Proc. 10th Int. Conf. Virtual Sys. MultiMedia (VSMM2004), Nov. 2004, Ogaki, Japan, pp. 564–573.

Generating a Triangular Mesh Adapted for Shape Reconstruction from Images

Atsutada Nakatsuji*, Yasuyuki Sugaya†, and Kenichi Kanatani†

*System Solutions Division II, NEC Engineering, Ltd.

†Department of Information Technology, Okayama University, Okayama, Japan

Abstract. In reconstructing 3-D from images based on feature points, one usually defines a triangular mesh that has these feature points as vertices and displays the scene as a polyhedron. If the scene itself is a polyhedron, however, some of the displayed edges may be inconsistent with the true shape. This paper presents a new technique for automatically detecting and eliminating such inconsistencies by using a special template. All the procedures do not require any thresholds to be adjusted. Using real images, we demonstrate that our method is efficient with high capability to correct inconsistencies.

1. Introduction

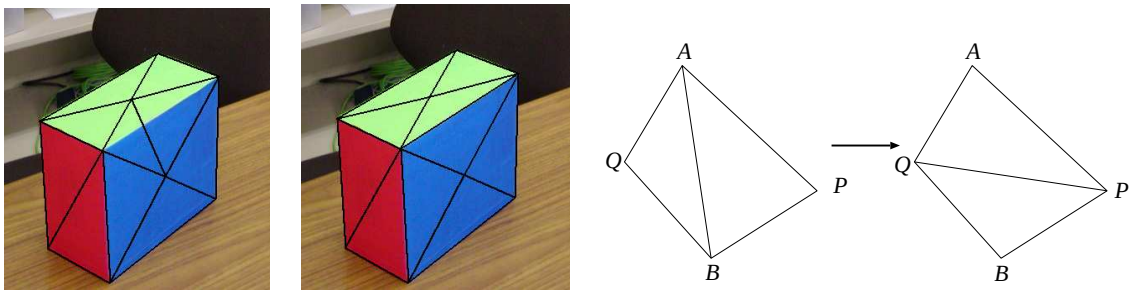
One of the most important problems for 3-D reconstruction from images is how to represent the reconstructed shape. If we use stereo vision using calibrated cameras, we can obtain a dense depth map over all the pixels. By inter-pixel interpolation, we can display the scene as a curved surface. Alternatively, we can use a technique called *space carving* [5] and represent the scene as an aggregate of colored voxels. More sophisticated methods, called by such names as *plenoptic representation* [1], *light field rendering* [6], and *lumigraph* [2], are to register all the light rays in the scene to generate new views seen from an arbitrary viewpoint.

For images taken by uncalibrated cameras, on the other hand, we first extract corresponding feature points from them and then compute their 3-D coordinates, whether we deal with a continuous video stream using a method such as the *factorization* [10] or impose the *epipolar geometry* [3] on separate images. Then, we define a triangular mesh that has the feature points as vertices and display the scene as a texture-mapped polyhedron.

The triangular mesh is usually generated by *Delaunay triangulation* [9] of feature points in a specified frame. This 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, and the vertices of polyhedral objects are likely to be chosen as feature points. In such a case, the edges of Delaunay triangulation may not coincide with the physical edges. If we use such triangulation for polyhedral representation, the displayed shape may be inconsistent with the true polyhedral shape.

See Fig. 1, for example. The Delaunay triangulation in Fig. 1(a) does not correctly represent the object shape. The triangulation in Fig. 1(b), on the other hand, correctly represents the object shape. The aim of this paper is to present a new technique for automatically transforming a given triangulation into a physically compatible one.

Sec. 2 and 3 describe the principle of our method. The details of the procedure are described in Sec. 4 $\sim$ 6. In Sec. 7, we show real image examples to demonstrate that our method is more efficient with higher capability to correct inconsistencies than a method based on a conventional criterion. The marked characteristic of our method is that no thresholds are required to be adjusted.



(a) (b) (c)

Figure 1: (a) Triangulation inconsistent with the object shape. (b) Triangulation consistent with the object shape. (c) Edge flipping.

2. Compatibility of Triangulation

Many studies have been done in the past for generating optimal triangular meshes. They are roughly classified into two categories: one is to replace a dense mesh by a coarser one without impairing the faithfulness of the shape representation; the other is to upgrade the descriptiveness of the shape by adding vertices and edges. For the former, Vogiatzis et al. [11] introduced stochastic annealing coupled with Bayesian estimation on the assumption that the object mostly consists of planar faces. For the latter, Yu et al. [12] refined the mesh by iteratively estimating both the object shape and the surface reflectance map.

To our knowledge, however, the only studies of optimizing edges for a given set of vertices are those of Morris and Kanade [7] and Perrier et al.[1] [8]. The basic principle for inconsistency detection is to compare the texture in corresponding triangular patches in different images. Suppose we have two images of a polyhedral object. If a triangular patch is defined on a planar surface of it, its texture in one image can be mapped onto the corresponding patch in the other image by an affine transformation[2]. Hence, the intensity difference[3] after the mapping should be zero in that patch. If not, the patch is not on a planar surface, so we "flip"[4] an appropriate edge into the diagonal position as illustrated in Fig. 1(c). Iterating this, we should end up with a triangular mesh compatible with the object shape [7].

In reality, however, the intensity difference is not exactly zero due to various disturbances such as inaccuracies of feature point matching, viewpoint dependent reflectance changes, and supposedly planar faces not being exactly planar. However, setting an appropriate threshold for judging planarity is very 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 between the corresponding pixel values (to be

[1]They also discuss splitting and merging of the mesh.

[2]Theoretically, corresponding patches in different views 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.

[3]In this paper, we consider color images and refer to the root-mean-square difference in the R, G, and B values simply as "intensity difference".

[4]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].

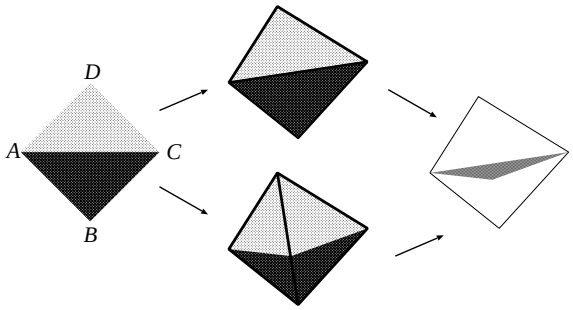


Figure 2: Two 3-D interpretations of a quadrilateral $ABCD$ and the predicted intensity difference.

maximized). In this paper, we show that the use of such a patch-based (dis)similarity measure is insufficient and present a more effective measure. Using real images, we demonstrate that the use of our measure is more efficient with higher capability to correct inconsistencies.

3. Principle of Inconsistency Detection

Given a triangular mesh over a polyhedral object scene, we hereafter say that an edge of the mesh is *correct* if it entirely lies on a planar face, and *incorrect* otherwise. By definition, an incorrect edge connects two points on different faces. We assume that the texture, color, or brightness of the object is different from face to face.

Figure 2 illustrates the principle of our incorrect edge detection. Consider the 2-D quadrilateral $ABCD$ in the left of Fig. 2. Two possibilities exist for interpreting it as a 3-D polyhedron: one with faces $\triangle ABC$ and $\triangle CDA$; the other with faces $\triangle ABD$ and $\triangle BCD$. Suppose we view this object from a different angle. If the former interpretation is the case, the object should look as shown in the upper middle; if the latter is the case, it should look as shown in the lower middle. The right figure shows their intensity difference (darker tones correspond to larger values; the white part has no difference).

Suppose another image shows its true projection viewed from that angle. The above observation implies that if we map the texture of the original image onto it, the intensity difference should look as shown in the right figure if the interpretation is incorrect, while no difference should arise if the interpretation is correct.

Figures 3(a),(b) are real images of a polyhedral object, on which a Delaunay triangulation (based on (a)) is overlaid; we selected the vertices at object corners by hand. Mapping the texture of Fig. 3(b) onto Fig. 3(a) patch by patch, we obtain Fig. 3(c). Figure 3(d) shows the intensity difference between Fig. 3(a) and Fig. 3(c). We observe narrow dark triangular regions that cross incorrect edges, as expected. We hereafter call such a region where the intensity difference markedly appears an *inconsistency region*.

4. Inconsistency Detection Template

The above observation leads to the idea of detecting inconsistency regions by a template *specifically designed to detect them*. Figure 4(a) shows our template (lighter tones correspond to larger values). It is defined over a square region $ORST$ of size $l \times l$

(a) (b) (c) (d)

Figure 3: (a),(b) Real images of a polyhedral object with a Delaunay triangulation (58 edges). (c) Texture mapping of (b) onto (a). (d) Intensity difference between (a) and (c). Darker tones correspond to larger values.

with the following value:

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

The template value is symmetric with respect to the diagonal OS and anti-symmetric 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. Figure 4(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[5].

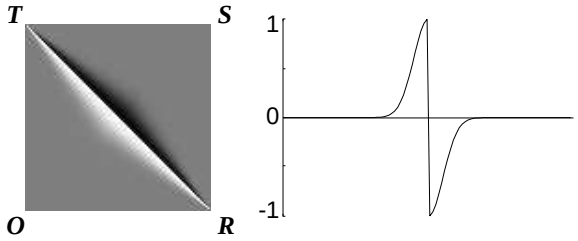
For a given edge, we map the intensity difference of the two triangles adjacent to it onto $\triangle OSR$ and $\triangle OST$ and compute the correlation (the sum of the product of corresponding pixel values) with this template. We set the template size l in such a way that the average area of the triangular patches in the input images is approximately $l^2/2$. The reason why we use an anti-symmetric template is that we do not know a priori on which side the inconsistency region appears; it should lie only on one side of the diagonal of the surrounding quadrilateral (see Figs. 2 and 3(d)). Since the intensity difference is nearly zero on the other side, we can detect the inconsistency region, on whichever side it lies, by computing the absolute value of the correlation. It also has the advantage of canceling small fluctuations in the intensity difference caused by texture mapping inaccuracy, since such fluctuations are expected to spread randomly and evenly over the quadrilateral region.

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 prevent texture mapping discrepancies caused by inaccuracies in locating feature points, since the periphery of a patch may be encroached by the texture of an adjacent patch.

5. Evaluation of Edge Incorrectness

Given an initial triangulation over two corresponding images, we first measure the degree of incorrectness $w(AB)$ of each edge AB using the template $T(x,y)$ of Eq. (1).

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



(a) (b)

Figure 4: (a) Inconsistency detection template. Lighter tones correspond to larger values. (b) Cross section along OS.

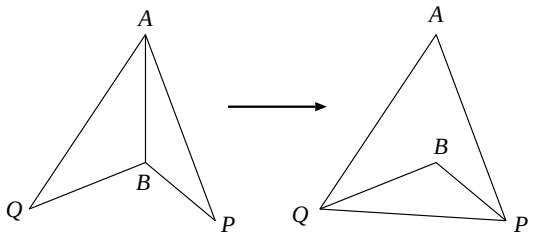


Figure 5: Edge flipping for a concave quadrilateral would result in a reversed patch.

For this, we make the computation symmetric with respect to the two images: instead of mapping the texture from one image onto the other and computing the intensity difference there as described earlier, we directly map the texture onto the template region $ORST$ by a homography and compute the intensity difference there. The procedure is 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 either image, meaning that we do not flip that edge (since a reversed patch would result; see Fig. 5).
3. Otherwise, map the texture in the quadrilateral $APBQ$ in the first image onto the template region $ORST$ by a *homography* and write down the intensity values there.
4. *Affinely* map the texture in $\triangle ABP$ and $\triangle ABQ$ in the second image onto $\triangle OSR$ and $\triangle OST$, respectively, and *subtract* the intensity values from the values written there.
5. Map the texture in the quadrilateral $APBQ$ in the second image onto the template region $ORST$ by a *homography* and *add* the intensity values to the values written there.
6. *Affinely* map the texture in $\triangle ABP$ and $\triangle ABQ$ in the first image onto $\triangle OSR$ and $\triangle OST$, respectively, and *subtract* the intensity values from the values written there.
7. Compute the correlation of the values written there with the template $T(x, y)$ of Eq. (1), and output its absolute value as $w(AB)$.

Here, we are assuming that at the time of generating the mesh each edge is classified either into a boundary edge with only one triangles on one side or into an internal edge with two triangles on both sides.

6. Procedure of Mesh Optimization

Given two images and corresponding feature points on them, we define a Delaunay triangulation using the feature points in the first image and isomorphically map it to the corresponding points in the the second image. Then, we compare the *signs* of corresponding triangular patches, where we define the sign of $\triangle ABC$ to be 1 if the order of A, B, and C is counterclockwise, -1 if clockwise, and 0 otherwise (i.e., degeneracy

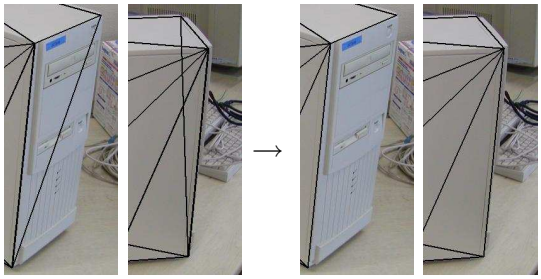


Figure 6: If one side of the reversed triangle is a boundary edge, we eliminate it.

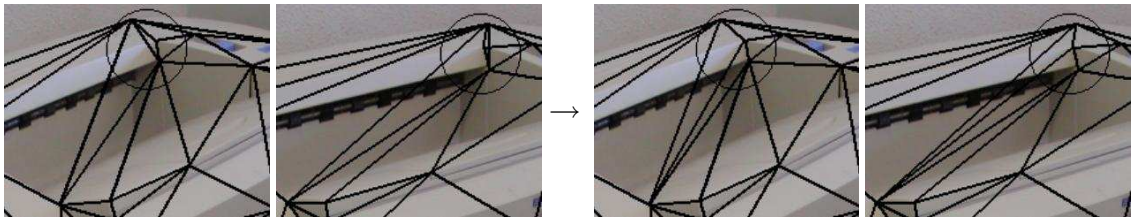


Figure 7: If the reversal occurs inside, we flip an appropriate side of the triangle.

into a line segment).

If the signs are different between the two images, the triangle in the second image is reversed. We dissolve such reversals as follows[6]. If one side of the reversed triangle is a boundary edge, we simply eliminate it (see Fig. 6). If the reversal occurs inside, we flip an appropriate side of the triangle, as discussed by Morris and Kanade [7] (see Fig. 7).

Once such patch reversals are resolved, no new reversals occur thereafter, since we check the concavity of the surrounding quadrilateral before flipping edges (Step 2 of the procedure in Sec. 5).

After resolving patch reversals, we do the following procedure:

1. Compute the incorrectness measure $w()$ for all the edges (Sec. 5).
2. Find the edge AB that has the largest value $w(AB)$.
3. Stop if $w(AB) = 0$.
4. Flip the edge AB to PQ as shown in Fig. 1(c) and compute $w(PQ)$.
5. If $w(PQ) > w(AB)$, eliminate the edge PQ and restore the edge AB. Then, let $w(AB) = 0$.
6. Otherwise, recompute $w()$ for edges PA, PB, QA, and QB, if $w()$ is not already 0, with respect to the new mesh configuration.
7. Go back to Step 2.

In this process, the value w itself has no absolute meaning; it is used only for comparison. Hence, no artificial thresholds need to be introduced. Since the largest value of $w()$ monotonically and strictly decreases at each flipping, and since edges once checked are not checked again, the above procedure terminates after all the edges are traversed once.

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, in particular

[6]Theoretically, the reversal can occur more globally, e.g., a large portion of the mesh can be reversed at the same time. In this paper, however, we assume that the scene can be triangulated in such a way that its projection onto any image do not contain reversed triangles and that initial reversals can be resolved by the operations described here.

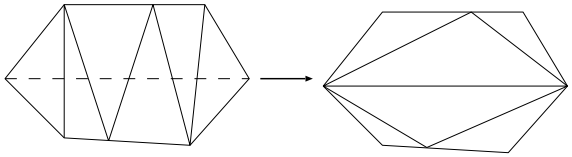
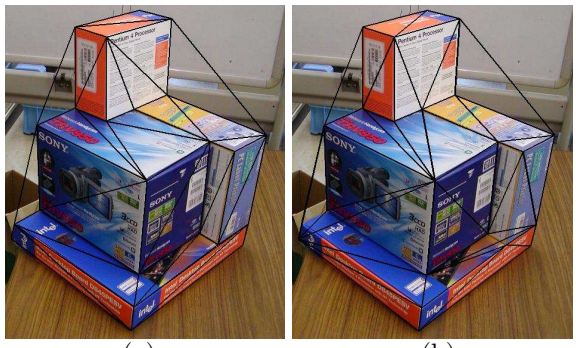


Figure 8: If many mesh edges cross one physical edge (drawn in dashed lines here), multiple steps of flipping is necessary for resolving the inconsistency. See also Fig. 11.



(a) (b)

Figure 9: Optimization of the mesh in Figs. 3(a),(b). (a) Our method (100% correct, 2 rounds, 3.43 sec). (b) Patch-similarity maximization (91.5% correct, 270 iterations, 9.05 sec).

when one physical edge is crossed by multiple mesh edges (Fig. 8; see also Fig. 11). So, we repeat the above procedure until the mesh configuration does not alter any further[7].

7. Experiments

Using real images, we compared the performance of our method with that of patch-similarity maximization. As the similarity measure, we adopted the normalized correlation used by Perrier et al. [8] rather than the sum of square differences used by Morris and Kanade [7]. This is because we have found that the Morris-Kanade measure often detects false intensity differences caused by reflectance changes depending on viewing angles[8].

In our experiment, we used a slightly modified implementation suitable for color images: instead of directly computing the normalized correlation, we computed the sum of square intensity differences after normalizing the intensity of each patch by a linear function[9] in such a way that the average intensity over the patch is 0 and their variance is 1 in both images. For maximizing the total similarity (minimizing the total dissimilarity, to be precise), we randomly selected edges and flipped them as long as the dissimilarity decreases. We stopped this when no effective flipping occurred for consecutive E times, where E is the number of the edges of the mesh.

Applying our optimization to Figs. 3(a),(b), we obtain the mesh shown in Fig. 9(a). The iterations converged in two rounds of the procedure of Sec. 6. The correctness and the computation time are also written in the caption, where the correctness is measured by (the number of correct edges)/(the number of non-boundary edges) in percentage. We used Pentium 4 3.2GHz for the CPU with 2GB main memory and Linux for the OS.

We compared this result with the patch-similarity maximization. Since the result changes at each trial because of the randomness of the search, we showed one typical

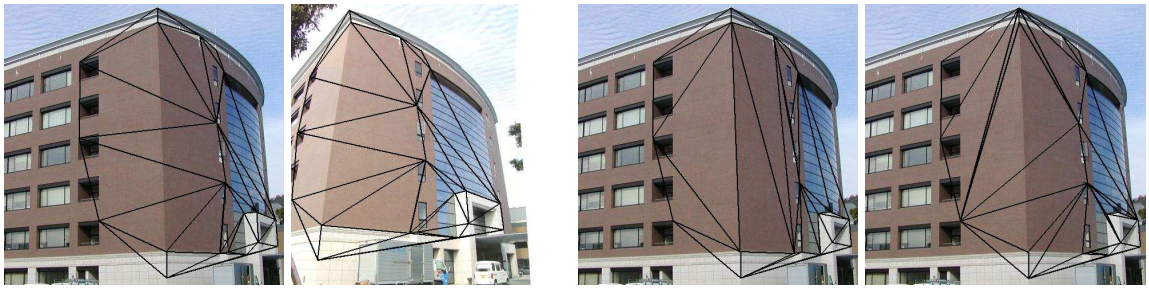
[7]We record the history of the flipping and stop the computation if the same configuration appears twice, which occurs very rarely, though.

[8]In contrast, our method uses an anti-symmetric template, so intensity differences uniformly caused by reflectance changes are canceled.

[9]We used color images and normalized the R, G, and B values by a *common* linear function so that the sum of the R, G, and B values over the patch is 0 and their variance is 1.

(a) (b) (c) (d)

Figure 10: (a),(b) Initial triangulation (31 edges). (c) Our method (100% correct, 3 rounds, 3.15 sec). (d) Patch-similarity maximization (75.2% correct, 159 iterations, 5.43 sec).



(a) (b) (c) (d)

Figure 11: (a),(b) Initial triangulation (47 edges). (c) Our method (100% correct, 3 rounds, 4.03 sec). (d) Patch-similarity maximization (96.2% correct, 313 iterations, 7.62 sec).

result in Fig. 9(b). The correctness, the number of iterations, and the computation time written there are their averages over 10 trials.

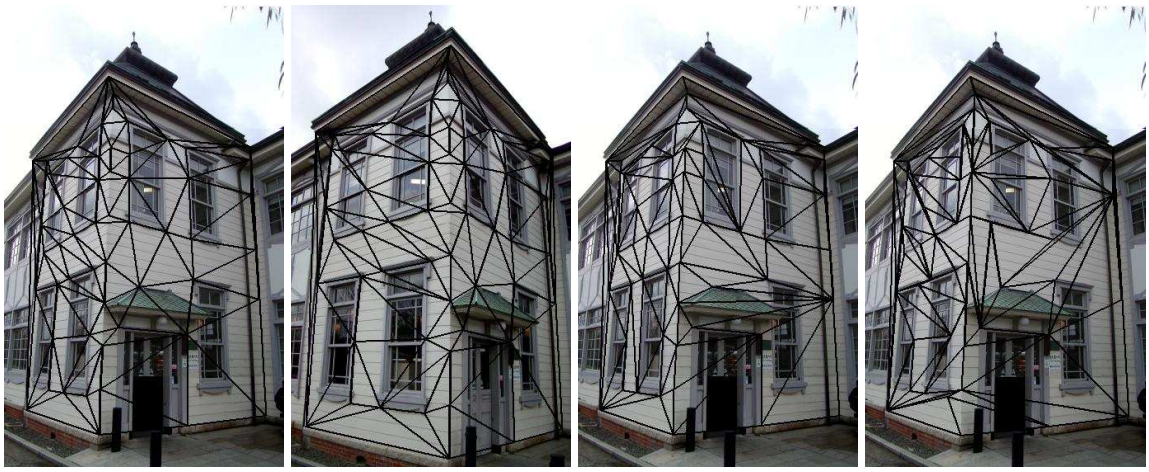
Figures 10∼13 show other real image examples. Today, many algorithms are available for automatically extracting and matching feature points, e.g., [4, 13]. However, our concern here is not the accuracy of automatic matching but the performance of mesh optimization, so we selected matching points by hand. In all examples, figures (a),(b) are input images with the initial Delaunay triangulation overlaid; figures (c),(d) are the results corresponding to Figs. 9(a),(b), respectively.

8. Observations

From the above results, we can see that our inconsistency detection template works very well. It is effective even if incorrect edges cannot be corrected by a single flipping operation (see Figs. 11(a),(b)). Although the inconsistency regions are not so marked as shown in Fig. 2, they still appear in the form of small narrow blobs crossing incorrect edges, so our template can also detect such inconsistencies very well.

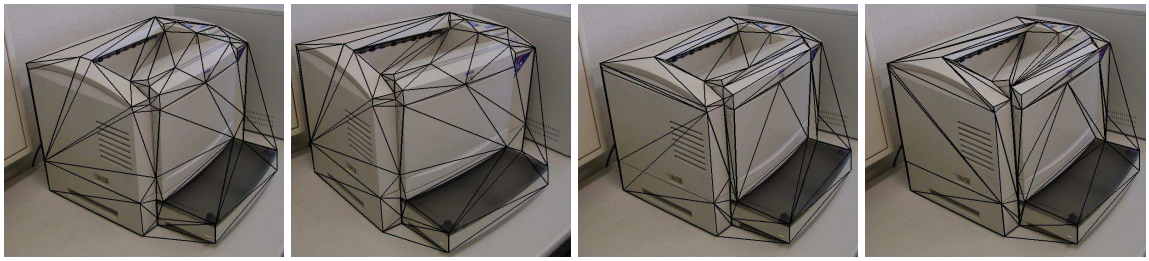
The effectiveness of our method is not restricted to exactly polyhedral scenes. It is also effective for objects that have curved surfaces as well (see Fig. 14): our method yields better polyhedral approximations.

Patch-similarity maximization, on the other hand, sometimes replace all incorrect edges correctly, but overall our method has higher performance in removing incorrect edges. This is perhaps because partially distorted texture is not sensitively reflected if the similarity is measured across the entire patch. In fact, we have found that the resulting meshes often have *much lower dissimilarity* than the desired configuration, meaning that the patch-based correlation is not a good measure of shape consistency.



(a) (b) (c) (d)

Figure 12: (a),(b) Initial triangulation (157 edges). (c) Our method (98.7% correct, 7 rounds, 11.80 sec). (d) Patch-similarity maximization (99.3% correct, 2611 iterations, 19.60 sec).



(a) (b) (c) (d)

Figure 13: (a),(b) Initial triangulation. (c) Our method (96.2% correct, 4 rounds, 11.96 sec). (d) Patch-similarity maximization (83.8%, 866 iterations, 19.70 sec).

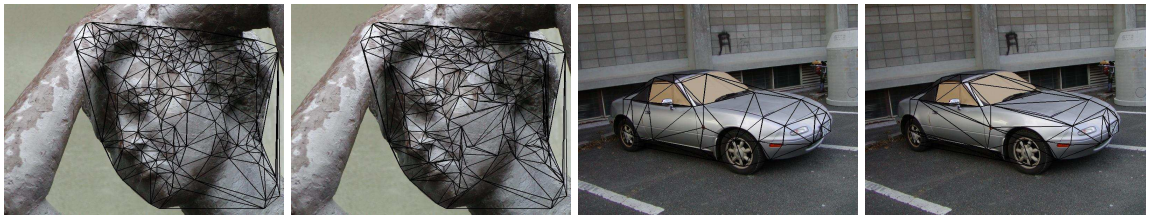
In contrast, our method focuses specifically on inconsistency regions where the texture difference is most conspicuous, thereby sharply detecting inconsistencies.

The computation time of patch-similarity maximization depends on when we stop the search, and there is no definite criterion for it. If use the stopping condition mentioned earlier, the patch-similarity maximization takes more time than our method, as we can see. Thus, our method is superior in both accuracy and efficiency.

9. Concluding Remarks

We proposed a new technique for automatically transforming a triangular mesh so that it is compatible with the physical object shape. To do this, we introduced a template that can sensitively detect shape inconsistencies. Our procedure does not require any thresholds to be adjusted. Using real images, we demonstrated that our method is more efficient with higher capability to correct inconsistencies than patch-similarity maximization.

In this paper, we used only pairs of images, but the principle can be straightforwardly extended to multiple images. Doing some experiments (not shown here for space shortage), we confirmed that the correctness can sometimes increase by using multiple images, but in most cases two separate views are sufficient.



(a) (b) (c) (d)

Figure 14: Triangulation of curved surfaces. (a),(c): Delaunay triangulation. (b),(d): optimized triangulation. The mesh in (a) was obtained using our automatic matching tool [4].

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. 15500113). The authors thank Masakazu Murata of Okayama University 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] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography—A factorization method," Int. J. Comput. Vision, vol.9, no.2, pp.137–154, Oct. 1992.

[11] 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.

[12] 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,

[13] 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., vol.78, pp.87–119, 1995.