Anexos

Apéndice A

Resultados adicionales

En este anexo se incluyen los resultados adicionales generados, producto de evaluar el método a distintas distancias de visualización para todos los modelos del capítulo de resultados con el objetivo de observar el comportamiento del filtrado a diferentes escalas.

En las Figuras A.1 y A.2 se muestra el tornillo, con detalles geométricos y de albedo, mientras que en las Figuras A.3, A.4, A.5 y A.6 se muestran las estatuas Lucy y Thai Statue.

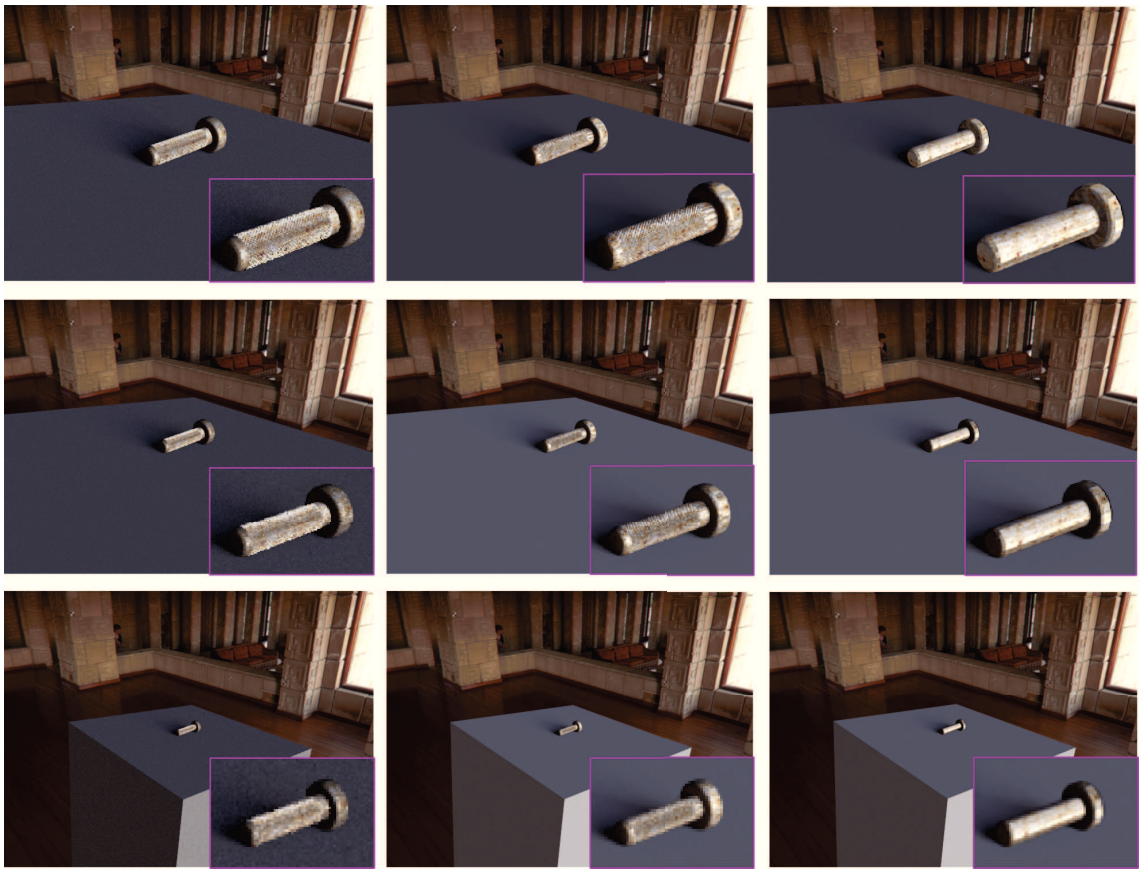

Figura A.1: Tornillo renderizado con mapa de ambiente, a diferentes distancias de visualización. Puede observarse cómo el método filtra correctamente la apariencia, preserva los detalles y evita el *aliasing*.

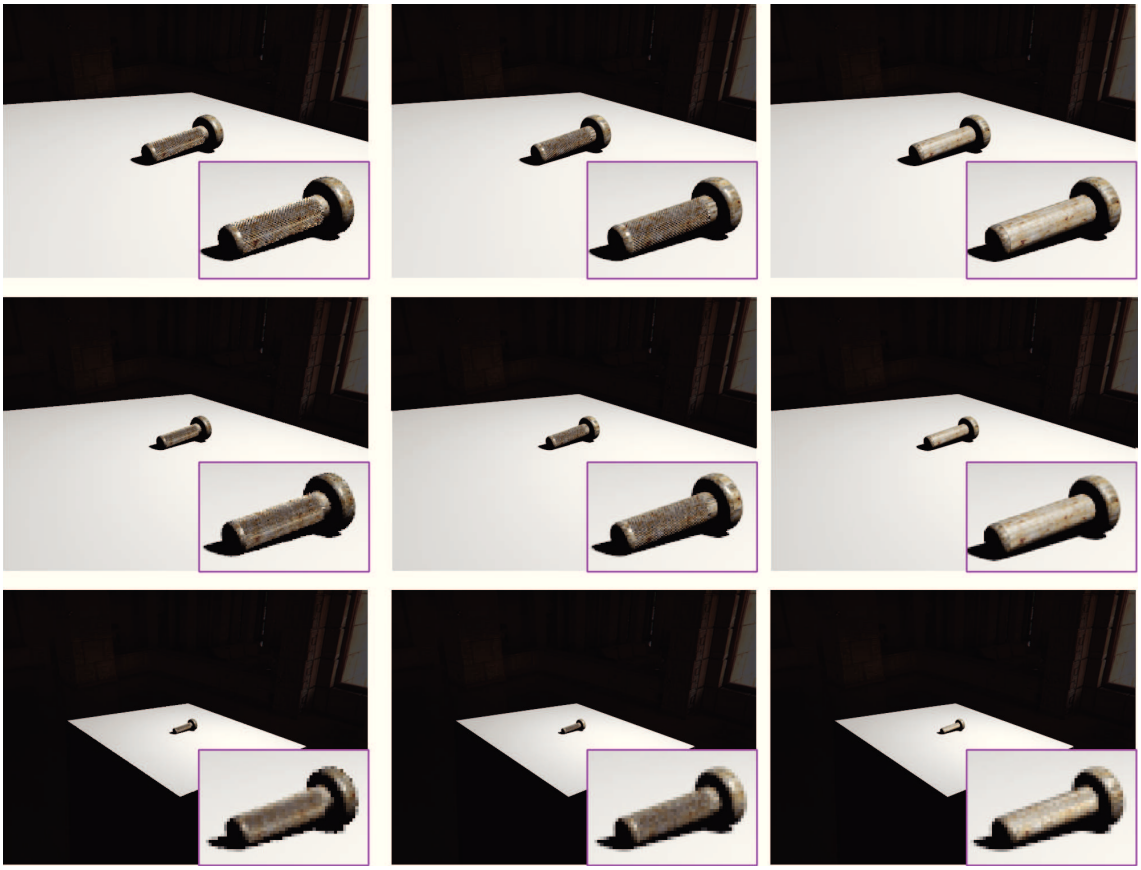

Figura A.2: Tornillo renderizado con luz puntual a diferentes distancias de visualización para observar el correcto funcionamiento del filtrado.

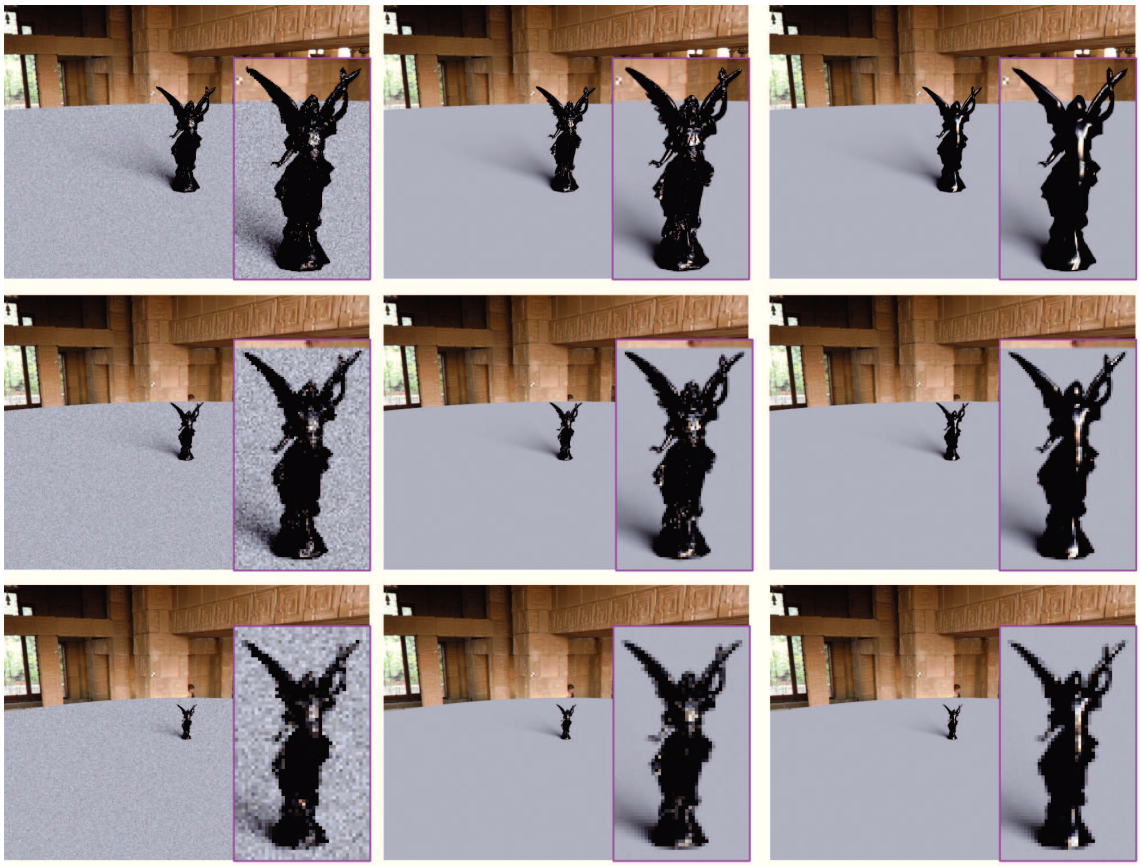

Figura A.3: Estatua Lucy renderizada con mapa de ambiente a diferentes distancias de visualización para observar el correcto funcionamiento del filtrado.

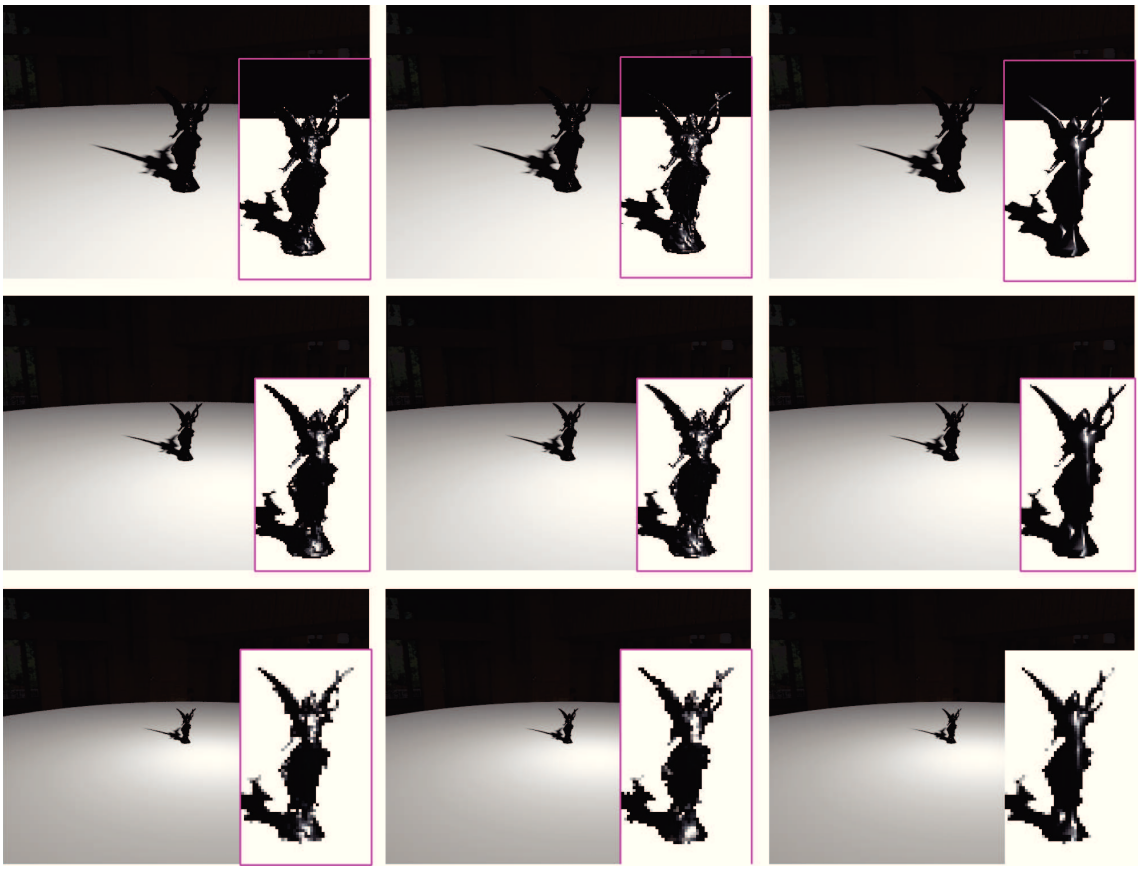

Figura A.4: Estatua Lucy renderizada con luz puntual a diferentes distancias de visualización para observar el correcto funcionamiento del filtrado.

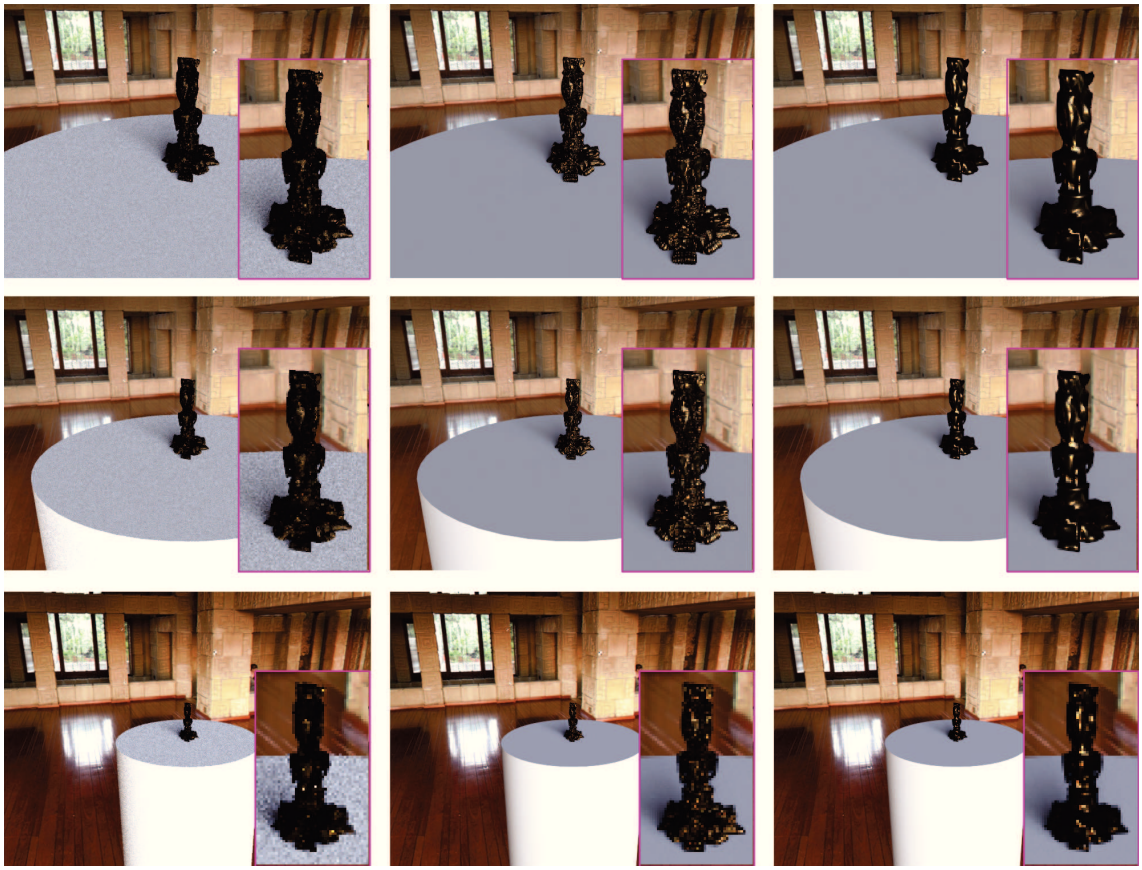

Figura A.5: Estatua Thai renderizada con mapa de ambiente a diferentes distancias de visualización para observar el correcto funcionamiento del filtrado.

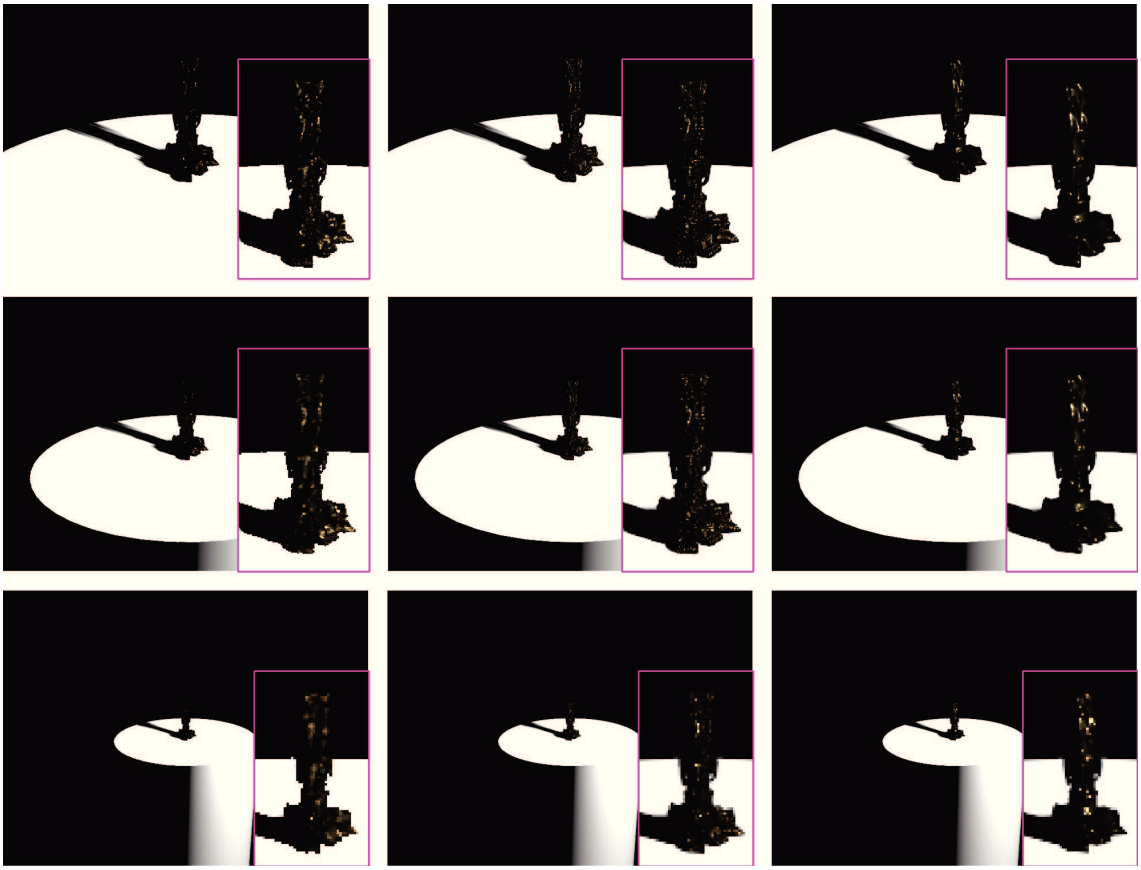

Figura A.6: Estatua Lucy renderizada con luz puntual a diferentes distancias de visualización para observar el correcto funcionamiento del filtrado.

Apéndice B

Implementación

En este anexo se describe el diseño de software realizado, para dar una buena intuición de la estructura y funcionamiento de la implementación. Se ha llevado a cabo sobre el trazador de rayos *Mjolnir*. Los diagramas que se muestran a continuación se centran en las clases más importantes, pudiendo obviar algunas de pequeño tamaño y focalizar así la atención en el funcionamiento global del sistema.

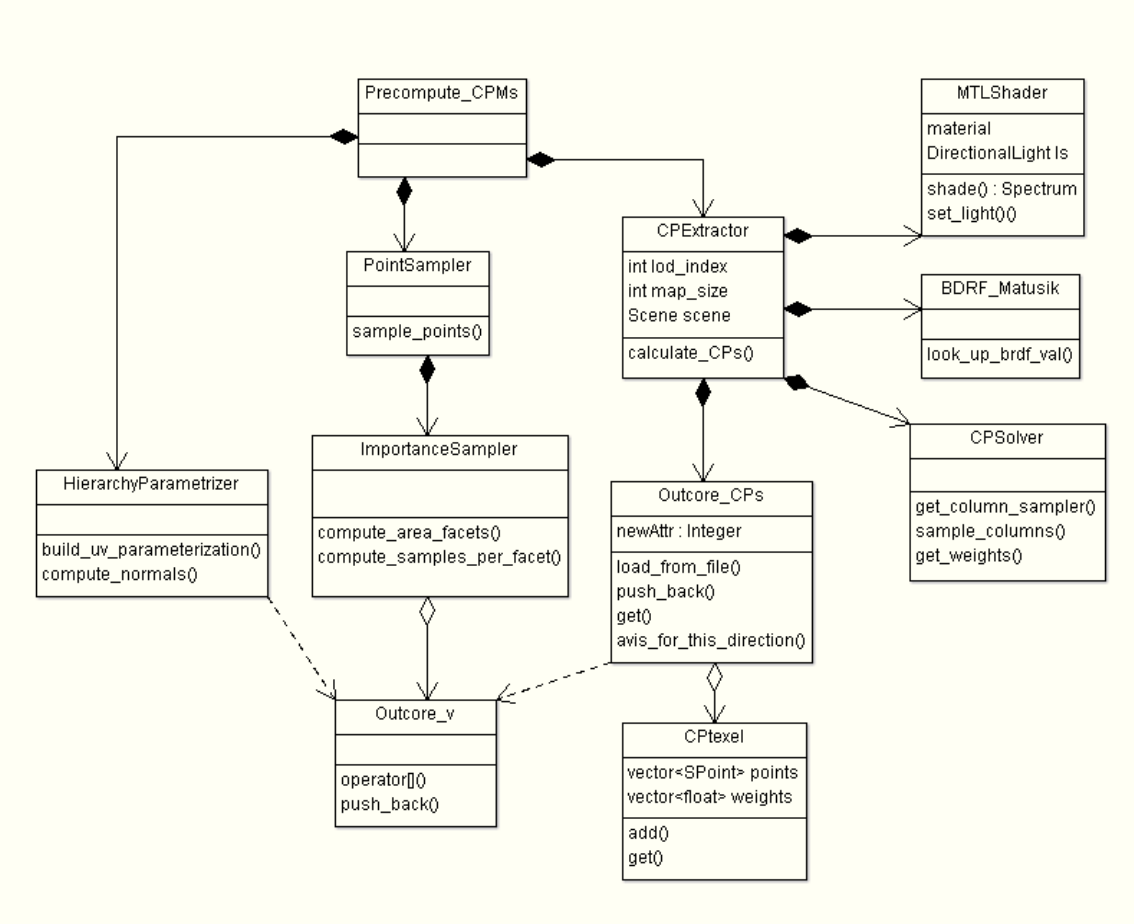

Figura B.1: Diagrama de clases de la fase de pre-cómputo.

Por otro lado, debido a la cantidad de datos que es necesario manejar simultáneamente en la aplicación, se han desarrollado estructuras *out-of-core* con un sistema de cache que

facilita la gestión eficiente de todos los datos que no caben en memoria.

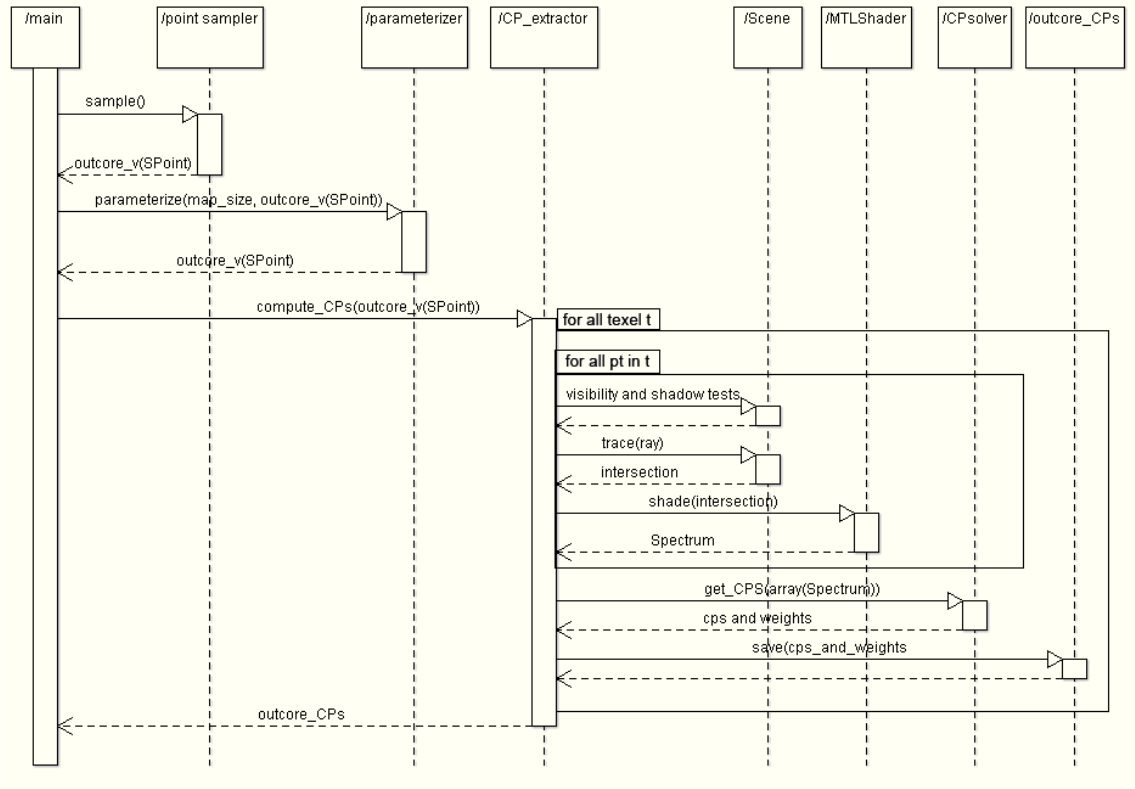

Figura B.2: Diagrama de secuencia de la fase de pre-cómputo.

Para generar el LOD de CPM, es necesario, para cada nivel, llevar a cabo un costoso pre-cálculo, compuesto de tres partes principales: distribución de puntos sobre la superficie, muestreo de radiancias en todos esos puntos para un conjunto determinado de pares de direcciones de luz y del observador, selección de puntos representativos, cálculo de pesos asociados y almacenamiento en memoria tanto de los puntos y pesos como del área visible proyectada de cada uno de los téxels.

En la Figura B.1 se muestra el diagrama de clases básico del algoritmo de precálculo, y en la Figura B.2 el diagrama de secuencia para esta fase. En primer lugar se lleva a cabo la distribución de muestras sobre M_org para, después, utilizar la parametrización para obtener la correspondencia téxels-muestras. Para cada téxel t, se evalua la radiancia en todos sus puntos, para el conjunto de direcciones de entrada y salida sobre la hemisfera. La matriz de radiancias tabuladas sirve de entrada al algoritmo de selección de columnas, que devuelve un subconjunto c con unos pesos asociados. El área visible proyectada se calcula aprovechando los rayos trazados para evaluar la reflectancia, y CPs y A_{vis} se almacenan en estructuras *out-of-core* para su posterior uso en tiempo de rénder.

En la Figuras B.3 y B.4 se exponen el diagrama de clases y de secuencia del algoritmo de render. En el bucle de muestras sobre la imagen, se proyectan rayos sobre la geometría simplificada. Una vez se produce la intersección con M_i, se accede, a través de la parametrización, al nivel de mipmap y al téxel t_i adecuados. Para todos los CPs dentro de t_i, se llevan a cabo los test de sombra y visibilidad, y se evalua su radiancia. La suma de las reflectancias pesadas, divididas por el A_{vis} de t_i para la dirección de la cámara

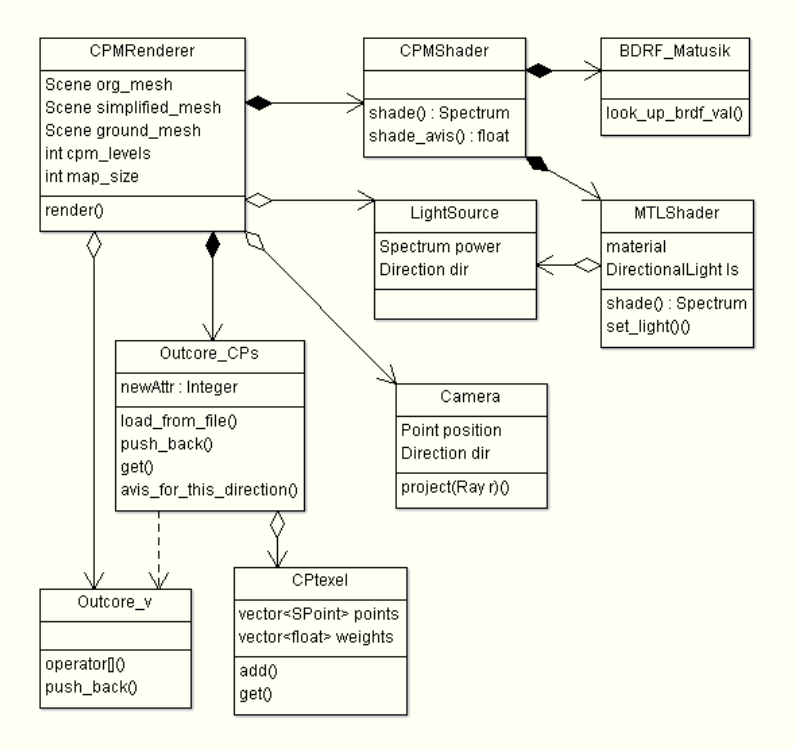

Figura B.3: Diagrama de clases del sistema de render.

correspondiente define el valor final del téxel.

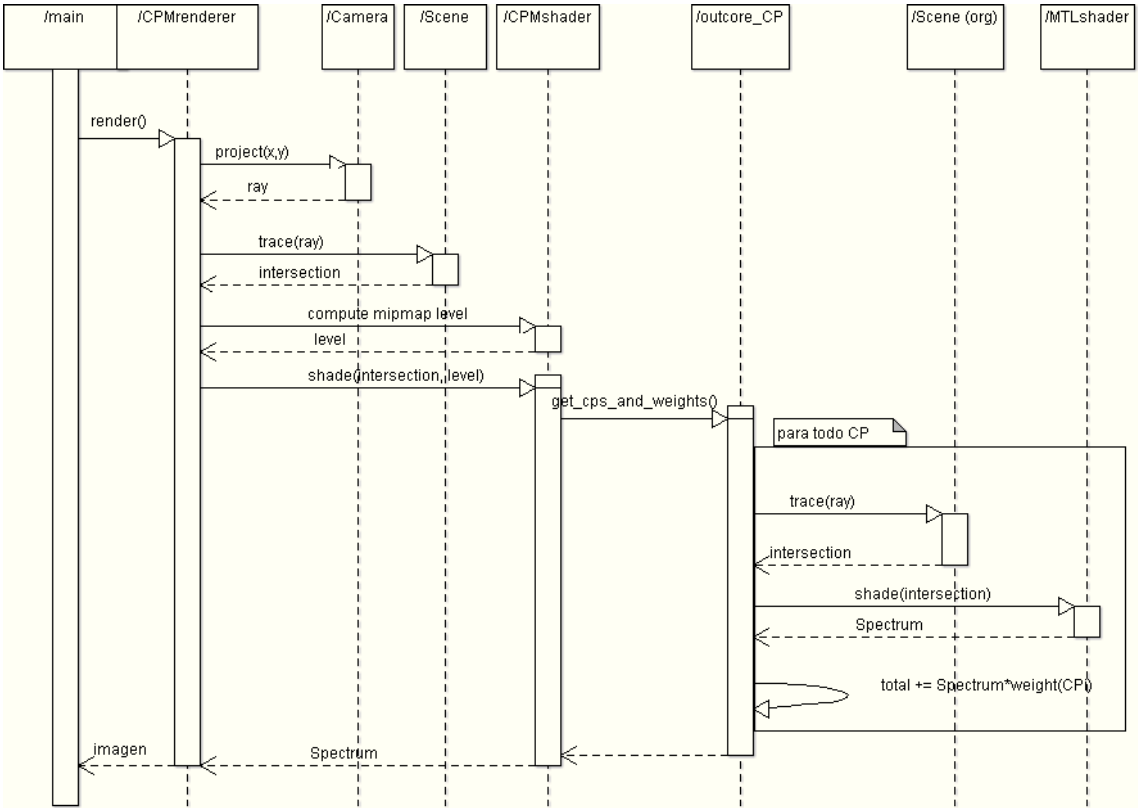

Figura B.4: Diagrama de secuencia de la fase de render.

Apéndice C

Coordenadas de textura y nivel de mipmap

Las coordenadas de textura están definidas en cada vértice de la malla e indican la correspondencia de ese punto con el mapa de textura en 2D. A la hora de renderizar, la resolución de dicho mapa de textura debe ser tal que la proyección de un téxel en el plano de la pantalla no sea mayor que un píxel.

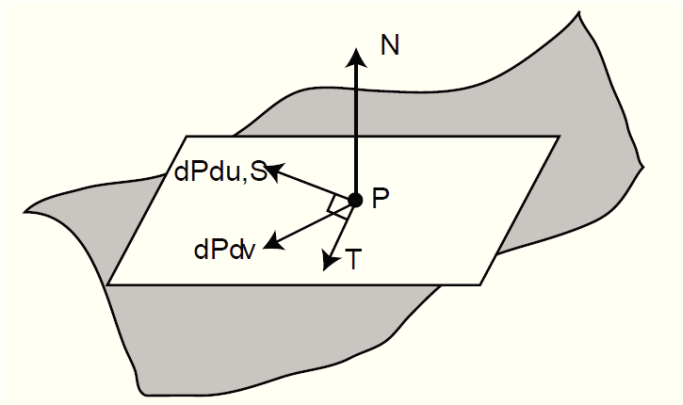

Figura C.1: Geometría diferencial en un punto P, donde N es la normal y dPdu y dPdv son las derivadas parciales de la superficie, $\frac{\partial p}{\partial u}$ y $\frac{\partial p}{\partial v}$. La geometría diferencial se define sobre el plano tangente T a la superficie en el punto P.

Para elegir el nivel de mipmap se necesita por tanto conocer la variación de u y v con respecto a las coordenadas x, y, z. Se hace uso de la geometría diferencial (Figura C.1) para conocer la cantidad de variación en textura con relación a la variación en geometría.

Los vectores s y t son ortogonales en el plano tangente a la superficie en p. Las derivadas parciales de la superficie, $\frac{\partial p}{\partial u}$ y $\frac{\partial p}{\partial v}$, también se encuentran en el plano tangente pero no son necesariamente ortogonales. La normal de la superficie n viene dada por el producto vectorial de $\frac{\partial p}{\partial u}$ y $\frac{\partial p}{\partial v}$. Los vectores $\frac{\partial n}{\partial u}$ y $\frac{\partial n}{\partial v}$ (no presentados aquí) guardan el cambio diferencial en la normal de la superficie en funciín de u y v.

El nivel del mipmap l viene definido por la siguiente expresión

$$l = log2 \left(max \left(\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2} \right) \right) \tag{C.1}$$

donde u y v son las coordenadas de textura, x e y las coordenadas cartesianas, y las derivadas parciales ∂ indican el diferencial de textura con respecto a la geometría.

Apéndice D

Characteristic Point Maps

Este anexo incluye la publicación en la que se basa este proyecto, *Characteristic Point Maps*, cuyos autores son Hongzhi Wu, Prof. July Dorsey y Prof. Holly Rushmeier, de Yale University, y que fue publicada en *Computer Graphics Forum* en 2009.

Eurographics Symposium on Rendering 2009
Hendrik P. A. Lensch and Peter-Pike Sloan
(Guest Editors)

Volume 28 (2009), Number 4

Characteristic Point Maps

Hongzhi Wu Julie Dorsey Holly Rushmeier

Computer Graphics Group, Yale University

Abstract

Extremely dense spatial sampling is often needed to prevent aliasing when rendering objects with high frequency variations in geometry and reflectance. To accelerate the rendering process, we introduce characteristic point maps (CPMs), a hierarchy of view-independent points, which are chosen to preserve the appearance of the original model across different scales. In preprocessing, randomized matrix column sampling is used to reduce an initial dense sampling to a minimum number of characteristic points with associated weights. In rendering, the reflected radiance is computed using a weighted average of reflectances from characteristic points. Unlike existing techniques, our approach requires no restrictions on the original geometry or reflectance functions.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

Rendering objects with high geometric and material variation at multiple scales is a challenging task. Efficient filtering techniques are needed to avoid aliasing. To address this issue, various level of detail representations have been proposed. Mesh simplification techniques (e.g. [Hop96]) generate geometry at multiple resolutions, and texture mipmap techniques [Wil83] pre-filter textures at various scales. Recently, Han et al. [HSRG07] have proposed a filtering method, where normal maps are used to represent fine-scale geometry. Unfortunately, these existing techniques are restricted in the variations of geometry and reflectance for which they can produce an accurate result. In this paper, we present a new approach, characteristic point maps (CPMs), to efficiently filter the appearance of models with arbitrary geometry and material. CPMs can be viewed as a precomputed object-space adaptive sampling method for efficient rendering.

Consider a problematic example, a cylinder with small-scale sharp ridges with alternating reflectance on various facets, as shown in Fig. 1. On the left is the result from densely sampling the full detailed object representation. In the center is a rendering produced by multi-sampling a simplified geometry combined with a normal map to represent the ridges. This is the ground truth result for normal map filtering methods (e.g. [HSRG07]). Clearly the result has not

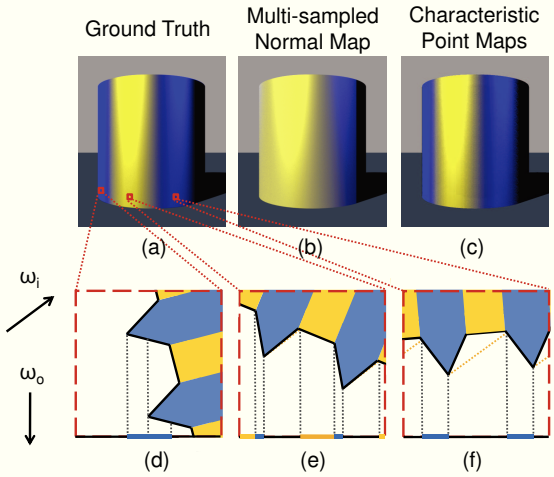

Figure 1: *Renderings of a cylinder using different methods. The ground truth (a) is obtained by expensive brute-force multi-sampling. Multi-sampling the simplified geometry with a normal map is shown in (b). Fig. 1(c) uses CPMs. Fig. 1(d-f) are top views of different parts of the original cylinder. The masking effect is shown in (d), and Fig. 1(f) illustrates the shadowing effect, both of which cannot be represented using normal maps and are well preserved by our CPMs. The lighting direction is ω_i, and the view direction is ω_o.*

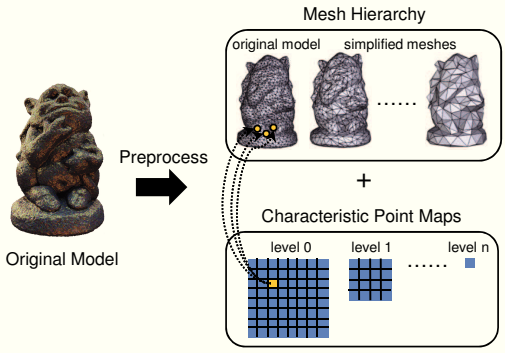

Figure 2: *Illustration of our representation. Each texel in one CPM contains characteristic points defined on the original model, which are chosen to preserve the filtered reflectance function.*

retained the appearance generated by the small scale geometry. The masking effects of the blue ridges obscuring the view of the yellow facets (Fig. 1(d)) that make the left most part of the cylinder appear blue are lost. The shadowing effects of the blue ridges that keep light from reaching the yellow facets on the right side of the cylinder (Fig. 1(f)) resulting in a blue color are also missing. Fig. 1(c) shows the rendering using our CPMs. The appearance resulting from facet masking and shadowing effects is retained.

The main challenge in generating a multi-scale reflectance representation for simplified geometry is to efficiently compute and represent the 6D spatially-varying filtered reflectance function – effectively the bidirectional texture function BTF [DvGNK99]. BTFs are useful when capturing real-world appearance, however they are notoriously difficult to compress or to represent with parametric functions [MMS*04]. With CPMs we demonstrate that the geometry and SV-BRDF that produce the full 6D function can be represented with a much smaller footprint.

In this paper, we represent an object as a simplified mesh hierarchy coupled with a CPM hierarchy (Fig. 2). Each texel in one CPM contains view- and lighting- independent characteristic points on the original object, whose density adapts to the complexity of the filtered reflectance functions. A hierarchy of CPMs is computed for a variety of scales. In rendering, the reflected radiance is rapidly computed using a weighted average of reflectances at individual characteristic points.

The major contribution of this paper is a framework that efficiently computes and adaptively represents a new hierarchical representation for any geometry and SV-BRDFs, using randomized column sampling on a matrix formulation derived from the rendering equation [Kaj86]. Unlike the normal map filtering method, we accurately incorporate shadowing and masking effects. We are also able to handle arbitrary BRDFs. We believe CPMs are the first structures that efficiently represent multi-scale reflectance functions for multi-resolution mesh hierarchy with arbitrary SV-BRDFs.

2. Previous Work

Hierarchy of Representations Our CPMs build on the basic idea of using different representations at different levels of detail. Kajiya [Kaj85] suggested a hierarchy of scales from geometry, bump/normal maps to BRDFs. Many researchers have explored the relationships between various scales. For example, Westin et al. [WAT92] obtained a densely-sampled BRDF from scattering events computed from fine-scale geometry. Becker et al. [BM93] computed smooth transition from displacement maps, bump maps to BRDFs.

Appearance-Preserving Mesh Simplification Geometric simplification techniques (e.g. [GH98, LT00, SSGH01]) maintain small scale details by using colored texture maps sampled from original objects. Most of these methods focus on minimizing the parameterization mapping distortion. However, the actual appearance may not be well-preserved as fine-scale geometry details are lost during simplification. Cohen et al. [COM98] introduced normal maps, which capture small scale surface orientation, in addition to texture maps. Cook et al. [CHPR07] proposed an algorithm to render complex aggregate details by randomly selecting a subset of the geometric elements which preserve the overall appearance. While the method works well for procedurally generated models, it is not clear how to extend the idea to more general cases.

Reflectance Filtering Han et al. [HSRG07] filtered a certain class of BRDFs with normal maps as the convolution of Normal Distribution Function and BRDF. They implicitly represent fine-scale geometry using a normal map, so neither shadowing nor masking effects are considered. Furthermore, there are limitations for handling multiple materials (e.g. only a linear combination of basis BRDFs is allowed). Tan et al. [TLQ*08] presented a mixture model that fits the filtered reflectance of Gaussian or cosine-based BRDFs using Expectation Maximization. Shadowing and masking effects are approximated using horizon mapping distribution. Particularly, masking effects are implemented by attenuating the unmasked appearance. Ma et al. [MCT*05] filtered BTF [DvGNK99] by applying Principle Component Analysis(PCA) to the BTF tabulation. However, in order to approximate high-frequency filtered reflectance functions faithfully, dense sampling of the 6D BTF is needed which is expensive both in time and space. Furthermore, it is challenging to extrapolate the filtered result beyond the resolution of BTF. In our method, conversion to a BTF representation is not required.

While we use points in our representation, our approach is qualitatively different from point-based rendering such as [SP04]. We do not represent the shape with points, and so are not concerned with the issues of visibility of point sets.

Symbol	Description
A	a surface
$L(x,\omega_o)$	reflected radiance at point x along direction ω_o
$\overline{L}(A,\omega_o)$	average reflected radiance of A along direction ω_o
$f_r(x,\omega_i',\omega_o')$	bidirectional reflectance distribution function at point x
$\overline{f}_r(A,\omega_i,\omega_o)$	average reflectance distribution function of A
$A_{vis}(\omega_o)$	visible subset of A when viewed from direction ω_o
$a_{vis}(A,\omega_o)$	visible projected area function of A
$f(x,\omega_i,\omega_o)$	apparent reflectance function at x
R	a matrix containing sampled spatially-varying reflectance functions of A
C	a matrix containing sampled columns from R
C^+	Moore-Penrose generalized inverse of C

Table 1: *Summary of the notation used in the paper.*

3. Characteristic Points

We select characteristic points to represent the light scattering properties of a surface using a *filtered reflected function*. In this section we define the filtered reflectance function, and then show how a matrix formulation is used to select characteristic points.

3.1. Preliminaries

We derive the equation for the effective average reflectance function. Note that throughout the paper, we focus on rendering under direct illumination only. First, the reflected radiance L at a single point x along direction ω_o is

$$L(x,\omega_o) = \int_{S^2} L_i(x,\omega_i)V(x,\omega_i)f_r(x,\omega_i',\omega_o')(n\cdot\omega_i)d\omega_i. \tag{1}$$

Here ω_i is the lighting direction, ω_o is the view direction, ω_i' and ω_o' are the same directions expressed in the local frame at x. V is the visibility function, which returns 1 if x is not blocked along the direction and 0 otherwise, and L_i is the incident radiance. f_r is the SV-BRDF, and n is the normal. In addition, $(\cdot)$ is the cosine of the angle between the two vectors, which is clamped to zero if it is negative.

Now if we are looking at surface A from a distance, the spatially averaged reflected radiance along direction ω_o is the average of all reflected radiance from visible part of A (Figure 3 illustrates the case). If we define $a_{vis}(A,\omega_o)$ as the visible projected area of A along direction ω_o, then we have

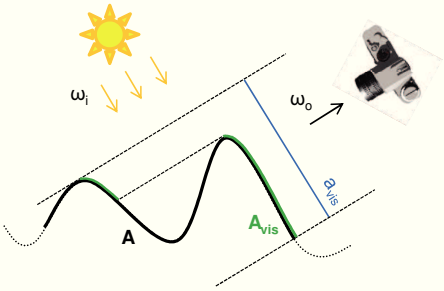

Figure 3: *Illustration of A, A_{vis} and a_{vis} in the filtered reflectance function derivation.*

the following equation:

$$\begin{aligned}\overline{L}(A,\omega_o) &= \frac{1}{a_{vis}(A,\omega_o)}\int_{A_{vis}(\omega_o)} L(x,\omega_o)dA_{vis}(\omega_o)\\ &= \frac{1}{a_{vis}(A,\omega_o)}\int_{A_{vis}(\omega_o)}\int_{S^2} L_i(x,\omega_i)V(x,\omega_i)\\ &\quad f_r(x,\omega_i',\omega_o')(n\cdot\omega_i)\,d\omega_i\,dA_{vis}(\omega_o),\end{aligned} \tag{2}$$

where $A_{vis}(\omega_o)$ is the subset of A which are visible from direction ω_o. Note that we use the differential dA_{vis}, not dA, because invisible (masked) parts do not contribute to the reflected radiance.

It can be verified that

$$dA_{vis}(\omega_o) = V(x,\omega_o)(n\cdot\omega_o)dA, \tag{3}$$

as one could think of A as composed of infinitely many infinitesimal discs. dA is the area of one disc, then $dA_{vis}(\omega_o)$ is just the visible projected area of the disc along direction ω_o. We further define the *apparent reflectance function* f as

$$f(x,\omega_i,\omega_o) = V(x,\omega_i)V(x,\omega_o)f_r(x,\omega_i',\omega_o')(n\cdot\omega_i)(n\cdot\omega_o). \tag{4}$$

Substituting Eq. 3 and 4 back into Eq. 2 gives

$$\begin{aligned}&\overline{L}(A,\omega_o)\\ &= \frac{1}{a_{vis}(A,\omega_o)}\int_A\int_{S^2} L_i(x,\omega_i)f(x,\omega_i,\omega_o)d\omega_i dA\\ &= \int_{S^2} L_i(\omega_i)\left(\frac{1}{a_{vis}(A,\omega_o)}\int_A f(x,\omega_i,\omega_o)dA\right)d\omega_i\\ &= \int_{S^2} L_i(\omega_i)\overline{f}_r(A,\omega_i,\omega_o)d\omega_i.\end{aligned} \tag{5}$$

In the above derivation, we assume that $\forall x\in A, L_i(x,\omega_i) = L_i(\omega_i)$. This assumption holds for distant lights (e.g. directional lights, environment maps), and we will discuss how to handle local lights in Sec. 4.2.

Finally, we obtain the equation for filtered reflectance function from Equation 5:

$$\overline{f}_r(A,\omega_i,\omega_o) = \frac{1}{a_{vis}(A,\omega_o)}\int_A f(x,\omega_i,\omega_o)dA. \tag{6}$$

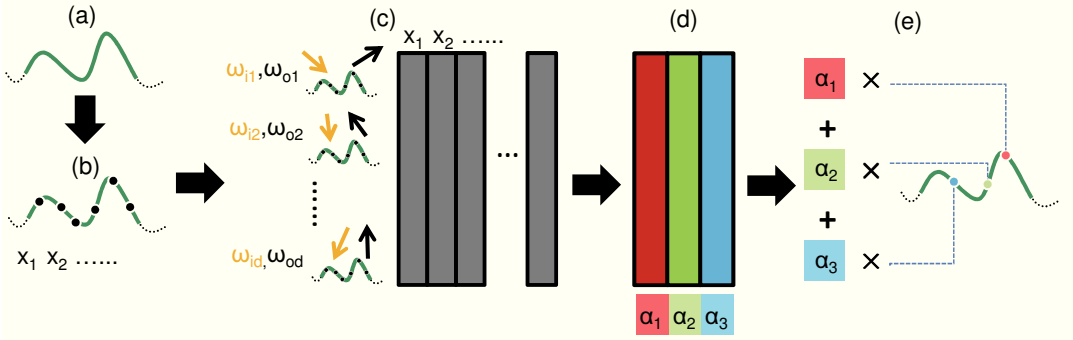

Figure 4: *Conceptual diagram for computing characteristic points and weights. The original surface (a) is sampled as a dense point set (b); (c) the reflectance for each point is computed for sampled incident and view directions and then stored in a matrix with one column per sample point and one row per incident/view direction pair; (d) the matrix is approximated by a small subset of columns with associated weights; (e) the filtered reflectance from a dense sampling of points on the original surface is approximated by a weighted sum of reflectance from the points associated with the selected matrix columns.*

3.2. Matrix Formulation of the Filtered Reflectance Function

To compute the filtered reflectance function in Equation 6, we first discretize the spatial integration into a summation:

$$\overline{f}_r(A,\omega_i,\omega_o) \approx \frac{1}{a_{vis}(A,\omega_o)} \sum_{j=1}^{m} f(x_j,\omega_i,\omega_o)\Delta A_j. \qquad (7)$$

Here we consider m points $x_j \in A$, each representing discrete area ΔA_j. This discretization gives a reasonable approximation as long as we use a sufficiently large m.

Next, we focus on efficiently computing the summation term in Eq. 7, and leave the $\frac{1}{a_{vis}(A,\omega_o)}$ term to Sec. 3.4. Observe that in the summation, f is evaluated m times for different input parameters. This could be expensive since m is typically large. Intuitively, if we could cluster "similar" f functions together, the summation could be approximated by evaluating f at only a few *characteristic points* $\hat{x}_k$, with appropriate weights α_k:

$$\sum_{j=1}^{m} f(x_j,\omega_i,\omega_o)\Delta A_j \approx \sum_{k=1}^{c} f(\hat{x}_k,\omega_i,\omega_o)\alpha_k, \qquad (8)$$

where $c \ll m$.

In order to find these characteristic points $\{\hat{x}_1,\hat{x}_2,\ldots,\hat{x}_c\}$ as well as their corresponding weights $\{\alpha_1,\alpha_2,\ldots,\alpha_c\}$, we take advantage of recent advances in low-rank matrix approximation theory by adopting the algorithm described in [DMM06]. To apply the algorithm, we further convert our problem to a matrix form by tabulating the summation term at d sampled incident and view direction pairs:

$$\begin{aligned}&\Big(\sum_{j=1}^{m} f(x_j,\omega_{i1},\omega_{o1})\Delta A_j,\ldots,\sum_{j=1}^{m} f(x_j,\omega_{id},\omega_{od})\Delta A_j\Big)\\ =&R\begin{pmatrix}1 & 1 & \ldots & 1\end{pmatrix}^T \qquad (9)\end{aligned}$$

where matrix R is equal to

$$\begin{pmatrix} f(x_1,\omega_{i_1},\omega_{o_1})\Delta A_1 & \ldots & f(x_m,\omega_{i_1},\omega_{o_1})\Delta A_m \\ f(x_1,\omega_{i_2},\omega_{o_2})\Delta A_1 & \ldots & f(x_m,\omega_{i_2},\omega_{o_2})\Delta A_m \\ \ldots & \ldots & \ldots \\ f(x_1,\omega_{i_d},\omega_{o_d})\Delta A_1 & \ldots & f(x_m,\omega_{i_d},\omega_{o_d})\Delta A_m \end{pmatrix}$$

Observe that each column of R is a tabulation of the summation term at point x_j at different sampled directions (ω_i,ω_o). Therefore, finding the characteristic points is equivalent to choosing *representative columns* in R.

It is important to note that in the current formulation we have introduced two simplifications that may limit the quality of our results. First, Eq. 5 effectively uses a box filter when integrating over the spatial domain. This does not eliminate high frequency signals as a low-pass filter would do. Second, the matrix formulation is based on the summation term in Eq. 7 only. As a result, our column sampling technique described next can at best optimize point selection with respect to the summation term, although we measure errors using the full equation 7. A new matrix formulation could be developed which also includes the $\frac{1}{a_{vis}}$ term. In future work we plan to explore methods that do not include these simplifications.

3.3. Randomized Matrix Column Sampling

We briefly describe the randomized column sampling algorithm in [DMM06] for selecting representative columns. Given any matrix R and $k \ll rank(R)$, the algorithm runs in $O(SVD(R))$ time and selects c columns of R as a new matrix C. Then the matrix CC^+R approximates R with relative error in terms of $\| R - R_k \|_F$, where R_k is the best rank k approximation to R in the Frobenius norm, C^+ is the Moore-Penrose generalized inverse, and $\| \cdot \|_F$ is the Frobenius norm. We show the pseudo-code in Tab. 2. (Interested readers are di-

1. Compute SVD of R as
$$R = U\Sigma V^T = U \begin{pmatrix} \Sigma_k & 0 \\ 0^T & \Sigma_{\rho-k} \end{pmatrix} \begin{pmatrix} V_k^T \\ V_{\rho-k}^T \end{pmatrix}$$
2. Compute p_i as
$$p_i = \frac{1}{3}\left(\frac{[(V_k)_{(i)}]^2}{\|V_k\|^2} + \frac{(V_k)_{(i)}(\Sigma_{\rho-k}V_{\rho-k}^T)^{(i)}}{(V_k)\cdot(\Sigma_{\rho-k}V_{\rho-k}^T)} + \frac{[(\Sigma_{\rho-k}V_{\rho-k}^T)^{(i)}]^2}{\|\Sigma_{\rho-k}V_{\rho-k}^T\|^2}\right)$$
3. Sample c columns from R according to $\{p_i\}$

Table 2: *Pseudo-code of the randomized matrix column sampling algorithm in [DMM06]. Here $\rho = rank(R)$, $\{p_i\}$ is the probability distribution to select a column from R.*

rected to the original paper for details on the algorithm as well as the proof.)

Once we have selected c representative columns of R, it is straightforward to compute corresponding weights α_k. We just substitute the approximation matrix CC^+R back into Eq. 9, which yields

$$\begin{aligned}
&\left(\sum_{j=1}^{m} f(x_j,\omega_{i1},\omega_{o1})\Delta A_j,\ldots,\sum_{j=1}^{m} f(x_j,\omega_{id},\omega_{od})\Delta A_j\right)\\
\approx& CC^+R\begin{pmatrix}1 & \ldots & 1\end{pmatrix}^T = C\begin{pmatrix}\alpha_1 & \alpha_2 & \ldots & \alpha_c\end{pmatrix}^T\\
=&\left(\sum_{k=1}^{c} f(\hat{x}_k,\omega_{i1},\omega_{o1})\alpha_k,\ldots,\sum_{k=1}^{c} f(\hat{x}_k,\omega_{id},\omega_{od})\alpha_k\right). \quad (10)
\end{aligned}$$

Hence we have

$$\begin{pmatrix}\alpha_1 & \alpha_2 & \ldots & \alpha_c\end{pmatrix}^T = C^+R\begin{pmatrix}1 & 1 & \ldots & 1\end{pmatrix}^T. \quad (11)$$

It is possible that some values of α_k are negative. While this formulation may produce a good approximation for all sampled (ω_i,ω_o) pairs, we have found in practice that the reflectance computed from these weights is very unstable, even for view and lighting directions that are slightly different from sampled ones. Therefore, we clamp negative weights to zero, which essentially encourages a result with non-negative weights. If negative weights cannot be avoided, our algorithm tries to find the result which gives best approximation after clamping the negative weights. Our experiments show good quality in computing reflectance for directions that are not sampled during precomputation.

As in many randomized algorithms, we repeat the above procedure a few times to improve the quality of the result. We measure approximation error using the squared distance between the average reflectance functions computed from R and from column representatives. A similar error measurement appears in [LFTG97].

In a sense, the column-sampling process can be viewed as expressing the summation in Eq. 7 in terms of a minimal set of basis functions $\cup\{f(\hat{x}_k)\} \subseteq \cup\{f(x_j)\}$, given approximation error constraint. Essentially we are exploiting the coherence in apparent reflectance functions defined at different points. Note that we cannot select CPs directly from the result of SVD, since we have to be able to select columns of the original matrix and the eigenvectors do not correspond to original columns anymore.

In a different context, Hašan et al. [HPB07] formalizes their problem also as column-based matrix sampling to reduce the number of lights in rendering computations. While their paper develops a rapid clustering method for fast previewing, which might be applicable to our problem, we are more concerned about approximation quality and therefore we would like to base our algorithm on a theoretical result that is proved to be optimal for any input, as shown in [DMM06]. In contrast, [HPB07] does not give formal proof, except for showing that their method is unbiased.

3.4. Visible Projected Area Function

In Eq. 7 the filtered reflectance function is represented as a $\frac{1}{a_{vis}(A,\omega_o)}$ term times a summation term. We have described the details of simplifying the summation term which leaves consideration of the a_{vis} term. Essentially $a_{vis}(A,\cdot)$ is a 2D spherical function whose computation requires costly global visibility calculations. Fortunately, a_{vis} can be precomputed and compressed for efficient evaluation during rendering. In precomputation, we render the original mesh on the GPU from a densely sampled set of directions ω_o. A high resolution texture is used to mark which part of the mesh corresponds to a texel in the CPM hierarchy. The rendering result is then read back from GPU to compute the visible projected area. In our experiments, this approach is an order of magnitude faster than an implementation solely based on CPU. Finally, we parameterize a_{vis} over a cube-map and compress each face of the cube-map using Haar wavelets. We choose Haar wavelets for its simplicity, good compression rate and rapid signal reconstruction.

4. Computing and Using Characteristic Point Maps

Given the method for computing characteristic points in Sec. 3, we describe in this section how an object is preprocessed into a CPM hierarchy and a mesh hierarchy. We then describe the corresponding rendering algorithm.

4.1. Preprocessing

Starting from an original model M_{org}, we apply existing geometry-based simplification techniques (e.g. [GH97]) to get a hierarchy of simplified meshes $\{M_1,M_2,\ldots\}$. We then establish a parameterization T on M_1 for the CPM hierarchy. The visible projected area functions a_{vis} for all texels in CPMs are computed using the method described in Sec. 3.4.

Next, we densely sample random points over the surfaces of M_{org}, denoted as X. Then, for every texel p in each mip level of the CPMs, we find its corresponding geometry $g(p)$ on M_{org} based on Euclidean distances. We densely sample

point set X_{M1} over M_1 and use the parameterization T to assign each of these points to a corresponding texel in the CPM. Then we sort X_{M1} into a k-d tree. Finally, for any point x on M_{org}, we look up in the k-d tree for the closest point in X_{M1}, and assign the associated texel to x. The union of all points belonging to texel p is defined as $g(p)$.

We select characteristic points for texel p from $g(p)$, using the technique introduced in Sec. 3.3. We determine the parameters required by the randomized matrix column sampling algorithm as follows. First, we select a minimum k such that the energy of the rank-k approximation matrix is above a user-specified percentage ψ (a typical value is 97%) of the total energy of R in the Frobenius norm (see Sec. 3.3 for more details on k). Second, we compute the number of CPs, c, using a binary-search like algorithm, based on a user-specified error threshold ε (a typical value is 30dB in terms of Signal-to-Noise Ratio(SNR)). We start with the interval $[k, c_m]$, where c_m is the maximum number of CPs supplied by the user. We use the median in the current interval as the number of columns to see whether we could approximate with an error below ε (see Sec. 3.3 on how we compute approximation error). Based on this result, we reduce the interval into its upper or lower half, and then repeat the above process until there is only one integer left, which is recorded as c. Third, the number of directional pair samples, d, is estimated based on the angular frequency of the filtered reflectance functions. A typical value of 4096 is sufficient to produce good results in most of our experiments.

The reflectance function is different for each color channel. One option is to use the weighted average of the sampled reflectances for each channel to select characteristic points. Then weights for each individual color channel are computed separately for better approximation quality. In cases where the reflectances in three color channels are highly decorrelated (e.g. the cylinder in Fig. 1), we compute characteristic points separately for each color channel.

We build the mipmap in a bottom-up fashion. Each level is directly computed from the initial point sample set due to the non-linearity of $\frac{1}{a_{vis}}$. Note that at higher levels of the mipmap, one texel could correspond to many point samples such that the matrix R (Sec. 3.2) is too large to fit in memory. To tackle this problem, we allow the user to specify a number n_{batch}, so that if the number of point samples for one texel exceeds n_{batch}, we process them in batches with at most n_{batch} points at a time. Theoretically this method does not produce a result as good as that by processing all point samples at the same time. In our experiments, we found that it gives satisfactory approximation quality while strictly following the memory constraint. Note that similar to the discussion at the end of Sec. 3.1, we also use a box filter here when building the hierarchy, which does not eliminate high frequency signals and may cause aliasing. In future work we would like to apply a low-pass filter in the mipmap generation.

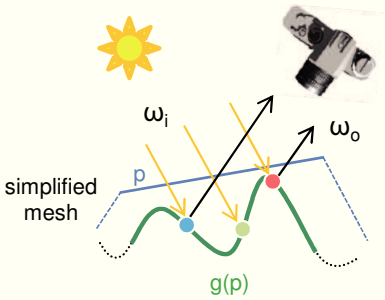

Figure 5: *Filtered reflectance reconstruction from characteristic points.*

After precomputation, each texel p in the CPMs consists of

1. Characteristic points selected from X
2. Corresponding weights
3. Wavelet coefficients for a_{vis}

In addition, we build a parameterization T_i on each of the meshes left in the hierarchy $M_i, i = 2,3,\ldots$ and use it to sample the parameterization T on M_1. Then the composite parameterization $T(T_i(\cdot))$ can be used to map points on M_i to texels in the CPMs .

4.2. Rendering

An object with CPMs can be rendered in a ray-tracer. Given a view and lighting configuration, we first determine the appropriate level i in the mesh hierarchy based on the screen-space projected area of the bounding volume of the original mesh. Then we proceed with ray-object intersection test and shadow test on M_i we have just selected. If an intersection point is found and it is not in shadow, we calculate the corresponding uv coordinates for the CPMs using the composite mapping $T(T_i(\cdot))$ described in Sec. 4.1. Screen-space derivatives for the CPMs are subsequently computed to determine the appropriate level j in the CPM hierarchy for rendering. Next, we reconstruct average reflectance from current texel p at mip level j. Let $g(p)$ denote the corresponding geometry of p on the original mesh M_{org} (see Fig. 5). From Eq. 7, 8 and 4, we immediately have

$$\overline{f}_r(\omega_i,\omega_o) \approx \frac{1}{a_{vis}(\omega_o)} \sum_{k=1}^{c} V(\hat{x}_k,\omega_i) V(\hat{x}_k,\omega_o) f_r(\hat{x}_k,\omega_i',\omega_o')(n\cdot\omega_i)(n\cdot\omega_o)\alpha_k \qquad (12)$$

For each characteristic point $\hat{x}_k$ in p, we sum up the product of its BRDF f_r with two visibility terms, two cosine terms and its weight α_k. The average reflectance is computed as dividing the summation by the visible projected area, which is obtained using inverse wavelet transform from corresponding wavelet coefficients. Note that the two visibility terms are computed on M_{org} to account for small-scale shadowing and masking effects which are not represented by M_i. Since

we have already performed a shadow test on the coarse geometry M_i for inter-object shadows, we can accelerate the computation of visibility terms by limiting the intersection test only to M_{org} for intra-object shadows. Note that when there are objects close to $g(p)$, we no longer do the initial shadow test on M_i and instead perform visibility tests at multiple points on $g(p)$, for better precision in inter-object shadows. If the test results indicate that $g(p)$ is partially occluded by other objects, we switch to rendering using the original representation as to our knowledge it is impractical to incorporate such cases during CPMs precomputation. Otherwise, $g(p)$ is either completely visible to the light or completely in shadow, so we continue to do the shading using CPMs.

Similar to traditional texture mipmaps, we could perform trilinear interpolation between neighboring texels and adjacent levels in the CPM hierarchy. Once we get the properly filtered reflectance function from the CPMs, the outgoing radiance is obtained using Eq. 5 by multiplying the reflectance with the incoming radiance.

In cases where the mip level j computed from screen-space derivatives is beyond the most detailed level in the CPM hierarchy (e.g. the viewer is too close to the object), we switch to multi-sampling of the original representation instead of using CPMs.

In addition, when the distant light assumption in Sec. 3.1 does not hold, we slightly change the process for choosing mip level j in CPMs to handle local lights. Specifically, we consider the ratio between the distance from the intersection point to the light and the size of the geometry covered by one texel in CPMs when determining the mip level. The idea is that this ratio should be large enough so that the incident direction ω_i is approximately constant across the texel.

5. Results

We conducted our experiments on a workstation with a 2.66GHz quad-core processor, 3GB memory and an nVidia 8800GT graphics card. When computing the a_{vis} function, a resolution of 6×64^2 is used for the cube map to sample directions. Applying Haar wavelets allowed us to use $4\% \sim 6.5\%$ of the original space to store a_{vis}. All images were rendered with a resolution of 512x512, using our own unoptimized Monte Carlo ray-tracer. We used only 4 eye rays per pixel for CPMs rendering, while 64-1024 rays were required in rendering the ground truth images with no aliasing. Timing results along with other details are listed in Tab. 3. For each scene we show rendering results for normal maps, ground truth (i.e. the original model, densely sampled), CPM and the original model rendered in approximately the same amount of time as the CPM (i.e. an "equal time budget" image).

In addition to the images shown in the paper, please refer to the accompanying video that shows smooth transitions through different mip levels. In each frame the boundaries between different mip levels are not visible, and there is no popping as the levels change between frames. We believe that the pixel level flickering may be due to the use of a box filter in our formulation.

The details of the cylinder scene is described in Sec. 1. The bolts scene (Fig. 6) is populated with bolts with many diamond-shaped bumps on the body, along with a high-frequency BRDF, *silver-metallic-paint-2* from [MPBM03]. In addition, spatial variation of reflectance is modeled using a texture of tainted metal. Our method preserves the complex shading variations along silhouettes of the bolts, while multi-sampled normal map method tends to give more uniform appearance over the body of the bolts.

In the wall scene, highly detailed geometry is used to model the fine scale features on the surfaces of the walls (see Fig. 6). A Lambertian BRDF model is used along with a color texture. The greatest challenge here is the high frequency spatial variation of visibility due to small-scale geometric details. Our method gives a good approximation by exploiting the spatial coherence of apparent reflectance functions, while normal-map method produces brighter results due to the lack of support for subtle shadowing and masking effects. Note that we tile basic blocks in the cylinder (Fig. 1) and the wall scene to reduce repetitive precomputation.

The scene in Fig. 7 consists of gargoyles made of bumpy surfaces plus a highly complex procedural shader, which employs cellular texture [Wor96] to simulate Cloisonné. Inside each cell, a Lambertian BRDF model with a particular color is defined. Along the boundaries among cells, a high-frequency *gold-metallic-paint* BRDF from [MPBM03] is used. Normal-map based methods not only ignore shadowing and masking effects, but also have difficulty in handling such complex materials. For example, the size of the representations in [HSRG07] grows linearly with the number of different materials, which is very inefficient in our case where there are many different materials but the apparent reflectance functions are coherent. It is unknown how to extend [TLQ*08] to efficiently handle multiple materials. By contrast, our representation is not tied to any specific type of BRDFs and could faithfully filter arbitrarily complex materials, as shown in Fig. 7.

We show how CPMs adapt to the complexity of filtered reflectance functions in Fig. 8. On the left is a rendering of the original bolt, and on the right shows the characteristic point density distribution over a simplified mesh. As expected, our method allocates a relatively large number of characteristic points in the middle part of the bolt body, where the visibility and the normal change rapidly. And there are few characteristic points at other parts, where the variation in geometry is small and the reflectance functions only vary by a constant (a color fetched from the tainted metal texture). We can view our method as a precomputed object-space adaptive sampling for efficient rendering.

In Fig. 9 we show bolts rendered at a resolution where

Model	#Faces	Precomputation Time			CPM		Rendering Time	
		a_{vis}	Reflectance Sampling	CPs Computation	Size	Finest Resolution	Ground truth	Our method
Cylinder	3072	12.5min	59.2min	485.9min	2.2MB	32x32	62.4min	2.4min
Bolt	10496	57.4min	366.3min	824.7min	21.0MB	128x128	69.1min	10.1min
Wall	491164	86.5min	856.6min	913.1min	28.0MB	128x128	49.9min	14.5min
Gargoyle	200000	203.6min	1063.9min	3572.3min	79.1MB	256x256	191.4min	52.1min

Table 3: *Timing results and various statistics from our experiments.*

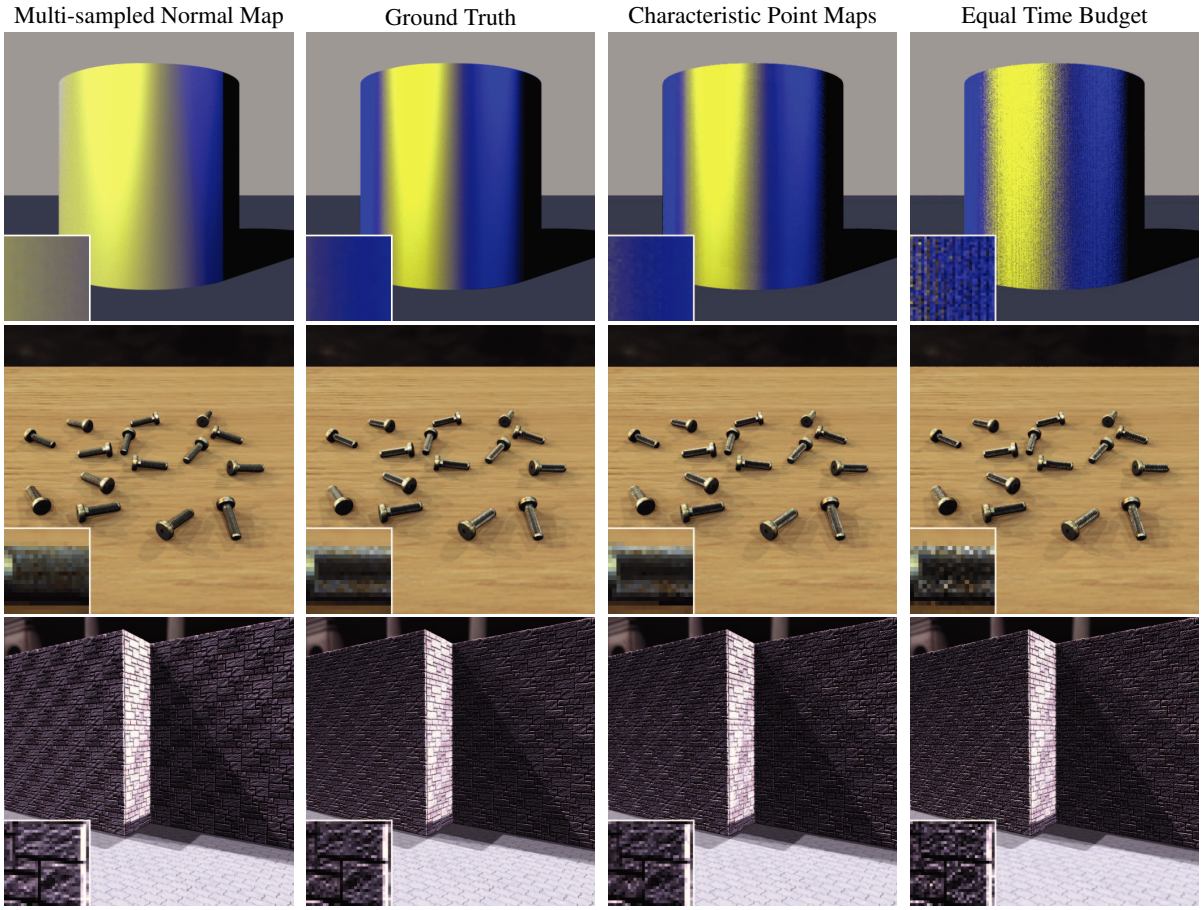

Figure 6: *Comparison of results using various methods. From left to right: multi-sampled normal-map renderings, ground truth renderings, renderings using CPMs and equal time budget renderings of original models. A magnification view is shown in the bottom left corner of each image.*

Figure 7: *Cloisonné gargoyles. From left to right: a close-up view of the micro structures of one gargoyle, a ground truth rendering, a rendering using CPMs and an equal time budget rendering of the original model.*

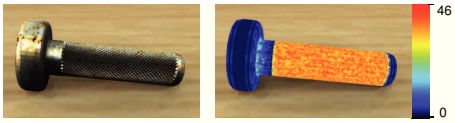

Figure 8: *Characteristic point density visualization. Left: a rendering of the original model. Right: characteristic point density distribution for one mip level in the CPM hierarchy. Using our method, the number of characteristic points automatically adapts to the complexity of the apparent reflectance functions.*

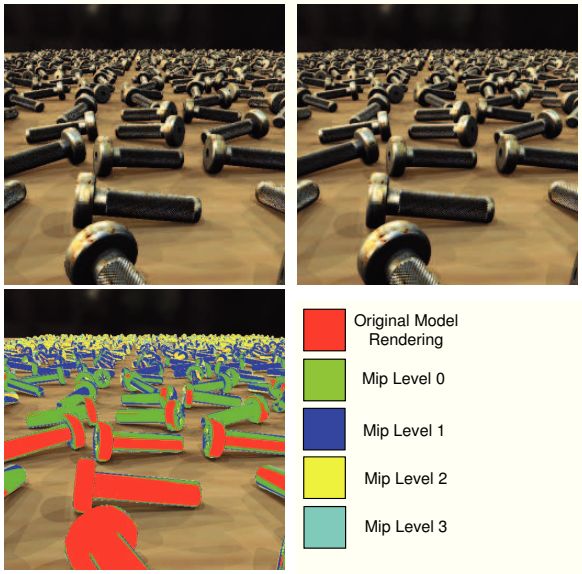

Figure 9: *An example that shows switching from full model to various levels of CPM representation. Top left: the image rendered with CPMs. Top right: the ground truth rendering of the original model. Bottom: the mip levels color coded.*

the representation used changes from the original model to various CPM levels. We use 4 eye rays per pixel for CPMs rendering and 64 eyes rays per pixel for the ground truth rendering.

6. Conclusions and Future Work

We have presented a general framework, Characteristic Point Maps, for efficiently computing and representing 6D spatially-varying average reflectance function for highly-detailed geometry along with complex BRDFs. Unlike existing reflectance filtering techniques, our method makes no assumption on the underlying geometry or BRDFs. We have demonstrated the ability of CPMs to accelerate the rendering process while maintaining image quality.

In future work, we would like to apply a low-pass filter in both filtered reflectance formulation and CPM mipmap generation to completely avoid aliasing. It would also be interesting to incorporate indirect illumination. In addition, applying our method to deformable objects would be useful future work.

Acknowledgements The authors would like to thank Sumanta Pattanaik, Li-Yi Wei and Ping Tan for useful discussions. This material is based upon work supported by the National Science Foundation under Grant No. 0528204.

References

[BM93] BECKER B. G., MAX N. L.: Smooth transitions between bump rendering algorithms. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM Press, pp. 183–190.

[CHPR07] COOK R. L., HALSTEAD J., PLANCK M., RYU D.: Stochastic simplification of aggregate detail. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 79.

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 115–122.

[DMM06] DRINEAS P., MAHONEY M. W., MUTHUKRISHNAN S.: *Polynomial Time Algorithm for Column-Row Based Relative-Error Low-Rank Matrix Approximation*. Tech. Rep. 2006-04, DIMACS, March 2006.

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (1999), 1–34.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 263–269.

[Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.

[HPB07] HAŠAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Trans. Graph. 26*, 3 (2007), 26.

[HSRG07] HAN C., SUN B., RAMAMOORTHI R., GRINSPUN E.: Frequency domain normal map filtering. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 28.

[Kaj85] KAJIYA J. T.: Anisotropic reflection models. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 15–21.

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1986), ACM Press, pp. 143–150.

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 117–126.

[LT00] LINDSTROM P., TURK G.: Image-driven simplification. *ACM Trans. Graph. 19*, 3 (2000), 204–241.

[MCT*05] MA W.-C., CHAO S.-H., TSENG Y.-T., CHUANG Y.-Y., CHANG C.-F., CHEN B.-Y., OUHYOUNG M.: Level-of-detail representation of bidirectional texture functions for real-time rendering. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM, pp. 187–194.

[MMS*04] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis and rendering of bidirectional texture functions. In *Eurographics 2004, State of the Art Reports* (Sept. 2004), Schlick C., Purgathofer W., (Eds.), pp. 69–94.

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Trans. Graph. 22*, 3 (2003), 759–769.

[SP04] SAINZ M., PAJAROLA R.: Point-based rendering techniques. *Computers & Graphics 28*, 6 (2004), 869 – 879.

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 409–416.

[TLQ*08] TAN P., LIN S., QUAN L., GUO B., SHUM H.: Filtering and rendering of resolution-dependent reflectance models. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 412–425.

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 255–264.

[Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH Comput. Graph. 17*, 3 (1983), 1–11.

[Wor96] WORLEY S.: A cellular texture basis function. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 291–294.