Proc. 10th Int. Conf. Comput. Vision (ICCV2005), Beijing, China, October 2005, Vol. 2, pp. 1148–1153.

Mesh Optimization Using an Inconsistency Detection Template

Atsutada Nakatuji*, Yasuyuki Sugaya† and Kenichi Kanatani†
*Internet Terminals Division, NEC Engineering, Ltd., Yokohama-shi, Kanagawa 224-0053 Japan
†Department of Computer Science, Okayama University, Okayama 700-8530 Japan
{nakatuji,sugaya,kanatani}@suri.it.okayama-u.ac.jp

Abstract

We propose a new technique for optimizing a triangular mesh for polyhedral representation of the scene in a video stream. We introduce a specially designed template that can effectively detect color and texture discontinuities. Using real images, we demonstrate that our method is superior to existing methods.

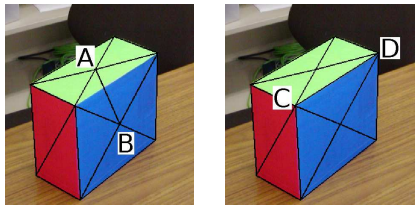
1. Introduction

One of the most important issues of 3-D reconstruction from images is how to represent the reconstructed shape. If calibrated stereo vision is available, we can obtain a dense depth map over all the pixels. If a technique called *space carving* [5] is used, the scene can be represented as an aggregate of colored voxels. More sophisticated methods, such as *plenoptic representation* [1], *light field rendering* [6], and *lumigraph* [2], register all the light rays in the scene for generating new views seen from an arbitrary viewpoint.

If images are taken by uncalibrated cameras, on the other hand, corresponding feature points are extracted, and their 3-D coordinates are computed [3]. Then, a triangular mesh is defined using the matched points, and the scene is displayed as a texture-mapped polyhedron. The triangular mesh can be automatically generated using *Delaunay triangulation* [9], which produces triangles of balanced sizes and shapes, suitable for polyhedral representation of a curved surface.

However, a serious problem occurs if the scene itself is a polyhedron. In man-made environments such as indoors and cities, most objects are polyhedra. If the vertices of polyhedral objects are chosen as feature points, some of the triangulation edges may not coincide with the physical edges. See Fig. 1, for example. The Delaunay triangulation in Fig. 1(a) does not correctly represent the true shape, but the triangulation in Fig. 1(b) correctly represents it.

The aim of this paper is to present a new technique for automatically transforming a given triangulation into a physically compatible one. Sec. 2 $\sim$ 4 describe the principle of our method. The procedure is described in Sec. 5. In Sec. 6, we show real image examples to demonstrate that our method is superior to existing methods. Sec. 7 concludes this paper.



(a) (b)

Figure 1. (a) Triangulation inconsistent with the true shape. (b) Triangulation consistent with the true shape.

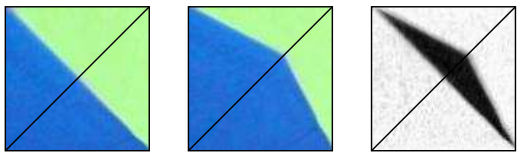
2. Compatibility of Triangulation

Many studies exist for generating optimal triangular meshes. They are roughly classified into two categories: to replace a dense mesh by a coarser one without impairing the faithfulness of the shape representation, or to upgrade the descriptiveness of the shape by adding vertices and edges. For example, Vogiatzis et al. [10] simplified a fine mesh by stochastic annealing, while Yu et al. [11] refined a coarse mesh by estimating the object shape and the surface reflectance map. These involve the *entire process of 3-D reconstruction* from sensor data acquisition to final display.

Here, we consider a third category: optimizing edges for a *given* set of vertices by image processing techniques *without using 3-D shape information*. By this approach, we can encapsulate this algorithm as an independent tool for all potential applications not limited to 3-D reconstruction.

The only existing studies in this category are those of Morris and Kanade [7] and Perrier et al. [8]. The basic principle is to compare the textures in corresponding triangular patches in different frames. If a triangular patch is defined on a planar surface, its texture in one frame can be mapped onto the corresponding patch in another by an affine transformation[1]. Hence, the intensity difference after the mapping should be zero. If

[1]Theoretically, corresponding patches in different frames are related by a *homography* [3], but as far as individual patches are concerned, as opposed to a global planar scene, the mapping can be approximated by an affine transformation with negligible differences.



(a) (b) (c)

Figure 2. (a) Mapping the inside of quadrilateral surrounding the edge AB in Fig. 1(a) into a square region by a homography. (b) Mapping the triangles adjacent to the edge AB in Fig. 1(a) separately into the same square region by affine transformations. (b) The difference of (a) and (b) (darker tones correspond to larger values).

not, the patch is not on a planar surface, so we "flip"[2] an appropriate edge (e.g., the edge AB in Fig. 1(a) is flipped into the edge CD in Fig. 1(b)). Iterating this, we should end up with a triangular mesh compatible with the object shape [7].

In reality, the intensity difference is not exactly zero due to inaccuracies of feature point matching and viewpoint dependent reflectance changes, but setting an appropriate threshold is difficult. So, Morris and Kanade [7] and Perrier et al. [8] iteratively flipped edges so as to maximize the similarity (or minimize the dissimilarity) between the textures of corresponding patches.

As the texture (dis)similarity measure, Morris and Kanade [7] used the sum of square differences of the corresponding pixel values (to be minimized), while Perrier et al. [8] used the normalized correlation (to be maximized).

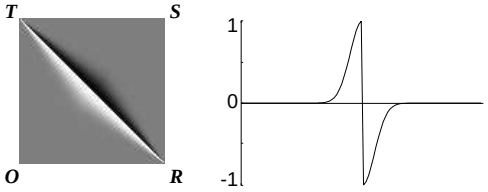
In this paper, we point out that their methods are in reality detecting not so much the appearance differences caused by viewing angles as *texture discontinuities over mesh edges*, which are the dominant clues to shape inconsistency. Based on this observation, we present a more efficient measure for directly detecting texture discontinuities and demonstrate that our method outperforms the existing methods.

3. Texture Discontinuity Detection

We say that an edge of the mesh is *incorrect* if it connects two points on different faces of the polyhedral object (e.g., the edge AB in Fig. 1(a)), and *correct* otherwise: a correct edge lies either within a planar face or along one of its boundaries (e.g., the edges in Fig. 1(b)). We assume that texture, color, or brightness (generically we call these simply "texture") is different from face to face. According to our experience, the methods of Morris and Kanade [7] and Perrier et al. [8] can effectively detect the shape inconsistency only when this assumption is satisfied.

Consider the edge AB in Fig. 1(a). We want to test

[2]Morris and Kanade [7] used the term "swap", but since only one edge is involved, we use the more mathematically accepted term "flip" after Perrier et al. [8].



(a) (b)

Figure 3. (a) Inconsistency detection template. Lighter tones correspond to larger values. (b) Cross section along OS.

if this edge is correct. If the inside of the quadrilateral surrounding this edge is mapped onto a square region by a *homography*, we obtain Fig. 2(a). If the triangles adjacent to this edge on both sides are mapped onto the same square region by affine translations individually, we obtain Fig. 2(b). Their difference is Fig. 2(c) (darker tones correspond to larger values).

We observe a dark triangular region crossing the diagonal that corresponds to the edge AB. We regard such a region as indicating the inconsistency of the edge. It is easily seen that no such region would appear if the same procedure is applied to the edge CD in Fig. 1(b).

Note that the edge CD in Fig. 1(b) is mapped to a straight line by a homography of the quadrilateral but generally not so if the two triangles adjacent to the edge AB in Fig. 1(a) are separately mapped by individual affine transformations.

Thus, the inconsistency of an edge is detected not by testing if texture discontinuity exists across *that edge* but by testing if texture discontinuity exists across the line, where *no mesh edge exists yet*, onto which the edge is to be flipped. If so, we flip that edge.

Of course, the discontinuity cannot be detected in this way if the surrounding quadrilateral happens to be a parallelogram. However, this can be resolved by using multiple (at least two) images, because if viewed from a different angle, the quadrilateral does no longer look like a parallelogram unless it is defined on a planar surface, in which case the edge is correct by definition.

4. Inconsistency Detection Template

In order to detect intensity difference as shown in Fig. 2(c), we define a template as depicted in Fig. 3(a), where lighter tones correspond to larger values. It is defined over a square region $ORST$ of size $l \times l$ with the following values[3]:

$$T(x,y) = \begin{cases} e^{-\frac{(x+y-l)^2}{2\alpha^2(x-y-l)^2}} & x+y<l, x \geq y \\ T(y,x) & x+y \leq l, x<y \\ -T(l-y,l-x) & x+y>l \end{cases} . \quad (1)$$

[3]We set the template size l in such a way that the average area of the triangular patches in the input images is approximately $l^2/2$.

This template is symmetric with respect to the diagonal OS and *antisymmetric* with respect to TR. The contour $T(x, y) =$ constant consists of two line segments starting from R and T and meeting on the diagonal OS. Fig. 3(b) shows the cross section along the diagonal OS: the Gaussian function of mean $l/\sqrt{2}$ and standard deviation $\alpha l\sqrt{2}$ cut in the middle and placed upside down on the right side[4].

In our experiment, we set the template value $T(x, y)$ to zero at the pixels on the diagonal TR and at the pixels within distance $0.02l$ pixels from the diagonal OS or from the boundary. This is to avoid the effect of misalignment of patch boundaries due to inaccuracies in locating feature points.

The reason we use an antisymmetric template is that we do not know a priori on which side the intensity difference appears; it should lie only on one side of the diagonal of the surrounding quadrilateral. If it appears on the other side, the difference has an opposite sign, but the template also has an opposite sign there, so the filter output always has the same sign. The antisymmetry also has the advantage of canceling intensity fluctuations in homogeneously textured region, since such fluctuations are usually uniformly distributed over the entire quadrilateral region.

5. Procedure for Edge Optimization

We assume that feature points are tracked through a video stream of M frames. An initial triangulation is defined by selecting one frame, defining a Delaunay triangulation over the feature points in that frame, and isomorphically mapping it to the rest of the frames.

5.1 Edge inconsistency measure

We measure the degree of inconsistency $w(AB)$ of edge AB as follows:

1. If the edge AB has only one adjacent triangle, let $w(AB) = -1$, meaning that AB is a boundary edge.
2. Let $\triangle ABP$ and $\triangle ABQ$ be the adjacent triangles. Let $w(AB) = 0$ if the quadrilateral $APBQ$ is concave in *some* frames.
3. Otherwise, define an $l \times l$ square region $ORST$ whose pixel values are reset to zero. Then, do the following for $\kappa = 1, ..., M$.
   (a) Map the texture in the quadrilateral $APBQ$ in the κth frame onto the square region $ORST$ by a *homography* and *add* to the pixel values written there.
   (b) *Affinely* map the texture in $\triangle ABP$ and $\triangle ABQ$ onto $\triangle OSR$ and $\triangle OST$, respectively, and *subtract* the pixel values from the values written there.
4. Output the correlation between the values written there and the template $T(x, y)$ as $w(AB)$.

[4]We experimentally found that $\alpha = 0.1$ can produce a good result.

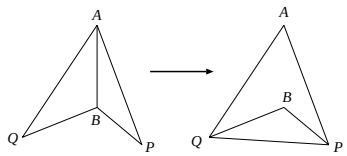


Figure 4. Edge flipping for a concave quadrilateral would result in a reversed patch.

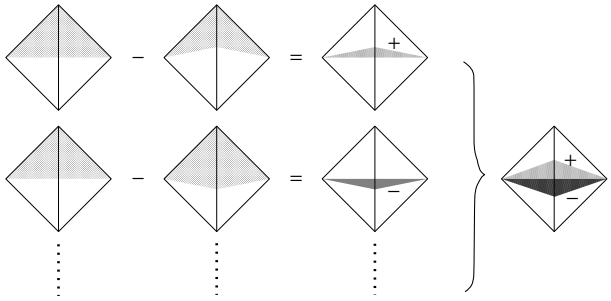


Figure 5. Principle of inconsistency detection.

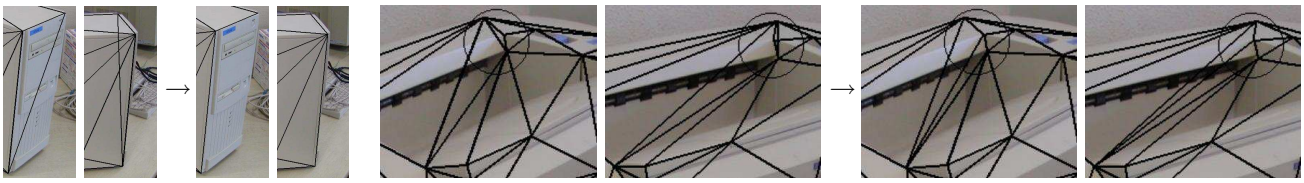
We assume that at the time of generating the initial mesh each edge is classified either into a "boundary edge" with one triangle on one side or into an "internal edge" with two triangles on both sides. The concavity check in Step 2 is for preventing patch reversal [7] (Fig. 4).

Fig. 5 illustrates the meaning of Step 3. If there is no texture difference across the diagonal onto which the edge might be flipped, no intensity difference will arise in the quadrilateral $APBQ$. If there is, intensity difference occurs within a narrow triangular region on either side of that diagonal. We can easily see that the difference has opposite signs depending on which side of the diagonal it appears. Accumulating them for all the frames, we end up with two narrow triangular regions with opposite signs (or possibly only one on one side). Hence, we can measure the strength of the total intensity difference by applying the antisymmetric template shown in Fig. 3.

For color images, we apply the above procedure for the R, G, and B values separately and compute the root-mean-square of the three outputs as $w(AB)$.

5.2 Optimization procedure

We compute the inconsistency measure $w()$ for all edges and look for the edge AB that has the largest value $w(AB)$. We stop if $w(AB) = 0$. Else, we flip the edge AB to PQ and compute $w(PQ)$. If $w(PQ) > w(AB)$, we eliminate the edge PQ, restore the edge AB, and let $w(AB) = 0$. Otherwise, we recompute $w()$ for edges PA, PB, QA, and QB, unless $w()$ is already 0, in the new mesh configuration. Then, we look for the edge $A'B'$ that has the largest value $w(AB)$ and



(a) (b)

Figure 6. (a) If one side of the reversed triangle is a boundary edge, we eliminate it. (b) If the reversal occurs inside, we flip an appropriate side of the triangle.

repeat the same procedure.

Since the largest value of $w()$ monotonically decreases at each flipping, and since edges once checked are not checked again, the above procedure terminates after all the edges are traversed. In this process, the measure $w()$ is used only for comparison, so no thresholds need to be introduced for judgment.

The above procedure can correct those incorrect edges that can be corrected by a single flipping operation. However, not all edges can be corrected that way when one physical edge is crossed by multiple mesh edges (see Fig. 9). So, we repeat the above procedure until the mesh configuration does not alter any further[5].

5.3 Removing patch reversals

If we define an initial Delaunay triangulation over a reference frame, patch reversal may occur in other frames. This can be detected by computing the *signs* of each triangular patch: the sign of $\triangle ABC$ is 1 if the order of A, B, and C is counterclockwise, -1 if clockwise, and 0 otherwise (degeneracy into a line segment).

If the sign is different in some frame, the triangle is reversed there. We dissolve this as follows. If one side of the reversed triangle is a boundary edge, we simply eliminate it (Fig. 6(a)). If the reversal occurs inside, we flip an appropriate side of the triangle (Fig. 6(b)), as discussed by Morris and Kanade [7].

Once such patch reversals are resolved, no new reversals occur thereafter, since we check the concavity of the surrounding quadrilateral before flipping edges.

6. Experiments

6.1 Real image examples

We show real image examples using two frames ($M = 2$). Figs. 7(a), (b) are real images of a polyhedral object, on which a Delaunay triangulation (based on (a)) is overlaid. Figs. 7(c) shows the mesh optimized by our method (superimposed on the first image). The iterations converged in two rounds of the procedure described in Sec. 4.2. The correctness and the computation time are written in the caption. The correctness is (the number of correct edges)/(the number of non-boundary edges) in percentage measured by visual inspection. We used Pentium 4 3.2GHz for the CPU with 2GB main memory and Linux for the OS.

[5]We record the history of the flipping and stop the computation if the same configuration appears twice, which occurs very rarely.

(a) (b) (c)

Figure 7. (a), (b) Real images of a polyhedral object with a Delaunay triangulation (58 edges). (c) Optimization by our method (100% correct, 2 rounds, 3.43 sec).

(a) (b) (c)

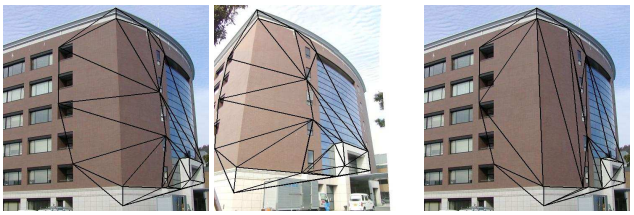
Figure 8. (a), (b) Initial triangulation (31 edges). (c) Optimization by our method (100% correct, 3 rounds, 3.15 sec).

Figs. 8∼11 show other examples along with the correctness, the number of rounds, and the computation time. We could automatically detect and match feature points [4, 12], but some mismatches are inevitable. Since our concern here is not the matching accuracy but the performance of mesh optimization, we selected matching points by hand.

From Figs. 8∼11, we can see that our method works very well. It is effective even when incorrect edges cannot be corrected by a single flipping operation (see Fig. 9). In such a case, the intensity difference region is not a marked triangle as shown in Fig. 2(c) but an obscure blob. Still, our method can correct such incorrect edges in the end after iterations.

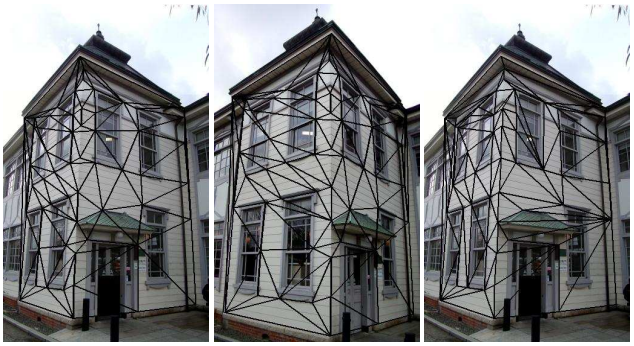
6.2 Comparisons

We compared our method with the methods of Morris and Kanade [7] and Perrier et al. [8]. Table 1 lists the correctness of the three methods for Figs. 8∼11.



(a) (b) (c)

Figure 9. (a), (b) Initial triangulation (47 edges). (c) Optimization our method (100% correct, 3 rounds, 4.03 sec).



(a) (b) (c)

Figure 10. (a), (b) Initial triangulation (157 edges). (c) Optimization by our method (98.7% correct, 3 rounds, 11.85 sec).

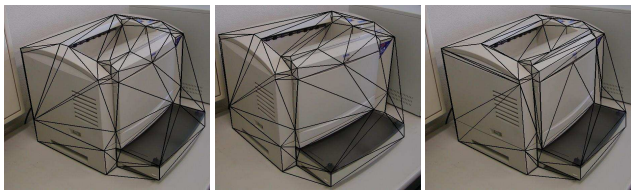
From this, we can see that our method is generally superior to the methods of Morris and Kanade [7] and Perrier et al. [8]. The exception is Fig. 10; our method was unable to correct one edge, because the surrounding quadrilateral is too small and the texture is too homogeneous.

The essential difference between the methods of Morris and Kanade [7] and Perrier et al. [8] is whether the intensity is normalized to cancel the illumination changes. From Table 1, we see that the method of Perrier et al. [8] generally results in lower correctness. This is because canceling illumination increases the similarity between patches, decreasing the discrimination capability. The exception in which the method of Perrier et al. [8] is superior is Fig. 9, in which considerable illumination changes exist between the images.

In contrast, our method is indifferent to interframe illumination changes, because we do not compare different frames; it only accumulates the evidence over the frames, so our method achieves 100% correctness.

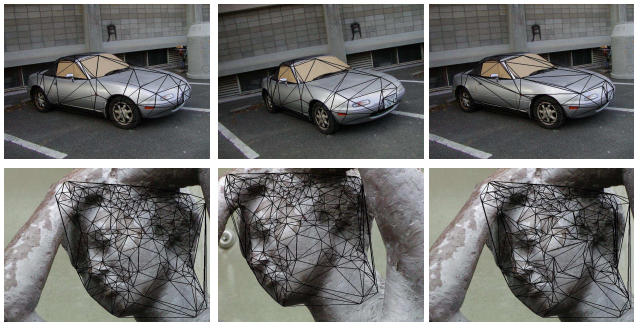
Table 1. Comparative Correctness (%).

	Morris & Kanade	Perrier et al.	our method
Fig. 7	100	95.7	100
Fig. 8	73.9	69.6	100
Fig. 9	89.2	91.9	100
Fig. 10	100	98.7	98.7
Fig. 11	92.4	85.7	96.2



(a) (b) (c)

Figure 11. (a), (b) Initial triangulation (114 edges). (c) Optimization by our method (96.2% correct, 7 rounds, 3.15 sec).



(a) (b)

Figure 12. Optimization for curved surfaces. (a) Initial triangulation. (b) Optimization by our method.

We also applied our method to curved surface scenes (Fig. 12). The feature points in the first example were generated and matched by the method of [4]. We see that our method yields better polyhedral approximations, though quantitative evaluation is difficult.

6.3 Patch-based similarity and correctness

We closely examined the example of Fig. 8 to see why the methods of Morris and Kanade [7] and Perrier et al. [8] are considerably inferior to ours. Fig. 13(a) plots in solid line for each iteration the total sum of square intensity differences between corresponding patches (the left scale); the dashed line plots the correctness in percentage (the right scale). Since the method of Morris and Kanade [7] minimizes this sum, it indeed decreases monotonically at each iteration, yet the correctness does not increase at the second iteration. Fig. 13(a) plots the intensity-normalized sum of squares to be minimized (equivalent to maximize the normalized correlation) corresponding to the method of Perrier et al. [8]. Again, it monotonically decreases, yet the correctness does not increase accordingly.

Thus, we conclude that the patch similarity measures of Morris and Kanade [7] and Perrier et al. [8] are not good indicators of the correctness of triangulation, whether illumination changes are canceled or not.

In contrast, our method focuses only on texture discontinuity in the region where it is potentially most

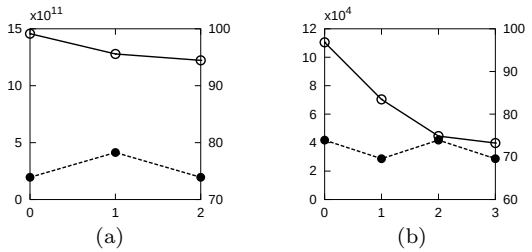


Figure 13. The number of iterations (vertical axis) vs. the total dissimilarity measure of the corresponding patches (left axis: —○—) and the correctness % (right axis: ...●...). (a) Morris and Kanade. (b) Perrier et al.

conspicuous, achieving 100% accuracy.

6.4 Use of video frames

We applied our method to video sequences as well. We manually specified corner points in the first frame and tracked them using the Kanade-Tomasi-Lucus algorithm[6]. Then, we defined a Delaunay triangulation over the middle frame, which is, according to our experience, least likely to cause patch reversals like those shown in Fig. 6. We optimized the mesh by (i) using the entire stream and (ii) using only the initial and the final frames.

According to our experiments (not shown here), the results using two frames are the same as using all frames. In contrast, the correctness somewhat deteriorates for the methods of Morris and Kanade [7] and Perrier et al. [8] if only two images are used. Overall, our method is superior to the methods of Morris and Kanade [7] and Perrier et al. [8].

7. Conclusions

We proposed a new technique for automatically transforming a given triangular mesh so that it is compatible with the physical object shape. Doing real video image experiments, we conclude:

1. Our method is superior to existing methods except when the patches are extremely small.
2. The use of the initial and the final video frames is sufficient rather than using all the frames.

The superiority of our method is because:

- The patch similarity measures of Morris and Kanade [7] and Perrier et al. [8] do not necessarily reflect the correctness of the mesh, while
- our method focuses on texture discontinuity in the region where it is potentially most conspicuous.
- Intensity normalization of Perrier et al. [8] deteriorates the accuracy except in the presence of strong illumination changes, while
- our method is indifferent to interframe illumination changes, because we do not compare different frames.

[6]Available at `http://vision.stanford.edu/~birch/klt/`

Like the methods of Morris and Kanade [7] and Perrier et al. [8], our method does not require any thresholds for judgment. Our method is also effective for curved surface scenes in generating a better polyhedral approximation.

Our method optimizes a given triangular mesh, but if some corners are missing, the resulting mesh does not represent a faithful 3-D shape however we optimize it. Also, too many feature points reduce efficiency. Perrier et al. [8] presented a scheme for removing points that produce extremely narrow triangles and adding new points inside large triangles. That type of process will be effective in practice.

Acknowledgments: This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under a Grant in Aid for Scientific Research C(2) (No. 17500112). The authors thank Daniel Morris of Northrop Grumman Corporation, U.S.A. for helpful comments and Masakazu Murata of Kumahira, Ltd., Japan, for helping the real image experiments.

References

[1] E.H. Adelson and J.R. Bergen, The plenoptic function and the elements of early vision, in M. Landy and J.A. Movshon (Eds.), *Computational Models of Visual Processing*, MIT Press, Cambridge, MA, U.S.A., pp. 3–20, Oct. 1991.

[2] S.J. Gortler, R. Gzreszczuk, R. Szeliski, and M.F. Cohen, The lumigraph, *Proc. SIGGRAPH*, pp.43–54, New Orleans, LA, U.S.A., August 1996.

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

[4] Y. Kanazawa and K. Kanatani, Robust image matching preserving global consistency, *Proc. 6th Asian Conf. Comput. Vision*, Jeju, Korea, Vol. 2, pp. 1128–1133, Jan. 2004.

[5] K. Kutulakos and S. Seiz, A theory of shape by space carving, *Proc. Int. Conf. Comput. Vision*, Kerkyra, Greece, pp.307–314, Sept. 1999.

[6] M. Levoy and P. Hanarahan, Light field rendering, *Proc. SIGGRAPH*, pp. 31–42, New Orleans, LA, U.S.A., August 1996.

[7] D.D. Morris and T. Kanade, Image-consistent surface triangulation, *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, Hilton Head, SC, U.S.A., Vol. 1, pp. 332–338, June 2000.

[8] J. S. Perrier, G. Agin, and P. Cohen, Image-based view synthesis for enhanced perception in teleoperation, in J. G. Verly (Ed.), *Enhanced and Synthetic Vision 2000, Proc. SPIE*, Vol. 4023, June 2000.

[9] F. Preparata and M. Shamos, *Computational Geometry*, Springer, Berlin, Germany, 1985.

[10] G. Vogiatzis, P. Torr, and R. Cipolla, Bayesian stochastic mesh optimization for 3D reconstruction, *Proc. British Machine Vision Conf.*, Norwich, U.K., Vol. 2, pp. 711–718, Sept. 2003.

[11] T. Yu, N. Xu, and N. Ahuja, Shape and view independent reflectance map from multiple views, *Proc. 8th Euro. Conf. Comput. Vision*, Prague, Czech., Vol. 4, pp. 602–615, May 2004.

[12] Z. Zhang, R. Deriche, O. Faugeras and Q.-T. Luong, A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry, *Artif. Intell.*, **78** (1995), 87–119.