Radiosity for Point-Sampled Geometry

Yoshinori Dobashi Tsuyoshi Yamamoto
Hokkaido University
Kita-ku Kita 14, Nishi 9,
Sapporo, 060-0814, Japan
{doba, yamamoto}@nis-ei.eng.hokudai.ac.jp

Tomoyuki Nishita
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku,
Tokyo, 113-0033, Japan
nis@is.s.u-tokyo.ac.jp

Abstract

In this paper, we propose a radiosity method for the point-sampled geometry to compute diffuse interreflection of light. Most traditional radiosity methods subdivide the surfaces of objects into small elements such as quadrilaterals. However, the point-sampled geometry includes no explicit information about surfaces, presenting a difficulty in applying the traditional approach to the point-sampled geometry. The proposed method addresses this problem by computing the interreflection without reconstructing any surfaces. The method realizes lighting simulations without losing the advantages of the point-sampled geometry.

1. Introduction

Recently, point-sampled geometry [14][15] has increased its importance in the computer graphics community. Traditionally, the shapes of objects have been represented by using surfaces as basic primitives, such as polygons. The point-sampled geometry uses points as the basic primitives. The shapes of objects are represented by a set of points. Information regarding connectivities between points is no longer stored. This simplifies data structures and reduces the memory requirement. Therefore, many methods have been developed [11][13][19].

This paper proposes a radiosity method for the point-sampled geometry to compute the interreflection of light. Traditionally, there are two approaches to compute the interreflection [6]. One of these is a probabilistic approach based on the Monte-Carlo simulation. A few methods using this approach have been proposed for the point-sampled geometry [1][16]. The other approach is based on the finite element method. This paper is not intended to discuss which approach is superior, but discusses the possibility of applying the finite element approach to the point-sampled geometry. To date, there have been no methods based on the finite element approach for the point-sampled geometry.

In the finite element approach, the energy transfer between small surface elements is calculated. The traditional radiosity methods calculate this by subdividing the surfaces into small elements such as quadrilaterals (or patches). However, traditional methods cannot be applied directly to the point-sampled geometry since it possesses no explicit information about surfaces. One possible solution is to reconstruct the surfaces from the point set and then to use traditional methods. However, this approach degrades the advantages of the point-sampled geometry. To address this problem, a new radiosity method that can handle the points directly is proposed here. We assume a point model is represented by a set of *surfels* [14]. A surfel is a point primitive consisting of its position and properties of the original surface around the point such as a reflectance and a normal vector. First, the proposed method computes an area represented by each surfel. To compute the area, we use a *tangent disk* assigned to each surfel [14]. Since the tangent disks generally overlap, we propose a method taking into account the overlap for computing the area of each surfel. We call the area an *effective area*. This allows the calculation of form factors between surfels. Then, the interreflection is computed, using the method proposed by Stamminger et al [17]. The proposed method uses points instead of patches. However, the intensity distribution may not be represented accurately by using only the original set of surfels. To address this, a method is proposed for adding new points adaptively.

Using the proposed method, the interreflection of light is simulated without degrading the advantages of the point-sampled geometry. In this paper, the reflectance of the surfaces of objects is assumed to be purely diffuse.

2. Related Work

Points were used as display primitives for the first time by Levoy and Whitted in 1985 [10]. Then, in 2000, the papers by Pfister et al. [14] and Rusinkiewica et al. [15] led to the recent development of many application methods for point-sampled geometry [19][8]. However, these methods consider only direct illumination, since their purpose is the display of point-sampled geometry.

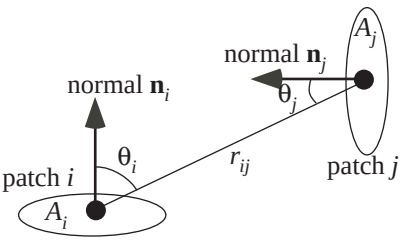


Figure 1. Calculation of form factors.

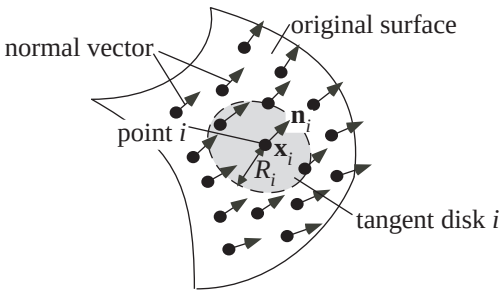


Figure 2. Surfel and tangent disk.

To simulate the global illumination effects, Shaufler et al. developed a ray-tracing method for point-sampled geometry [16]. This method computes an intersection between a ray and the point-sampled geometry by placing a disk at each point. However, Adamson et al. pointed out a problem with this method, and they proposed an improved method [1]. Global illumination can be simulated by combining these methods with the idea of the Monte-Carlo simulation.

However, there have been no methods based on the finite element approach to point-sampled geometry. To employ the finite element approach, the object surfaces must be divided into patches. This makes it difficult to apply traditional approaches to point-sampled geometry. This paper gives a solution to this problem.

3. Concept of Radiosity Calculation

One of the simplest approaches is to subdivide surfaces of objects into patches and then compute energy transfers between the patches. This results in solving the following linear equations (for details, see [3]).

$$B_i = E_i + \sum_{j=1, j\neq i}^{n} \rho_j V_{ij} F_{ij} B_j \quad (i = 1, \cdots n), \qquad (1)$$

where n is the number of the patches, B_i is the radiosity of patch i, E_i is the initial emittance of patch i, ρ_j is a diffuse reflectance of patch j, and V_{ij} is a visibility between patches i and j. F_{ij} is a form factor between patches i and j. When the size of the patch is sufficiently small, F_{ij} is expressed approximately as:

$$F_{ij} = \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} A_j, \qquad (2)$$

where, as shown in Fig. 1, r_{ij} is a distance between patches i and j, θ_i and θ_j are the angles between a line connecting these patches and normal vectors of the patches i and j, respectively.

The proposed method uses the surfels instead of the patches. It is assumed the domain represented by each surfel is sufficiently small and the radiosity is constant in the domain. This is generally true since the point-sampled geometry usually consists of a large number of points. This assumption allows the use of Eq. (2) to compute the form factors between the surfels.

4. Basic Idea

As shown in Fig. 2, the input data to the proposed method is a set of points, where it is assumed that normal vector $\mathbf{n}_i$ and diffuse reflectance ρ_i have been obtained for each point $\mathbf{x}_i$. Pfister et al. called this type of point primitive a surfel [14]. This notation is used in the following. Objects represented by the surfels are typically displayed by assigning a tangent disk to each surfel. As shown in Fig. 2, the tangent disk i is perpendicular to normal $\mathbf{n}_i$ with its center at $\mathbf{x}_i$. Its radius R_i is determined so that there are no gaps between the surfels. We can use the maximum distance between points in a small neighborhood around surfel i [14].

First, to compute the form factors between surfels, information is required regarding the area (A_j) used in Eq. (2). The simplest way to compute the area is to use an area of the tangent disk. However, this leads to an overestimation of the total area of a point-sampled object since the tangent disks generally overlap. Here, to address this, an accurate method is proposed for the computation of an effective area of each surfel.

Next, for efficient radiosity computation, a hierarchical structure is constructed by recursively grouping neighboring surfels. Surfels in the lowest level correspond to the original set of points. The interreflection of light is computed by using the method developed by Stamminger et al [17]. However, the intensity distribution may not be represented by using only the original point set. This is especially true for the boundaries of shadows. Thus, during the radiosity computation, new surfels are adaptively added if the difference of radiosities in a small neighborhood exceeds a specified threshold. After the radiosity computation, a final image is rendered by using a hardware-accelerated point rendering method [9].

The procedure of the proposed method is summarized as follows.

1. Computation of effective areas of the surfels.
2. Construction of the hierarchical structure.

3. Calculation of the interrefelction of light.
4. Creation of final images.

Steps 1 and 2 are done in a pre-process.

The following sections describe steps 1 through 3. For step 4, see [9].

5. Computation of Effective Area

This section describes the method for computing the effective area. The following subsections describe the idea with reference to Fig. 3. First, the basic idea is explained using a simple case where the normal vectors of neighboring surfels are the same (Section 5.1). Next, the idea is extended to general cases (Section 5.2). Then, a hardware acceleration of the area computation is described (Section 5.3).

5.1. Simple Case

As shown in Fig. 3(a), let us assume a differential surface element at $\mathbf{y}$ on tangent disk i. The area of the differential element is $\Delta A(\mathbf{y})$. The number of disks that overlap at $\mathbf{y}$ is m ($m = 3$ in Fig. 3(a)). Clearly, the total area S_{total} represented by these disks is obtained by integrating $\Delta A(\mathbf{y})$ over the domain of these disks. That is,

$$S_{total} = \int_{\Omega_m} \Delta A(\mathbf{y}) d\mathbf{y}, \tag{3}$$

where Ω_m indicates points inside the disks. On the other hand, we assume the total area is the sum of the effective areas of m surfels, that is, $S_{total} = S_1 + S_2 + \cdots + S_m$, where S_i is the effective area of surfel i. To determine the effective area, we introduce an influence function $\xi_i(\mathbf{y})(0 < \xi_i(\mathbf{y}) < 1.0)$ for each surfel i. Using this function, we define the effective area S_i as:

$$S_i = \int_{D_i} \xi_i(\mathbf{y}) \Delta A(\mathbf{y}) d\mathbf{y}, \tag{4}$$

where D_i indicates points inside disk i. It is reasonable that the influence of the surfel i on the surface element should increase if the distance from the surfel position to the surface element becomes short. Furthermore, $\xi_1(\mathbf{y}) + \xi_2(\mathbf{y}) + \cdots + \xi_m(\mathbf{y})$ must be equal to 1 in order to satisfy $S_{total} = S_1 + S_2 + \cdots + S_m$. Therefore, we use the following function for the influence function.

$$\xi_i(\mathbf{y}) = w(r_i(\mathbf{y})) / \sum_{j=1}^{m} w(r_j(\mathbf{y})), \tag{5}$$

$$w(r_i(\mathbf{y})) = \begin{cases} \exp(-\kappa r_i(\mathbf{y})/R_i) & (r_i(\mathbf{y}) \le R_i) \\ 0 & (r_i(\mathbf{y}) > R_i) \end{cases}, \tag{6}$$

where κ is a constant specified by the user and $r_i(\mathbf{y})$ is the distance between $\mathbf{y}$ and $\mathbf{x}_i$ (see Fig. 3(a)). The parameter κ controls the influence of each surfel. We use 3.0 for κ in the examples shown in Section 8. Clearly, the influence function expressed by the above equations satisfies the condition, $\xi_1(\mathbf{y}) + \xi_2(\mathbf{y}) + \cdots + \xi_m(\mathbf{y}) = 1$, and therefore, $S_{total} = S_1 + S_2 + \cdots + S_m$. The effective area is given by putting Eqs. (5) and (6) into Eq. (4).

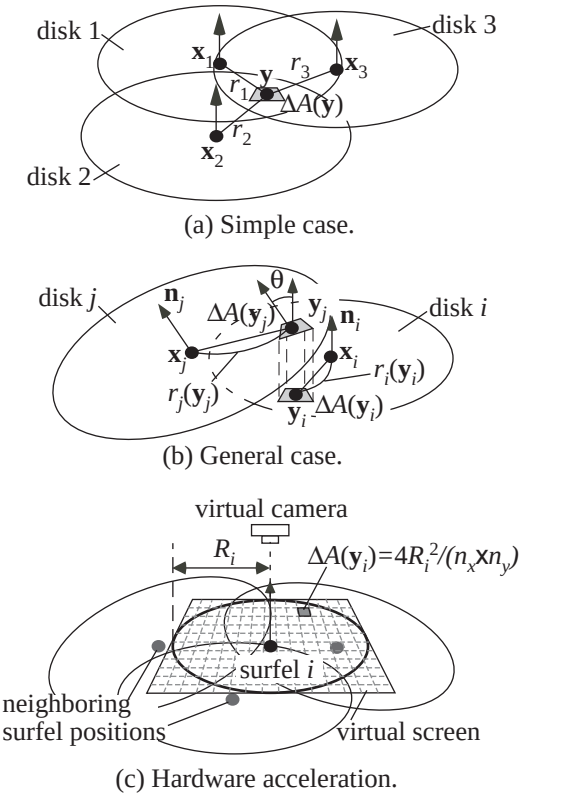


Figure 3. Computation of effective area.

When the normal vectors of the surfels are the same, their effective areas are obtained by the above method. However, in general, the normal vectors are different and the tangent disks do not exist on the same plane (see Fig. 3(b)). In the next subsection, we extend the above idea to the general case.

5.2. General Case

As mentioned above, the tangent disks do not exist on the same plane generally. We assume there are m surfles around surfel i. Fig. 3(b) shows two surfels i and j among them. A problem in this general case is the area ΔA of the differential surface element used in Eq. (4). In the simple case (Fig. 3(a)), the area of the differential element is the same for all surfels overlapping at $\mathbf{y}$, since the disks are on the same plane. However, in the general case, this is not true. We resolve this problem as follows. First, we consider a differential element at $\mathbf{y}_i$ on disk i as shown in Fig. 3(b). Then we consider a corresponding differential element at $\mathbf{y}_j$ on disk j. The position $\mathbf{y}_j$ is at the intersection between disk j and a ray from $\mathbf{y}_i$ in the direction of $\mathbf{n}_i$ (see Fig. 3(b)). We use

the average of these differential areas instead of ΔA in Eq. (4). The details are described below.

Let us denote the areas of the differential elements at $\mathbf{y}_i$ and $\mathbf{y}_j$ are $\Delta A(\mathbf{y}_i)$ and $\Delta A(\mathbf{y}_j)$, respectively. $\Delta A(\mathbf{y}_j)$ is expressed as $\Delta A(\mathbf{y}_j) = \Delta A(\mathbf{y}_i)/\cos\theta = \Delta A(\mathbf{y}_i)/(\mathbf{n_i} \cdot \mathbf{n}_j)$, where θ is an angle between $\mathbf{n}_i$ and $\mathbf{n}_j$. Then the weighted average $\Delta\bar{A}(\mathbf{y}_i)$ of $\Delta A(\mathbf{y}_1), \Delta A(\mathbf{y}_2), \cdots$, and $\Delta A(\mathbf{y}_m)$ is calculated. The weighting function expressed by Eq. (6) is used again. That is, $\Delta\bar{A}(\mathbf{y}_i)$ is calculated by the following equation.

$$\Delta\bar{A}(\mathbf{y}_i) = \frac{\sum_{j=1}^{m} w(r_j(\mathbf{y}_j))\Delta A(\mathbf{y}_j)}{\sum_{j=1}^{m} w(r_j(\mathbf{y}_j))}, \quad (7)$$

$$\Delta A(\mathbf{y}_j) = \Delta A(\mathbf{y}_i)/(\mathbf{n}_i \cdot \mathbf{n}_j), \quad (8)$$

where $r_j(\mathbf{y}_j)$ is the distance between the center of surfel j and the surface element on disk j (see Fig. 3(b)). We use the weighted average $\Delta\bar{A}(\mathbf{y}_i)$ instead of $\Delta A(\mathbf{y})$ in Eq. (4). The effective area S_i of surfel i is then expressed by the following equation.

$$S_i = \int_{D_i} \frac{w(r_i(\mathbf{y}_i))\Delta\bar{A}(\mathbf{y}_i)}{\sum_{j=1}^{m} w(r_j(\mathbf{y}_j))} d\mathbf{y}_i. \quad (9)$$

5.3. Hardware Acceleration

We make use of graphics hardware for fast computation of the effective areas expressed by Eq. (9). In the following, details of our algorithm are described with regard to the computation of the effective area of surfel i.

To use graphics hardware, a virtual camera is placed in the direction of the normal vector of surfel i as shown in Fig. 3(c). The reference point of the camera is the surfel position $\mathbf{x}_i$ and the parallel projection is specified. A virtual screen is placed on the same plane of the tangent disk of surfel i (see Fig. 3(c)). The size of this screen is $2R_i \times 2R_i$, where R_i is the radius of the tangent disk. The screen is divided into $n_x \times n_y$ pixels. Each pixel corresponds to the differential surface element on surfel i. The differential area $\Delta A(\mathbf{y}_i)$ in Eq. (8) is $\Delta A(\mathbf{y}_i) = 4R_i^2/(n_x \times n_y)$. We assume that each pixel of the frame buffer consists of R, G, B and α components.

First, surfels around surfel i are extracted. This can be done efficiently using the kd-tree [7]. Among them, only the surfels whose normal vectors are similar to that of surfel i are extracted. This excludes surfels that are considered to be on different surfaces. Those surfels should not be used for the computation of the effective area of surfel i. For this process, we use the second condition of Eq. (10) for building the hierarchical structure (see Section. 6).

Next, the weighted average $\Delta\bar{A}(\mathbf{y}_i)$ is calculated by using Eq. (7). The computation of Eq. (7) is equivalent to rendering an image by using the surface splatting method [19] after assigning $\Delta A(\mathbf{y}_j)$ (Eq. (8)) to intensity of surfel j. We use a hardware-accelerated surface splatting method [9] to speed up the computation. In [9], the normalization corresponding to the denominator of Eq. (7) is processed only at every surfel instead of every pixel for real-time display. In our method, the normalization is processed at every pixel for accurate computation. Only surfel i and its neighboring surfels are taken into account for this image generation.

The image is created as follows. First, a quadrilateral, whose side length is the diameter of the tangent disk, is placed at the position of each surfel. Then, a texture representing the weighting function expressed by Eq. (6) is mapped onto each quadrilateral. Each of RGBα components stores the same value. The quadrilaterals are rendered after multiplying RGB components of the texture by $\Delta A(\mathbf{y}_j)$ and their colors are accumulated using additive color blending functions. In this image generation process, the intensity of each component of the weight texture and the intensity representing $\Delta A(\mathbf{y}_j)$ must be scaled appropriately since most graphics hardwares quantize them with 8 bit precision. The scaling factor must be carefully chosen to reduce the quantization errors. We determined the scaling factor experimentally so that any overflows do not occur. This problem can be resolved by using video cards, such as nVidia GeForce FX, that support floating point texture/frame buffer formats. After the rendering, RGB components of each pixel in the frame buffer store the numerator of Eq. (7) (each of RGB components stores the same value) and α component stores the denominator. The pixel values are then read back from the frame buffer to the main memory.

Finally, the following computations are repeated for each pixel. The average area $\Delta\bar{A}(\mathbf{y}_i)$ expressed by Eq. (7) is computed by dividing the R component by the α component. Then, the distance between surfel position $\mathbf{x}_i$ and a point corresponding to each pixel is computed to obtain a weight $w(r_i(\mathbf{y}_i))$ in Eq. (9). The integrand of Eq. (9) is obtained by multiplying $\Delta\bar{A}(\mathbf{y}_i)$ by $w(r_i(\mathbf{y}_i))$ and then dividing the result by the α component. The sum of the results of the above computations for all pixels is the effective area S_i of surfel i.

6. Building Hierarchical Structures

The hierarchical structure is constructed by grouping surfels with almost the same normal vectors in a small neighborhood. There have been several such methods for building the hierarchy [2][5][12]. We use the simplest of these approaches, where the surfels are grouped by searching the neighbors. We explain the basic idea briefly using Fig. 4. Note that Fig. 4 shows a two-dimensional case for simplicity.

First, surfel i is randomly selected. Next, surfel j ($j \neq i$), satisfying the following conditions, is classified into the

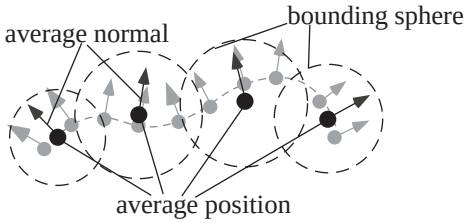


Figure 4. Building hierarchy.

same group (see Fig. 4).

$$|\mathbf{x}_i - \mathbf{x}_j| < b, \quad \mathbf{n}_i \cdot \mathbf{n}_j > \epsilon, \tag{10}$$

where b is equal to double the radius of the tangent disk i ($b = 2R_i$) and ϵ is specified by the user ($|\epsilon| < 1.0$). ϵ is a threshold for the difference between the normal vectors. We use the kd-tree [7] for an efficient search of surfels satisfying the above conditions. Next, a surfel that has not been grouped is selected and the grouping process is repeated. These processes are repeated until all surfels are grouped.

Next, for each group, an average position and an average normal of the surfels are computed as shown in Fig. 4. The total area of the surfels in the group is also assigned to each group. Then a bounding sphere is generated at the average position. The radius of the sphere is computed so that all the surfels in the group are included inside the sphere. Next, all the groups are reclassified by applying the conditions of Eq. (10) to the average position, the average normal, and the bounding sphere. In this case, b is set to $2.0\times$ (Radius of bounding sphere). The hierarchy is constructed by repeating the above processes recursively.

7. Radiosity Calculation

In the computation of the interreflection of light, we need to determine the form factors and the visibilities between any pairs of surfels. The form factor F_{ij} between surfels i and j is calculated by Eq. (2) using their positions, normal vectors, and the effective areas. The visibility V_{ij} is determined by an intersection test between all tangent disks and a line segment connecting surfels i and j. This intersection test is efficiently examined by making use of the hierarchical structure.

We use the method proposed by Stamminger et al [17] for calculating the interreflection. First, the interreflection is calculated by using the surfels corresponding to the top level (the coarsest level) of the hierarchy. The form factors are calculated by using the average positions, the average normals and the total areas. Next, the error is evaluated. The groups with large errors are replaced by their children. Next, the interreflection is calculated again. These processes are iterated until the error becomes less than a specified threshold, ζ.

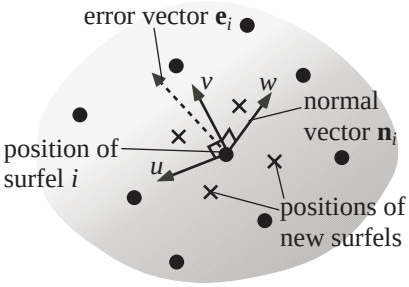


Figure 5. Adding new surfels.

In the above procedure, the error evaluation method needs to be defined. Moreover, the surfels at the leaf nodes of the hierarchy (i.e., surfels corresponding to the original point set) may not be sufficient to represent the intensity distribution. To address this problem, new surfels are adaptively added. Details are described in the following subsections.

7.1. Error Evaluation

The error is evaluated by using differences of radiosties in a small neighborhood around surfel i. First, surfel j satisfying Eq. (10) around surfel i are extracted. A group is replaced by its children if the following condition is satisfied.

$$\max_j |d_{ij}| > \zeta, \quad d_{ij} = B_j - B_i, \tag{11}$$

where B_i and B_j are the radiosities of surfels i and j, respectively.

7.2. Adaptive Addition of Surfels

Surfel i, corresponding to the leaf node of the hierarchy, is deleted when Eq. (11) is satisfied. Then four new surfels are added around surfel i. These four surfels are inserted as children of surfel i in the hierarchy. This idea is similar to the adaptive subdivision scheme that is often used in the traditional patch-based radiosity [4]. The addition of the surfels is recursively processed for the surfels at the leaf nodes. The positions of the new surfels are determined as follows.

It is expected that the accuracy is improved by placing surfels along the boundary of the shadows. In addition, it is clear that adding surfels at the same position is inefficient. Therefore, the following two conditions are taken into account when determining the position of the new surfels.

(1) Surfels should be placed along the direction that is perpendicular to the gradient of the intensity.

(2) Surfels should be placed as uniformly as possible.

In the following, we explain our method using Fig. 5. The shading in Fig. 5 indicates the intensity distribution around surfel i.

First, to take into account condition (1), surfel $j(j = 1, \cdots m)$ satisfying Eq. (10) around surfel i are extracted. m is the number of surfels around surfel i. Then, we define an error vector $\mathbf{e}_i$ as the weighted sum of the normal vectors from surfel i toward surfel j. That is,

$$\mathbf{e}_i = \sum_{j=1}^{m} d_{ij} \frac{(\mathbf{x}_j - \mathbf{x}_i)}{|\mathbf{x}_j - \mathbf{x}_i|}, \tag{12}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are positions of surfels i and j, respectively, and d_{ij} is the difference of the radiosities (see Eq. (11)). As shown in Fig. 5, $\mathbf{e}_i$ nearly indicates the steepest direction of the gradient of intensity at surfel i. If the magnitude of $\mathbf{e}_i$ is zero, we can use an arbitrary vector perpendicular to normal $\mathbf{n}_i$, instead of $\mathbf{e}_i$. Using this vector, we define a local coordinate system uvw with an origin at surfel i. The normal vector $\mathbf{n}_i$ corresponds to w axis. u axis is determined so that it is perpendicular to both w axis and the error vector $\mathbf{e}_i$. Then v axis is set to the direction perpendicular to both u and w axes. Using this coordinate system, we add four surfels at positions $(-cR_i, cR_i, 0.0)$, $(cR_i, cR_i, 0.0)$, $(cR_i, -cR_i, 0.0)$, $(-cR_i, -cR_i, 0.0)$ (see Fig. 5). $c(0.0 < c < 1.0)$ is a constant specified by the user and R_i is the radius of the tangent disk. c controls the distance from the surfel i to the new surfels. We use 0.3 for c in the examples shown in Section 8. In this way, new surfles are generated at positions reflecting the gradient of the intensity at surfel i. That is, condition (1) described above is taken into account. Radii of the tangent disks of the new surfels are set to half of the tangent disk i. Their effective areas are quarter of that of surfel i. Normal vectors and reflectances are determined by interpolation, using the neighboring surfels.

Next, to satisfy condition (2), we modify the positions of the new surfels. The point repulsion method [13][18] is used for this modification. This method modifies the positions by considering forces between the surfels. The magnitudes of the forces depend on the distances between the surfels. When the surfels are close, repulsive forces are generated. When the surfels are sufficiently far apart, no forces are generated. The surfels are moved toward the direction of the forces. Then the forces are computed again. These processes are repeated until the iterations reach a specified number. This method prevents the positions of the surfels from being gathered in a small region. Since the initial positions are determined by the previous process, the surfels are moved to the positions that satisfy both of the two conditions.

Strictly, the hierarchy should be reconstructed since the newly added surfels may distort the hierarchy. However, it is very time-consuming and, in general, the distortion is quite small. In our experiment, it has not caused any serious problems. Therefore, we do not reconstruct the hierarchy. We modify only the size of the bounding sphere of each group in the hierarchy.

7.3. Combining with Polygonal Models

In this subsection, we mention a way to incorporate polygonal models into our method. As we mentioned previously, the traditional radiosity methods subdivide surfaces of objects into small patches. Our method can handle the polygonal models by placing surfels at centers of the patches. In this case, we assign the areas of the patches to the effective areas and we use circumscribed disks as the tangent disks.

8. Examples

In this section, we first discuss the validity of the proposed method by using a simple example. Next, the proposed method is applied to complex examples. All examples shown in this section are calculated using a desktop PC (Pentium4 2.8GHz) with a nVidia GeForce4.

8.1. Experimental Results

We have applied the proposed method to a scene shown in Fig. 6. There are three spheres in a room. The spheres are represented by surfels and the walls (including the ceiling and the floor) are polygons. The length of each side of the walls is 3.5 m and the radii of the spheres are 0.8 m, 0.4 m, and 0.2 m. Each of the walls is subdivided into 400 patches and each of spheres is sampled by 642 points. We created a hierarchical structure with four levels by setting ϵ in Eq. (10) to 0.8.

First, we computed the surface area of the biggest sphere in order to verify the validity of the area computation method proposed in Section 5. The sum of the effective areas of all the surfels is $8.23m^2$. The true value is $8.04m^2$ and therefore the relative error is 2.63 %. This implies that our method can calculate the effective areas with sufficient accuracy.

Next, four area light sources are placed on the ceiling and the interreflection of light is simulated. Fig. 6(a) shows the resulting image. Fig. 6(b) shows the positions of surfels. As shown in Figs. 6(a) and (b), the surfels are adaptively added according to the intensity gradients. To verify the accuracy of the proposed method, we calculated the interreflection by using the same scene sampled more densely (1,600 points for each wall and 2562 points for each sphere). The resulting image is shown in Fig. 6(c). We calculated the differences in intensity between Figs. 6(a) and (c). Fig. 6(d) shows the difference image where 100 % means the difference is 255. As shown in Fig. 6(d), the errors are less than 10 %, implying that the accuracy of the proposed method is adequate.

Regarding the calculation time, the calculation of the effective areas took 5 seconds, the construction of the hierarchy took 1 second, and the interreflection took 32 seconds.

The results shown in this subsection demonstrate the effectiveness of the proposed method.

8.2. Practical Examples

The proposed method is applied to complex examples shown in Fig. 7.

First, Fig. 7(a) shows two bunnies placed in a room. Textures of a marble and a leopard patterns are mapped onto these bunnies. Each of the bunnies consists of 34,834 surfels. It took 68 seconds to compute the effective areas. As in Fig. 6, the walls of the room are polygons and each of them is subdivided into 400 patches. There are four area light sources as shown in Fig. 7(a). The radiosity calculation took 47 seconds. The both bunnies have become greenish due to the reflections from the walls. In addition, the marble bunny has become slightly yellowish due to the reflections from the leopard bunny.

Next, the proposed method is applied to a lighting simulation of a gallery as shown in Fig. 7(b). The statues in the gallery are represented by surfels and walls are polygons. The total number of surfels is 351,686. To verify the effects of the interreflection in detail, the images of some of the statues are shown in Fig. 7(c). It took 6 minutes to compute the effective areas for these statues. There are three area light sources and five spotlights (some of them are not displayed in this image). The geometries of the light sources are not taken into account in the radiosity computation. The computation time of the radiosity solution is 36 minutes. The soft lighting effects are achieved due to the interreflection and a very realistic image is created.

9. Conclusions

In this paper, a method has been discussed for computing the interreflection of light for point-sampled geometry. Both the computation method for the effective areas of surfels and its acceleration using graphics hardware have been proposed. This enables the computation of the form factors between the surfels. An efficient computation of the interreflection has been realized by applying the hierarchical radiosity method to point-sampled geometry. To capture the rapid changes in intensity, a method has been proposed for adding surfels adaptively.

An important task for the future is to take into account specular reflections for the creation of more realistic images. The proposed method may be combined with many traditional radiosity methods that can handle the specular reflections.

References

[1] A. Adamson and M. Alexa. Ray tracing point set surfaces. *Proc. Shape Modeling International 2003*, pages 272–279, 2003.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Leving, and C. T. Silvia. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[3] M. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Morgan Kaufman Publishing (ISBN 0121782700), 1993.

[4] M. F. Cohen, D. P. Greenberg, D. S. Immel, and P. J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Application*, 61(2):26–35, 1986.

[5] C. Dachsbacher, C. Vogelgsang, and M. Stamminger. Sequential point trees. *ACM Trans. on Graphics (Proc. SIGGRAPH 2003)*, 23(3):657–662, 2003.

[6] C. Damez, K. Dmitriev, and K. Myszkowski. State of art in global illumination for interactive applications and high-quality animations. *Computer Graphics Forum*, 21(4):55–77, 2003.

[7] H. W. Jansen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. *Proc. SIGGRAPH' 98*, pages 311–320, 1998.

[8] A. Kalaiah and A. Varshney. Modeling and rendering points with local geometry. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):30–42, 2003.

[9] H. P. L. Ren and M. Zwicker. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. *Computer Graphics Forum*, 21(3):461–470, 2002.

[10] M. Levoy and T. Whitted. The use of points as a display primitive. *Technical Report TR 85-022, The University of North Carolina at Chapel Hill, Dept. of Computer Science*, 1985.

[11] M. Pauly and M. Gross. Spectral processing of point-sampled geometry. *Proc. SIGGRAPH 2001*, pages 379–386, 2001.

[12] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. *Proc. IEEE Visualization 2002*, pages 163–170, 2002.

[13] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM. Trans. on Graphics (Proc. SIGGRAPH 2003)*, 23(3):641–650, 2003.

[14] H. Pfister, M. Zwicker, J. V. Baar, and M. Gross. Surfels: Surface elements as rendering primitives. *Proc. SIGGRAPH 2000*, pages 335–342, 2000.

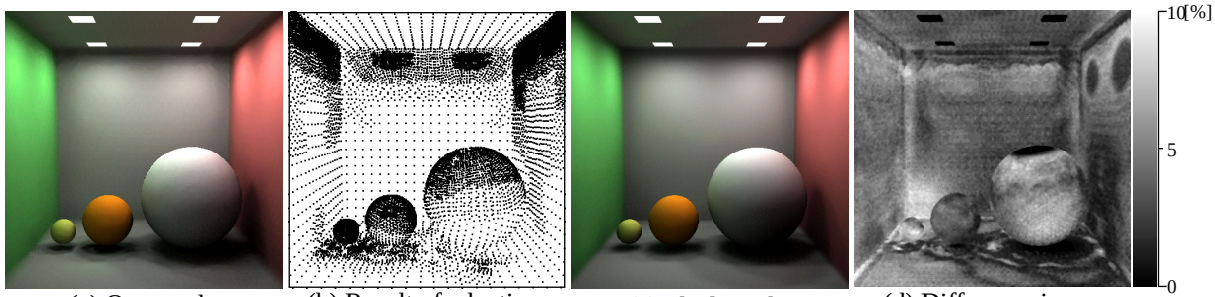
[15] S. Rusinkiewica and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. *Proc. SIGGRAPH 2000*, pages 343–352, 2000.

[16] G. Shaufler and H. W. Jansen. Ray tracing point sampled geometry. *Proc. Eurographics Workshop on Rendering 2000*, pages 319–328, 2000.

[17] M. Stamminger, A. Scheel, and H. P. Seidel. Hierarchical radiosity with global refinement. *Proc. Vision, Modeling, and Visualization 2000*, pages 263–271, 2000.

[18] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics (Proc. SIGGRAPH' 92)*, 26(3):52–64, 1992.

[19] M. Zwicker, H. Pfister, J. V. Baar, and M. .Grossn. Surface splatting. *Proc. SIGGRAPH 2001*, pages 371–378, 2001.



(a) Our result. (b) Result of adaptive addition of surfels. (c) Ideal result. (d) Difference image.

Figure 6: Experimental results.

(a) Two texture mapped bunnies. (b) Gallery of point models.

(c) Images of the statues shown in Fig. 7(b).

Figure 7: Practical examples.