Thirteenth Eurographics Workshop on Rendering (2002)
P. Debevec and S. Gibson (Editors)

Microfacet Billboarding

Shuntaro Yamazaki Ryusuke Sagawa Hiroshi Kawasaki Katsushi Ikeuchi Masao Sakauchi

Institute of Industrial Science, University of Tokyo

Abstract

Rendering of intricately shaped objects that are soft or cluttered is difficult because we cannot accurately acquire their complete geometry. Since their geometry varies drastically, modeling them using fixed facets can lead to severe artifacts when viewed from singular directions. In this paper, we propose a novel modeling method, "microfacet billboarding," which uses view-dependent "microfacets" with view-dependent textures. The facets discretely approximate the geometry of the object and are aligned perpendicular to the viewing direction. The texture of each facet is selected from the most suitable texture images according to the viewpoint. Microfacet billboarding can render intricate geometry from various viewpoints. We first describe the basic algorithm of microfacet billboarding. Also, we predict artifacts generated due to the use of discrete facets and we analyze the necessary sampling interval of the geometry and texture for regarding the artifacts as negligible. In addition to the modeling method, we have implemented a real-time renderer by a hardware-accelerated technique. To evaluate the efficiency of our method, we compared it with traditional texture mapping to a mesh model, and showed that our method has a great advantage over the former in rendering intricately shaped objects.

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Graphics Systems I.3.3 [Computer Graphics]: Display algorithms I.4.10 [Image Processing and Computer Vision]: Volumetric K.8.1 [Personal Computing]: Graphics

1. Introduction

Realistic representation of real-world scenes and real objects by computer graphics techniques is important in industry and academia, for example, movie production, architecture, digital museums, and cultural anthropology. Although there is much research on this topic, it still remains a challenging problem. Among the number of reasons for this difficulty, we consider complicated surface reflectance attributes and intricate geometry to be the most crucial.

Recently, several researchers [1, 2] have developed methods for precise modeling of real-world objects; however, they are mainly concerned with the surface reflectance model and rigid geometry rather than intricate and soft geometry such as a cluster of leaves, or fur. A natural scene usually contains many objects which have intricate shapes or soft geometry; therefore, without consideration of such intricate geometry, it is almost impossible to achieve realistic rendering of real-world scenes.

In this paper, we deal with the issue of how to render objects realistically when their geometry cannot be completely acquired. Since our target is real-world objects and scenes, the problem can be divided into two parts, modeling and rendering. For modeling, we propose an octree-based data structure and discrete microfacets to approximate the actual geometry which cannot be correctly acquired using conventional range sensors or a stereo-algorithm. For rendering, we propose a view-dependent geometry and a view-dependent texture technique to synthesize a realistic image in real time using graphics hardware accelerators.

The central idea of our proposed method is based on the following observation: while objects with intricate and soft geometry cannot be acquired by the conventionally used methods of modeling, such as stereo-modeling, visual hull modeling [3], or scanning by laser range sensors, they can usually be imaged with extremely high resolution using charge-coupled devices (CCD) sensors, and such images can be efficiently used to increase the reality of images synthesized by image-based rendering (IBR).

The concept of our method is shown in Figure 1. In this

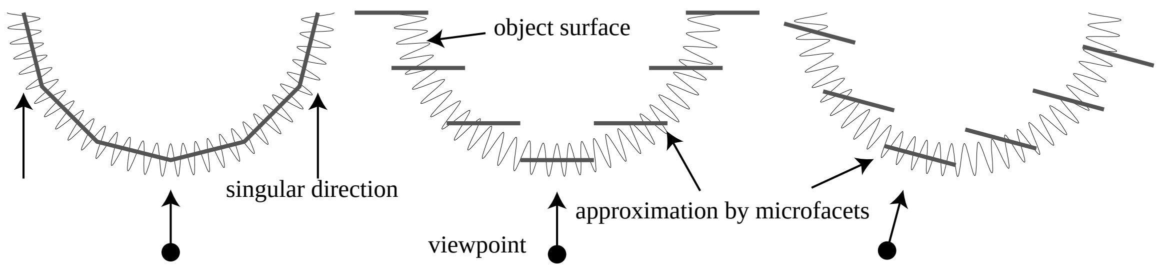


Figure 1: *Left: Existing methods of using fixed facets for modeling the geometry of an object. It is difficult to use these methods to represent intricate geometry, particularly on occluding boundaries. Center: The facets used in our proposed method are kept perpendicular to the direction of view. Right: When viewed from another point, the facets are again perpendicular.*

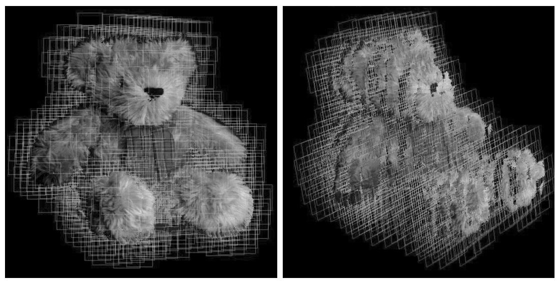


Figure 2: *Left: Frontal view of microfacets. Right: Side view of the rendering result.*

method, the surface of the object is approximated by a set of "microfacets" onto which the texture images of the object are mapped. As shown in this figure, all facets are aligned perpendicular to the viewing direction even when the viewpoint or the viewing direction[†] changes. The texture image mapped to each facet is selected according to the viewpoint. The left figure in Figure 2 is an example of the rendering result. The right figure shows a side view of the microfacets viewed from another (unusual) viewpoint, for clarity of the explanation. In the figure, the artifacts are observed on the boundary of microfacets. We have estimated the upper bound of such visual artifacts with regard to the size of facets, and show that they can be successfully eliminated.

In the following section, we first review the past research on geometric representation and IBR techniques. Then we describe the basic algorithm of our method in Section 3, followed by actual implementation in Section 5. Section 6 concerns our experiments performed using our method which successfully renders intricately shaped geometry. Finally, we discuss the efficiency of our method and summarize this work.

[†] In the rest of this paper, the term *viewpoint* implies both the point and the direction of the viewer.

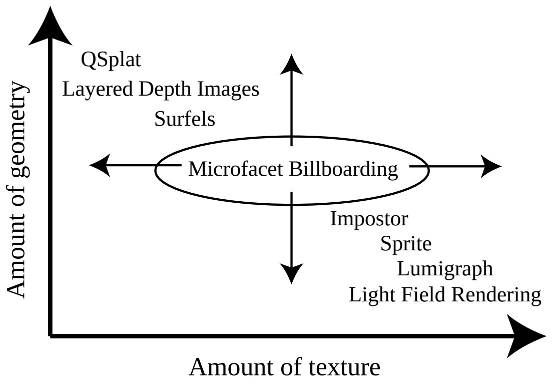


Figure 3: *Relationship of our microfacet billboarding with other related work.*

2. Related Work

Figure 3 shows the relationship of our microfacet billboarding with other related work in terms of geometry and texture. The vertical axis indicates the amount of geometry used for rendering and the horizontal axis indicates the amount of texture image used for rendering.

Considering our method from geometrical aspects, if we use a large number of geometry primitives and assign a simple color to a primitive, our technique is similar to conventional surface-based rendering and point-based rendering [4, 5, 6, 7]; Qsplat [4] yields a multiresolution representation based on a bounding sphere hierarchy and uses squares or ellipses as rendering primitives. Shade *et al.* [7] proposed a data structure called layered depth images (LDIs), which are rendered using splatting. Chang *et al.* [8] proposed a LDI tree, which is a hierarchical representation of LDIs in an octree manner. Surfels [5] yields a hierarchical structure based on the LDI tree and utilizes elliptical tangent disks with texture in the rendering pipeline. These point-based rendering methods used discrete primitives such as circles, ellipses and rectangles for rendering. On the other hand, if we use a small number of large primitives and map a detailed texture to each primitive, our method becomes similar to image caching or sprites [9, 10, 11]. In these methods, the results of slower rendering are cached and new views are generated by warping. Layered impostors [12, 13] divide an object in multiple layers according to the distance from the viewpoint and render the object from new viewpoints by warping the layers. Meyer and Neyret [14] rendered complex repetitive scenes using a volume renderer with view-dependent texture to switch textures according to the viewing direction.

Similar to the last two methods, our method also applies view-dependent textures for rendering, thereby making it possible to synthesize the details of small geometry realistically without a complete model. The application of multiple images to compensate for geometric complexity and to realize realistic image synthesis is also adopted in existing image-based methods; therefore, our technique can naturally be considered an IBR technique.

Although IBR has great potential for synthesizing a realistic image [15, 16], two major drawbacks, which prevent IBR from being applied for widespread use, are known: huge input data size and lack of interactive use. One solution to these problems is to use geometric data. By using the geometry, we can remove the redundancy of the input data set and consequently decrease the data size without degrading the quality of the output image [17]. Furthermore, the traditional polygon-based method can be applied for IBR using geometry.

The surface light field [18, 19] is one of the implementations of IBR with geometry. It employs an efficient analysis of the surface properties of the object and accomplishes clever data compression. The method allows the user to synthesize an image with a large degree of freedom of view and lighting conditions. However, in this method, precise geometry is needed. The unstructured lumigraph [20] is another implementation for IBR with geometry. This method permits the use of an approximate geometry for rendering, and also includes an efficient camera selection algorithm. The view-dependent texture mapping technique [21], which can yield realistic large-scale scenes, can also be considered as a technique of IBR with geometry .

Although these proposed methods solve many of the problems arising in a simple IBR implementation, using consistent geometry to define the surface of an object sometimes causes serious artifacts in the resultant images (i.e., double images and ghosting), particularly when the target object consists of intricately shaped geometry which cannot be captured precisely. Our method, unlike the traditional IBR with geometry, adopts multiple geometries for single objects and has good potential to overcome the above problems.

3. Microfacet Billboarding

3.1. Outline

The whole process of microfacet billboarding can be separated into two steps: the modeling process and the rendering process. They are outlined in the following pseudocodes.

```
ModelingProcess
output: MultiresVolumetricModel $G_m$
output: TextureImages $I_t$
local: PolygonalModel $G_p$
local: ColorImages $I_c$
local: VoxelModel $G_v$
  $G_p \leftarrow$ AcquireGeometry()
  $I_c \leftarrow$ AcquireColorImage()
  $G_v \leftarrow$ ApproximateGeometry($G_p$)
  $G_m \leftarrow$ ConstructMultiresolution($G_v$)
  $I_t \leftarrow$ GenerateTexture($G_p$, $I_c$)
```

In the modeling process, first the surface model and color images of the object are acquired. Then the surface is resampled into a set of voxels which are replaced by microfacets in the rendering process. The voxel representation is extended to a multiresolutional structure by using octree representation if necessary. In parallel, we generate the range images taken from the surface model at all camera positions where color images are taken. The range images are used for texture clipping in the final step of the rendering process.

```
RenderingProcess
input: MultiresVolumetricModel $G_m$
input: TextureImages $I_t$
input: Viewpoint $V$
local: Microfacets $M$
local: Microfacet $m$
local: TextureImage $t$
  $M \leftarrow$ GenerateMicrofacets($G_m$, $V$)
  for all $m$ in $M$ do
    $t \leftarrow$ SelectTexture($V$, $I_t$)
    $m \leftarrow$ MapTextureImage($m$, $t$)
    RenderMicrofacet($m$)
  end for
```

In the rendering process, the object is rendered by a set of microfacets with color texture. First, view-dependent microfacets are generated, then the color image of the object is mapped onto each of them. The texture image mapped onto the facet is dynamically clipped according to the distance to the object in the rendering pipeline of graphics hardware.

The modeling process may take some time, therefore it is assumed to not be accomplished in real time. On the other hand, once the model is prepared, our proposed method can render the object in real time by taking advantage of acceleration by graphics hardware. The details of the processes are described in the following sections.

3.2. Modeling by Discrete Geometry

3.2.1. Acquisition of Geometry and Texture

First, the geometry and texture of the object are acquired. For geometry, we build the surface model of the object in the form of a polygonal mesh. We used the point cloud measured by the scanner in our experiments presented in this paper, since our first objective in starting this research is realistic rendering of intricately shaped objects by making the best use of incomplete geometry acquired using a laser range scanner. Our method, however, is also applicable to models obtained by other methods of modeling, such as stereo- or visual hull modeling. For texture, we use color images of the object taken from several viewpoints. We assume that the color images are already aligned with the geometric model.

The accuracy of the geometry can have a significant effect on the result of rendering. In general, the more accurately the geometry is measured, the more reliable the image rendered at the viewpoint apart from the point where the texture image is taken. On the other hand, a model whose geometry is incompletely measured can be rendered realistically by introducing an image-based approach, because the accuracy of the texture image is independent of the intricacy of the surface of the object. Our method focuses on rendering such incomplete models, and provides a means of control of continuous transition between model-based rendering and image-based rendering, as shown in Figure 3.

3.2.2. Approximation of Geometry

For rendering models of incomplete geometry, we approximate the geometry by resampling the obtained surface model. The model is sampled in three-dimensional space and discretized into voxels. The value of the voxel is binary, that is, the value is set to be 1 if the surface intersects the voxel, otherwise it is set to be 0. The optimal interval of sampling depends on the accuracy of the geometry. In the rendering process, each of the voxels is rendered by a small surface patch, called a microfacet, if the value of the voxel is 1. A detailed explanation of the generation of microfacets is described in Section 3.3.1. In this volumetric representation, the geometry of the object is roughly approximated by a set of microfacets, while the detailed appearance of the object is represented by the texture mapped onto them.

3.2.3. Construction of Multiresolutional Representation

Although extending the representation of the approximated geometry to the multiresolutional structure is an optional step, it is useful in two points. First, it enables the viewer to control the extent of the approximation. If the measured geometry is incomplete, for example, a surface which has many holes or much noise, well approximated geometry also suffers from incorrectness, which can decrease the quality of the rendered result. If the degree of approximation can be controlled, the viewer can determine the appropriate level of approximation manually by trial and error. Second, it enables automatic control of the levels of detail, as described in Section 5.4.

Since our representation of the surface model is based on the volumetric partition of the space, it can be easily extended to multiresolutional representation. We can construct an octree representation by repeatedly doubling the sampling interval for voxels.

When the shape of an object is represented by a single voxel and approximated by a single microfacet, the method is equivalent to such a fully image-based one as QuickTime VR [22]. Conversely, when the octree is subdivided into the size of the points in the original point cloud, the rendering becomes closer to splatting [6, 4].

3.2.4. Generation of Texture Images

Because the geometry of the model is resampled and replaced by a set of flat microfacets, the depth of the object is quantized into a value corresponding to the depth of each facet. If the size of polygons (which corresponds to the interval of resampling) is sufficiently large, the flatness of the facets becomes apparent, which causes visual artifacts when the viewpoint moves.

One approach to removing these artifacts is to clip the texture according to the depth of the object. In order to determine the depth of the object for each texel in the texture, range images with the same resolution as the corresponding color images are generated for each color image. Since we assume that we have acquired a surface model of the object and color images from various viewpoints aligned with the model, we can generate the range image of the model from the viewpoint where each color image was taken. When we render a microfacet of depth D from the viewpoint of a color image, a pixel of the color image which has depth d is rendered by

$$\begin{cases} \text{opaque color} & \text{if} \quad |d - D| < \frac{W}{2} \\ \text{transparent color} & \text{otherwise,} \end{cases} \tag{1}$$

where W is the width of the space of which the facet takes charge. Figure 4 shows an example of a result of clipping. Nearby pixels are bright while ones farther away are dark in the range images. In the clipped image, black pixels are clipped and not mapped to microfacets (actually, they are mapped as transparent).

It is worth noting that although we use range images acquired using a laser range scanner to build the surface model, the generation of range images described in this section is still necessary because the positions where color images are taken are generally not the same as those where range images used to build the surface model are taken.

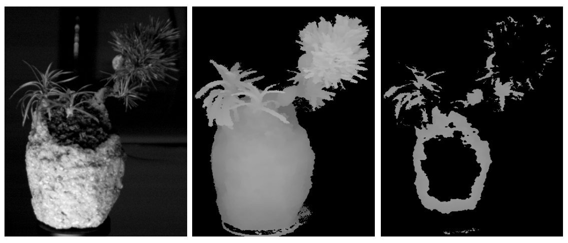


Figure 4: *(left) One of the color images. (center) Range image from the same viewpoint. (right) Clipped result of the range image.*

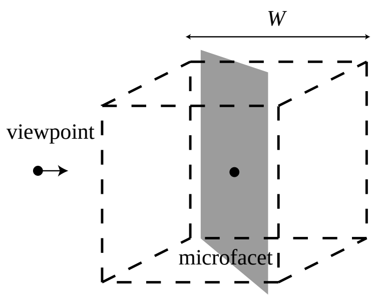


Figure 5: *A microfacet is a slice which intersects the center of the voxel and is vertical to the viewing direction.*

3.3. Rendering by Microfacet Billboarding

3.3.1. Generation of Microfacets

A microfacet is defined as a slice which intersects the center of the voxel and is vertical to the viewing direction. Each microfacet represents the approximated surface inside the voxel occupied by an object.

The choice of the shape used to represent a microfacet can have a significant effect on the quality of the final image. With regard to the analogy of splatting techniques [4], a square or an ellipse is possible. In our method, a square is chosen because a quadrilateral polygon is simple enough to render and to map a texture efficiently using standard graphics hardware. The width of the square is defined by the size of the voxel (see Figure 5). Namely, if the size of a voxel is W, the microfacet of $\sqrt{3}W$ will cover the voxel.

The quadrilateral facet is usually rendered as a group of small triangles since a small region of the texture image is mapped onto them. It, however, can be rendered as a point with a single color if the size of the facet is sufficiently small compared with the width of texture sampling. We implemented the renderer using triangles and that using points, and we compare their performance in Section 6.2.

3.3.2. Selection of Texture

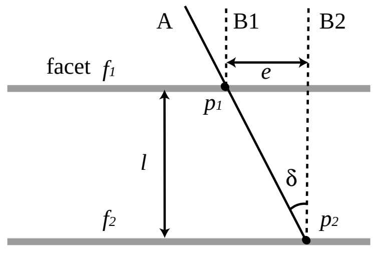
The texture mapped onto a microfacet is selected or generated from input images according to the angle formed by the viewing direction for rendering and the camera direction of input images. The simplest way to generate texture is to select the image whose camera direction is *nearest* to the current viewing direction. The distance between directions is defined by the angles they form. Let a unit vector parallel to the current viewing direction be $\boldsymbol{v}$ and a unit vector parallel to the i-th camera direction be $\boldsymbol{c}_i$, then the distance d_i between the directions is defined as $d_i = cos^{-1}(\boldsymbol{v} \cdot \boldsymbol{c}_i)$. With every change of viewing direction, distance d_i for each i is calculated; then, the camera position i which has minimum d_i is selected and the i-th camera image is mapped to the facet.

Another way to generate texture is interpolation. For every change of viewpoint, the distance between viewing direction and each camera direction is calculated. The parameters used for interpolating images are determined by the distances for all images; then, some or all of the images are blended according to the values. For smooth rendering, blending parameters should change continuously between 0 and 1. If the camera positions are distributed spherically around the object, blending the three nearest camera images can accomplish smooth rendering. If the cameras are distributed uniformly in three-dimensional space, selecting the four nearest camera images can enable smooth blending. Generally, since it is not necessarily feasible to position the cameras uniformly in space, optimal selection of the camera images can involve complicated problems, as discussed in [20]. The method of interpolation takes account of only blending, and not warping, therefore, coarse input images can result in such artifacts as ghost images. Adopting a more sophisticated interpolation technique such as view morphing [23] may produce a smoother and more accurate result.

3.3.3. Mapping of Texture Images

The selected camera images are mapped onto the facets by perspective projection. When an image is mapped, we post-process it by clipping the area which should be on the facet, using the range image described in Section 3.2.4. For every texel in the texture image, the depth d to the object is fetched from the corresponding range image, then d is compared to the distance D to the point to which the texel is mapped on the microfacet. If equation (1) is satisfied, the texel is mapped with an opaque color; otherwise the point on the facet is rendered as transparent.

Assume that the viewpoint is on the line of view of a selected camera, then the distance to the microfacet is equal for every point on the facet. In this case, texture can be clipped by comparing range data with fixed D for each facet. The distance D for a single facet, however, can vary since microfacets rotate according to the viewpoints. Consequently D must be changed for every point, even on a facet. In addition, d is defined in the camera direction, and not in the viewing direction, therefore, D must be calculated as the distance from the camera to the facet, instead of that from the viewpoint to the facet.



A: viewing direction of texture image
B: viewing direction of rendering

Figure 6: *A single ray of texture image A is projected onto two microfacets p_1, p_2, which are rendered at different points on the synthesized image.*

4. Analyses of Visual Artifacts

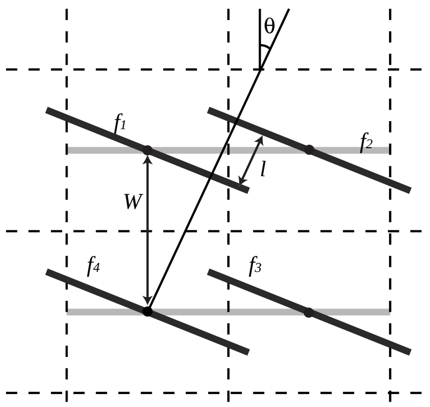
In our method, we use a set of facets to represent real objects with small geometries. Therefore, visual artifacts can occur depending on size, shape and orientation of microfacets. Here, we employ several analyses for these visual artifacts to derive an optimum sampling interval for the microfacet. Furthermore, we study the orientation of the microfacets, which also affects distortion in the final image.

4.1. Sampling Interval for Microfacet

If we synthesize a new image using a texture which is acquired from a different viewpoint, artifacts can occur in the transition between microfacets because a pixel of the texture image can be projected onto different pixels of the synthesized image. Figure 6 shows this situation. Consider two microfacets f_1 and f_2 at an interval of l. If the difference in the viewing direction of rendering and that of the texture image is δ, a ray of the texture image A is projected onto these two microfacets at points p_1 and p_2, respectively. These two points p_1 and p_2 are projected onto different points of the synthesized image, in the directions of B_1 and B_2, respectively. We can see the same texture on both microfacets f_1 and f_2. The error e on the synthesized image is

$$e = l \tan \delta. \tag{2}$$

This equation depicts how the artifact depends on the sampling interval of facets, l, and that of camera images δ. Namely, if the geometry becomes finer, or if the number of camera images becomes larger, the artifact becomes smaller. This issue of the sampling of geometry and texture is discussed in detail by Chai *et al.* [17] To generate the rendering result in which the artifacts are negligible, we should use a sufficient number of facets and camera images according to equation (2). When the model is rendered onto the screen, the projection of e can be regarded as the error of screen space. Therefore the error on the screen can be kept smaller than 1 pixel by generating facets at sufficiently small intervals.



▬ billboarding facets
▬ fixed facets

Figure 7: *In the case of billboarding facets, the orientation of facets is perpendicular to the viewing direction. In contrast, the orientation of facets is perpendicular to the direction of the axis in the case of fixed facets.*

4.2. Orientation of Microfacets

Another important factor for visual artifacts is the orientation of the microfacet. In this section, we evaluate the distortion for both billboarding and fixed facets. To calculate the actual distortion in a final rendered image, we assume that the orientation of facets is perpendicular to the direction of the axis in the case of fixed facets. Figure 7 illustrates the array of billboarding facets when the angle between the axis and the viewing direction is θ. For fixed facets, the distance between overlapping facets along the viewing direction is always $W \tan \theta$. On the other hand, for the billboarding facets, since the size of the facets is $\sqrt{3}W$, facet f_1 overlaps

$$\begin{cases} f_2 & \text{if} \quad 0 \le \theta < \frac{1}{8}\pi \\ f_3 & \text{if} \quad \frac{1}{8}\pi \le \theta < \frac{3}{8}\pi \\ f_4 & \text{if} \quad \frac{3}{8}\pi \le \theta < \frac{1}{2}\pi, \end{cases} \tag{3}$$

where $0 \le \theta < \frac{1}{2}\pi$ is satisfied in two-dimensional space. Therefore, the distance between adjacent billboarding facets is kept less than W. Since the distance l for billboarding facets in equation (2) is less than that for fixed facets, the error e caused by the use of discrete facets is kept smaller using billboarding microfacets rather than fixed facets.

The rendering method using the fixed facets resembles the layered impostors [12, 13] or slice-based volume rendering [24]. Since the interval between facets is constant, the double images caused by incomplete texture clipping can be prevented. The billboarding method, however, has an advantage over the fixed facet method, particularly in rendering intricately shaped surface models, because the reproduced scene

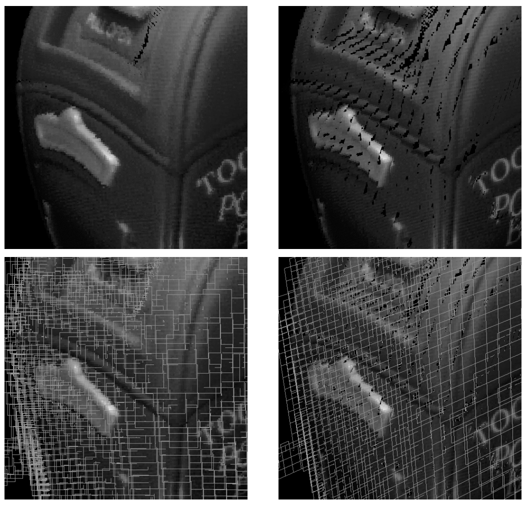


Figure 8: *Comparison of the rendered images using billboarding facets and fixed facets. The same number of facets were drawn in each figure.*

viewed apart from the camera positions may have many gaps and holes between facets due to insufficient sampling of texture and incomplete geometry. In the billboarding method, changing the direction of primitives can successfully fill the gaps, as shown in Figure 8.

5. Implementation Details

The modeling process described in Section 3.2 is straightforward. Some techniques, however, are necessary for increasing the quality of rendering results. In this section we describe the preprocessing of texture images and range images.

The rendering by microfacet billboarding can be effectively implemented by using the OpenGL graphics library. Microfacets are rendered by polygons with textures, taking advantage of rendering acceleration enabled by graphics hardware. One difficulty in the process is a texel-wise operation which is necessary to perform the texture clipping described in Section 3.3.3. In this section, we show that it can be accomplished by using the pixel shading function of modern graphics hardware, and explain an implementation of texture clipping on PC graphics hardware. In addition, we also present a technique of controlling visual artifacts generated due to the use of discrete facets, using the level-of-detail control of microfacets.

5.1. Range Image Processing

Since we assume that the geometry of the object is not necessarily acquired completely, the range images generated by projecting the obtained surface model can have holes which cause undesired deletion of the texture by texture clipping. Therefore we apply a morphological filter to the obtained range image in order to remove the holes.

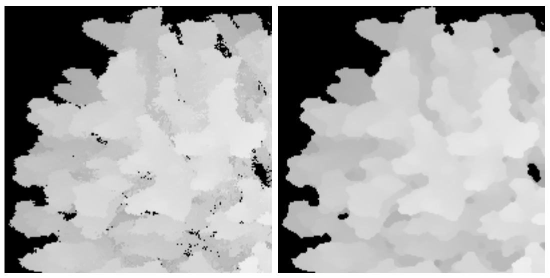


Figure 9: *The holes in a range image can be removed by applying a morphological filter. (left) Original range image. (right) Filtered range image.*

The filter is based on a fixed size of the window. For each pixel in the image, we assign a certain size of window whose center is the pixel. Then the value of each pixel is iteratively modified according to the values of the pixels within the window. In our adopted filter, the value of the pixel is replaced by the median of the values of the nonhole pixels within the window if the modified pixel is detected as a hole. To avoid excessive modification of the image, we also count the number of holes in a window, and fill the hole if the number is less than a certain threshold.

In our experiments, we use a 3×3 pixel window and the threshold is set to 5 pixels. An example of filtering range images is shown in Figure 9.

5.2. Alpha Estimation

The texture images used to render the object are taken using an ordinary camera; the images include background which should be removed before or when rendering. If the object has rigid geometry such as a smooth surface, it is unnecessary to remove the background because it is removed by texture clipping using depth information. However, since we focus on rendering intricately shaped objects, background can be remove not by clipping, but by alpha estimation.

To estimate the alpha value, many efficient methods have been proposed [25, 26] to date. In the work described in this paper, we can measure the background and the lighting conditions precisely; therefore, simple applications of past methods are expected to work well. Here, we adopt Ruzon-Tomasi's Knock-Out method to estimate the alpha value.

Most research methods require a known boundary between the foreground and the background; some of them are determined by the chromakey method and some of them are determined manually. At this time, we assume that we already have an approximate geometry which is calibrated to

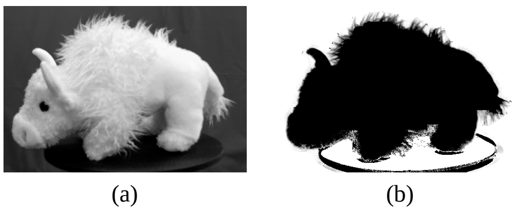


Figure 10: *(a) Originally captured object. (b) Extracted alpha matte.*

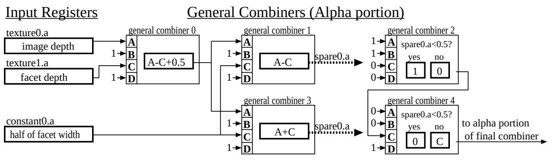


Figure 11: *Register combiners setup for texture clipping by substituting alpha value according to equation (1).*

the image; therefore, boundary detection is robust. We perform the following process to estimate the alpha value.

1. Divide image into foreground, boundary and background areas using approximated geometry
2. Generate color value clusters for the background and foreground in RGB space using the k-means algorithm. Then construct a network between background and foreground color clusters
3. Plot a pixel value of the boundary area into RGB space and search for the nearest link. Each node can be estimated as a background color and a foreground color. The ratio between foreground-color-to-pixel and background-color-to-pixel gives the alpha value.

This is a simpler method than other recently proposed methods [25, 26], but the whole process can be done automatically and the advantage is significant.

Figure 10 shows the result of alpha estimation. Figure 10 (a) is an original image and (b) is the alpha matte extracted from (a). We can see that intricately shaped geometries are successfully rendered with the assumed alpha values.

5.3. Texture Clipping

In order to clip texture during the run time, we utilize the flexible function of texture operation available on modern PC graphics hardware. First, since the range image for texture clipping has the same resolution as corresponding color images, we can store the depth of the range image in the alpha channel of the color image. Thereby, for each color image we generate the RGBz textures which contain color texture in RGB channels and depth d in equation (1) in the alpha channel before rendering.

As previously mentioned, we render a microfacet as a single quadrilateral to which a selected texture is mapped. When the polygon is rasterized and the corresponding texel is fetched, the alpha value of the texel is compared with the depth of the microfacet, and then replaced with 1 if the pixel is within the voxel represented by the microfacet, otherwise it is set to 0. Then, the polygons are rendered using alpha blending. As a result, the area of texture which satisfies the depth test is mapped with an opaque color, and the other area is mapped as transparent and has no effect on the result of rendering.

This alpha substitution can be performed within the rendering pipeline of graphics hardware using a programmable texture shader when microfacets are rendered. In our current implementation, the functions *register combiners* and *texture shader* available on nVidia's GeForce3 GPU or newer are utilized.

The register combiners take texel values fetched by preceding texel fetch units through input registers, modify them, and then write back to output registers. We also set and refer to some constants through constant registers. To accomplish texture clipping, we set up the textures so that a texel from RGBz texture is loaded to the first texture register, the depth D in equation (1) to the second texture register, and the width w to a constant register.

The connection of register combiners for alpha substitution is depicted in Figure 11. Although current register combiners cannot support the comparison between two arbitrary floating point values, which is necessary in our method, we construct the function of comparison by connecting the two multiplexers shown on the two rightmost combiners in Figure 11. As mentioned in Section 3.3.3, since the depth D can vary for each point on the facet, the value must be generated in the rendering pipeline. To load the depth of the microfacet, we set it at the vertices as texture coordinates, then load the interpolated texture coordinate as the texel value by using the `pass_through` function of the texture shader.

5.4. Controlling Visual Artifacts

Since the geometry of models is approximated by a set of quadrilateral polygons, visual artifacts might be observed as discontinuities of texture if the size of facets is too large. The appropriate size of facets can be determined by considering the size of the screen, as mentioned in Section 4. Figure 12 shows the level-of-detail control of microfacets. When models are viewed from a sufficiently distant position, large facets are acceptable because the discontinuity of texture due to the large facets cannot be observed, as shown on the left in Figure 12. The closer the viewing position is to the object, the smaller the size of microfacets that must be used, as shown in the center and on the right in Figure 12. Obviously, smaller facets lead to larger numbers of generated microfacets, which slows down the rendering to some extent. The problem, however, is not fatal because simple techniques of

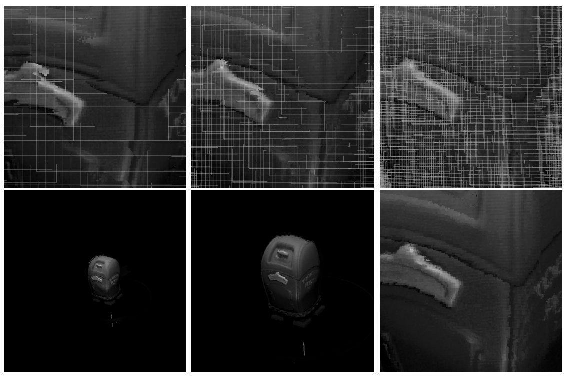


Figure 12: *Level-of-detail control of microfacet billboarding.*

visibility culling, such as viewport clipping, can successfully stop the slowing of rendering at certain speeds.

6. Experimental Results

We implemented a microfacet billboarding renderer on a standard PC equipped with nVidia's GeForce4 Ti4600 GPU. Microfacets are rendered as squares of texture-mapped polygons using the OpenGL graphics library and its extensions taking advantage of the rendering acceleration of graphics hardware.

6.1. Rendering Objects by Microfacet Billboarding

A stuffed cow with soft and intricate geometries was modeled and then rendered, as shown in Figure 13. The model is furry over a large part of its surface and it is difficult to measure and to represent its geometry completely.

In our experiment, the geometry and texture of the object are acquired at the same time, in the form of a sequence of range images and texture images, using the VIVID 900 laser range scanner. The number of images captured for the model is 36. Although our method has no limitation on the position of cameras, we assume the cameras are placed along a circle because a turning lathe was used to create the model in this experiment. This circular camera position confines the area in which the viewpoint can be positioned to a two-dimensional plane which contains the circle of the camera positions. However, the information on the geometry of the object expands this two-dimensional plane to three-dimensional space with the distortion estimated in Section 4.

For the geometry, first the range images are converted into a set of polygonal patches. Then they are aligned in three-dimensional space using a simultaneous registration method [27], followed by adaptive resampling into an octree of the signed distance field by distance transformation. Figure 13 (a) shows the surface model extracted from the volumetric data using the robust surface reconstruction algorithm proposed by Wheeler *et al.*[28] The surface model is so intricate, particularly for the fur, that the texture-mapped surface of the model cannot reproduce correct visibility, as is shown in Figure 13 (b). Therefore, we approximate its geometry using a set of microfacets. The microfacets are generated at the center of the leaf octants which intersect the surface of the object.

For the texture, the images are first processed by the alpha estimation described in Section 5.2. Since our implementation of texture clipping uses the alpha channel of texture in an unconventional manner, extracted alpha values cannot be dynamically applied during rendering. Accordingly, we assume that the object is rendered on a background of a single color, then the texture images are superimposed on the background using the extracted alpha matte in the preprocess. When micofacets are rendered, the modified images are used as texture images mapped onto the microfacets.

Figure 13 (c) shows a result of microfacet billboarding rendering. The microfacets for this model are placed on a $16 \times 16 \times 16$ sampled grid, and the number of facets actually generated is 756. Figure 13 (d) shows the facets without texture. The color of the facet identifies the camera currently selected for the facet. When the viewpoint moves, the sudden change of the camera selection causes a severe visual artifact. To avoid this artifact, the texture mapped onto a microfacet is generated by interpolating two nearby camera images, as shown in Figure 13 (e). Since the cameras are positioned circularly and equidistantly around the object, we can render the view from any viewpoint by blending two selected camera images. Two input images whose camera directions are the nearest to the viewing direction are selected and interpolated to generate texture. The blending parameters for the images are determined considering $cos^{-1}(\boldsymbol{v} \cdot \boldsymbol{c}_i)$. Finally, Figure 13 (f) shows the final result of microfacet billboarding.

6.2. Performance Analysis

Since the result of a traverse of the octree is cached for every change of resolution, the speed of rendering depends only on the number of microfacets to be generated in the resolution. An object was rendered in various resolutions using (a) quadrilateral polygons with view-dependent textures, (b) points with view-dependent colors, and (c) quadrilateral polygons with accelerated view-independent textures, as shown in Table 1. Contrary to our expectation, a comparison of (a) and (b) in Table 1 indicates that the shape of microfacets has little effect on rendering performance. The reason is that the bottleneck of the rendering process is not the rasterization of microfacets in graphics hardware, but the camera selection for view-dependent texture. The calculation for camera selection can be accelerated by limiting the number of candidates for the selection. With every change

Table 1: *Example of rendering performance. The top two rows indicates the size of data. The bottom rows show the number of rendered frames per second (FPS) when microfacets are represented by (a) quadrilateral polygons with view-dependent textures, (b) points with view-dependent colors, and (c) quadrilateral polygons with accelerated view-dependent textures.*

Size of volume	32^3	64^3	128^3	256^3
# of facets	2460	11207	50560	208289
FPS				
(a)	>30	23.8	5.2	1.2
(b)	>30	24.4	5.5	1.4
(c)	>30	>30	26.3	8.1

of viewpoint, we generate a cone which includes all possible directions from the viewpoint to microfacets, then some texture images are selected as the candidates for camera selection using the cone. This simple technique can increase the performance of rendering, as shown in row (c) in Table 1. Since our current implementation is not optimized for performance, adopting more sophisticated techniques of camera selection or other processes may accelerate the performance. Further improvement of performance is not discussed in this paper.

It is worth noting that the depth in the range images is quantized as integer in $[0, 255]$ due to the limitation of graphics hardware. Therefore, microfacets finer than those generated from a volume of more than 256^3 have the same quality of texture.

The required memory size for microfacet billboarding depends on both the number of facets and the number of cameras. For geometry, one microfacet requires three 32-bit floating points to represent its position in space. The normal and rotation of a microfacet are determined dynamically when it is rendered. Therefore, the required memory size for geometry is roughly estimated as $24 \times \{\text{number of facets}\}$ for each resolution of the octree. For texture, one texture image requires $w \times h$ pixels which have 4 channels (RGBz) in 8-bit precision. Therefore, the required memory size for texture is roughly estimated as $4 \times w \times h \times \{\text{number of images}\}$.

6.3. Comparison with the Surface Model

It is common for models with clear and static surfaces to be rendered by methods using polygons with texture, therefore, we should compare our method with such methods. One of the most suitable techniques of rendering objects with intricate geometry is the combination of visual hull modeling and view-dependent/independent texture mapping. The visual hull algorithm can generate a model of the approximate geometry of the object independently of the intricacy of the surface of the object, because this algorithm utilizes only the silhouette line of the object. However, the detail of an object which has an intricate silhouette cannot be modeled correctly.

Figure 14 shows a result of the comparison. Several images of a plant in a pot are captured, and one of them is shown on the left. First, the geometry of the object is generated using visual hull algorithm, and represented by a polygonal mesh (center), and microfacets (right). The silhouette line of the object is extracted from the range images capured using a laser range scanner. Then, both models are rendered using view-dependent texture mapping. Although the quality of both models is poor, microfacet billboarding reproduces the real view much better than does the surface-based algorithm.

7. Future Work

In future work, we plan to develop a new method for each process of microfacet billboarding. In geometry approximation, adaptive approximation according to the object shape may be considered. Also, we must evaluate view-dependent segmentation of geometry, such as screen-aligned segmentation. In texture generation, since the algorithm of camera selection becomes elaborate in the case of an intricate configuration of cameras, we must develop a flexible algorithm for camera selection. Moreover, the algorithm for planning image acquisition will be important. In the present experiments, we considered only a fixed light environment. It is necessary to acquire images in various environments to develop a more flexible model.

8. Summary

In this paper, we proposed a novel modeling method, “microfacet billboarding.” Microfacet billboarding approximates intricate geometry using view-dependent facets, and renders the object using view-dependent texture mapping. Since the facets remain perpendicular to the viewing direction, microfacet billboarding can render intricately shaped objects, such as fur and trees, from various viewpoints. We described the basic algorithm and our current implementation. We also estimated artifacts generated due to the use of discrete facets, and analyzed the required sampling interval of geometry and texture. Finally, we performed experiments and evaluated our method through comparison with conventional texture mapping to a mesh model. We concluded that microfacet billboarding is highly advantageous for rendering intricately shaped geometry.

9. Acknowledgments

We are grateful to the reviewers for the 13th Eurographics Workshop on Rendering for their valuable comments, suggestions and correction of errors. This work was performed under the auspices of the Japan Science and Technology Corporation.

References

1. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D scanning of large statues. In *Proc. SIGGRAPH '00*, pages 131–144, 2000. 1
2. K. Ikeuchi, Y. Sato, K. Nishino, R. Sagawa, T. Nishikawa, T. Oishi, I. Sato, J. Takamatsu, and D. Miyazaki. Modeling cultural heritage through observation. In *Proc. First Pacific-Rim Conference on Multimedia*, Dec. 2000. 1
3. A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994. 1
4. S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. SIGGRAPH '00*, pages 343–352, 2000. 2, 4, 5
5. H. Pfister, M. Zwicker, J. Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proc. SIGGRAPH '00*, pages 335–342, 2000. 2
6. J. Grossman and W. Dally. Point sample rendering. In *Proc. the 9th Eurographics Workshop on Rendering*, pages 181–192, 1998. 2, 4
7. J. Shade, S. Gortler, L. He, and R. Szeliski. Layered depth images. In *Proc. SIGGRAPH '98*, pages 231–242, 1998. 2
8. C. Chang, G. Bishop, and A. Lastra. LDI tree: A hierarchical representation for image-based rendering. In *Proc. SIGGRAPH '99*, pages 291–298, 1999. 2
9. J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Proc. SIGGRAPH '96*, pages 75–82, 1996. 3
10. X. Decoret, G. Schaufler, F. Sillion, and J. Dorsey. Multi-layered impostors for accelerated rendering. In *Proc. Eurographics '99*, pages 61–73, 1999. 3
11. M. Wimmer, P. Wonka, and F. Sillion. Point-based impostors for real-time visualization. In *Proc. the 12th Eurographics Workshop on Rendering*, pages 163–176, 2001. 3
12. G. Schaufler. Per-object image warping with layered impostors. In *Proc. the 9th Eurographics Workshop on Rendering*, pages 145–156, 1998. 3, 6
13. G. Schaufler. Image-based object representation by layered impostors. In *Proc. Symposium on Virtual Reality Software and Technology*, pages 99–104, 1998. 3, 6
14. A. Meyer and F. Neyret. Interactive volumetric textures. In *Proc. the 9th Eurographics Workshop on Rendering*, pages 157–168, 1998. 3
15. M. Levoy and P. Hanrahan. Light field rendering. In *Proc. SIGGRAPH '96*, pages 31–42, 1996. 3
16. S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. SIGGRAPH '96*, pages 43–54, 1996. 3
17. J. Chai, X. Tong, S. Chan, and H. Shum. Plenoptic sampling. In *Proc. SIGGRAPH '00*, pages 307–318, 2000. 3, 6
18. K. Nishino, Y. Sato, and K. Ikeuchi. Eigen-texture method: Appearance compression based on 3D model. In *Proc. CVPR '99*, volume 1, pages 618–624, 1999. 3
19. D. Wood, D. Azuma, W. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Steutzle. Surface light fields for 3d photography. In *Proc. SIGGRAPH '00*, pages 287–296, 2000. 3
20. C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. SIGGRAPH '01*, pages 425–432, 2001. 3, 5
21. P. Debevec, Y. Yu, and G. Boshokov. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proc. SIGGRAPH '96*, pages 11–20, 1996. 3
22. E. Chen. QuickTime VR — an image-based approach to virtual environment navigation. In *Proc. SIGGRAPH '95*, pages 29–38, 1995. 4
23. S. Seitz and C. Dyer. View morphing. In *Proc. SIGGRAPH '96*, pages 21–30, 1996. 5
24. M. Brady, K. Jung, H. Nguyen, and T. Nguyen. Two-phase perspective ray casting for interactive volume navigation. In *Proc. Visualization '97*, 1997. 6
25. Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *Proc. CVPR '01*, volume 2, pages 264–271, 2001. 7, 8
26. M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *Proc. CVPR '00*, volume 1, pages 24–31, 2000. 7, 8
27. P. Neugebauer. Geometrical cloning of 3d objects via simultaneous registration of multiple range images. In *Proc. Shape Modeling and Application '97*, pages 130–139, 1997. 9
28. M. D. Wheeler, Y. Sato, and K. Ikeuchi. Consensus surfaces for modeling 3d objects from multiple range images. In *Proc. ICCV '98*, pages 917–924, 1998. 9

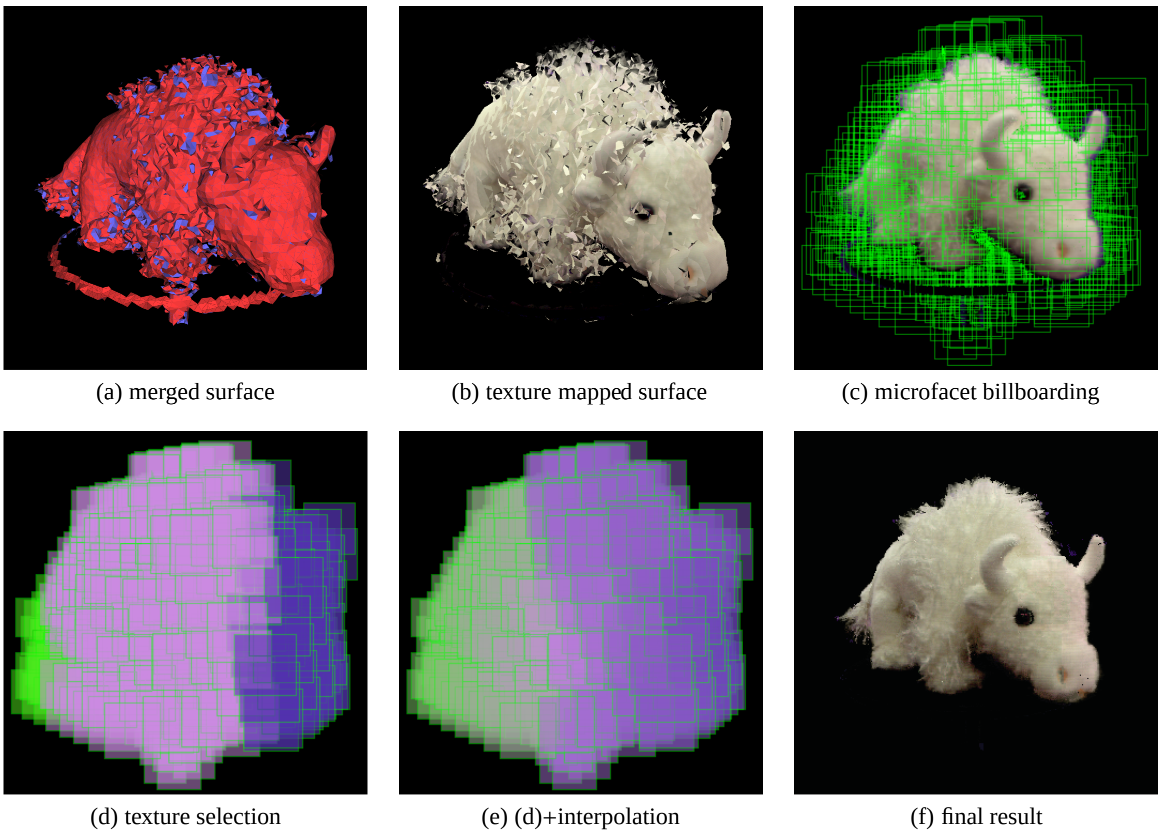


Figure 13: *These are the images obtained by microfacet billboarding rendering of a stuffed cow. (a) The acquired surface model of the object. (b) The image rendered by the surface model with the textures. (c) The result of rendering by microfacet billboarding using 756 facets. (d) Each microfacet is colored according to the index of the selected camera. (e) The textures mapped onto facets are generated by interpolation of several textures. (f) Final result of rendering by microfacet billboarding.*

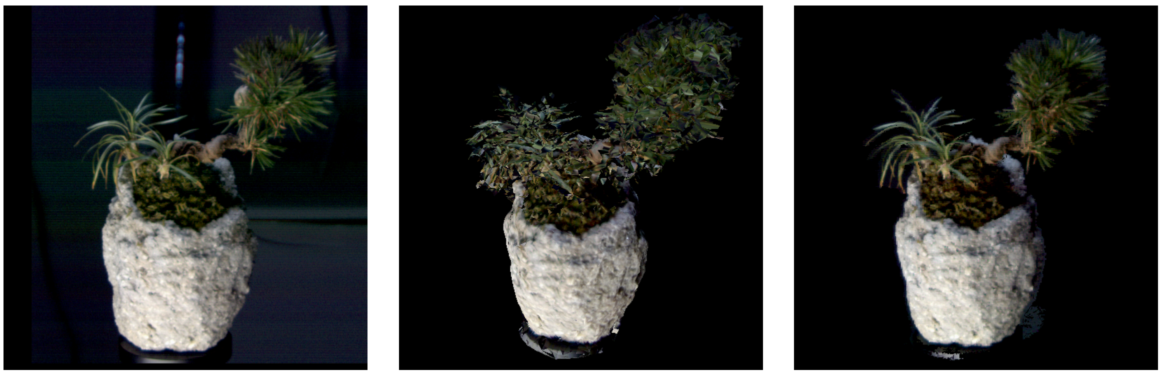


Figure 14: *The texture mapped to a microfacet is interpolated when viewed at a location between camera positions. (left) An image which is not used for rendering. (center) The image synthesized by texture-mapped surface rendering at the position of the image on left. (right) The image synthesized by microfacet billboarding, which successfully reproduced intricately shaped needles of the plant.*