# User-guided Shape from Shading to Reconstruct Fine Details from a Single Photograph

Alexandre Meyer Hector M. Briceño Saïda Bouakaz

Université de Lyon, LIRIS[**], France.

**Abstract.** Many real objects, such as faces, sculptures, or low-reliefs are composed of many detailed parts that can not be easily modeled by an artist nor by 3D scanning. In this paper, we propose a new shape from shading (SfS) approach to rapidly model details of these objects such as wrinkles and reliefs of surfaces from one photograph. The method first determines the surface's flat areas in the photograph. Then, it constructs a graph of relative altitudes between each of these flat areas. We circumvent the ill-posed problem of shape from shading by having the user set if some of these flat areas are a local maximum or a local minimum; additional points can be added by the user (*e.g.* at discontinuous creases) – this is the only user input. We use an intuitive mass-spring based minimization to determine the final position of these flat areas and a fast-marching method to generate the surface. This process can be iterated until the user is satisfied with the resulting surface. We illustrate our approach on real faces and low-relief photographs.

## 1 Introduction

Despite recent advances in surface modeling and deformation, creating photorealistic 3D models remains a difficult and time consuming task. Many real objects, such as people, faces, sculptures, masks or low-reliefs are composed of many detailed parts that can not be easily modeled by an artist. Alternatively, 3D scanning technology is still an expensive process. While much work has been devoted to using several photographs to build 3D models, or to rendering new views from many photographs, little work has been done to address the problem of modeling objects from a single photograph.

The fine aspects of these surfaces appear on photographs as a variation of shading. The methods that recover these features from the shading are called *S*hape from Shading (SfS). Nevertheless, it has been shown that this is an ill-posed problem [1, 2]: a solution does not necessarily exist and when it exists, it is not unique meaning that different surfaces may have produced a given image. Figure 1 illustrates this point: the image on the left may have been produced by both objects on the right.

In this paper, we propose a new practical Shape from Shading method which can be applied to a real photograph to help on the difficult problem of modeling fine aspect such as wrinkles and reliefs of surfaces. The ill-posed of shape from shading is solved by asking an user to decide whether some areas orthogonal to the viewing direction are a

[**] Laboratoire d'InfoRmatique en Images et Systèmes d'information UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon.

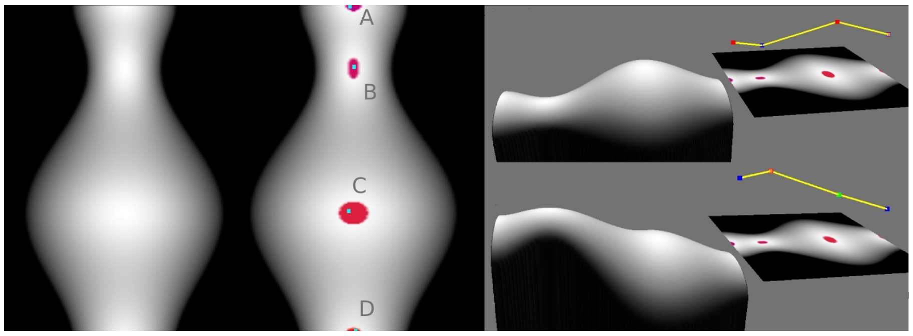

**Fig. 1.** Shape from Shading is an ill-posed problem: two different surfaces may produce the same shaded image. The left image may have been produced by several surfaces shown on the on the right. The highlights correspond to the flat areas. The upper right image was produced considering highlights A and C as peaks, and the lower right image considering only highlight B as a peak. Other combinations are possible.

local extrema (local maximum or local minimum) or not. The user information is propagated during a minimization process based on a mass-spring simulation. This mass-spring minimization has the advantage of presenting a graphical visualization which allows the user to interact during the computation according to his knowledge. Indeed, we have noticed that many minimization approaches like simulated annealing [3] which are fully automatic do not offer any convenient way of correcting reconstructed surface if it is incorrect. With our approach, the user can visually follow the intuitive mass-spring minimization and correct any errors in the subsequent reconstruction. Moreover, each time he wants to see the potential reconstructed surface, the fast marching method [4, 5] generates it in few seconds.

## 2 Related Work

The problem of surface reconstruction from single image can vary in the degree of user interaction: from fully automatic approaches to interactive modeling. Hoiem *et al.* in [6] proposed a fully automatic system that creates a 3D model of the scene made up of several texture-mapped planar billboards. Their approach captures the global geometry of the scene as planar. On a single image without any underlying lines (or planes), for example a face, it seems difficult to obtain better results without using shading information. The Shape from Shading (SfS) problem has been widely studied in the computer vision area. See [7–9] for a survey on SfS methods. The SfS problem is known as a difficult problem because of his ill-posedness [2]. Consequently, few approaches have been tried on real photographs. Courteille *et al.* in [10] propose a method to set flat a photograph of a curved sheet of paper in order to facilitate the character recognition. Prados *et al.* in [2, 11] have tried a method based on taking into account the light attenuation on face photographs with relatively good results. Zhu *et al.* in [12] tackle the ambiguities of shape from shading by a semi-definite programming relaxation process which flip patches and adjust heights until the result surface has no kinks.

To our knowledge, beside Zeng *et al.* [13] SfS approaches are mostly fully automatic in spite of the ill-posed nature of the problem. Zeng's method asks the user to enter some normal in order to determine in which direction the slope is going up to a local maximum. Once all local maxima are computed, they compute the relative altitude between each of them and the fast-marching algorithm generates the surface. Similarly to Zeng *et al.* , we believe that the ill-posed aspect may only be tackled with user interaction. Comparing to Zeng's approach we differ in the user input aspect. In Zeng's approach, the user has to enter enough normals to capture the small variation of surface such as wrinkles on a forehead. Our approach automatically computes all flat areas and mass-spring minimization is more visual. Thus, the user may interact more directly on the data (the mass-spring graph for us) and may explore different solutions as illustrated on Fig. 1 with the two plausible configurations.

## 3 Formulation and Overview of our Approach

Our technique takes as input a color image and produces as output an heightmap which is an image where each pixel stores a distance to the underlying plane. Input images are obtained with a camera using a flash as then only light source.

The first step of our approach is to compute the shading image from the RGB colored photograph. For that, we convert each RGB pixel into YUV color space and keep the luminance Y as shading. A more elaborate solution based on assumption of shading continuity is one proposed by Funt *et al.* in [14]: reflectance changes are located and removed from the luminance image by thresholding the gradient at locations of abrupt chromaticity change. More recently, Tappen *et al.* in [15] proposed a solution based on both color information and a classifier trained to recognize gray-scale patterns.

Once the shading image is computed, we have the classical SfS problem. In this part, 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 the camera direction by a point light source at infinity.

Our approach is decomposed as follows:

1. For each pixel of the RGB photograph, extract the shading value by converting it into YUV and taking the Y value (or with [14]'s or [15]'s methods).
2. Detect the flat regions (highlights) in the image which will become the vertices to our relative altitude graph which will guide the user-interaction and the reconstruction (Sect. 4).
3. Using fast-marching (for a recap see Sect. 3.1) we compute the relative altitude difference between the vertices in the relative-altitude graph (Sect. 4).
4. The user defines few vertices as peaks or saddles.
5. The position of the remaining vertices is computed by solving a spring-mass system over the vertices (Sect. 5), and a new surface is quickly computed
6. If features on the surface do not have a vertex associated with them, it is possible for the user to add new vertices to the relative-altitude graph.
7. These three last steps can me re-iterated until the user is satisfied with the reconstructed surface.

### 3.1 Shading Image Formation Model and Fast-Marching

We first review the shading image $I(x, y)$ formation model for a 3D Lambertian object. In our approach the camera and the light-source have the same position which we consider at infinity from the object; we then define the light source direction as $L = (0, 0, 1)$. The surface normal direction is given by $N_{x,y} = (\frac{\partial z_{x,y}}{\partial x}, \frac{\partial z_{x,y}}{\partial y}, 1)$. Notice that $N$ is not normalized. The shading image is the dot product of the light and the normalized surface normal, it is computed by:

$$I_{x,y} = L.\frac{N_{x,y}}{||N_{x,y}||} = \frac{1}{\sqrt{\frac{\partial z_{x,y}}{\partial x}^2 + \frac{\partial z_{x,y}}{\partial y}^2 + 1}}$$

With $\nabla z_{x,y} = (\frac{\partial z_{x,y}}{\partial x}, \frac{\partial z_{x,y}}{\partial y})$ we get $||\nabla z_{x,y}|| = \sqrt{I_{x,y}^{-2} - 1}$

This equation is known as the *Eikonal equation* which can be solved by the numerical algorithm *fast marching* which we recap coarsely here. More detailed information can be found in [4, 5, 16].

At initialization, all pixel's altitude $z_{x,y}$ are set to $\infty$ beside few pixels whose altitudes are known. All *known pixels* are put into a priority queue ordered by their altitude: smaller altitude first. The algorithm extracts pixels from the priority queue until it is empty. Starting from a known pixel, the altitude of its four-connected neighbors is updated and added to the queue. The altitude $z_{x,y}$ of pixel $(x, y)$ is updated as follows:

- Let $z_1 = min(z_{x-1,y}, z_{x+1,y})$ and $z_2 = min(z_{x,y-1}, z_{x,y+1})$.
- If $|z_1 - z_2| < ||\nabla z_{x,y}||$ then $z_{x,y} = \frac{z_1+z_2+\sqrt{2\times\nabla z_{x,y}^2-(z_1-z_2)^2}}{2}$
  else $z_{x,y} = min(z_1, z_2) + ||\nabla z_{x,y}||$;

## 4 Relative Altitude Graph

Our minimization process described in the next section is based on a relative-altitude graph mapped on the shading image. This section is dedicated to this graph computation.

**Vertices Detection** We define a graph over the photograph where the vertices correspond to highlights (high intensity values) on the shading image. These points correspond to singular points on the surface where the gradiant is 0 (*e.g.* local minima, local maxima, saddle) On the shading image, several adjacent pixels may have the same intensity, we thus consider only one area, and thus one vertex in the graph. Since we assume an orthographic camera, a vertex represent a flat area of the surface orthogonal to the viewing direction. We named them *OVD areas* for Orthogonal to the Viewing Direction. All these OVD areas are parallel because of the orthographic assumption. Theoretically, these OVD areas have a maximal shading value. Since on real photograph, light attenuation may appear, we define them by a threshold $T_{shading}$. To compute these OVD areas we consider all regions of 4-connected pixels with equivalent values.

The OVD areas are computed by a depth-first search algorithm on the shading image interpreted as a graph: two pixels are neighbor if their shading values are equal. Among all these areas, we keep as OVD areas only those with a shading value greater than $T_{shading}$ and where its neighboring area is less bright. On the vase image of Fig. 2, this algorithm finds four OVD areas which intuitively are the four highlight spots.

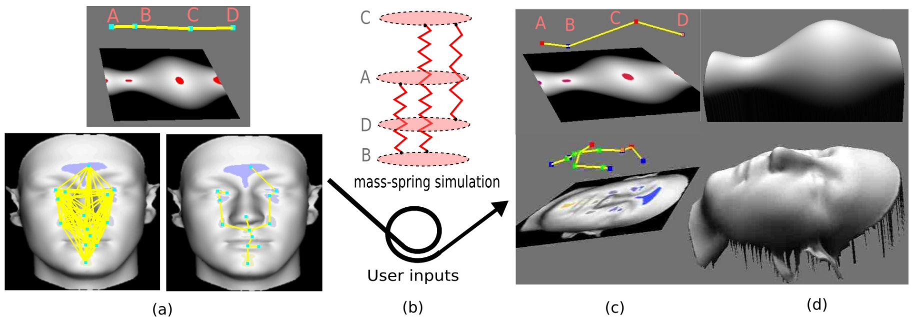

**Fig. 2.** (a) Each highlight of the shading image gives a vertex in the relative-distance graph. Fast marching algorithm computes the relative distance between each pair of vertices. The graph is simplified into a minimum spanning tree. (b) An iterative process of mass-spring simulation/user inputs on the graph (c) runs until the user is satisfied by the reconstructed surface (d). Notice that, since our graph represents the relative-altitude (and not euclidean distance), each vertex can move only in a column (change altitude) during the mass-spring simulation.

**Edges and Relative Altitude Computation** We now determine edges and their weight which are the relative-altitude between vertices. For each vertex $v_i$ of the graph, we compute the relative altitude to all other vertices by fast marching: We set to zero the altitude of the pixel $p_i$ under the considered vertex $v_i$. The altitude of all other pixels is set to $\infty$. We run the fast marching algorithm as described on Sect. 3.1. The altitude of all other pixels will be lower than $v_i$ because we only descend from this pixel. We then look at the altitude of pixels that correspond to the other vertices $v_j, \quad j \neq i$. We use this difference to set the weight of the edge $e_{ij}$ to be the relative altitude difference between vertex $v_i$ and $v_j$. Notice the fact that relative altitude difference between two vertices might be wrong if there is a inflection point between them, this situation will be addressed in next section by a simplification of the graph. We iterate this process for each vertex until we get the weights of all the edges between all the vertices. After this process, we do not know if a vertex is, for example, a local maximum, local minimum, saddle; we only know its relative altitude to its neighbors.

## 5 Mass-spring Simulation and User Interaction

The relative altitude graph is mapped onto the shading image, then it is simplified and converted into a mass-spring network which will serve as a visual aid and a way for the user to correct the minimization process.

Our initial complete graph is composed of $C_2^n = \frac{1}{2}n(n-1)$ edges, $n$ being the number of vertices (OVD areas). However, the relative altitude between two *far* vertices (in the sense of Euclidean distance) is probably incorrect and should be discarded: the monotonic descent assumption used for the relative altitude calculation does not hold if the path between the two vertices crosses a valley, a saddle or a ridge. Thus, a relative altitude is only meaningful for adjacent vertices, the graph can be reduced to a subset of edges. For the same reasons that Zeng *et al.* in [13], we simplify the graph by its minimum spanning tree using Prim's algorithm [17]. The number of edges is thus reduced to $n$. Since all vertices are directly/indirectly connected to each other by the tree, the user can seed a minimization process to find their absolute altitude. Notice, that our approach of computing a complete graph which is then simplified is simple to setup whereas determining directly which vertices are neighbors would have been error-prone.

The weight of each edge is the (relative-)altitude difference between its two vertices but the sign of this relative altitude is unknown: we do not know which vertex is above the other. In others words, we do not which vertices are local maximum, which are local minimum, and which ones are saddle. Thus, we ask a user to select some vertices and to move them up or to the down according to his knowledge of the target surface, this will serve as the initial condition to the minimization process. For example on a face, user will move up the vertex corresponding to the nose. In order to respect the relative altitude constrains between vertices and to propagate user's information to the remaining vertices we build a mass-spring network. This saves the user from having to adjust the altitude of *all* vertices. Each vertex becomes a mass which will be able to move only in the $z$ direction as its (x,y) position is fixed. Indeed, we consider only relative-altitude between vertices (and not Euclidean distance) this simulation is like a single column of mass linked by springs as illustrates on Fig. 2. All vertices have a mass of 1 meaning any vertex is more important than another. Each edge of the spanning tree becomes a spring having a rest-length equal to the relative altitude between its two vertices.

This mass-spring network is animated by an Euler-explicit integration [18] until it stabilizes. This simulation has the two advantages: propagating user inputs and being visually intuitive for the user. Indeed, even before the stabilization, the user may want to interact by moving vertices according to his knowledge of the surface. Moreover, each time he wants to see the potential reconstructed surface, the fast marching method generates it in less than a second. This iterative process "mass-spring simulation/user interaction on the vertices" is running until the reconstructed surface satisfies the user.

Our interactive method allows to deal with surfaces with sharp edges: meaning local minimum or maximum without highlight. Sharp edges are points where the surface is $C_0$ continuous but piecewise $C_1$ continuous meaning where the gradient is not smooth. For instance, at the line of the junction of the lips, the surface changes its orientation without producing a flat area with a highlight. Thus, our method allows the user to add a vertex to the relative-altitude graph which will allows a change in the surface orientation. Fig. 3 illustrates this feature: on the left the graph without user intervention produce incoherent lips whereas on the right, after the addition of the blue vertex, our method produces correct lips.

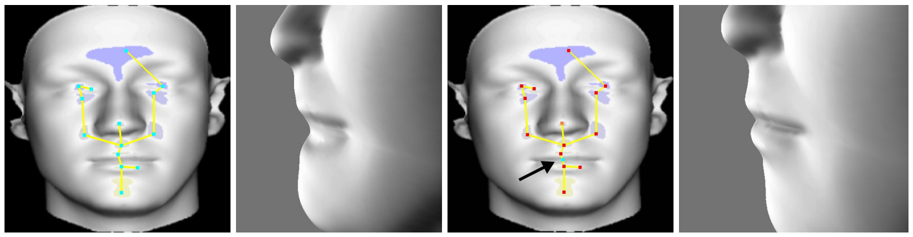

**Fig. 3.** A surface may have sharp edges corresponding to local minimum or maximum without producing highlights. For example, at the line of the junction of the lips, the surface changes its orientation without forming a flat area. To correct this, the user can add vertices to the graph which will allow a change in the orientation. On the left, the graph without user intervention produces incoherent lips whereas on the right, after adding a vertex (blue), our method produces correct lips.

## 6 Results

On Fig. 5 we show results from real photographs to demonstrate that our technique is well suited to image-based modeling. This surface is reconstructed from one input photograph in few minutes by an user: graph generation takes around $30$ seconds on a Intel Centrino Laptop for approximately $70$ vertices, surface generation by fast-marching takes less than a second for an image of $300 \times 200$. For the faces, the user has to interact with fewer vertices, between $2$ and $10$. We illustrate our concept on face photographs to show the capability of our technique to capture fine wrinkles of the skin which are difficult to obtain with multi-view approaches. Once the surface is reconstructed as an heightmap, we use the surface normal to extract a pseudo intrinsic color of each pixel by solving the diffuse equation $(R, G, B)_{image} = (R, G, B)_{intrinsic} \times N.L$ with $L = (0, 0, 1)$. Thus, a textured image of the surface is obtained by combining the intrinsic color and the shading computed with surface normal. Notice, that if the light is similar to the original photograph (position and color) we should obtain similar results.

The heightmap produced by our system is easily triangulated to a mesh. Nevertheless, it can also be directly included as displacement map, for instance to produce realistic scene of low-relief walls as illustrate on Fig. 4.

**Limitations.** Our method allows the reconstruction of fine details, nevertheless there are some limitations with our system. First, the reconstructed surface might have some kinks at the surface jonctions during the fast marching process. On Fig. 5, kinks near the eyebrows are present due to the difference of albedo between the skin and the eyebrows. At the end of [13], Zeng *et al.* propose a method to fix this kind of surface incoherency by a minimization process. Second, if there are too many fine details, the amount of user input can become important. It is concevaible to add heuristics, to modify the mass-spring simulation or to use hierarchical approach to alleviate this limitation. Additionnally, the algorithm supposes that the surface is C1 continuous, thus discontinuities in the surface can be hard to capture. This problem is partially mitigated by allowing the user to add vertices to the relative altitude graph.

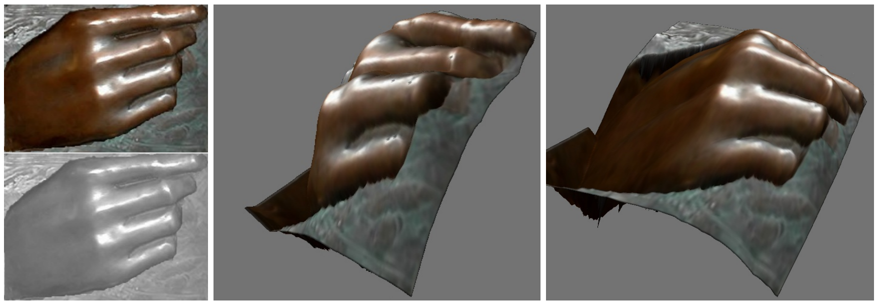

**Fig. 4.** On the left, the original photograph and the computed shading image. On the right, the reconstructed surface representing a low-relief hand. After an empirical test on this image, we do not perform any particular process to manage the specular aspect, nor to take into account that the wall behind the hand has probably a different albedo than the hand.

## 7 Conclusion and future work

Starting from a single color image (a photograph), we have presented an intuitive method for user reconstruction of surfaces which may have produced the image. Our method is interactive, guided by the user it may reconstruct different surfaces for a same input image. It allows to explore different SfS solutions in case of doubt. For example, the image on the left of Fig. 1 which may have been produced by the two surfaces on the right. This exploration facility allows the user to interactively, quickly, and easily reconstruct the surface of a given object with only one photograph. The ambiguity around the global shape of the photographed object is hard to resolve automatically without any a priori knowledge, so we ask the user to specify few local extrema (maximum or minimum). Since reconstructed surface is computed in less than few seconds, it is easy for the user to converge to a surface. Manual intervention is only needed to reconstruct the global shape whereas fine part of the surface is automatically extracted.

Finally, we believe that a little user interaction can help to reconstruct many real objects. Thus, SfS approaches may be practically included in 3D mesh modelers [1] by defining a shape by example paradigm. In the future, we also would like to combine SfS approaches to global surface reconstruction based on multiple-views.

[1] such as Maya(Alias Wavefront), 3D Studio Max(Discreet) or Image Modeler(Realviz).

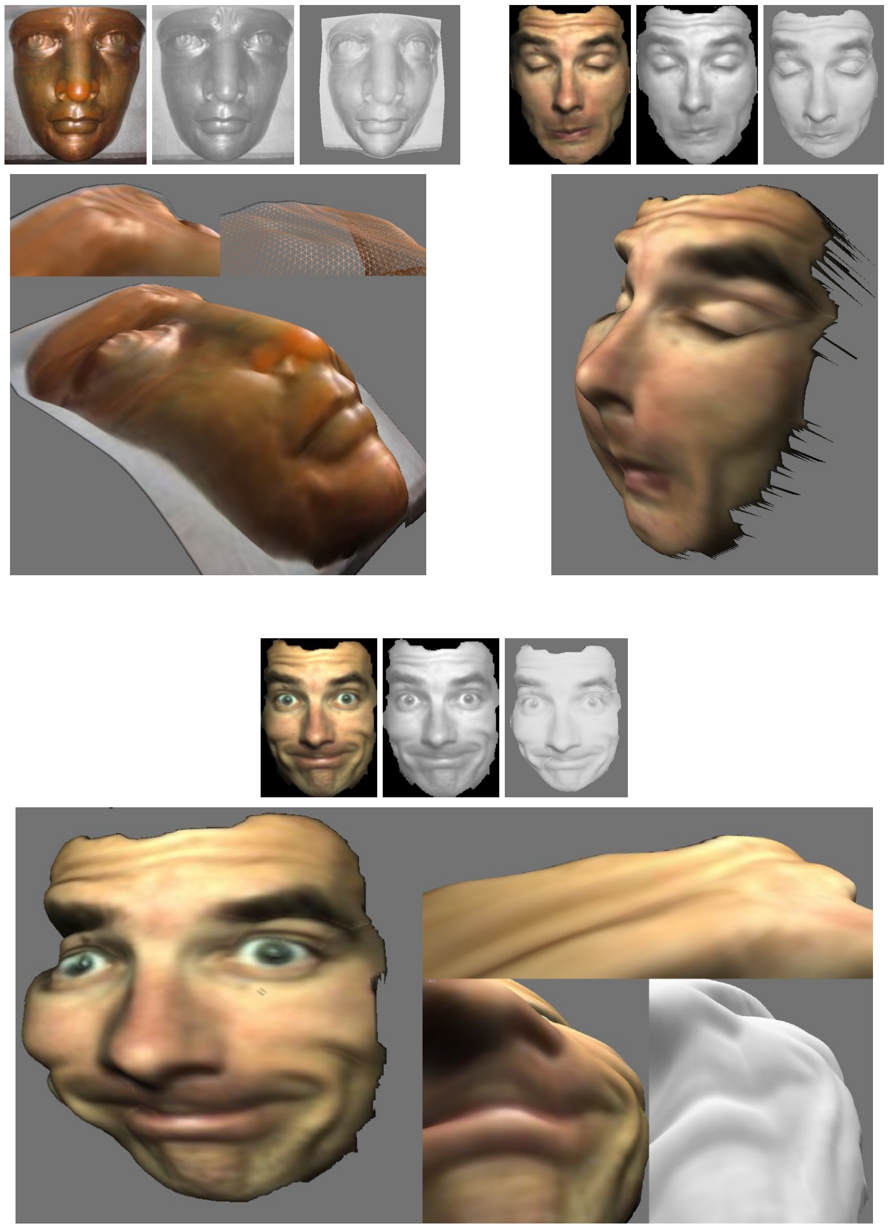

**Fig. 5.** Fine detail of facial expression are hard to capture because of the wrinkles of the skin. Using shading information, our technique allows to capture them from a single photograph. We show the original image, the computed shading image used to reconstruct the surface, a rendering of the reconstructed image with only the shading computed using the normal and some results with the color texture. Top left is a photo downloaded from the web. Others are extracted from a $640 \times 480$ video sequence. Notice that Zeng *et al.* in [13] propose a minimization method to fix the kinks of the surface due to fast marching imprecision like the ones present near the eyebrows (bottom).

## References

1. Durou, J.D., Mascarilla, L., Piau, D.: Non-Visible Deformations . In: 9th International Conference on Image Analysis and Processing - ICIAP'97 , Florence, Italie, 17/09/1997-19/09/1997. (1997)
2. Prados, E., Faugeras, O.: Shape from shading: a well-posed problem ? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, California. Volume II., IEEE (2005) 870–877
3. Courteille, F., Durou, J.D., Morin, G.: A global solution to the sfs problem using b-spline and simulated annealing. In: ICPR. (2006)
4. Sethian, J.A.: A Fast Marching Level Set Method for Monotonically Advancing Fronts. Proceedings of the National Academy of Sciences of the United States of America **93**(4) (1996) 1591–1595
5. Kimmel, R., Sethian, J.A.: Optimal Algorithm for Shape from Shading and Path Planning. Journal of Mathematical Imaging and Vision **14**(3) (2001) 237–244
6. Hoiem, D., Efros, A.A., Hebert, M.: Automatic photo pop-up. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, New York, NY, USA, ACM Press (2005) 577–584
7. Kozera, R.: An overview of the shape from shading problem. Machine Graphics and Vision (1998)
8. Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.: Shape from Shading: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(8) (1999) 690–706
9. Durou, J.D., Falcone, M., Sagona, M.: A Survey of Numerical Methods for Shape from Shading. Rapport de Recherche 2004-2-R, Institut de Recherche en Informatique de Toulouse, Toulouse, France (2004)
10. Courteille, F., Crouzil, A., Durou, J.D., Gurdjos, P.: Shape from shading for the digitization of curved documents. In: Machine Vision and Applications. (2006)
11. Prados, E., Camilli, F., Faugeras, O.: A unifying and rigorous shape from shading method adapted to realistic data and applications. Journal of Mathematical Imaging and Vision **25**(3) (2006) 307–328
12. Zhu, Q., Shi, J.: Shape from shading: Recognizing the mountains through a global view. In: CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2006)
13. Zeng, G., Matsushita, Y., Quan, L., Shum, H.Y.: Interactive Shape from Shading. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (volume I), San Diego, California, USA (2005) 343–350
14. Funt, B.V., Drew, M.S., Brockington, M.: Recovering shading from color images. In: ECCV '92: Proceedings of the Second European Conference on Computer Vision, London, UK, Springer-Verlag (1992) 124–132
15. Tappen, M.F.: Recovering intrinsic images from a single image. IEEE Trans. Pattern Anal. Mach. Intell. **27**(9) (2005) 1459–1472 Member-William T. Freeman and Member-Edward H. Adelson.
16. Ho, J., Lim, J., Yang, M.H.: Integrating Surface Normal Vectors Using Fast Marching Method. In: Proceedings of the 9th European Conference on Computer Vision (volume III). Volume 3953 of Lecture Notes in Computer Science., Graz, Austria (2006) 239–250
17. PRIM, R.C.: Shortest connection networks and some generalizations. In: Bell Syst. Tech. J. 36, 1389- 1401. (1957)
18. Desbrun, M., Schröder, P., Barr, A.: Interactive animation of structured deformable objects. In: Graphics Interface. (1999) 1–8