A faster MRF optimization based method for Shape from Shading using Gibbs sampling with quadruplet cliques.

Markus Louw *Fred Nicolls*

Department of Electrical Engineering
University of Cape Town, South Africa
markus.louw@gmail.com

Abstract

This paper extends the MRF formulation approach developed in [7] and [6] for solving the shape from shading problem. Our method extends the Gibbs sampling approach to solve an MRF formulation which characterizes the Shape from Shading (SFS) problem under Lambertian reflectance conditions (the algorithm is extensible to other lighting models). Our method uses a simpler set of energy functions (on point quadruplets), which is faster to converge, but less accurate.

1. Introduction and Literature Review

Two surveys, [12] (1999), and [5] (2004) describe the history of Shape from Shading algorithms. In the former, SFS approaches are classified into minimization (e.g. [10]), propagation (e.g. [8]), local (e.g. [9]), or linear (e.g. [11]) 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 [12].

This work builds on that of [6], called Gibbs Multi-Scale Projective Multi-Res SFS with Occlusion handling (GMPM-SFS), in which a Markov Random Field formulation for the labels of points on a lattice is developed. In that case we minimize 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.

In this paper we change the energy function which is minimized by increasing the clique sizes of the Markov Random Field. This approach requires us to treat the observed reflectance map data (image) as if each pixel were the reflectance of light off a single plane through the four corner vertex nodes about the pixel. Using Gibbs sampling means there is a computational speed increase (if we used LBP, it would take much longer). This algorithm is called Gibbs Multi-Scale Projective Multi-Res with clique quadruplets SFS, or GMPM4-SFS.

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

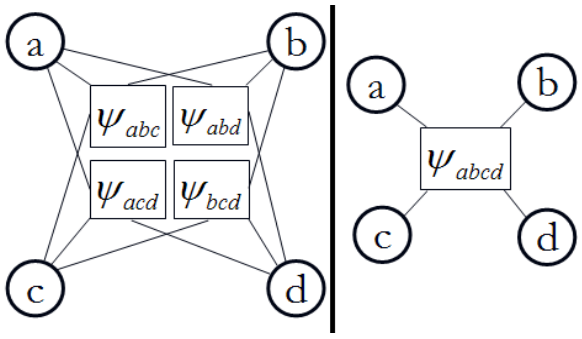

Figure 1: On the left is shown the original MRF topology w.r.t. energy terms over cliques of corner vertex node triplets. On the right is shown our new energy term, associated with the quadruplet and represented by a square. It is connected by lines to their corresponding corner vertex nodes.

the surface which takes the normal at point x as an argument. The surface normal may be calculated as

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

where

$$p = \partial u/\partial x_1 \tag{3}$$

and

$$q = \partial u/\partial x_2 \tag{4}$$

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

$$R(\overrightarrow{n}(x)) = w \cdot \overrightarrow{n}(x) \tag{5}$$

Hereafter (until section 6), without loss of generality (but assuming all surface points are visible to both camera and light source) we assume the light source is in the same direction as the camera, which produces an orthogonal projection.

3. MRF formulation to solve SFS

This algorithm calculates an optimal set of labels for the height at each corner vertex on the image. A corner vertex occurs at the corner of a pixel; 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

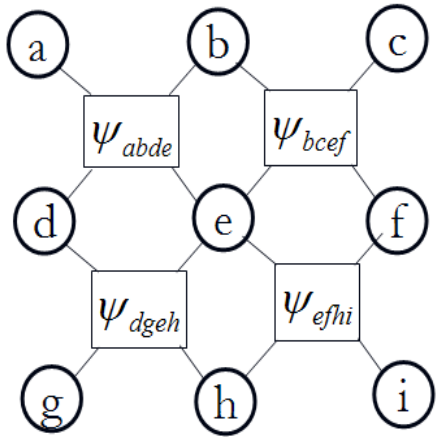

Figure 2: Depiction of the locations of 9 corner vertex nodes (round) over 4 pixels. The energy terms associated with each quadruplet are represented by squares. The energy terms are connected by lines to their corresponding vertex nodes.

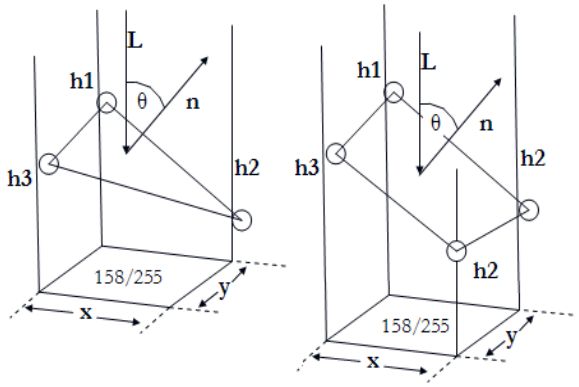

Figure 3: On the left is the plane generated by corner vertex nodes each at a particular height. The inner product of the plane's normal and the light source's direction gives the pixel intensity at the pixel corresponding to those three corner vertex nodes (Lambertian reflectance model), which in this diagram is 158/255. On the right, is shown the best fit/average plane given four corner vertex nodes.

a probability to be assigned to that configuration for that triplet, given the observed reflectance for that image region. A diagram for the topology for this scheme with pixels, corner vertex nodes, and the corresponding energy terms for each quadruplet, is shown in Fig. 2. 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. 3. Next we define a Markov Random Field (MRF) on this set of vertex nodes X, given the image data Y and explicit range data Z (which gives a prior probability for the height of the surface at a particular location on the surface):

$$P(X|Y,Z) \propto \prod_{\substack{i,j,k,l \\ i<j<k<l}} \exp(-\psi^t_{ijkl}(x_i,x_j,x_k,x_l,y_i))... \prod_i \exp(-\psi_i(x_i,z_i)) \quad (6)$$

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

$$\psi^t_{ijkl}(x_i,x_j,x_k,x_l,y_i,L) = |y_i - |\overrightarrow{n}\cdot\overrightarrow{L}|| \quad (7)$$

where y_i is the pixel intensity of the pixel contained by the three vertex nodes; x_i, x_j, x_k, x_l are the corner vertex node labels.

We now describe two ways for generating a plane normal from four 3D points. Since we want to approximate the surface of the object interior to these four points as a plane (so that we may use its normal to calculate the reflectance), we may either use Singular Value Decomposition (SVD) to fit the best plane through the four points, or for each triplet in the quadruplet, calculate the normal, then average the four normals.

3.1. Fitting a plane with Singular Value Decomposition

We recall that in homogenous coordinates, when a 3D point $\mathbf{X} = [x \;\; y \;\; z \;\; 1]$ lies on a plane $\mathbf{p}$, the inner product is zero, i.e. $\mathbf{X}\cdot\mathbf{p} = 0$. Therefore, to calculate the plane with the smallest least squares error through the four points X_i, we calculate the SVD:

$$USV^T = \text{SVD}\begin{pmatrix}\mathbf{X}_1\\ \mathbf{X}_2\\ \mathbf{X}_3\\ \mathbf{X}_4\end{pmatrix} \quad (8)$$

after which the coeffecients of $\mathbf{p}$ are the elements of the last column of V^T. While there are advantages to using this method we found the computation time per SVD evaluation to be much too high, and therefore used the following method:

3.2. Averaging the plane normals over each of four interior triplets

If we have four points, x_i, x_j, x_k, x_l, we take each triplet in turn $(x_i, x_j, x_k,)$, (x_i, x_j, x_l), (x_i, x_k, x_l), (x_j, x_k, x_l), and form the normal for the plane through the triplet. E.g. for (x_i, x_j, x_k), 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 on which the three corner vertex nodes lie:

$$p = \partial u/\partial x_1 = (x_j - x_i)/\partial x_1 \quad (9)$$

and

$$q = \partial u/\partial x_2 = (x_j - x_i)/\partial x_2. \quad (10)$$

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. Repeating this for the other triplets, we calculate the average plane normal $\overrightarrow{n}$.

As in [7] we can extend the energy function to include static scene/moving light source 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. 7 above to be:

$$\psi^t_{ijkl}(x_i, x_j, x_k, x_l, y_i, \overrightarrow{L}) = \sum_{n=1}^{N} |y_{ni} - |\overrightarrow{n} \cdot \overrightarrow{L_n}||, \quad (11)$$

where N is the number of images (one for each light source), 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 light source direction in the n_{th} image.

3.3. 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

$$p(X_i = l) \propto \exp(-(h(X_i, l) - u(X_i))), \quad (12)$$

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.

4. Minimizing energy of MRF using Gibbs sampling

The labels of nodes in a Markov Random field may be estimated using Gibbs sampling.

1. for $t \leftarrow 1..M$
2. for $i \leftarrow 1..N$
3. $j \leftarrow$perm(i)
4. Collect energies $E(X_j = k|N(X_j))$ of each possible label k of current node X according to Eqn. 15
5. Calculate the probabilities of each state given the energy for each state: $p(X_j = k|N(X_i)) = f(E(X_j = k|N(X)), kT(t))$
6. Choose a state for this node $L(X_j)$ by randomly sampling from the pdf for the states of this node.
7. end
8. end

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

$$f(E(X = k|N(X)), kT(t)) = \frac{\exp(-E(X = k|N(X))/kT(t))}{\sum_{n=1}^{M} \exp(-E(X = n|N(X))/kT(t))} \quad (13)$$

The temperature schedule we used was

$$kT(t) = \text{maxTemp} - \frac{t}{M}(\text{maxTemp} - \text{minTemp}), \quad (14)$$

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. A node's state probability depends only on the energy terms in its Markov neighbourhood, which we calculate as:

$$\begin{aligned} E(X = z|N(X)) = \sum_{i=1}^{n(X)} \sum_{j=1}^{n(X)} \sum_{k=1}^{n(X)} \cdots \\ \psi_{X,N(X,i),N(X,j),N(X,k)}(z, L(N(X,i)), L(N(X,j)), \\ L(N(X,k))) \cdots \\ +\psi_{N(X,i),X,N(X,j),N(X,k)}(L(N(X,i)), z, L(N(X,j)), \\ L(N(X,k))) \cdots \\ +\psi_{N(X,i),X,N(X,j),N(X,k)}(L(N(X,i)), L(N(X,j)), z, \\ L(N(X,k))) \cdots \\ +\psi_{N(X,i),N(X,j),N(X,k),X}(L(N(X,i)), L(N(X,j)), \\ L(N(X,k)), z), \end{aligned} \quad (15)$$

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 $\psi_{WXYZ}(\cdot)$ is so specified that its value is zero if given nodes $WXYZ$ there is no energy term over nodes W, X, Y, Z (i.e. if they are not corners of the same square surface region).

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

The MRF formulation allows us to use a coarse-to-fine multi-resolution manner (as in [7], [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, and in this way "home in" on a closer approximation of the correct value.

6. Image projections

As in [6], we use a height label parameterization where the state on a corner vertex node corresponds to its depth behind the camera plane (Fig. 4, left). This formulation is general in that the same parameterization works wherever the camera is

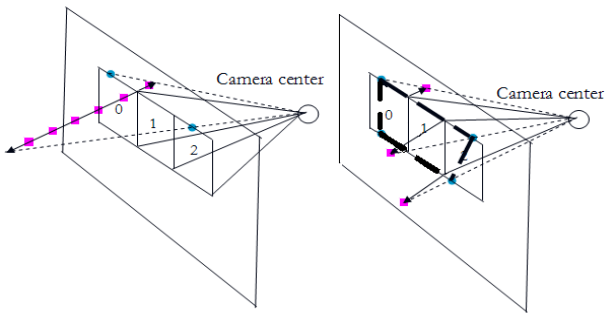

Figure 4: On the left, the label of a corner vertex node refers to its height perpendicular to the image plane. On the right, we see that if the labels parameterize the perpendicular distance from the image plane, different values for any connected quadruplet may cause the interior quadrilateral to span many image pixels.

in the scene, since corner vertex nodes may be associated with different image pixels at different elevations (Fig. 4, right).

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 (Eqn. 7) now becomes:

$$\psi^t_{ijkl}(x_i, x_j, x_k, x_l, P, \overrightarrow{Y}, \overrightarrow{L}) = \cdots$$
$$|a(Y, P \cdot D(x_i), P \cdot D(x_j), P \cdot D(x_k), P \cdot D(x_l)) - |\overrightarrow{n} \cdot \overrightarrow{L}|| \quad (16)$$

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. 11 for multiple light-sources becomes

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

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

6.1. Smoothing

If we are using a small number of images, and if the energy terms used is that are simply those shown in Fig. 2 and written in Eqn. 6, it is likely that the algorithm may converge to a solution (digital elevation map) with undesirable high frequency components. As in [7], we can use two types of smoothing terms, viz. smoothing with quadruplets of varying size (see Fig. 5), and smoothing with collinear point triplets (see Fig. 6). A corresponding energy term for each of these may be added. Details for the energy terms may be found in [7].

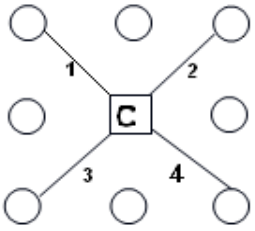

Figure 5: The connectivity of a quadruplet energy term which spans four pixels and may be used for smoothing.

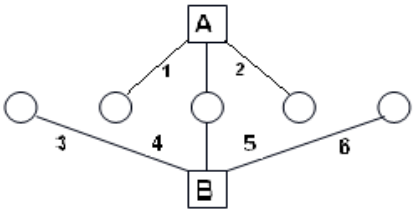

Figure 6: Energy term topology for first and second order smoothing on collinear triplets of corner vertex nodes. The corner vertex nodes are circles and energy terms are squares. Energy term "A" enforces smoothing over a small scale, while energy term "B" enforces smoothing over a larger scale. The numbers correspond to pixels (there is one pixel interior to four corner vertices).

7. 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).

7.1. 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 (Fig. 7).

Tables 1 and 2 shows the results of the algorithm run with

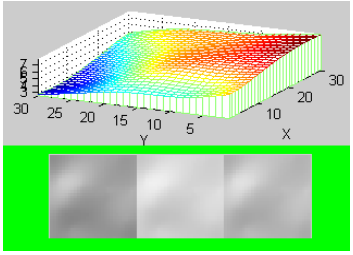

Figure 7: An example of a synthetic surface and its corresponding images from the same camera, with different lighting directions.

Image Width Num iters	40	50	60	70
1000	812.3	1551.2	1576.5	5062.7
5000	756.1	1168.8	1500.4	4334.1
10000	676.9	1089.5	1445.2	4049.6

Table 1: Three images given (reflectance maps for 3 light sources). Three height resolutions, two spatial resolutions. Entries in the table indicate the average error between the calculated and synthesized surfaces.

Image Width Num iters	40	50	60	70
1000	640.0	1135.5	1199.67	3678.2
5000	587.8	848.8	1125.4	3248.5
10000	525.9	800.5	1049.1	2946.3

Table 2: Four images given (reflectance maps for 4 light sources). Three height resolutions, two spatial resolutions, 30 labels per node. Entries in the table indicate the average error between the calculated and synthesized surfaces.

spatial multi-resolution, with 3 and 4 reflectance maps. In all of our trials, the correct boundary conditions along the perimeter of the surface were given as hard constraints to the algorithm. Each of the tables is populated with average error entries which were calculated as follows:

$$e = \frac{1}{N} \sum_{t=1}^{N} \sum_{i=1}^{M} |(g(i) - c(i))|, \tag{17}$$

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 (we used between 5 and 30 trials per entry). In Table 3, we see the performance of the previous version of algorithm (GMPM-SFS). In theory, only a coarser approximation may be reached as the energy function is based on a coarser plane approximation, although we couldn't demonstrate this with synthetic data as the time required for convergence of GMPM-SFS is prohibitive.

7.2. Algorithm running time

The algorithm running times for different numbers of iterations may be viewed in Table. 4. The running time is roughly of order

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 3: GMPM-SFS algorithm run with spatial multi-resolution: Three reflectance maps were used, four resolutions, 30 labels per node. Entries in the table indicate the average error between the calculated and synthesized surfaces (these results taken from [6]).

Image Width Num iters	40	50	60	70
1000	7	12	16	24
5000	32	58	76	116
10000	68	148	154	219

Table 4: Time taken for multiscale version of GMPM4-SFS, in minutes, for 3 resolutions per height map, 30 labels per node, 1 energy term, 3 reflectance maps. (Run on AMD Athlon 2.4 GHz).

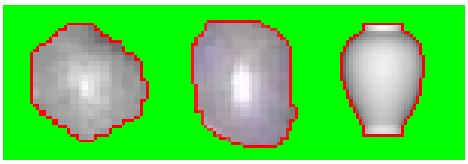

Figure 8: Two bubbles and the classic vase shape.

$O(N \cdot M \cdot L \cdot R)$, 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 L possible height labels (here we assume a fixed number of energy terms, different energy terms have different complexity.).

7.3. Real Data

For the real data, we took pictures of some some bubbles from mineral ore flotation cells (Fig. 8, 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 applying convolution with a Gaussian kernel.

8. Conclusion

The algorithm has been tested in its functioning at a spatial multiresolution level, and with a projective camera (projective rendering of generated hypothetical surfaces). Like GMPM-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. Larger images can be processed using this method (in a given time), than with GMPM-SFS. Different reflectance models per surface can be easily accounted for in the energy terms. The algorithm supports a projective cam-

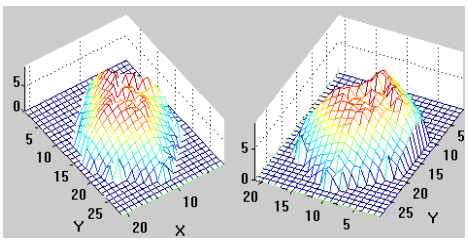

Figure 9: Bubble reconstruction. Note that the bubble is distorted. GMPM4-SFS does not run well when given a single image, and the distortion is due to the plane approximation necessary for a quadruplet energy term.

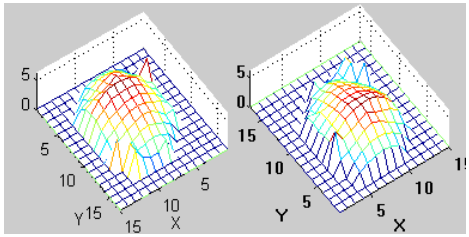

Figure 10: Bubble reconstruction.

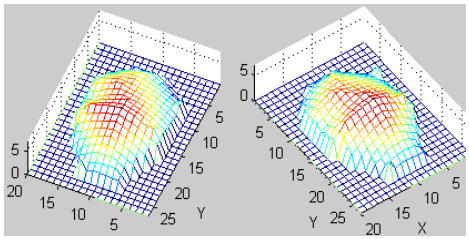

Figure 11: Vase reconstruction. Note that the vase is distorted. GMPM4-SFS does not run well when given a single image, and the distortion is due to the plane approximation necessary for a quadruplet energy term.

era model, though we have not developed local or global self-occlusion.

9. Future Work

We are currently experimenting with using the method here proposed to gain a good initial estimate of the surface, then continuing refining the surface further using the energy function (still with Gibbs sampling) described in [6], which is more exact. This method takes us one step closer to incorporating MRF based SFS for improving dense stereo calculation.

10. 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.

11. References

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

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

[3] M. G. Crandall and P.L. 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] Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape from shading: A survey with benchmarks. *CVIU*, 2004.

[6] M. Louw and F. Nicolls. A spatially multiresolution, `mrf` optimization based approach to the shape from shading problem. *In Proc. 7th IASTED Int. Conf. Visualization, Imaging, and Image Processing*, 2007.

[7] 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.

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

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

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

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

[12] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *PAMI*, 1999.