EUROGRAPHICS '99 / P. Brunet and R. Scopigno (Guest Editors) *Volume 18 (1999), Number 3*

Efficient and Handy Texture Mapping on 3D Surfaces

Kenji Matsushita and Toyohisa Kaneko

Department of Information and Computer Sciences, Toyohashi University of Technology, Toyohashi, Japan
kaneko@tutics.tut.ac.jp

Abstract

There has been a rapid technical progress in three-dimensional (3D) computer graphics. But gathering surface and texture data is yet a laborious task. This paper addresses the problem of mapping photographic images on the surface of a 3D object whose geometric data are already known. We propose an efficient and handy method for acquiring textures and mapping them precisely on the surface, employing a digital camera alone. We describe an algorithm for selecting a minimal number of camera positions that can cover the entire surface of a given object and also an algorithm to determine camera's position and direction for each photograph taken so as to paste it to the corresponding surfaces precisely. We obtained a matching accuracy within a pixel on a surface through three experimental examples, by which the practicability of our method is demonstrated.

1. Introduction

Rendering a three-dimensional (3D) model requires 3D surface data and surface textures. Optical scanners such as a Cyberware Scanner are commonly used for acquiring such data from a real object. Most of the scanners set an object on a rotating table[1], or rotate a camera around a stationary object, as adopted by Cyberware. A laser beam slit is commonly used for shape measurement; for texture acquisition, however, an optical system including a fluorescence lamp is additionally required. Many inexpensive scanners often lack the capability of texture acquisition since it is costly to equip two optical systems.

On the other hand, a CT (Computer Tomography) equipment, which is commonly used for scanning the human body, may be used as a device for shape measurements, which obviously has no capability of acquiring textures. Consequently, the measured surface data requires textures to be added on hand.

For these reasons, there is a need and technical problem of adding texture data to a 3D object of known geometrical data. To the problem, we propose a method for manually positioning a digital camera without the use of precise mechanical devices, and that for mapping textures taken from the camera onto the surfaces. It will be shown that a minimal set of camera positions to cover the entire surfaces can be specified once the surface data are available.

1.1. Related Works

In the field of computer vision, estimation of the position and shape of an object of interest has been an important subject of research[2]. A stereo matching is a common technique, where textures on an object are usually acquired simultaneously with shape measurement. It is however difficult to apply this technique to an object that possess no distinct textures for matching.

Suenaga and Watanabe[3] developed color data acquisition capability for a rangefinder that scans objects cylindrically. Debevec et.al.[4] reported acquisition of a shape and color for an architectural building consisting solely of polyhedrons. Following the work, he and his coworkers[5] proposed view-dependent texture mapping (VDTM) to make an animation with varying a viewpoint and also smooth blending between three texture images. Sato et.al.[6] proposed a method for acquiring textures, reflection properties, and shapes by using a projector-type rangefinder. A number of researches[8, 9, 10, 11, 12] were concerned with measuring the geometry of occluded parts of an object with a rangefinder which is manipulated by a robot arm.

There are some related works in a medical area, especially in surgery simulation[13]. For facial surgery simulation, a color texture has to be mapped on the 3D surface obtained from CT data. Koch et.al.[14] used CT data for facial reconstruction and employed a Cyberware for acquiring facial tex-

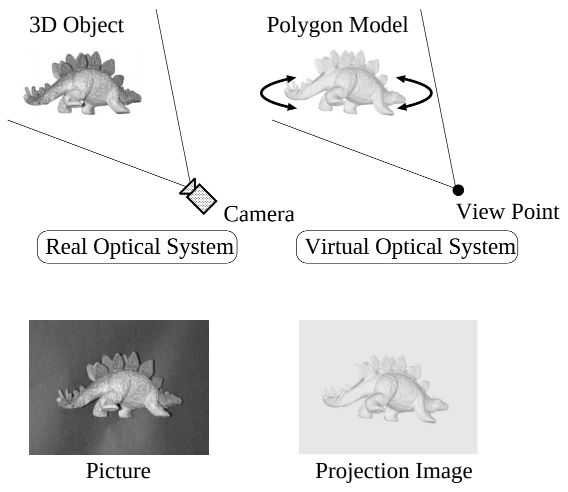

Figure 1: *Real and Virtual Camera System*

tures. They used 3D-to-3D matching for aligning 3D surfaces from a Cyberware with those from CT. Haider et.al.[15] proposed use of color photographs by 3D-to-2D matching based upon the registration with manually selected landmarks.

1.2. The approach of this work

Our approach to the problem is to compute a minimal number of pictures of a real object, based upon the known surface data of a real object on hand, to take pictures resembling the predicted pictures as closely as possible by just holding a camera with a hand or held on a tripod, and finally to map textures from the pictures on the corresponding surfaces as precisely as possible.

The basic concept of our approach may be explained by the setting of two optical systems: a real and virtual camera system as shown in Figure 1. The virtual camera system simulates the real camera system by using the known geometrical surface data of the object. We then vary the position of the virtual camera in a uniform spacing around the virtual object. We next identify the surface polygons visible from each camera's position, and finally determine a minimal number of camera positions that efficiently cover the entire surfaces of the object. A real digital photograph is taken so as to coincide with the virtual photograph of the object calculated at each virtual camera position. This is easily done by comparing the image taken by a real camera with the reference image generated by calculations. This manual method may cause a slight positioning error; it, however, ensures precise alignment by adjusting the position and direction of the virtual camera. Finally the textures from the photographs are mapped on the corresponding surfaces precisely by using the geometric relation obtained from the matching.

The advantage of our method is that only a commercially available digital camera is required for obtaining entire surface textures. Another advantage is that an object of complicated shape can be handled by just moving the camera closer to the object.

It is generally assumed here that object surfaces are diffused (not specular) and reasonably fragmented. We will investigate primarily the following four topics:

1) Selection of a minimal number of camera positions,

2) Mapping of the photographs to the corresponding surfaces,

3) Recovery of textures, and

4) Composition of two textures taken from different camera positions.

We describe the above topics subsequently, and present experiments, discussion, and conclusion.

2. Selection of Camera Positions

The virtual camera is set around an object as shown in Figure 1, as it is directed toward the object's center. We consider 320 equally spaced camera positions, and select a minimal number of the positions from them in order to cover the whole surface textures.

2.1. Visibility condition and visibility matrix

We assume that pictures of an object are taken with a light shed of the camera such as an electric flash or a strobe. Consider a surface seen from a direction indicated by a view vector as shown in Figure 2. Let the angle between the normal direction of the surface and the view vector be θ. We call the surface is visible (from the camera) if $\theta < \alpha$, where

$cos\alpha =$ target resolution/camera resolution.

For example, if the camera's resolution is two times of the target resolution, which depends upon the CT resolution for the case of CT, α is 60[degree].

Furthermore we carry another visibility check for occlusion. We employ a Z-buffer[16], executing distance computation two times. In the first cycle we register the distance from each view position to each surface. Actually the three corners of a triangular surface are projected on the Z-buffer plain and each pixel inside the surface is assigned an interpolated distance. (Note here a buffer size of 640x480 is used. See the experiment section for details.) In the second cycle, we check whether each surface is occluded or not based upon its content: namely a surface is occluded if its distance is farther than the buffer content. This check is actually carried out for each pixel of a surface polygon. Even if a single

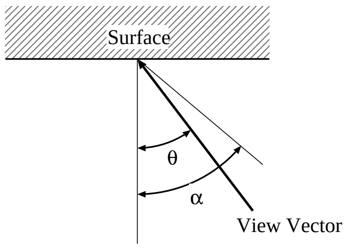

Figure 2: *Visibility Condition*

pixel is occluded, the surface is judged to be occluded. An occluded surface is not visible, as a matter of fact.

After the above two checks, we can define a matrix V, whose (i,j)th component is denoted by v_{ij}, and set $v_{ij} = 1$ if the j-th surface from the i-th camera position is visible and $v_{ij} = 0$ otherwise. We call the resulting matrix *the visibility matrix*.

2.2. Exhibition Surfaces

For applications such as virtual museums, an exhibited object may be seen only from walking persons. Then only a limited portion of the object surfaces may be seen. Textures of polygonal surfaces near the top or underneath can be neglected. The surfaces which can not be seen from any viewing position can be deleted for further processing. However, in this paper we will deal with the case of entire polygonal surfaces since the above case can be easily handled as a special case.

2.3. Edge Sensitivity

As was mentioned in section 1.2, we use the profile of a 3D object for matching it with that in a 2D photograph. When we take a picture, we place a sheet of uniform color as the background. The profile of the object in a photograph then can be easily obtained by applying a proper threshold, and a derivative operation provides a binary edge profile for matching. On the other hand, the profile of a 3D object seen from a camera position is given as the projected (binary) image of all the polygons on a plane having the same resolution as that of the real camera.

When we match two profiles in order to identify exact camera position and direction, it is very important to measure profile changes with respect to the movement of a camera. The resulting measure is called the edge sensitivity. To estimate the edge sensitivity at the i-th camera position, we employ only the surface polygons of the virtual object. Let F_i and F_j be the edge profiles at the i-th and j-th camera position, respectively, where the j-th camera position is a neighboring position of the i-th position as illustrated in Figure 3. For the i-th camera position, we define the profile sensitivity δ_i as follows:

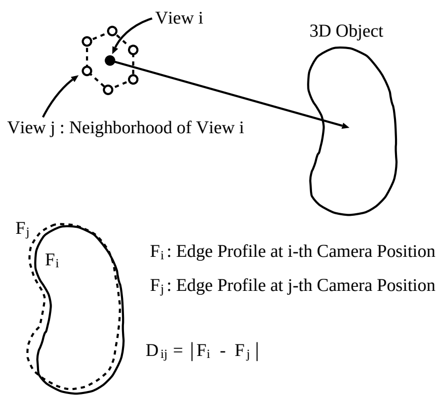
View i
3D Object
View j : Neighborhood of View i
F_j
F_i
F_i : Edge Profile at i-th Camera Position
F_j : Edge Profile at j-th Camera Position
D_ij = |F_i - F_j|

Figure 3: *Edge Sensitivity*

$$\delta_i = \frac{1}{n} \min_j \{D_{ij}\}, \qquad (1)$$

where D_{ij} is the number of edge pixels in F_i whose locations differ from those in F_j, and n is the total number of edge pixels of F_i. Note that $0 \leq \delta_i \leq 1$. For instance, at any camera position the edge sensitivity is zero for a complete globe. Therefore we should not employ a camera position at which the edge sensitivity is below a threshold δ_t. How to determine this threshold will be described in the experiment section later.

2.4. Selection algorithm for camera positions

We first select the camera positions at which the edge sensitivity exceeds a certain threshold δ_t for further processing. The selection will be done based upon the visibility matrix V, whose (i,j)th element v_{ij} is 1 if the i-th surface is visible from the j-th camera position. Therefore we select a set of camera positions so that each column includes at least a single element of $v_{ij} = 1$.

An exhausting method for such selection is to check all the possible combinations of camera positions, by starting any combination of two positions. However, the method becomes impractical according to the increase of the number of surface polygons.

To solve the problem, we developed the following heuristic algorithm:

(1) Count non-zero elements in each column.

(2) Let J be the column, which yields the smallest number at (1).

(3) Calculate the total area S_i among all the row vectors whose v_{iJ} is non-zero, and select the camera position which yields the largest S_i.

(4) Set zero to all the elements of the selected row and

Polygon j		1	2	3	4	5	6	7	8
Area of Polygon j		82	35	19	23	42	56	72	64
	1	1	0	0	1	1	1	1	0
	2	0	0	1	1	0	0	0	1
View i	3	1	1	0	0	0	0	1	0
	4	0	1	0	1	1	1	0	0
	5	0	0	1	0	1	0	1	0

Figure 4: *Initial State*

Polygon j		1	2	3	4	5	6	7	8
Area of Polygon j		82	35	19	23	42	56	72	64
	1	1	0	0	1	1	1	1	0
	2	0	0	1	1	0	0	0	1
View i	3	1	1	0	0	0	0	1	0
	4	0	1	0	1	1	1	0	0
	5	0	0	1	0	1	0	1	0
		2	2	2	3	3	2	3	1

Figure 5: *1st Selection*

Polygon j		1	2	3	4	5	6	7	8
Area of Polygon j		82	35	19	23	42	56	72	64
	1	1	0	*	*	1	1	1	*
	2	0	0	*	*	0	0	0	*
View i	3	1	1	*	*	0	0	1	*
	4	0	1	*	*	1	1	0	*
	5	0	0	*	*	1	0	1	*
		2	2	*	*	3	2	3	*

Figure 6: *2nd Selection*

Polygon j		1	2	3	4	5	6	7	8
Area of Polygon j		82	35	19	23	42	56	72	64
	1	*	0	*	*	*	*	*	*
	2	*	0	*	*	*	*	*	*
View i	3	*	1	*	*	*	*	*	*
	4	*	1	*	*	*	*	*	*
	5	*	0	*	*	*	*	*	*
		*	2	*	*	*	*	*	*

Figure 7: *3rd Selection*

mark the polygon number (#) that includes $v_{ij} = 1$ by * where a 1 is found.

(5) Return to the process (1) until all the entries are marked by *.

The rationale of this algorithm may be understood by an example illustrated in Figures from 4 to 8, where the row entry indicates the surface polygon #, and its area. The column entry identifies the view #, or the camera position #. Figure 4 lists the initial state of the visibility matrix. First we vertically search columns and count the number of 1's. Since the eighth column is 1, the smallest number, the first selection is view #2(see Figure 5). The reason is that polygon #8 is visible only from view #2. Then since this view is already selected, we set all the elements of view #2 to be zero and mark columns or polygons #3, 4,and 8 by *, since these polygons are already taken. Next we again count the number of 1's vertically and four columns are detected to have a total count of 2. Then identify the views intersecting with the four columns of an entry of 1. In this case view#1, 3, and 4 are identified (see Figure 6). When this happens, our policy is to choose the view yielding the largest total surface area. So we calculate their total surface areas: 138, 117, and 91, respectively. Therefore we select the view with the largest area, that is, view #1. Again we set all the entries of view #1 to be zero and mark the column or polygon #1, 5, 6, and 7 by *. After the second selection, we again count vertically, finding only the second column of 1's. By repeating the above process, the third choice is view #3 (see Figure 7). Then all the entries are filled with *, signaling the end of the process (see Figure 8). In summary, we selected view#2, 1, and 3. In this case it can be easily shown that the resulting selection agrees with that by the exhaustive search.

In the above algorithm, the area of a polygon to calculate the total surface area is an intrinsic one, which is independent of view positions. A possible improvement is to use the area seen from each position.

3. Image Matching for Estimating Real Camera Position

3.1. Taking Pictures

For taking a picture from each selected camera position, a precise mechanical equipment may be used to fix the camera as specified. Here our method does not use such a device but a digital camera alone because such an equipment is expensive and also limits the camera's position (e.g. a fixed-length arm of a Cyberware). First, behind the object we place a distinct background sheet. For each camera position which is already identified, we render the virtual photographic image (using a common CG technique) to be seen from its position. We then manually move and adjust the real camera so that its image on the LCD monitor is as closer to the CG image as possible. A tripod may be used to hold a position, and we may take several pictures around each position for safety, in order to select the best matching picture afterwards.

	Polygon j	1	2	3	4	5	6	7	8
	Area of Polygon j	82	35	19	23	42	56	72	64
View i	1	1	0	0	1	1	1	1	0
	2	0	0	1	1	0	0	0	1
	3	1	1	0	0	0	0	1	0
	4	0	1	0	1	1	1	0	0
	5	0	0	1	0	1	0	1	0

Finally Selected Camera Positions

Figure 8: *Final Result*

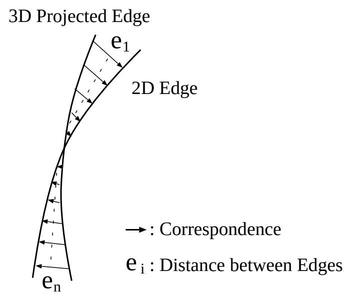

Figure 9: *Edge Correspondence*

3.2. Matching Algorithm

After taking all pictures, the next process is to match the real photographs with the corresponding profiles of the 3D object. As previously mentioned, pictures taken manually often cause misalignment. In order to compensate the misalignment, the position of the 3D object is adjusted algorithmically until exact matching is achieved.

We here use a modified version of Besl and MacKay's iterative algorithm[17]. Let us call the edge profile in a real 2D photograph as a 2D edge image and the edge profile of the virtual 3D object to be seen from a camera position as a 3D edge image, which is two-dimensional, though. Then we define the distance between the two edge profiles as follows(see Figure 9):

$$E = \frac{1}{n}\sum_{j=1}^{n} e_j \qquad (2)$$

For each edge pixel of the 2D edge image, we usually identify the pixel of the 3D edge image which is geometrically the closest. However mismatchings are detected for the edges curved as shown in Figure 10. To avoid this situation, we divide the entire region into 4x4 subregions, and the matching is done between each corresponding subregion. In addition, this modification reduces the cost of computation because the search for the closest pixel is confined to the subregion.

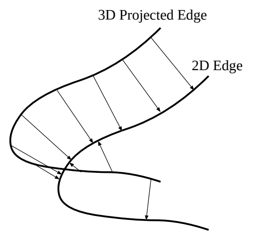

Figure 10: *Erroneous Correspondence*

Note here that there are six degrees of freedom for controlling a virtual camera: a translational vector T_x, T_y, T_z, and rotational angle R_x, R_y, R_z. To adjust these parameters, we employ iterative calculations of a Simplex method[18]. We employ a coarse incremental step initially, and then gradually reduce the step size when the minimization of E with a step size reaches an oscillatory mode. (We also tried Powell's method[18], which provided almost identical results.)

For each parameter we assign a small incremental value u (u_T for translations, and u_R for rotations). Details of these values are in the experiment section. The process of this matching is as follows:

(1) Set initial values for six parameters of virtual camera.

(2) Compute E given in equation(2).

(3) Compute E with current value $\pm\, u_T$, and $\pm\, u_R$.

(4) Replace the current value with the value yielding a smaller E.

(5) Repeat (3) and (4) until the value of E stops decreasing (or becomes oscillatory).

(6) If the value of E is below a thresold, stop the iteration, otherwise go to (7).

(7) Reduce u by a half and return to (2).

4. Filling Holes

For an object of complicated shape, there may be some invisible surface polygons from distant camera positions. If there is a column vector whose elements are entirely 0, the corresponding polygon is invisible from the camera. We call such polygons as holes, which are classified into three types as illustrated in Figure 11. Type 1 is an isolated polygon which is surrounded by visible polygons, Type 2 is a series of polygons which are surrounded by visible polygons except one or two invisible polygons, and Type 3 has at least one invisible polygon internally. For a Type 1 or Type 2 hole, we will replace each invisible polygon by the average of visible polygons in its direct neighborhood, provided that the invisible polygon contains a small area (say, up to several pixels). For Type 3 or for the case that a large invisible polygon is

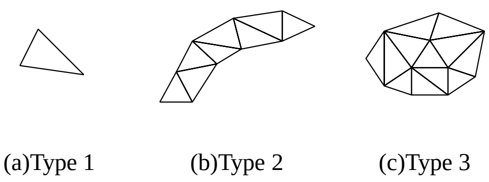

Figure 11: *Types of Holes*

Table 1: *Size of Objects*

	D[mm]	H[mm]	W[mm]
giraffe	148.8	243.1	53.9
dinosaur	132.8	61.4	46.9
dwarf	205.1	264.4	120.3

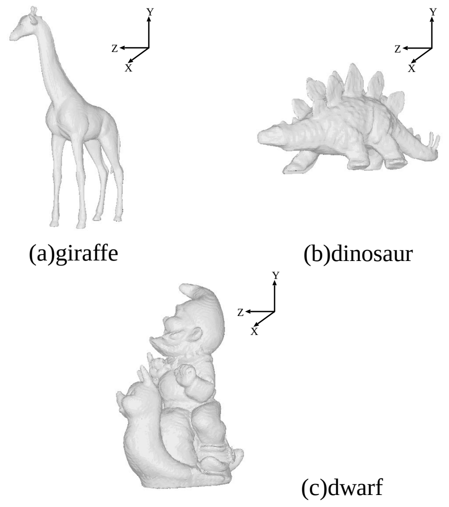

Figure 12: *Rendered figures*

involved, our policy is to take pictures again, usually, from closer distances.

There is a possibility that Type 3 holes are generated due to a deviation of the specified camera position. A remedy is to retake pictures, although the total process time will increase. To avoid this situation, we may take several pictures at each designated camera position for safety as was mentioned.

5. Texture Mapping

Some of the surface polygons are visible from more than two camera positions. To avoid sudden changes in color, we employ blending [5] whose weights are determined by areas as follows:

$$\begin{cases} W_1 = \dfrac{A_1}{A_1+A_2} \\ W_2 = \dfrac{A_2}{A_1+A_2}, \end{cases} \quad (3)$$

where A_1 and A_2 stand for the areas of a polygon seen from view#1 and #2, respectively. Texture mapping will be done for each surface polygon. Each polygon defined on the 3D coordinate will be projected on a 2D plain based upon the exact real camera position and direction which have been obtained as the result of matching.

6. Experiments

We used three plastic toys shown in Figure 14: a giraffe, a dinosaur, and a dwarf. These figures were scanned by a CT scanner and polygonized by applying the method of marching cubes[19] to the CT data. The polygonized surfaces are shown in Figure 12. The resolution of CT data is 1[mm] along the axial direction and .625[mm] along the radial direction or on the slice.

For taking pictures, we used a handheld digital camera, Nikon COOLPIX900, which has a resolution of 1280 by 960 pixels and has a focus distance of 17.4[mm]. Initially we computed its fan-out angle by placing a measure at two distances (50[cm] and 100[cm]). The fan-out angle is required to equate the virtual camera system with the real one.

However, the resolution of images to be processed internally was set to be 640x480, which is a half of the camera's resolution along each direction. This value which is convenient for processing was chosen considering the resolution of the CT data in addition. The Z-buffer for occlusion computation, images to compute edge sensitivity, hypothetical CG images, and final output images have this resolution. (The original textures taken by the camera alone are in the higher resolution of 1280x960.)

We varied the number of camera positions at 80, 320, and 1280, and found that 80 positions were too coarse and 1280 was unnecessarily too minuscule. The camera to each object was set at the minimum distance such that any part will never disappear in the picture. Actually the approximate distance was 130[cm] for the dwarf, 110[cm] for the giraffe, and 60[cm] for the dinosaur toy.

With all the 320 camera positions, we computed edge sensitivity and tabulated a histogram for each toy. The histogram for the dinosaur is shown in Figure 13. Based upon the histogram, we set the threshold for δ_t, which was mentioned in Section **2.3**. In this case, $\delta_t = 0.4$ was chosen so that about 30 percent of the camera positions were the candidates. Then using the algorithm given in Section **2.3**, we identified 7 camera positions for the giraffe, 8 for the dinosaur, and 20 for the dwarf. However, the number of po-

Table 2: *Number of Holes*

	Type 1	Type 2	Type 3
giraffe	83	17	0
dinosaur	97	64	0
dwarf	18	175	88

sitions for the dwarf is insufficient so that several (actually five) closer positions are needed. This process for finding new positions and photographing are carried out after photographing of the initial 20 pictures is completed. When no satisfactory set of camera positions to cover the entire surfaces is found, we reduce the threshold δ_t gradually until such a set is found. (In the above toys, with a starting value of $\delta_t = 0.4$, this reduction process was not needed.)

Then we produced a hypothetical CG image seen at each camera position. An example for the dinosaur is shown in Figure 15. The camera was directed to the object center with such a distance and location that the image taken by the digital camera is as equal to the hypothetical image as possible.

Using the matching process described in Section **3.2**, we matched the profile of the real camera picture with a corresponding 3D edge by adjusting the virtual camera's position and direction. The initial step sizes of u_T and u_R are 5.0. More explicitly speaking, 5[mm] for translation and 5[degree] for rotation. Note here that all the translation and rotation values are measured in terms of units in [mm] and [degree], respectively, in the iteration. The number of repetitions for decreasing the incremental value u was 10, which means that the initial value of u is halved ten times or divided by 1024 at the end.

After the entire process was carried out with the first series of camera positions, we counted the number of resulting holes: Type 1, 2, and 3 holes separately. Table 2 lists these numbers for the three toys. Since surfaces of the dwarf are the most rugged and complicated, there remained Type 3 holes, but there were no Type 3 holes in the other toys. The number of Type 2 holes in the dinosaur is quite high due to the fact that there are a number of narrow and slender slits on its surfaces. In order to eliminate Type 3 holes in the dwarf, we tried a second series of camera positions by decreasing the distance from the camera to the toy by 25% and 50% of the original distance. After the camera selection algorithm described in Section **2.4** was executed with these 640 new camera positions, we obtained five new camera positions with which the same process was repeated. Finally the number of Type 1, 2, and 3 holes in the dwarf toy were 17, 58, and 0, respectively.

Figure 17 illustrates the effect of blending. We see abrupt transition in the cheek of the dwarf as shown in the Figure

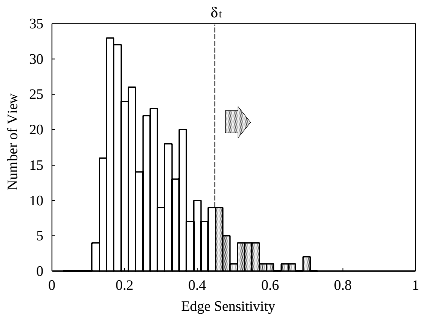

Figure 13: *Histogram of Edge Sensitivity*

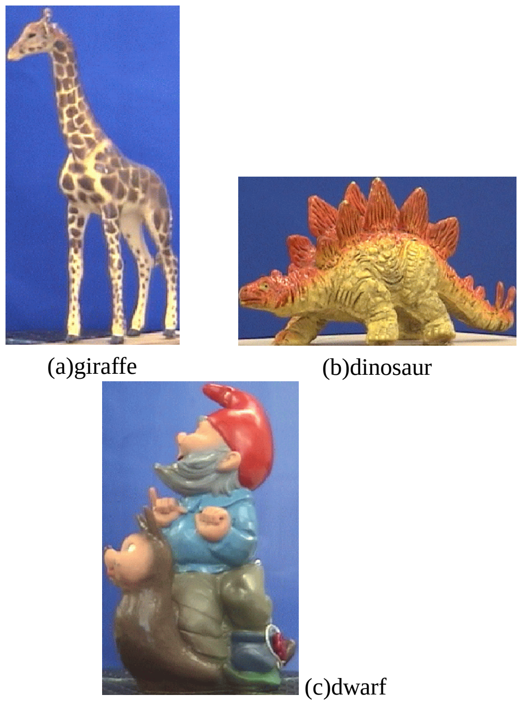

Figure 14: *Photographs of three plastic toys*

17(a) with blending OFF. On the other hand, the color transition is smooth with Figure 17(b). Figure 18 shows the final figures after all the processes. For comparison with the original photographs shown in Figure 14, we took the same camera positions. It is seen that resemblance of the giraffe and dinosaur is very good. However, the original dwarf has specular surfaces but the reconstructed dwarf does not show specularities. This is probably due to the fact that textures are averaged on most of its surfaces.

To evaluate the texture mapping results, we compared a head portion of the dwarf shown in Figure 19. Figure 19(a) is the original image and Figure 19(b) is the reconstructed image. It is found that the displacement is generally within a

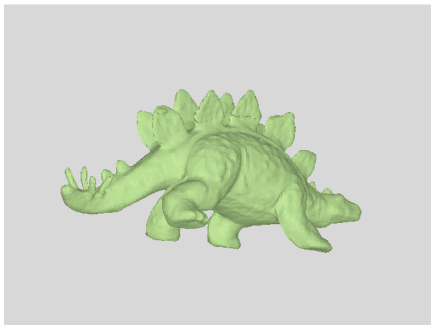

Figure 15: *A Reference Image*

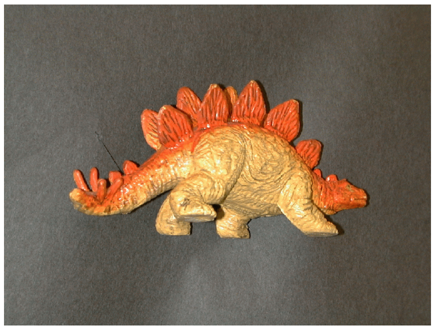

Figure 16: *A Digital Camera Image*

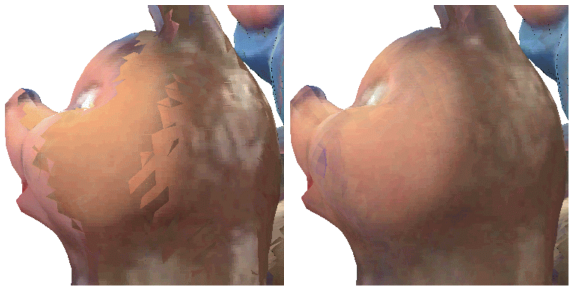

Figure 17: *Blending Effect*

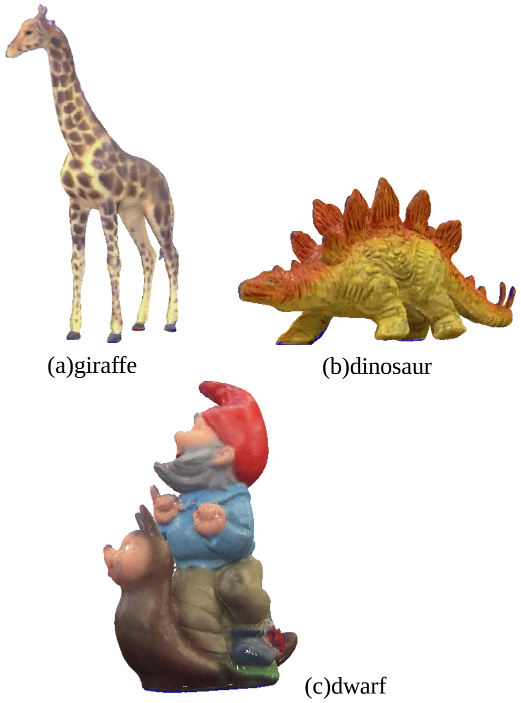

Figure 18: *The final Results*

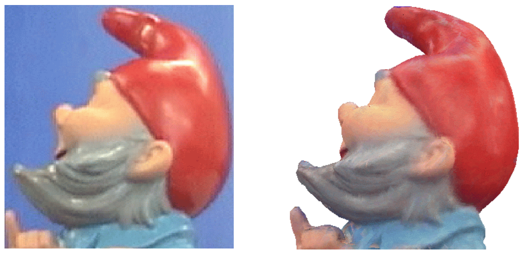

Figure 19: *Figures for Evaluation*

pixel and never beyond two pixels. In terms of real distance scale, the displacement was .25[mm], for the giraffe .1[mm] for the dinosaur, and .28[mm] for the dwarf in case of one pixel displacement.

7. Conclusion and Discussion

We described a method for acquiring textures and for matching them on 3D surface data precisely. Three experimental examples demonstrate the practicability of our method, by which textures can be easily and precisely added. It is a surprising finding that the number of views needed to cover the entire surfaces of our three specimen, giraffe, dinosaur, and dwarf are only seven, eight, and 25, respectively.

The method described here is not applicable to an object whose profile does not change much with varying camera positions, for example, a globe and a cylinder. Edge sensitivity mentioned in Section **3.2** can be used to judge whether the presented method is applicable or not. Based upon our experiments, our method is applicable if edge sensitivity δ is larger than 0.3.

We considered the situation that camera is always directed towards the gravity center of a given object. This restriction is not always to be followed. Particularly for a second series of camera positions to eliminate type 3 holes, the camera can be directed to a specified direction not towards the center.

The required condition is that edge sensitivity of the scene is sufficiently large(i.e. above 0.3) so as to identify the camera position and direction. For a large object which can not be contained in an exposure of a camera, we may take a number of exposures to cover the entire area as long as sufficient edge sensitivity is obtainble.

The presented method uses only the external profiles of an object. If clear internal edges in both shape and the corresponding picture exist, they may be exploited for matching. A human face[20] is an example, where edges in the eyes, noses, and mouth are used.

Our future work is to speed up the matching operations described in section 3. Presently the entire process for an object takes about five to six hours on a SUN ULTRA 1 computer. The most time consuming operation is matching the profile of a picture with a 3D profile. If this matching could be done within a minute or less, we could do the whole process iteratively in real-time. So if Type 3 holes are found, we can eliminate them by just adding new camera positions on the spot.

In the presented work the criteria for the camera position selection is the existence of at least one view for each surface polygon. We plan to investigate the possibility of at least two or more views for each surface, which are situated reasonably apart mutually. Another view in the direct neighborhood is not useful.

We observed some specular reflections on some of the dwarf surfaces. A topic to investigate is to exclude surfaces perpendicular to the camera direction when light and camera are oriented in the same direction like a flash light. More generally, we should exclude surfaces where the light and camera direction are in a mirror reflection relation. Furthermoe it will be a challenging problem to consider local specular properties[6, 7].

8. Acknowledgement

Our thanks are due to Professor K. Katada, Fujita Health University, for providing CT data of the three toys used in the experiment.

References

1. Y. Sato, H. Kitagawa, and H. Fujita, Shape Measurement of Curved Objects Using Multiple Slit-ray Projections, *IEEE Tr. on PAMI*, Vol.PAMI-4,No.6,pp.641-646,1982
2. R. Klette et.el., Computer Vision, Three-dimensional Data from Images, Springer, 1998
3. Y. Suenaga and Y. Watanabe, A method for Synchronized Acquisition of Cylindrical Range and Color Data, *IEICE Tr. on Information and Systems*,Vol.E-74,No.10,pp.3407-3415,1991
4. P.E. Debevec, C.J. Taylor, and J. Malik, Modeling and Rendering Architecture from Photographs; A Hybrid Geometry-and Image-based Approach, *Proc. of SIGGRAPH 96*,pp.11-20, Aug. 1996
5. P. Debevec, Y. Yu, and G. Borshukov, Efficient View-dependent Image-based Rendering with Projective Texture-mapping, *9th Eurographics workshop on rendering*, pp.105-116, 1998
6. Y. Sato, M.D. Wheeler, and K. Ikeuchi, Object Shape and Reflectance Modeling from Observation, *Proc. of SIGGRAPH 97*, pp.379-387, Aug. 1997
7. K. Ikeuchi and K. Sato, Determining Relectance Properties of an Object Using Range and Brightness Images, *IEEE Tr. on PAMI*, Vol.PAMI-13, No.11, pp.1139-1153,1991
8. Y. Sakaguchi, H. Sato, K. Sato and S. Inokuchi, Acquisition of Entire Data Based on Function of Range Data, *IEICE Tr. on Information and Systems*,Vol.E-74,No.10,pp3417-3422,1991
9. C.K. Cowan and P.D. Kovesi, Automatic Sensor Placement from Vision Task Requirements, *IEEE Tr. on PAMI*, Vol.PAMI-10,No.3,pp.407-416,1988
10. P. White and F.P. Ferrie, From Uncertainty to Visual Exploration, *IEEE Tr. on PAMI*, Vol.PAMI-13,No.10,pp.1038-1042,1991
11. Y. Sato and M. Ohtsuki, Three-Dimensional Shape Reconstruction by Active Rangefinder, *Proc. CVPR'93*,pp.142-147, 1993
12. M. Ashida, S. Dan, and T. Kitahashi, Inferring Appropriate Camera Positions for Invisible Parts, *Proc. ACCV'93*,pp.692-695, 1993
13. C.R. Maurer,Jr., G.B. Aboutanos, B.M. Dawant et.al., Registration of 3D Images using Weighted Geometrical Features, *IEEE Tr. on Medical Imaging*,Vo.15,No.6,pp.836-849, Dec.1996
14. R.M. Koch, M.H. Gross, F.R. Carls, D.F. von Bueren, G. Fankhauser, and Y.I.H. Parish, Simulating Facial Surgery using Finite Element Models, *Proc. of SIGGRAPH 96*,pp.421-429, Aug. 1996
15. A.M. Haider, E. Takahashi, and T. kaneko,a 3D Human Face Reconstruction from CT Image and Color-Photographs *IEICE Tr. on Information and Systems*,Vol.E-81,No.10,pp1095-1102,1998
16. J.D. Foley, A. van Dom, S.K. Feiner, and J.F. Hughes: Computer Graphics, Priciples and Practice, 2nd edition, *Addison-Wesley Publishing Company*, 1992
17. P.J. Besl and N.D. McKay, A Method for Registration of 3-D Shapes, *IEEE Tr. on PAMI*, Vol.PAMI-14,No.2,pp.239-257, 1992
18. W.H. Press et.al. Numerical Recipes in C, The Art of Scientific Computing, *Cambridge University Press*, 1988
19. W. Lorensen and H. Cline, Marching cubes: a High Resolution 3D Surface Construction Algorithm, *ACM Computer Graphics*, 21(4):163-170, 1988
20. A.M. Haider, and T. kaneko, Automatic Reconstruction

of 3D Human Face from CT and Color-Photographs *Proc. of MVA-98 (IAPR Workshop on Machine Vision Applications)*, pp.127-130, Nov. 1998, Chiba, Japan