SHAPE FROM COLOR CONSISTENCY USING NODE CUT

Ning Xu, Tianli Yu and Narendra Ahuja

Department of Electrical and Computer Engineering and Beckman Institute
University of Illinois at Urbana-Champaign, Urbana, IL61801, USA

ABSTRACT

In this paper, we present a method to estimate the shape of a three dimensional (3D) object by enforcing color consistency in the views of object points in images acquired from different viewpoints. The estimated shape and color information of the object can be used to render new views of the object. We first construct a 3D voxel space and assign to each voxel a photo inconsistency value, and then represent this voxel space using a node capacitated graph. We use s-t node cut to obtain a surface that minimizes the summation of photo inconsistency values of surface voxels. Experimental results for simulated and real objects are provided. We also simulate how the calibration errors of camera intrinsic and extrinsic parameters affect the performance of our algorithm.

1. INTRODUCTION

In this paper, we present a shape from color consistency method to estimate the three dimensional (3D) model of a real world object given its images acquired from different viewpoints. Three dimensional models of real world objects are very important to manufacturing systems, virtual reality systems and multimedia applications.

Computer vision techniques have a long history of use in the reconstruction of 3D graphical models of real world objects [1, 2, 3]. They use regular images obtained from multiple viewpoints and recover a 3D description of the object from them. Two basic classes of algorithms can be distinguished. The first class computes depth maps from individual viewpoints and then registers the depth maps into a single 3D surface model. To obtain the depth maps from different viewpoints, various methods such as depth from focus [4] or defocus [5], stereo vision [1], structure from motion [6], and shape from shading [7] have been proposed, each of which has different degrees of applicability in different conditions. The second class of algorithms is based on volumetric representation and is often referred to as shape from silhouettes [8, 9], and shape from photo consistency [10, 11, 12]. Our proposed algorithm belongs to the second class.

Reconstructing 3D object models can also be posed as a global minimization problem. There have been many approaches in the literature to formulate stereo, from two views or multiple views, as a global minimization problem and solve the problem using graph cuts [13, 14, 15, 16]. These graph-cut based approaches result in improved depth maps. Graph cut is also applied in volumetric representations to minimize the energy given two sets of images, one with the object and one with only the background [17], yielding another solution to the problem of shape from silhouettes.

In this paper, we present an approach to reconstruct 3D object models using photo consistency information in volumetric representation. Each voxel in the 3D space is assigned a photo inconsistency value according to the input images. Then the 3D reconstruction problem is formulated as an optimization problem to minimize the photo inconsistency of the object surface voxels. We represent the 3D voxel space, with photo inconsistency value assigned to each voxel, as a node capacitated graph and apply s-t node cut on this graph to obtain a surface that minimizes the summation of photo inconsistency values of surface voxels.

Section 2 presents our approach in detail. Section 3 provides experimental results. Section 4 presents concluding remarks and discussion.

2. OUR APPROACH

2.1. Related graph theory

2.1.1. Basic concepts

In this section, we will give a graph theoretic description of $s-t$ minimum cut. Let a flow network $G=(V,E)$ be a connected graph with vertex set V and edge set E. Each edge $(u,v)\in E$ has a nonnegative capacity $c(u,v)\geq 0$. If $(u,v)\notin E$, we assume that $c(u,v)=0$. Two vertices in V are distinguished: a source s and a sink t. A cut (S,T) of the flow network G is a partition of V into S and $T=V-S$ such that $s\in S$ and $t\in T$. The capacity of a cut is defined as the summation of the capacities of the edges across the cut, i.e. $c(S,T)=\sum_{u\in S,v\in T}c(u,v)$. The $s-t$ minimum cut problem is to find a cut in G that separates s and t with the smallest capacity. This problem is very closely related to the max-flow problem in graph theory. A flow in G is a real-valued function $f: V\times V\rightarrow R$ that satisfies the following properties [18]:

1. for all $u, v \in V$, $f(u, v) \leq c(u, v)$;
2. for all $u, v \in V$, $f(u, v) = -f(v, u)$;
3. for all $u \in V - s, t$, $\sum_{v \in V} f(u, v) = 0$.

The value of a flow f from s is defined as

$$|f| = \sum_{v \in V} f(s, v).$$

In the maximum-flow problem, we are given a flow network G with a source s and a sink t, and we wish to find a flow with maximum value from s to t. There is an important correspondence between flows and cuts in networks, as we can see in the max-flow min-cut theorem as follows:

[Ford-Fulkerson Theorem [19]] The maximum flow from a vertex s to vertex t, $|f|$, is equal to the value of the capacity $c(s, t)$ of the minimum cut separating s and t.

With this theorem, the $s - t$ minimum cut problem can be solved using existing max-flow algorithms.

The minimum cut considered in this paper is required to separate multiple source nodes from multiple sink nodes. An operation called *node identification* [20] enables us to use $s - t$ minimum cut algorithms to solve the multi-source multi-sink minimum cut problem by simply identifying the multiple sources as a single source and multiple sinks as a single sink, respectively. There is also another standard approach to transfer a multi-source multi-sink minimum cut problem into a single $s - t$ minimum cut problem in the literature [18] by adding a new source node s and connecting it to all sources with weight infinity and adding a new source node t and connecting it to all sink nodes with weight infinity.

2.1.2. Node capacitated graph

In some cases, it is more convenient to formulate a problem as a minimum $s-t$ cut problem on a node capacitated graph $G = (V, E)$, i.e., each node $v \in V$ has a nonnegative capacity $c(v) \geq 0$. An $s - t$ node cut in such a node capacitated graph is a node set $C \subset V$, and by removing C from G, s and t are disconnected. Neither s nor t can be contained in C (otherwise, it is called separation instead of cut [21]). The capacity $c(C)$ of a node cut C is summation of the capacities of all nodes in $C \subset V$, i.e. $c(C) = \sum_{v \in C} c(v)$. An $s - t$ minimum node cut yields a minimum capacity of all possible node cuts that separate s and t.

There is a standard technique [19] which can transform node connectivity graph $G(V, E)$ to directed edge connectivity in another graph $\bar{G}(\bar{V}, \bar{E})$ with edge capacity function $\bar{e}$. For each node $v \in V$, there are two corresponding nodes v_1 and v_2 in the node set $\bar{V}$ of the new graph $\bar{G}$. Edge $(v_1, v_2) \in \bar{E}$ is capacitated with $\bar{e}(v_1, v_2) = c(v)$. For all $(u, v) \in E$, the corresponding (u_2, v_1) and (v_2, u_1) will be the edges of $\bar{E}$ with capacity $\bar{e}(u_2, v_1) = \bar{e}(v_2, u_1) = \infty$. An example of this transform is shown in Fig. 1.

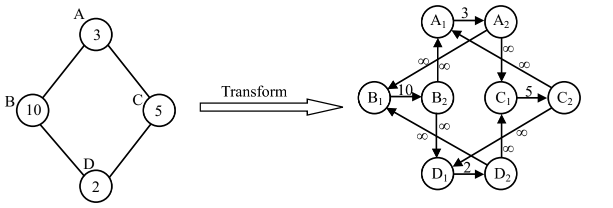


Fig. 1. Transform from node capacitated graph to corresponding edge capacitated digraph.

2.1.3. S-t node cut algorithm

With the techniques to transform a node capacitated graph $G(V, E)$ into an edge capacitated graph $\bar{G}(\bar{V}, \bar{E})$, $s-t$ minimum node cut on G can be solved by applying $s - t$ minimum edge cut on corresponding $\bar{G}$.

Let s and t be the source node and sink node in $G(V, E)$, and $C(s, t) \subset V$ is the $s - t$ node cut on $G(V, E)$, then

$$C(s, t) = \{v | (v_1, v_2) \in \bar{C}(\{s_1, s_2\}, \{t_1, t_2\})\},$$

where $\bar{C}(\{s_1, s_2\}, \{t_1, t_2\})$ is the $s - t$ minimum cut in $\bar{G}(\bar{V}, \bar{E})$, with $\{s_1, s_2\}$ as sources and $\{t_1, t_2\}$ as sinks.

2.2. Constructing Node Capacitated Graph

In order to apply s-t minimum node cut, we construct a node capacitated graph and formulate the 3D modelling problem as a minimization problem on the constructed graph.

We first construct a voxel space and assign each voxel a photo inconsistency value. We divide the 3D space into a grid of voxels, each voxel x is assigned a photo inconsistency value $P(x)$ according to the colors $C_i(x), i = 1, 2, ..., N$, where $C_i(x)$ is the color of the pixel in image I_i projected from voxel x. Theoretically, we want to set the photo inconsistency value $P(x)$ based on a subset of $\{C_i(x)|i = 1, 2, ..., N\}$, i.e.,$\{C_j(x)|j \in J\}$, where $J \subseteq \{1, 2, ..., N\}$ and voxel x is visible in each image $I_j, j \in J$. However, without knowing the real object surface, it is impossible to check the visibility of voxel x to each camera. To avoid the problem of visibility checking, we assume that each surface point of the object is visible to at least M cameras and cluster the M nearest colors, out of the N possible colors associated with voxel x, under the condition that the corresponding M cameras should lie on one side of the voxel x. The variance of these M nearest colors is set as the photo inconsistency value $P(x)$ of the voxel x.

Then we represent this 3D voxel space as a node capacitated graph. Each voxel x corresponds to a node $v \in V$ in the graph $G(V, E)$. Two nodes u, v are connected, i.e. $(u, v) \in E$, if their corresponding voxels are neighbors to each other. Here, we use 6-connection neighborhood. For each node v, we set $c(v) = P(x)$, where $c(v)$ is the node capacity of node v corresponding to voxel x. After the graph is constructed, the 3D reconstruction problem becomes a problem of minimizing the summation of photo inconsistency values of the surface voxels.

2.3. Minimization using s-t node cut

To solve the above minimization problem, we use an s-t node cut algorithm. The algorithm can be described as follows:

1. Define a near plane and a far plane such that the object surface lies between them.

2. Use node identification to identify nodes corresponding to the voxels on the near plane as a single node s, and identify nodes corresponding to the voxels on the far plane as a single node t.

3. Apply $s-t$ node cut on the resulting node capacitated graph and transform the resulting cut into a surface in the 3D space.

4. For rendering, assign each voxel on the resulting surface an average color from all visible cameras.

Thus, we obtain a surface that is globally optimal within the voxel space and estimate the colors of the surface voxels.

2.4. Implementation considering computation limits

Since graph cut algorithms are usually memory and time intensive for large graphs. The maximum number of nodes that can be processed is limited by the available memory. Smaller voxel sizes yield better resolution of estimated surface but larger number of nodes. For improving resolution under the memory limitation, we use small voxel size and iteratively compute minimum cut in a part of the graph, which represents the volume around the resulting surface from previous iteration.

To do this, we first obtain an initial plane parallel to the near plane and far plane and across the object surface. Then we iteratively perform the following steps: 1. Dilate the current surface to obtain a neighborhood. 2. Extract the near surface and far surface of the dilated neighborhood and identify all the nodes corresponding to voxels on the near surface as a single node s, and identify all the nodes corresponding to voxels on the far surface as a single node t. 3. Apply s-t node cut to compute a minimum surface. 4. Repeat step 1-3 until the resulting surface has been obtained previously. This iterative scheme will converge to a surface that is globally optimal within its own neighborhood. The proof of convergence is analogous to the proof provided in [20].

The dilation process we use consists of a number of single binary dilations with a structuring element which is $3 \times 3 \times 3$ tensor of 1. For example, if the initial surface is a sphere with radius of 100, after 5 single dilation steps, the inner surface will be a sphere with radius of 95 while the outer surface will be a sphere with radius of 105. Other structuring elements could also be used. The number of single dilation in each step will determine the size of the neighborhood space D_i. This number is selected according to the size and shape of the object.

3. EXPERIMENTS

In this section, we present our experimental results and also simulate how the calibration errors of camera intrinsic and extrinsic parameters affect the performance of our algorithm.

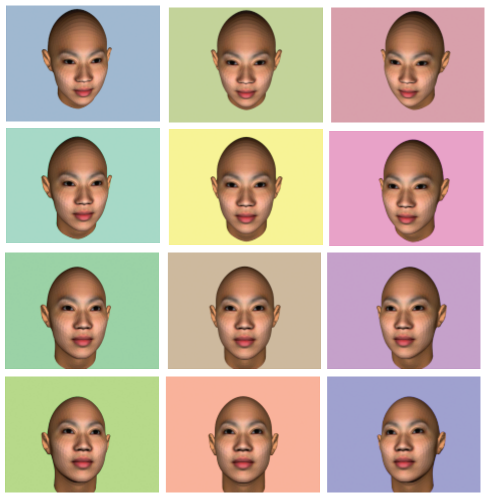


Fig. 2. Twelve input images of experiment 1.

3.1. Experiment 1

The object in our first experiment is a synthetic human face. We capture 12 images in front of the face as input images. The size of each image is 640×480. Input images are simulated with a 3D head model using 3D Studio Max. Five single dilations are performed in the direction perpendicular to the plane. Each voxel in 3D space is assumed to be visible to at least 8 cameras. Fig. 2 shows all the 12 input images. Fig. 3 shows some new images rendered from the reconstructed face model.

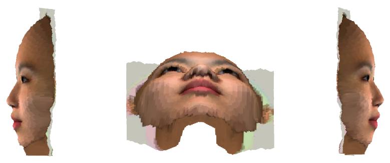


Fig. 3. Three rendered new images from the reconstructed 3D model. Note that we are viewing the object from angles very different from those used to acquire the input images.

3.2. Simulating calibration errors

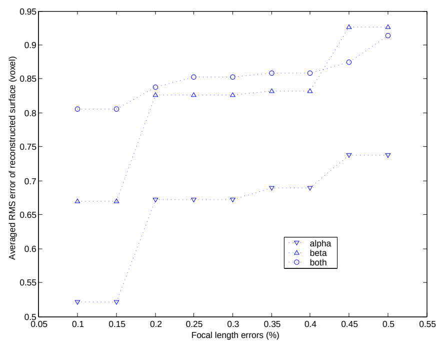
The above results are obtained with calibration matrixes free of error. In order to examine how our algorithm works in real applications, we simulate the camera calibration errors as follows.

3.2.1. Camera intrinsic parameters

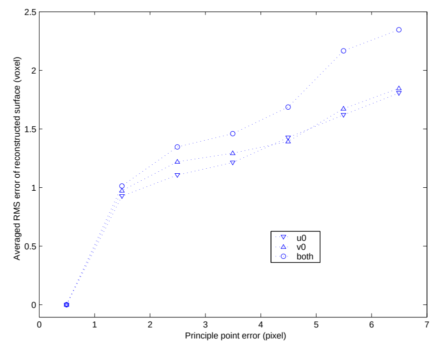
Camera intrinsic parameter matrix is usually defined by

$$\begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where α and β are focal lengths along image u and v axes, and u_0 and v_0 are the coordinates of the principle point. Here, we do not consider the skewness of the two image axes since compared to the error caused by other parameters, the calibration errors from skewness are negligible.



(a)

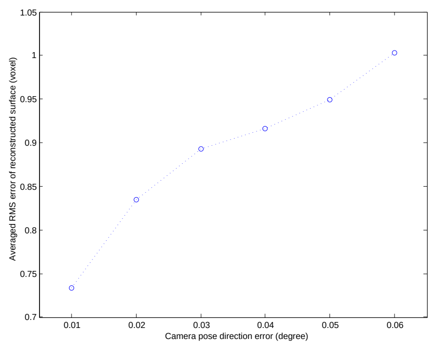


(b)

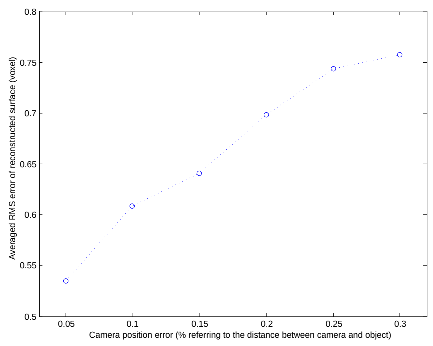
Fig. 4. (a) Reconstruction errors v.s. calibration errors of focal lengths. (b)Reconstruction errors v.s. calibration errors of image principle points.

We use relative error for α and β, and absolute error for u_0 and v_0. We vary the error level of α and β from 0.1% to 0.5%, and vary the error level of u_0 and v_0 from 0.5 pixel to 6.5 pixels. The reconstruction errors are measured as average RMS error of the depth of surface voxels. Experiments are performed for the following six error combinations: α only, β only, both α and β, u_0 only, v_0 only, and both u_0 and v_0. Fig. 4 shows the errors of reconstructed surface v.s. the simulated errors on these camera intrinsic parameters. As indicated in [22], the errors in α and β are less than 0.3% and the errors in u_0 and v_0 are around 1 pixel. For these achievable accuracy of intrinsic parameters, the average RMS errors of our algorithm are smaller than 1 voxel.

3.2.2. Camera extrinsic parameters



(a)



(b)

Fig. 5. (a) Reconstruction errors v.s. camera direction errors. (b) Reconstruction errors v.s. calibration errors on camera position errors.

Camera extrinsic parameters include rotation matrix R and translation vector T. We simulate these errors by varying camera look-at vector L_i and position vector T_i, where $i = 1, 2, ..., 24$. We use absolute error for L_i in terms of angle and use relative error for T_i in terms of percentage referring to the distance between camera and object. We vary the error level of L_i from $0.01°$ to $0.06°$, and vary the error level of T_i from 0.05% to 0.3%. For each error level, we perform 10 independent trials with random errors of that level for all cameras and the results shown are the average. Fig. 5 shows the errors of reconstructed surface v.s. simulated errors on these camera extrinsic parameters. As indicated in [23], the achievable accuracy is about 0.02° for L_i and about 0.1% for T_i. For these achievable calibration accuracies, the reconstruction errors of our algorithm are also smaller than 1 voxel.

3.2.3. Performance with calibration errors for both intrinsic and extrinsic parameters

With the achievable accuracies of calibration parameters as mentioned above (i.e. 0.3% for α and β, 1 pixel for u_0 and v_0, 0.02° for L_i and 0.1% for T_i), we perform 100 independent experiments. The mean of the average RMS error is 1.0575 voxels and the standard deviation is 0.0699. This results shows that our iterative node cut algorithm is robust to camera calibration errors. Some rendered images from new viewpoints are shown in Fig. 6. Visually there is little difference from the rendering results with parameters free of calibration errors.

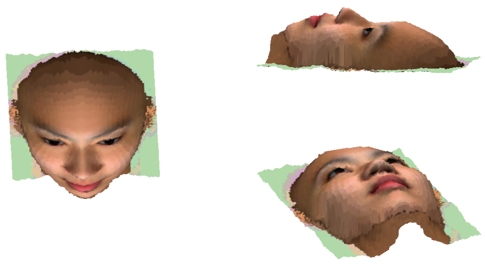


Fig. 6. Three rendered new images from the reconstructed 3D model. Note that this model are reconstructed after considering the possible calibration errors.

3.3. Experiment 2

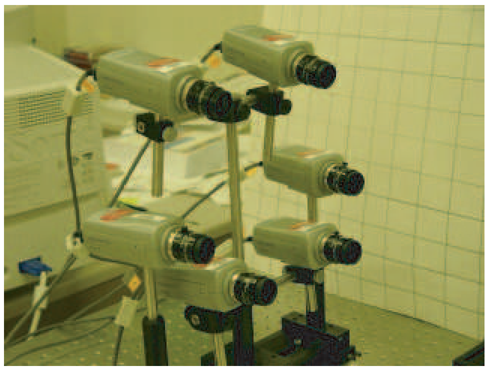


Fig. 7. A six-camera system.

We set up a real six-camera system to model human face. Fig. 7 shows the system setup. We calibrate the cameras using the Camera Calibration Toolbox for Matlab [24]. Each surface point is assumed to be visible to at least three cameras. Fig. 8 shows all the six input images. Fig. 9 shows some new images rendered from the reconstructed face model.

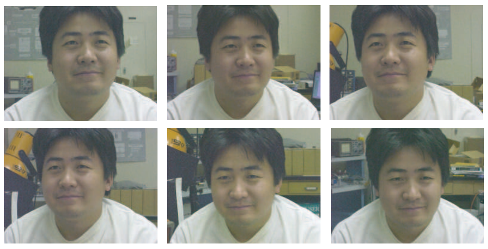


Fig. 8. Six input images of experiment 3.

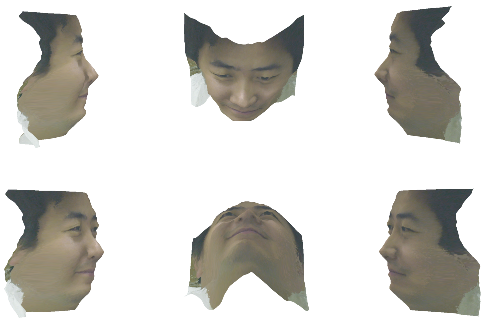


Fig. 9. Six rendered new images from the reconstructed 3D model.

4. CONCLUSIONS AND DISCUSSION

In this paper, we have presented a shape from color consistency method to generate 3D object model given multiple calibrated pictures acquired from different viewpoints. Node capacitated graph is used to represent the 3D voxel space with photo inconsistency value assigned to each voxel. Our node cut based algorithm is able to reconstruct a surface that is globally optimal within its neighborhood, in the sense that the summation of photo inconsistency values of the surface voxels is minimized.

The disadvantage of our approach is that the graph cut algorithm is memory exhausting and time consuming if the node number of the graph is large. Currently we are using P4-1.8GHz desktop with 1GB memory, the running time of the algorithm is within several minutes when the graph has one million nodes. Using more nodes will cause out of memory error. However, our algorithm is coded in Matlab. No particular effort is made to minimize the memory occupancy. (Although where it is possible to achieve significant improvements in execution time, routines are coded as

MEX files in C, for example, the s-t min cut routine.) Ongoing research includes how to seamlessly combine other depth information in this iterative node cut algorithm to obtain better results.

Furthermore, if the reflectance model of object surface can be modelled as combination of diffuse reflection and specular reflection, using clustering of M nearest colors in our algorithm will be able to handle the specular reflection problem if we assume that each surface point can be seen by at least M cameras which are not in the specular reflection directions.

5. REFERENCES

[1] S.T. Barnard and M.A. Fishler, "Computational stereo," *Computing Surveys*, vol. 14, no. 4, pp. 554–572, 1982.

[2] A. Black and A. Yuille, Eds., *Active Vision*, MIT Press, 1993.

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

[4] P. Grossmann, "Depth from focus," *Pattern Recognition Letters*, vol. 5, pp. 63–69, Jan. 1987.

[5] A. Pentland, "A new sense of depth of field," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-9, pp. 523–531, July 1987.

[6] J. Weng, N. Ahuja, and T.S. Huang, "Optimal motion and structure estimation," *IEEE PAMI*, vol. 15, no. 9, pp. 864–884, 1993.

[7] B.K.P. Horn, *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View*, Ph.D. thesis, Massachusetts Inst. Of Technology, 1970.

[8] S. Srivastava and N. Ahuja, "An algorithm for generating octrees from object silhouettes in perspective views," *Proc. IEEE Workshop Computer Vision*, pp. 363–365, Dec. 1987.

[9] S. Sullivan and J. Ponce, "Automatic model construction, pose estimation, and object recognition from photographs using triangular splines," *ICCV*, pp. 90–95, Jan. 1998.

[10] Charles Dyer, *Foundations of Image Understanding*, chapter Volumetric Scene Reconstruction from Multiple Views, pp. 469–489, Kluwer, 2001.

[11] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. of Computer Vision*, vol. 35, no. 2, pp. 151–173, 1999.

[12] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, no. 3, pp. 199–218, July 2000.

[13] H. Ishikawa and D. Geiger, "Occlusions, discontinuities, and epipolar lines in stereo," *ECCV*, 1998.

[14] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *International Conference on Computer Vision*, vol. I, pp. 377–384, 1999.

[15] Vladimir Kolmogorov and Ramin Zabih, "Multi-camera scene reconstruction via graph cuts," *ECCV*, 2002.

[16] C. Buehler, S.J. Gortler, M.F. Cohen, and L. McMillan, "Minimal surfaces for stereo," *ECCV*, pp. 885–899, 2002.

[17] Dan Snow, Paul Viola, and Ramin Zabih, "Exact voxel occupancy with graph cuts," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 345–352, 2000.

[18] Thomas Cormen, Charles Leiserson, and Ronald Rivest, *Introduction to Algorithms*, McGraw–Hill Companies, 1990.

[19] L. Ford and D. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.

[20] Ning Xu, Ravi Bansal, and Narendra Ahuja, "Object segmentation using graph cuts based active contours," *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 46–53, June 2003.

[21] Andras A. Benczur, "Counterexamples for directed and node capacitated cut-trees," *SIAM Journal on Computing*, vol. 24, no. 3, pp. 505–510, 1995.

[22] Zhengyou Zhang, "A flexible new technique for camera calibration," *Microsoft Research Technical Report MSR-TR-98-71*, December 1998.

[23] Jean-Yves Bouguet and Pietro Perona, "3d photography on your desk," in *ICCV*, 1998, pp. 43–52.

[24] Jean-Yves Bouguet, "Camera calibration toolbox for matlab," http://www.vision.caltech.edu/bouguetj/calib_doc/.