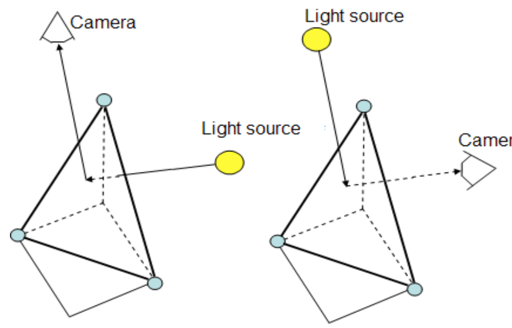

Figure 5. On the left, a triangle is hidden from the light source but is visible to the camera. This surface region should therefore not appear to be illuminated by the light source. Similarly, on the right, the pixel(s) on the triangle should be dark because the light source and camera are on different sides of the triangle.

$$\psi^t_{ijk}(x_i, x_j, x_k, P, \overrightarrow{Y}, \overrightarrow{L}) = \sum_{n=1}^{N} \cdots$$
$$|a(Y_n, P \cdot D(x_i), P \cdot D(x_j), P \cdot D(x_k)) - |\overrightarrow{n} \cdot \overrightarrow{L_n}||, \quad (15)$$

where Y_n is the n_{th} image and $\overrightarrow{L_n}$ is the n_{th} light source direction.

8 Simple occlusion of surface from light sources

The triangle face formed by three connected vertex nodes is not always visible to both the camera and the light source. In Fig. 5 we show a triangle which is visible to the camera but not to the light source. The (expected) value for this pixel should therefore not involve the summation of the reflectance intensity for this light source (this pixel should be dark).

8.1 Simple occlusion with affine camera, multiple light sources

Under the affine camera assumption, the energy term from Eqn. 9 becomes

$$\psi^t_{ijk}(x_i, x_j, x_k, y_i, \overrightarrow{L}, C) = \sum_{n=1}^{N} |y_{ni} - t|, \qquad \text{where}$$

$$t = \begin{cases} |\overrightarrow{n} \cdot \overrightarrow{L_n}| & \text{if } \overrightarrow{n} \cdot \overrightarrow{L_n} < 0 \\ & \text{and } \overrightarrow{n} \cdot (D(x_i) - \overrightarrow{C}) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

In the above equation, $D(x_i) - \overrightarrow{C}$ is the vector from the camera center $\overrightarrow{C}$ to the 3D location of one of the corner vertex nodes (we assume the points are relatively close together, so this condition is sufficient to determine that the surface is facing the camera).

8.2 Simple occlusion with projective camera, multiple light sources

The following is the energy term for triplets where a projective camera with simple occlusion and multiple light sources is used.

$$\psi^t_{ijk}(x_i, x_j, x_k, P, \overrightarrow{Y}, \overrightarrow{L}, C) =$$
$$\sum_{n=1}^{N} |a(Y_n, P \cdot D(x_i), P \cdot D(x_j), P \cdot D(x_k)) - t|, \quad (17)$$

where

$$t = \begin{cases} |\overrightarrow{n} \cdot \overrightarrow{L_n}| & \text{if } \overrightarrow{n} \cdot \overrightarrow{L_n} < 0 \\ & \text{and } \overrightarrow{n} \cdot (D(x_i) - \overrightarrow{C}) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (18)$$

where Y_n is the n_{th} image and $\overrightarrow{L_n}$ is the n_{th} light source direction.

9 Results

We tested the algorithm on real and synthetic data: the synthetic data was created by generating random smooth surfaces, and calculating the image of those shapes under the Lambertian model. For real data we used images of everyday objects, and some images of individual froth bubbles from a mineral ore flotation cell (ground truth was unavailable for these).

9.1 Synthetic Data

For this synthetic data, we generate some smooth surfaces and supply random lighting directions and the camera parameters for a single projective camera to render the intensity map of the image under Lambertian assumptions (as in Fig. 6). In Table 1 are shown results for the algorithm run without spatial multi-resolution. We display the average errors for the generated vs. reconstructed surfaces. Table 2 shows the results of the spatial multi-resolution version of the algorithm. In all of our trials, the correct boundary conditions along the perimiter of the surface were given as soft constraints to the algorithm. Each of the tables is populated with average error entries which were calculated as follows:

$$e = \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{M} |(g(i) - c(i))|, \quad (19)$$

where $g(i)$ is the true height at corner node vertex i, $c(i)$ is the height calculated for that corner vertex node by the algorithm, M is the number of corner vertex nodes, and N is the number of trial runs.

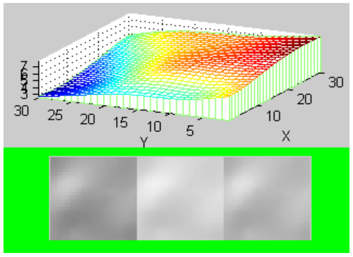

Figure 6. An example of a synthetic surface and its corresponding images from the same camera, with different lighting directions.

Image Width Num iters	8	24	30	40	50
8000	12.2	145.3	1054.7	1117.4	2326.7
12000	10.9	135.7	960.8	953.8	2082.3
20000	9.3	121.4	772.5	780.3	1841.1
35000	8.8	116.4	453.3	601.5	1720.3

Table 1. Algorithm run without spatial multi-resolution: Three reflectance maps were used, four resolutions. Entries in the table indicate the average error between the calculated and synthesized surfaces.

Image Width Num iters	8	24	30	40	50
8000	12.4	114.3	814.4	938.5	1402.8
12000	11.2	102.8	715.1	852.6	1155.5
20000	9.3	99.9	402.2	603.2	842.4
35000	8.8	85.3	302.1	403.1	650.1

Table 2. Algorithm run with spatial multi-resolution: Three reflectance maps were used, four resolutions. Entries in the table indicate the average error between the calculated and synthesized surfaces. Our tests suggest that with more iterations and more resolutions, the error would decrease further, at the expense of more computation time.

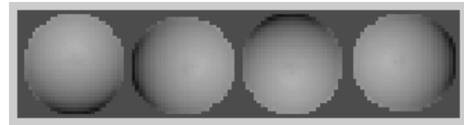

Figure 7. An egg (real smoothed images) with illumination from four different directions.

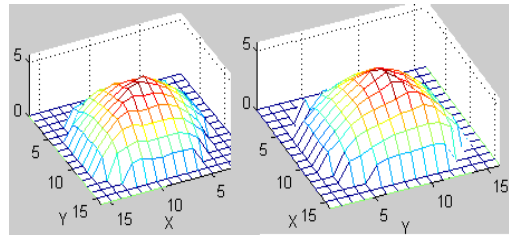

Figure 8. Reconstruction of egg in Fig.7 (this object was far from Lambertian; it was necessary to do some histogram equalization on the images first to approximate a Lambertian surface. On a synthetic egg-shaped surface the algorithm performed fine).

9.2 Real Data

For the real data, we took pictures of some objects: an egg (Fig. 7) and some bubbles from mineral ore flotation cells (Fig. 9, left, middle). The algorithm was run at various height resolutions, and the run times at each of the resolutions is shown. In the real images, noise was ameliorated by convolving them with a Gaussian kernel. We can see the reconstructions from these images in Figs. 8, 10, 12. The reconstruction in Fig. 10 is poor because we haven't modelled the reflectance well. For the bubbles, only an overhead image was available.

9.3 Algorithm running time

The algorithm running times for different numbers of iterations may be viewed in Table. 3. The running time is roughly of order $O(N \cdot M \cdot S \cdot R)$, where N is the number of iterations over each corner vertex node, for each of R resolutions, if there are M corner vertex nodes, each with S possible height labels (here we assume a fixed set of energy terms, different energy terms have different complexity).

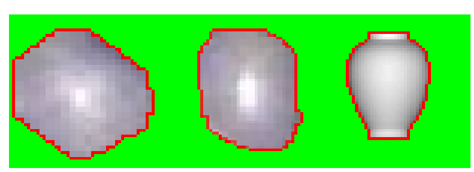

Figure 9. Two bubbles and the classic vase shape.

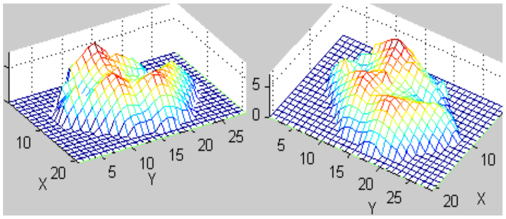

Figure 10. Poor reconstruction of left bubble in Fig. 9. Smoothing energy terms were used.

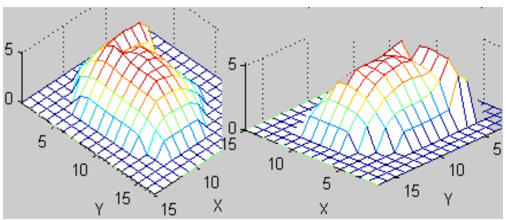

Figure 11. Good reconstruction of middle bubble in Fig. 9. Smoothing energy terms were used.

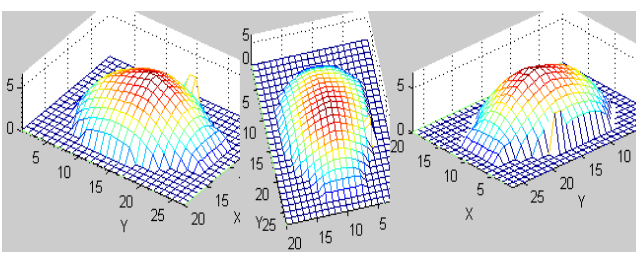

Figure 12. Good reconstruction of (synthetic) vase in Fig. 9. Smoothing energy terms were used.

Image Width Num iters	10	20	30	40	50
8000	4	14	34	60	89
12000	6	23	50	88	138
20000	10	38	84	147	232
35000	16	73	146	254	398

Table 3. Time taken for algorithm, in minutes, for 4 resolutions per height map, 20 labels per node, 1 energy term, 3 reflectance maps. (Run on Intel Centrino 1.86 GHz).

10 Conclusion

The algorithm has been tested in its functioning at a spatial multi-resolution level with a projective camera (projective rendering of generated hypothetical surfaces). Like LBP-SFS [6], this method can incorporate both hard and soft constraints on the boundary conditions of the surface and range data at points on the surface. It also provides a natural way incorporate multiple reflectance maps. Different reflectance models per surface can be easily accounted for in the energy terms. This algorithm requires less computation time and storage than LBP-SFS, and the support for projective camera is more appropriate for real world problems. However, with current hardware and large images, the algorithm is very slow to converge.

11 Acknowledgements

The authors are grateful for the financial support given by the National Research Foundation of South Africa, and Anglo Platinum Ltd and Rio Tinto via the Centre for Minerals Research at the University of Cape Town.

References

[1] B.K.P.Horn. *Obtaining Shape from Shading Information*. McGraw-Hill, 1975.

[2] A. Bruss. The eikonal equation: Some results applicable to computer vision. *Journal of Mathematical Physics*, pages 890–896, 1982.

[3] M. G. Crandall and P. Lions. Viscosity solution of hamilton-jacobi equations. *Trans. Amer. Math. Soc.*, pages 1–42, 1983.

[4] P. Daniel and J.-D. Durou. *From Deterministic to Stochastic Methods for Shape from Shading*. In Proc. 4th Asian Conf. on Comp. Vis., 2000.

[5] J.-D. Durou, M. Falcone, and M. Sagona. Numerical methods for shape from shading: A survey with benchmarks. *CVIU*, 2004.

[6] M. Louw, F. Nicolls, and D. Bradshaw. A loopy belief propagation approach to the shape from shading problem. *In Proc. 2nd Int. Conf. on Computer Vision Theory and Applications*, 2007.

[7] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamiltonian-jacobi formulations. *J. Comput. Phys.*, pages 12–49, 1988.

[8] A. Pentland. Local shading analysis. *IEEE. transactions on Pattern Analysis and Machine Intelligence*, pages 170–187, 1984.

[9] R. Szeliski. Fast shape from shading. *Computer Vision, Graphics, Image Processing: Image Understanding*, pages 129–153, 1994.

[10] P. Tsai and M. Shah. Shape from shading using linear approximation. *Image and Vision Computing Journal*, pages 187–198, 1994.

[11] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *PAMI*, 1999.