CGI2012 manuscript No.
(will be inserted by the editor)

Real-Time Rendering of Deformable Heterogeneous Translucent Objects using Multiresolution Splatting

Guojun Chen · Pieter Peers · Jiawan Zhang · Xin Tong

the date of receipt and acceptance should be inserted later

Abstract In this paper, we present a novel real-time rendering algorithm for heterogenous translucent objects with deformable geometry. The proposed method starts by rendering the surface geometry in two separate geometry buffers –the irradiance buffer and the splatting buffer– with corresponding mipmaps from the lighting and viewing directions respectively. Irradiance samples are selected from the irradiance buffer according to geometric and material properties using a novel and fast selection algorithm. Next, we gather the irradiance per visible surface point by splatting the irradiance samples to the splatting buffer. To compute the appearance of long-distance low-frequency subsurface scattering, as well as short-range detailed scattering, a fast novel multiresolution GPU algorithm is developed that computes everything on the fly and which does not require any precomputations. We illustrate the effectiveness of our method on several deformable geometries with measured heterogeneous translucent materials.

Keywords Translucency · Real-Time Rendering · Image-Space Splatting · Heterogeneous · Deformable

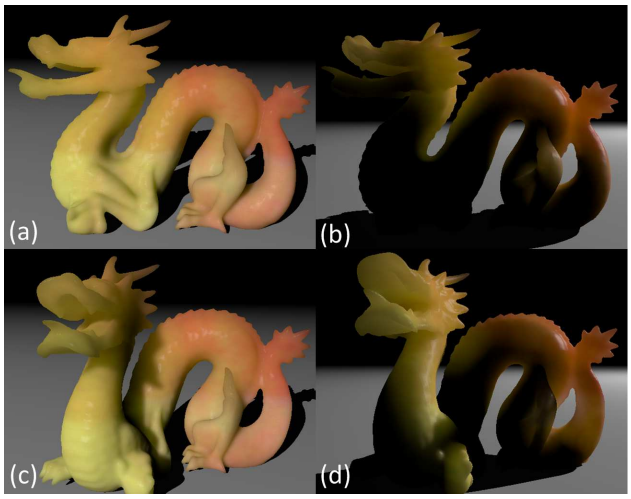

Fig. 1 Rendering of a deformable dragon geometry with measured heterogeneous translucent wax [18]: **(a), (b)** before deformation, and **(c), (d)** after deformation. Note how the visual impact of translucency varies with shape deformations.

1 Introduction

Subsurface light transport plays an important role in the complex appearance of many real world materials such as skin, milk, marble, etc. The impact of subsurface scattering on the appearance of translucent materials is striking, and the faithful visual reproduction is essential for producing photoreal CG imagery. The appearance of a heterogeneous translucent material is due to the complex light interactions induced by material variations inside the object volume and changes to the object geometry. Despite the enormous progress in rendering techniques for such heterogeneous translucent materials, real-time rendering of deformable objects with heterogeneous translucent materials is still a challenging problem.

Prior work on rendering translucent objects can be roughly categorized in volumetric, surface-based, and image-space methods. Volumetric methods [7, 11, 20, 23] directly simulate the light transport through the volume. Although some volumetric methods achieve in-

Guojun Chen, Jiawan Zhang
Tianjin Univerisity

Pieter Peers
College of William & Mary

Xin Tong
Microsoft Research Asia, Tianjin University

teractive rates [20,23], they require expensive precomputations that make these methods not suited for deformable objects. Surface-based methods model the subsurface scattering by the bidirectional subsurface scattering reflectance distribution function (BSSRDF) [15] over the object surface. The appearance is then the result of the convolution of incoming radiance with the BSSRDF. However, these methods either fix the geometry [8,18,21,22,24], or assume that the translucent material is homogeneous [1–3,6,10]. Recently, Shah et al. [17] proposed a real-time image-space method for rendering deformable translucent objects with homogeneous materials based on irradiance splatting and gathering. While efficient for homogeneous translucent materials, rendering speeds quickly degrade for heterogeneous mixtures of short and long scattering ranges. The first requires a high resolution irradiance buffer to capture all the lighting details, while the latter requires large splatting kernels.

In this paper we present a novel multiresolution image-space real-time rendering algorithm for deformable objects with optically thick, heterogeneous translucent materials. Similar to Shah et al. [17], we construct two geometry buffers of the object viewed from the camera and light source respectively. However, instead of using these geometry buffers directly, we create two mipmaps that we refer to as the irradiance buffer and the splatting buffer for the lighting and view directions respectively. Next, we apply a novel efficient scheme that selects irradiance samples from different mipmap levels from the irradiance buffer based on local geometrical and material variations. A multiresolution splatting algorithm is employed to accumulate the irradiance at different resolutions. The key observation is that subsurface scattering of light decreases exponentially with distance. Thus, the BSSRDF exhibits less variation for distant sample points. Consequently, a piecewise-linear approximation of the BSSRDF can use larger segments further away from the center. Based on this observation, we splat the irradiance samples in a multiresolution splatting buffer depending on surface distance. The final image is then the result of accumulating the splats across all scales.

In summary, the main technical contributions are:

1. an adaptive irradiance sampling scheme that automatically adjust the sampling rate in regions with different scattering ranges; and
2. a multi-resolution splatting algorithm specifically designed for efficient rendering of heterogeneous translucent materials.

As a result, the proposed method provides an efficient algorithm for real-time rendering of heterogeneous translucent objects with deformable geometry, such as the one shown in Figure 1.

2 Related Work

Prior work in real-time rendering of translucent materials can roughly be subdivided in three categories: volumetric methods, surface-based methods, and image-space methods. The proposed method is also an image-space method. We will therefore also review a related fourth category, namely image-space global illumination methods, that employs similar strategies as the ones employed in the proposed solution.

Volumetric Methods. Subsurface rendering methods in this category directly simulate the light transport inside the object volume with known material properties. Simulating the full light transport inside the volume is computationally very expensive. However, Stam [19] noted that for optically thick materials, the light transport inside the volume can be approximated by a diffusion process. Haber et al. [7] propose to solve the diffusion equation over a regular grid inside arbitrarily shaped objects with an embedded boundary discretization. Wang et al. [20] solve the diffusion equation over a 6-connected PolyGrid embedded in the object volume on the GPU. This was further improved by using a 4-connected QuadGraph [23]. However, these methods still require expensive precomputations to discretize the object volume, and are therefore not suited for real-time rendering of deformable translucent objects.

Surface-based Methods. A second category of rendering methods expresses the appearance of translucent objects as the convolution of the BSSRDF and the incoming radiance over the surface. Precomputed Radiance Transfer (PRT) methods [8,21,22,24] exploit the linearity of light transport, and precompute and recombine (based on the target illumination) the subsurface scattering effects of the fixed geometry object under a small set of basis lighting conditions. Due to the precomputations, these methods are not suited for deformable shapes. Mertens et al. [11] employ a hierarchical boundary element method for solving the light transport integral. However, their method is limited to interactive frame rates for moderately tessellated deformable objects. All of the above methods are restricted to homogeneous translucent materials. Recently, Song et al. [18] presented a surface-based method for heterogeneous translucent materials by adaptively sampling the irradiance and BSSRDF over the surface using an octree. However, this method is also restricted to rigid objects and only achieves interactive frame rates.

Image-space Methods. This category's methods compute the convolution between the BSSRDF and surface

irradiance in image space. Lensch et al. [10] decompose the BSSRDF as a local and a global response. The former is computed by texture filtering on the GPU, while the latter is approximated by integration of vertex-to-vertex throughput factors using the CPU. Dachsbacher and Stamminger [3] extended standard shadow mapping for translucent materials: contributions of irradiance samples are gathered for every pixel from a surface irradiance map rendered from the lighting direction. Chang et al. [2] perform importance sampling in texture space for translucent rendering. Recently, Shah et al. [17] proposed to employ a novel image-space splatting technique for real-time rendering of homogeneous translucent objects. All the above methods are specifically designed for the rendition of homogeneous translucent materials. The proposed method, also an image-space method, differs in that it employs an adaptive irradiance sampling and multiresolution splatting method which enables efficient rendering of heterogeneous translucent materials.

Image-space Global Illumination. The proposed method bares some similarity to image-space global illumination methods where indirect illumination is computed as the integral of the BRDF and the irradiance due to a set of Virtual Point Lights (VPLs). Dachsbacher et al. [4] employ reflective shadow maps to create VPLs to model single-bounce indirect lighting and gather the contributions via splatting [5]. Nichols et al. extend on this by clustering VPLs [12] and hierarchical splatting [13,14]. While the proposed method also employs similar hierarchical data structures, key differences are the sampling and splatting strategies used to accommodate for differences in the density of the incident radiance between both: VPLs are sparsely distributed through the scene and their contribution covers the whole screen; Conversely, in the case of heterogeneous subsurface scattering, the density of the irradiance samples is much higher, and their contribution covers a local screen region.

3 Background

Subsurface Scattering. The Bidirectional Subsurface Scattering Reflectance Distribution Function (BSSRDF), an 8D function, models subsurface light transport as the ratio of the outgoing radiance L with the incident flux Φ between two surface points: $\mathbf{x}_i$ the incident surface location, and $\mathbf{x}_o$ the exitant surface location, with ω_i and ω_o the incoming and outgoing light directions at the respective surface points:

$$S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) = \frac{dL(\mathbf{x}_o, \omega_o)}{d\Phi(\mathbf{x}_i, \omega_i)}. \tag{1}$$

Often, the BSSRDF is further partitioned into a single scattering term S_s and a multiple scattering term S_m:

$$S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) = S_s(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) + S_m(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o). \tag{2}$$

For highly scattering, optically thick materials, the BSSRDF is dominated by the multiple scattering component, which can be further decomposed as [9]:

$$S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) \approx S_m(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) = \frac{1}{\pi} F_t(\mathbf{x}_i, \omega_i) R_d(\mathbf{x}_i, \mathbf{x}_o) F_t(\mathbf{x}_o, \omega_o), \tag{3}$$

where F_t is the Fresnel transmittance term, and R_d is the diffuse scattering function, a 4D function that encodes the light transport between pairs of surface points due to subsurface scattering. Note that, while for homogeneous materials R_d can be reduced to a 1D profile by writing it as a function of the distance between $\mathbf{x}_i$ and $\mathbf{x}_o$, for heterogeneous no such reduction is possible.

BSSRDF Evaluation. The outgoing radiance due to subsurface scattering at a surface point $\mathbf{x}_o$ can be expressed by:

$$L_o(\mathbf{x}_o, \omega_o) = \int_A \int_\Omega S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) L_i(\mathbf{x}_i, \omega_i) |\mathbf{n}(\mathbf{x}_i) \cdot \omega_i| \mathrm{d}\omega_i \mathrm{d}A(\mathbf{x}_i), \tag{4}$$

where A is the object surface, Ω is the hemisphere of incident directions around $\mathbf{x}_i$, and $\mathbf{n}(\mathbf{x}_i)$ the surface normal at $\mathbf{x}_i$. Note that this equation requires us to compute an integral over the object surface. In order to make this more amendable for real-time computations on graphics hardware, we approximate Equation (4) by an evaluation in image space as opposed to the object surface. First, the radiant exitance $B(\mathbf{x}_o)$ is evaluated for every outgoing pixel $\mathbf{x}_o$:

$$B(\mathbf{x}_o) = \sum_{\mathbf{x}_i} R_d(\mathbf{x}_i, \mathbf{x}_o) E(\mathbf{x}_i) \Delta A_{\mathbf{x}_i}, \tag{5}$$

where $\mathbf{x}_i$ is the set of all surface pixels visible to the light source, $\Delta A_{\mathbf{x}_i}$ is the surface area covered by $\mathbf{x}_i$. The total irradiance $E(\mathbf{x}_i)$ received at $\mathbf{x}_i$ is computed by:

$$E(\mathbf{x}_i) = \int_\Omega F_t(\omega_i) L_i(\omega_i) |\mathbf{n}(\mathbf{x}_i) \cdot \omega_i| \mathrm{d}\omega_i. \tag{6}$$

The final outgoing radiance for a given view direction ω_o is then:

$$L(\mathbf{x}_o, \omega_o) = \frac{F_t(\omega_o)}{\pi} B(\mathbf{x}_o). \tag{7}$$

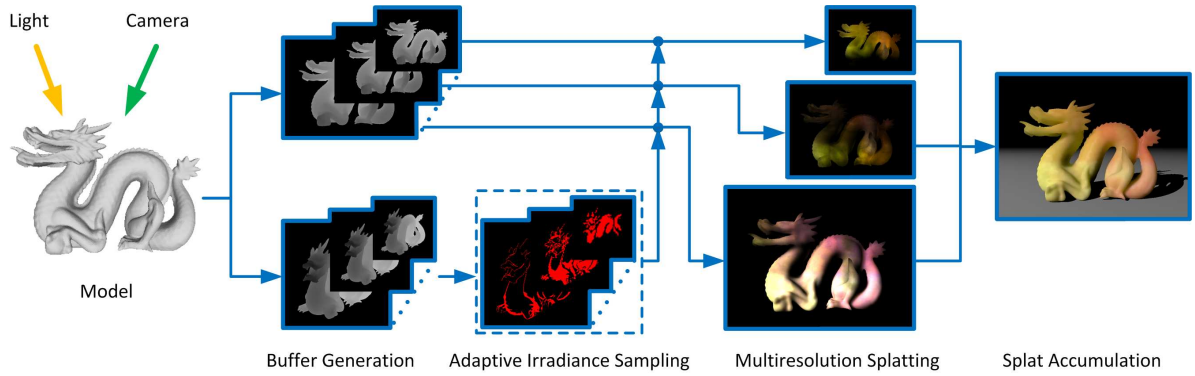

Fig. 2 Algorithm overview. Given a view direction V and lighting direction L, our algorithm renders the results in four steps: **(1)** buffer generation, **(2)** adaptive irradiance sampling, **(3)** multiresolution splatting, and **(4)** splat accumulation.

Compact Representation. Directly storing the full 4D diffuse scattering function R_d for heterogeneous translucent materials requires significant storage, and is only practical for low spatial resolutions. We therefore employ the SubEdit representation [18], which factors the diffuse scattering function R_d into a product of two local scattering profiles $P_{\mathbf{x}}(r)$ defined as a radial function around a surface point $\mathbf{x}$:

$$R_d(\mathbf{x}_i, \mathbf{x}_o) = \sqrt{P_{\mathbf{x}_i}(r) P_{\mathbf{x}_o}(r)}, \tag{8}$$

where r is the distance between the incident and exitant surface points $\|\mathbf{x}_i - \mathbf{x}_o\|$. The scattering profile $P_{\mathbf{x}}(r)$ defined at each surface point $\mathbf{x}$ is represented as a piecewise linear function with n segments:

$$\ln P_{\mathbf{x}}(r) = \hat{P}_{\mathbf{x}}(r) = (1 - w_x^k)\hat{P}_x^k + w_x^k \hat{P}_x^{k+1} \tag{9}$$

for $kr_s < rn < (k+1)r_s$, where r_s is the maximum scattering radius and $\hat{P}_x^k$ is the value of the scattering profile at $r_k = kr_s/n$. $w_x = nr/r_s - k$ is the linear interpolation weight.

4 Rendering Algorithm

Our algorithm takes as input an object geometry and a BSSRDF (in the SubEdit [18] representation) mapped over the surface. Given a view direction V and lighting direction L, we then generate a rendition in four steps (see also Figure 2):

1. **Buffer Generation:** A hierarchical irradiance and a splatting buffer are computed for the lighting and view directions respectively;
2. **Adaptive Irradiance Sampling Selection:** Irradiance samples are selected from the hierarchical irradiance buffer according to material properties and geometrical variations in a local region around the point of interest;
3. **Multiresolution Splatting:** The selected samples are then splat at different levels into the splatting buffer; and
4. **Splat Accumulation:** The splatted irradiance samples are gathered across the different scales to produce the final result.

Buffer Generation. To construct the *irradiance buffer*, the object is rendered from the lighting direction, and the resulting depth, normals and texture coordinates are stored in a texture (512×512 resolution in our implementation). Next, we generate an n-level mipmap (i.e., the irradiance buffer) from the resulting texture using the standard hardware mipmap construction function. Level 0 refers to the finest resolution, and level $n-1$ refers to the coarsest resolution. To efficiently detect geometrical variations (needed in step 2 in Figure 2), we also construct an additional mipmap, similar to Nichols et al. [12], to record the minimum and maximum depth values covered by each pixel in the hierarchical irradiance buffer. We initialize the minimum and maximum depth value at the finest level with the corresponding depth value at the finest irradiance level. The minimum/maximum depth value at level $k+1$, is then the minimum/maximum depth value of the four children at level k.

To construct the *splatting buffer*, the object is rendered from the normal view direction and stored in a texture of the same resolution as the frame buffer (i.e., 800×600 in our implementation). Again, a mipmap is created to form the hierarchical splatting buffer. However, instead of the standard mipmap construction functions, we employ a median filter to preserve sharp geometrical features. In particular, we pick the texel with median depth in every 2×2 neighborhood, and copy the corresponding normal, texture coordinates and depth.

Adaptive Irradiance Sampling Selection. To adaptively select irradiance samples, a coarse to fine scan is performed on the hierarchical irradiance buffer, and

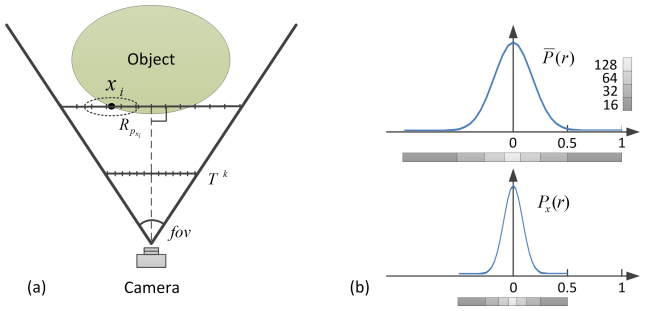

Fig. 3 Determining the effective rendering range. **(a)** Definition of $T(x_i)$, which accounts for the projection of the scattering profile from visible surface points to splatting buffer pixel resolution T^k. **(b)** The minimal effective rendering ranges of two scattering profiles for different sampling resolutions. The top plot shows the mean scattering profile, while the bottom plot shows a scattering profile at $\mathbf{x}$.

selection is based on material properties, geometrical discontinuities, and surface orientation. If a sample is marked valid at a specific level, all descendants at finer levels are discarded for selection. In particular, a sample $\mathbf{x}$ is a valid irradiance sample if it does not include sharp geometrical discontinuities and the texture sampling rate exceeds the required sampling rate for the material at $\mathbf{x}$:

$$z_{max}(\mathbf{x}) - z_{min}(\mathbf{x}) < z_0 \text{ and } T > M(\mathbf{x}), \tag{10}$$

where $z_{max}(\mathbf{x})$ and $z_{min}(\mathbf{x})$ are the maximum and minimum depth values respectively, z_0 is a user specified threshold (set to 0.03 times the scene's bounding sphere radius in our implementation), and T is the texture resolution at the sample's current level. The required material sampling rate $M(\mathbf{x})$ is defined by:

$$M(\mathbf{x}) = \frac{\alpha R_w}{R_{P_\mathbf{x}} |\mathbf{n}(\mathbf{x}) \cdot L(\mathbf{x})|}, \tag{11}$$

where $R_{P_\mathbf{x}}$ is the effective scattering range of the scattering profile $P_\mathbf{x}$ at $\mathbf{x}$, which is defined as the radius that preserves 95% percent of the energy of the full scattering profile. R_w is the diameter of object bounding sphere. R_w togeter with α, a user-specified scale factor (set to 15 in our implementation), relates the scattering range scale to irradiance buffer pixel size. Finally, the term $|\mathbf{n}(\mathbf{x}) \cdot L(\mathbf{x})|$ accounts for surface orientation, and relates local surface area to pixel size.

Multiresolution Splatting. The subsurface scattering contributions of the selected irradiance samples to visible surface points are computed by a novel multiresolution splatting method. A key observation is that the magnitude of subsurface light transport decreases exponentially with distance. Consequently, the resulting absolute radiance variation due to a single irradiance sample also decreases with distance. We therefore approximate the contributions of an irradiance sample at different scales depending on the distance between the incident and exitant surface points. Practically, we splat the irradiance samples in a high-resolution splatting buffer for rendering the subsurface contributions of nearby surface points, and employ lower-resolution splatting buffers for rendering more distant surface points. This reduces the total computational cost, especially for BSSRDFs with a large scattering range.

More formally, we accumulate and splat irradiance samples $\mathbf{x}_i$ to the splatting buffer $B^k(\mathbf{x}_o)$ for exitant pixel $\mathbf{x}_o$ at splatting level k as follows. If the distance d between the corresponding surface point of $\mathbf{x}_o$ and the irradiance sample $\mathbf{x}_i$ falls within the effective rendering range $[R^k_{min}(\mathbf{x}_i), R^k_{max}(\mathbf{x}_i)]$, then we update the splatting buffer as:

$$B^k(\mathbf{x}_o) \mathrel{+}= \sqrt{P_{\mathbf{x}_o}(d) P_{\mathbf{x}_i}(d)} E(\mathbf{x}_i) \Delta A_{\mathbf{x}_i}, \tag{12}$$

where, $P_{\mathbf{x}_o}$ and $P_{\mathbf{x}_i}$ are the SubEdit scattering profiles [18], $E(\mathbf{x}_i)$ is the irradiance of sample $\mathbf{x}_i$, and $\Delta A_{\mathbf{x}_i}$ is the surface area covered by $\mathbf{x}_i$.

The effective rendering range $[R^k_{min}(\mathbf{x}_i), R^k_{max}(\mathbf{x}_i)]$ is computed by:

$$\begin{aligned} R^k_{min}(\mathbf{x}_i) &= R_{P_{\mathbf{x}_i}} \bar{R}(\mathbf{x}_i), \\ R^k_{max}(\mathbf{x}_i) &= R^{k+1}_{min}(\mathbf{x}_i), \end{aligned} \tag{13}$$

with $R^0_{min}(\mathbf{x}_i) = 0$, and $R^{n-1}_{max}(\mathbf{x}_i) = R_{P_{\mathbf{x}_i}}$ the effective scattering range. $\bar{R}(\mathbf{x}_i)$ is the minimal effective rendering radius:

$$\bar{R}(\mathbf{x}_i) = \{r | (\frac{1}{T(\mathbf{x}_i)} \sum_{\hat{r}=r}^{1} \bar{P}(\hat{r}) \hat{r} - \int_r^1 \bar{P}(r) r dr) < \varepsilon\}, \tag{14}$$

where:

- ε is a user-set error-tolerance threshold, empirically set to 0.001 in our implementation.
- $\bar{P}(r)$ is the mean scattering profile defined on the normalized scattering range $[0, 1]$ and computed by $\bar{P}(r) = \frac{1}{N_{\mathbf{x}_i}} \sum_{\mathbf{x}_i} P'_{\mathbf{x}_i}(r)$, where $N_{\mathbf{x}_i}$ is the total number of scattering profiles. The normalized scattering profile $P'_{\mathbf{x}_i}(r)$ at x_i is defined as:

$$P'_{\mathbf{x}_i}(r) = C(\mathbf{x}_i) P_{\mathbf{x}_i}(r R_{P_{\mathbf{x}_i}}), \tag{15}$$

  with the normalization constant:

$$C(\mathbf{x}_i) = \frac{R^2_{P_{\mathbf{x}_i}}}{\int_0^{R_{P_{\mathbf{x}_i}}} P_{\mathbf{x}_i}(r) r dr}. \tag{16}$$

- $T(\mathbf{x}_i)$ accounts for the projection of the scattering profile from visible surface points to splatting buffer pixel resolution T^k, and which is defined as: $T(\mathbf{x}_i) = \frac{T^k R_{P_{\mathbf{x}_i}}}{2 \tan(fov/2) z(\mathbf{x}_i)}$, with $z(\mathbf{x}_i)$ the depth at $\mathbf{x}_i$.
- $\hat{r} = \{r' | r' = r + \frac{m}{T(\mathbf{x}_i)},\ m = 0, 1, 2, \ldots, \lfloor (1-r) T(\mathbf{x}_i) \rfloor\}$, which uniformly samples the range $[r, 1]$.

We precompute the mean scattering profile $\bar{P}(r)$ and the minimal effective rendering radius $\bar{R}$ for different splatting buffer resolutions T and store the results in a texture lookup table. During rendering, we employ this table for computing the effective rendering range of each x_i at run time. Figure 3(b) shows the minimal effective rendering range precomputed for $\bar{P}(r)$ compared to a scattering profile with the effective scattering range $R_{P_{\mathbf{x}_i}} = 0.5$.

Splat Accumulation. The splatting buffers $B^k(\mathbf{x}_o)$ contain the exitant radiance at pixel $\mathbf{x}_o$ due the subsurface scattering at different non-overlapping scattering ranges. These splatting buffers need to be combined to obtain the final rendered image. Starting from the coarsest level and ending at the finest level, the splatting buffer B^{k+1} is upsampled to match the resolution at level k and is subsequently accumulated with B^k. However, naively upsampling consecutive levels can introduce visual artifacts, because neighboring pixels in the image may originate from different parts on the object. We employ a bilateral upsampling scheme to take geometrical and material cues in account:

$$B^k(\mathbf{x}_o) = \frac{\sum_{\mathbf{x}_j \in N^{k+1}(\mathbf{x}_o)} w^{k+1}(\mathbf{x}_o, \mathbf{x}_j) B^{k+1}(\mathbf{x}_j)}{\sum_{\mathbf{x}_j \in N^{k+1}(\mathbf{x}_o)} w^{k+1}(\mathbf{x}_o, \mathbf{x}_j)}, \quad (17)$$

where $N^{k+1}(\mathbf{x}_o)$ is the set of four pixels nearest to $\mathbf{x}_o$ at the level $k+1$. The weighting function $w^{k+1}(\mathbf{x}_o, \mathbf{x}_j) = w_p^{k+1}(\mathbf{x}_o, \mathbf{x}_j) w_z^{k+1}(\mathbf{x}_o, \mathbf{x}_j) w_m^{k+1}(\mathbf{x}_o, \mathbf{x}_j)$ is the product of three factors:

1. A standard bilinear interpolation weight w_p^{k+1};
2. A geometry interpolation weight w_z^{k+1} that accounts for sharp geometrical discontinuities and which is defined as: $w_z^{k+1}(\mathbf{x}_o, \mathbf{x}_j) = e^{-\lambda_z \frac{|z(\mathbf{x}_o) - z(\mathbf{x}_j)|}{R_w}}$, where $\lambda_z = 8.0$ is the sharpness factor; and
3. A material interpolation weight w_m^{k+1} that accounts for sharp discontinuities in the material, and which is defined as: $w_m^{k+1}(\mathbf{x}_o, \mathbf{x}_j) = e^{-\lambda_m |R_{P_{\mathbf{x}_o}} - R_{P_{\mathbf{x}_j}}|}$, with $\lambda_m = 8.0$.

The final rendering result is obtained by computing the outgoing radiance from the upsampled radiant exitance for a given view direction V using Equation (7). Specular highlights are added based on the geometrical information stored in the splatting buffer.

Hardware Implementation Details.

The algorithm proposed in the previous sections is well suited for computation on the GPU with multiple rendering passes. After construction of the hierarchical irradiance and splatting buffer, a vertex array is created in which each vertex corresponds to a sample in the hierarchical irradiance buffer which in turn is stored as a texture. For each splatting buffer at all levels, we render the vertex array with the adaptive sampling implemented in vertex shaders. In particular, we follow the strategy of Nichols et al. [12] to check the samples at all levels simultaneously (see also Algorithm 1). Next, the effective rendering range is computed in a vertex shader for all valid irradiance samples (i.e., not rejected in the prior step) as well as its corresponding splatting kernel in a geometry shader. After rasterization, we compute the contribution of each splatting kernel in a pixel shader and store the result in the radiance exitance buffer. After the radiance exitance buffer at all resolutions are generated, a final rendering pass is performed to accumulate the radiance at the exitance buffer at different scales using the bilateral upsampling scheme in a pixel shader. Note that to improve efficiency of the GPU implementation, the adaptive sampling scheme is executed several times in order to generate splatting kernels for each scale.

Algorithm 1 Adaptive Sample Selection

Input: A sample $\mathbf{x}_i^k$ in the hierarchical irradiance buffer at level k.
Output: Valid irradiance samples for splatting.
if $z_{max}(\mathbf{x}_i^k) - z_{min}(\mathbf{x}_i^k) < z_0 \parallel T^k > M(\mathbf{x}_i^k)$ **then**
  Discard.
end if
if $z_{max}(\mathbf{x}_i^k) - z_{min}(\mathbf{x}_i^k) \geq z_0 \parallel T^{k+1} > M(\mathbf{x}_i^{k+1})$ **then**
  Discard.
end if
Mark $\mathbf{x}_i$ as valid.

5 Discussion and Results

Performance. We implemented our algorithm on a PC with an Intel Xeon 5080 3.73 GHz CPU and an AMD Radeon HD5870 graphics card with 1 GB of graphics memory. Table 1 summaries the relevant statistics and rendering performance of all scenes shown in this paper. All results are rendered in real-time and at a resolution of 800×600, and with a 512×512 irradiance buffer at the finest scale. We employ a three-level mipmap for both the irradiance buffer and the splatting buffer.

The scattering ranges of the scenes summarized in Table 1 vary significantly and in general contain a large maximum range. Without adaptive sampling and without multiresolution splatting (6th column), a large splatting kernel and high sampling rate is required and a significant performance hit is incurred. Multiresolution splatting without adaptive sampling (7th column) and vice versa (8th column) yields a significant speedup. Enabling both multiresolution splatting and adaptive sampling (9th column) further increases the overall performance.

Object + Material	Figure	Mesh resolution (triangles)	Material resolution (texels)	Scattering range (bounding sphere radius)	No accel. (fps)	Multi-res. (fps)	Adapt. samp. (fps)	Full accel. (fps)
Dragon + Yellow Wax	Fig. 1	560K	110×110	$0.12 \sim 0.3$	2.8	10	14	31
Buddha + Art Wax	Fig. 7	100K	145×275	$0.04 \sim 0.15$	14	22	48	57
Bunny + Marble	Fig. 8	70K	277×277	$0.06 \sim 0.25$	5.6	14	36	66
Bird + Artificial Stone	Fig. 9	54K	224×214	$0.15 \sim 0.30$	2.0	16	10	60
Parent + Jade	Fig. 10	30K	504×504	$0.10 \sim 0.20$	5.9	12	30	49
Elephant + Blue Wax	Fig. 11	20K	212×138	$0.08 \sim 0.35$	3.7	17	19	53

Table 1 Statistics and rendering performance for the results shown in this paper. Note that the lower performance for the dragon scene is due to the overhead of deformation computations.

Object	Buffer Gen.	Multires. Splatting			Acc. Gath.
		Level 0	Level 1	Level 2	
Dragon	42.3%	16.9%	23.7%	15.8%	1.3%
Buddha	7.6%	40.0%	39.0%	11.8%	1.6%
Bunny	8.1%	39.7%	35.0%	15.2%	2.0%
Bird	5.3%	13.8%	12.5%	66.4%	2.0%
Parent	8.1%	24.0%	51.2%	14.5%	2.2%
Elephant	5.9%	33.2%	26.8%	31.7%	2.4%

Table 2 The relative performance of each the components. The relative timings of the multiresolution splatting step are further subdivided by multiresolution level.

The performance gain due to multiresolution splatting varies significantly amongst different scenes. Table 2 further details the relative performance variations for the different levels in the multiresolution splatting scheme. For example, for the *bird* scene, level 2 in the multiresolution splatting accounts for 66% of the time due to the smooth nature of the scattering profiles which can be well approximated by a coarse sampling and thus most energy is splat in the lowest resolution buffer. Also note that in such a case the multiresolution approach achieves a 6× speed up (Table 1 col. 8 versus col. 9). In contrast, for the *Buddha* and *bunny* scene, most energy is splat in levels 0 and 1, and consequently, only a modest speed (2×) is obtained.

Comparisons. Figure 4 compares the visual accuracy of proposed method with ground truth visualizations of several scenes. As can be seen, the obtained results are visually indistinguishable from the ground truth.

The method of Shah et al. [17] is most similar to ours. We will discuss some of the key differences that are crucial for faithfully reproducing the appearance of heterogeneous translucent materials.

Similar to [17], the required sampling rate employed in the adaptive irradiance sampling scheme is inversely proportional to the effective scattering range of the material. However, different from the sampling scheme in [17], which assumes a fixed effective scattering range, our adaptive sampling method considers spatial variations in scattering range, as well as geometric variations (i.e., discontinuity and orientation). The impact of the two geometrical terms on the final appearance is illustrated in Figure 5. Insufficient irradiance samples around sharp geometrical variations can lead to incorrect shading variations, especially when ignoring geometrical discontinuities. Furthermore, undersampling of surface regions orthogonal to the lighting direction can result in ringing artifacts in the rendered results.

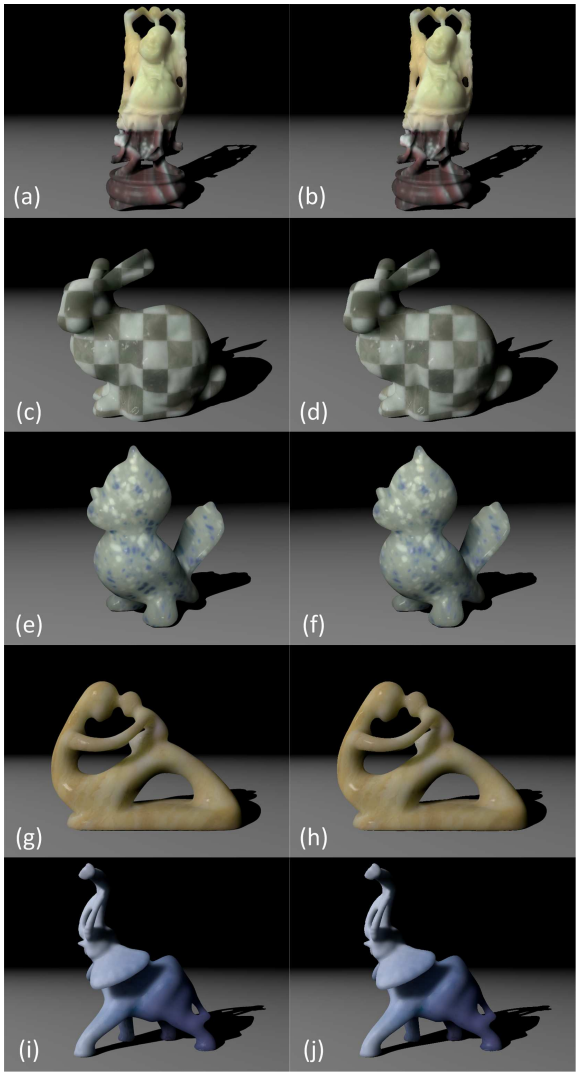

Fig. 4 A comparison of the visual accuracy of the ground truth results **(left)** to results obtained with the proposed method **(right)**.

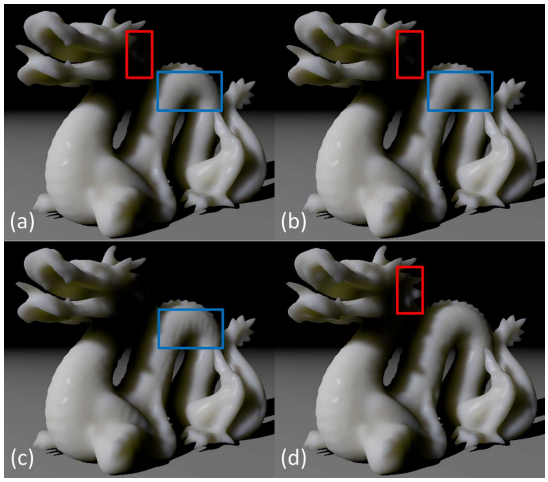

Fig. 5 A comparison of the impact of the two geometrical terms (i.e., discontinuity and orientation) in the adaptive sampling scheme. **(a)** Ground truth. **(b)** Result with both discontinuity and orientation taken in account. **(c)** Artifacts (highlighted in the blue box) due to sampling based on geometry discontinuities only. **(d)** Artifacts (highlighting in the red box) due to sampling based on orientation only.

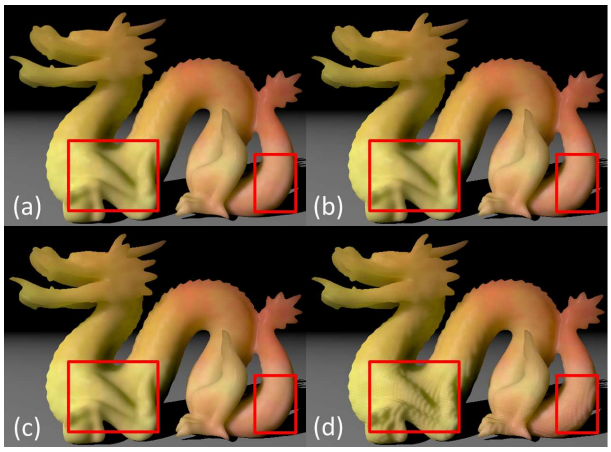

Fig. 6 Comparison of our method with the image-space algorithm of Shah et al. [17]: **(a)** Ground truth, **(b)** Our method (40 FPS), **(c)** The method of Shah et al. [17] with similar visual quality (3FPS). **(d)** The method of Shah et al. [17] at a comparible rendering speed as ours. However, at this rate the result exhibits significant visual artifacts.

By taking both properties in account, our method is able to generate convincing renditions of heterogeneous translucent objects in real-time.

A qualitative comparison of the method of Shah et al. [17] extended to heterogenous translucent materials and the presented technique is shown in Figure 6. Our method is able to attain real-time rendering speeds, while maintaining visual accuracy compared to the ground truth. The method of Shah et al. [17], however, suffers either from degraded quality when maintaining rendering speed, or from degraded performance when maintaining visual quality.

Additional Results. Figure 7 shows renditions of an artist-modeled heterogeneous wax BSSRDF [16] mapped onto the well-known Buddha model. Despite the complex geometry of the Buddha model, and the large scattering range variations in the wax material, our method is able to generate convincing results in real-time for different lighting and viewing conditions.

Figure 8 displays the Stanford Bunny model with a measured marble chessboard material [16]. Note that the presented technique is able to preserve sharp features and the subtle detail variations of the material in the rendered image. Figure 9 shows the bird model with a measured artificial stone material [18].

It is straightforward to use our method on deformable shapes and temporally varying translucent materials, since our method does not require any precomputations and employs an image-space irradiance gathering scheme. Figure 10 illustrates the case of temporally varying material properties, where the material on the sculpture model is changed from measured jade [18] to another artist modeled material. Our rendering algorithm is able to dynamically handle changes in scattering range between the two materials in real-time. Figure 11 illustrates the case of deformable geometry, where a measured blue wax material [18] is mapped onto the elephant model, and where the shape of the trunk is changed in real-time. Please see the supplemental video for additional results.

Limitations. As with other image-space rendering algorithms, irradiance samples selected at one frame may be different from the ones selected in a subsequent frame when the lighting is changing. Theoretically, this can lead to temporal flickering, especially for objects with complex geometry. In our experiments, temporal artifacts are almost invisible, which we partially ascribe to the high resolution of the irradiance buffer and the novel adaptive sampling scheme.

6 Conclusion

In this paper, we present a real-time algorithm for rendering deformable, heterogeneous translucent objects. Key to our algorithm is the use of two *hierarchical geometry buffers*, the irradiance buffer and the splatting buffer, for computing subsurface scattering light transport in image space. Irradiance samples are adaptively selected, filtered and rendered to the resulting image using a novel hierarchal splatting algorithm. Our method does not require any precomputations, hence it is ideally suited for rendering translucent objects with deformable shapes and dynamic material properties in real-time.

Our algorithm currently only supports point and directional light sources. A possible strategy for adding environmental lighting support would be to use depth peeling when computing the hierarchical irradiance buffer [17]. The main challenge would be to compute the irradiance at each surface point of a freely deformable geometry. Furthermore, we would also like to investigate combinations of our algorithm with other image-space global illumination methods for rendering a full global illumination solution, in real-time, of dynamic scenes including both opaque and translucent objects.

References

1. Nathan A. Carr, Jesse D. Hall, and John C. Hart. Gpu algorithms for radiosity and subsurface scattering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '03, pages 51–59, 2003.
2. Chih-Wen Chang, Wen-Chieh Lin, Tan-Chi Ho, Tsung-Shian Huang, and Jung-Hong Chuang. Real-time translucent rendering using gpu-based texture space importance sampling. *Computer Graphics Forum*, 27(2):517–526, 2008.
3. Carsten Dachsbacher and Marc Stamminger. Translucent shadow maps. In *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03, pages 197–201. Eurographics Association, 2003.
4. Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 203–231, New York, NY, USA, 2005. ACM.
5. Carsten Dachsbacher and Marc Stamminger. Splatting indirect illumination. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, I3D '06, pages 93–100. ACM, 2006.
6. Eugene d'Eon, David P. Luebke, and Eric Enderton. Efficient rendering of human skin. In *Rendering Techniques*, pages 147–157, 2007.
7. Tom Haber, Tom Mertens, Philippe Bekaert, and Frank Van Reeth. A computational approach to simulate subsurface light diffusion in arbitrarily shaped objects. In *Proceedings of Graphics Interface 2005*, GI '05, pages 79–86, 2005.
8. Xuejun Hao and Amitabh Varshney. Real-time rendering of translucent meshes. *ACM Trans. Graph.*, 23:120–142, April 2004.
9. Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 511–518. ACM, 2001.
10. Hendrik P. A. Lensch, Michael Goesele, Philippe Bekaert, Jan Kautz, Marcus A. Magnor, Jochen Lang, and Hans peter Seidel. Interactive rendering of translucent objects. In *In Proceedings of Pacific Graphics 2002*, pages 214–224, 2002.
11. Tom Mertens, Jan Kautz, Philippe Bekaert, Hans-Peter Seidelz, and Frank Van Reeth. Interactive rendering of translucent deformable objects. In *Proceedings of the 14th Eurographics workshop on Rendering*, EGRW '03, pages 130–140. Eurographics Association, 2003.
12. Greg Nichols, Jeremy Shopf, and Chris Wyman. Hierarchical image-space radiosity for interactive global illumination. *Comput. Graph. Forum*, 28(4):1141–1149, 2009.
13. Greg Nichols and Chris Wyman. Multiresolution splatting for indirect illumination. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 83–90. ACM, 2009.
14. Greg Nichols and Chris Wyman. Interactive indirect illumination using adaptive multiresolution splatting. *IEEE Trans. Vis. Comput. Graph.*, 16(5):729–741, 2010.
15. F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Radiometry. chapter Geometrical considerations and nomenclature for reflectance, pages 94–145. Jones and Bartlett Publishers, Inc., , USA, 1992.
16. Pieter Peers, Karl vom Berge, Wojciech Matusik, Ravi Ramamoorthi, Jason Lawrence, Szymon Rusinkiewicz, and Philip Dutré. A compact factored representation of heterogeneous subsurface scattering. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 746–753. ACM, 2006.
17. Musawir A. Shah, Jaakko Konttinen, and Sumanta Pattanaik. Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE Computer Graphics and Applications*, 29:66–78, 2009.
18. Ying Song, Xin Tong, Fabio Pellacini, and Pieter Peers. Subedit: a representation for editing measured heterogeneous subsurface scattering. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 31:1–31:10. ACM, 2009.
19. Jos Stam. Multiple scattering as a diffusion process. In *In Eurographics Rendering Workshop*, pages 41–50, 1995.
20. Jiaping Wang, Shuang Zhao, Xin Tong, Stephen Lin, Zhouchen Lin, Yue Dong, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph.*, 27:9:1–9:18, March 2008.
21. Rui Wang, Ewen Cheslack-Postava, Rui Wang, David P. Luebke, Qianyong Chen, Wei Hua, Qunsheng Peng, and Hujun Bao. Real-time editing and relighting of homogeneous translucent materials. *The Visual Computer*, 24(7-9):565–575, 2008.
22. Rui Wang, John Tran, and David Luebke. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM Trans. Graph.*, 24:1202–1207, July 2005.
23. Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes. *Computer Graphics Forum*, 29:497–506, 2010.
24. Kun Xu, Yue Gao, Yong Li, Tao Ju, and Shi-Min Hu. Real-time homogenous translucent material editing. *Computer Graphics Forum*, 26(3):545–552, 2007.

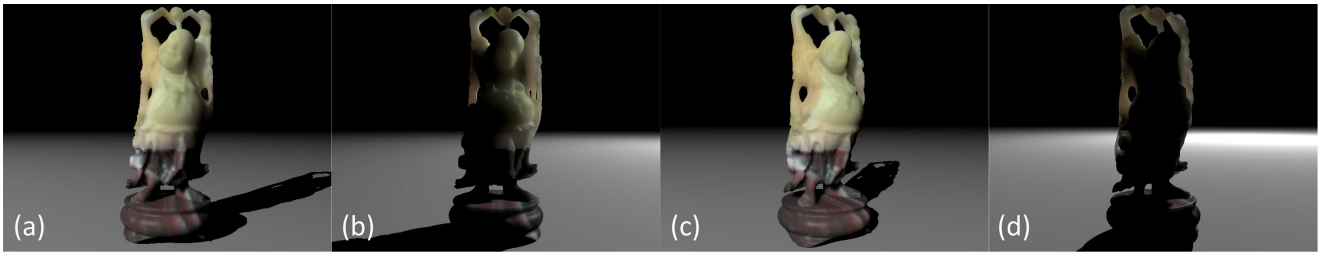

Fig. 7 An artist-modeled heterogeneous wax material [18] applied to the Buddha model.

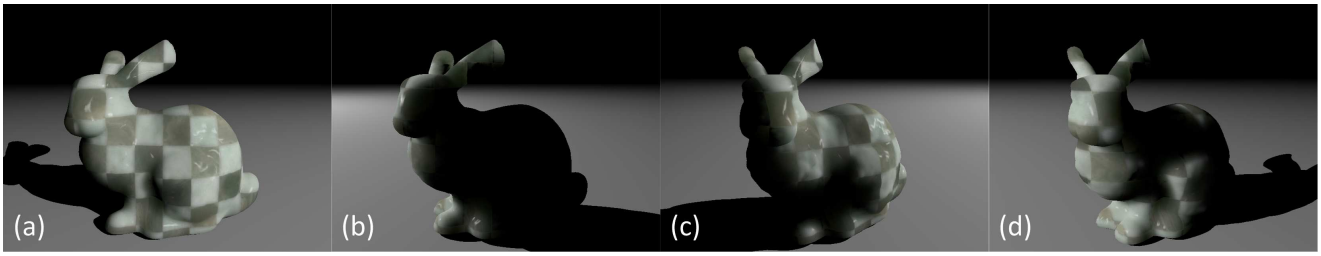

Fig. 8 Measured marble BSSRDF [16] mapped on the Stanford Bunny model.

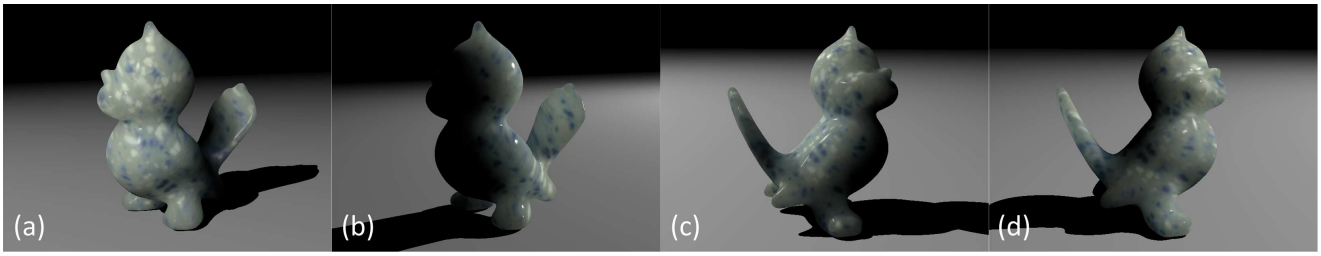

Fig. 9 A measured artificial stone material [18] applied to the bird model.

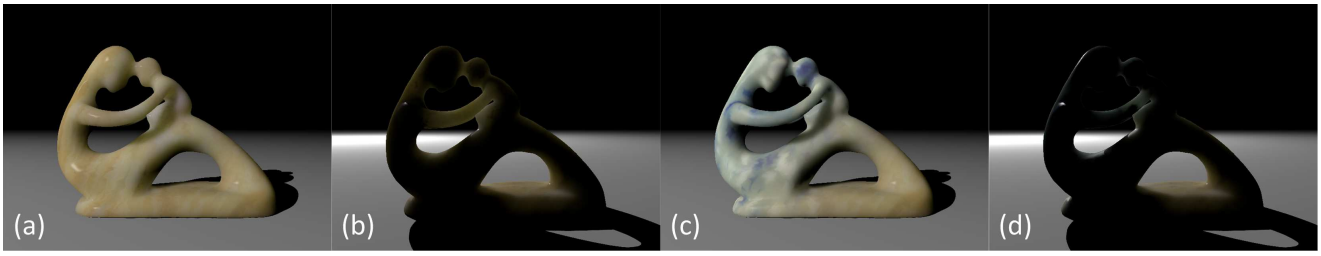

Fig. 10 A real-time user-altered subsurface scattering material mapped onto the sculpture model. **(a),(b)** measured marble [18] visualized from different lighting directions. **(c),(d)** the same model, but with a novel user-edited heterogeneous translucent material. All appearance effects are faithfully reproduced in real-time.

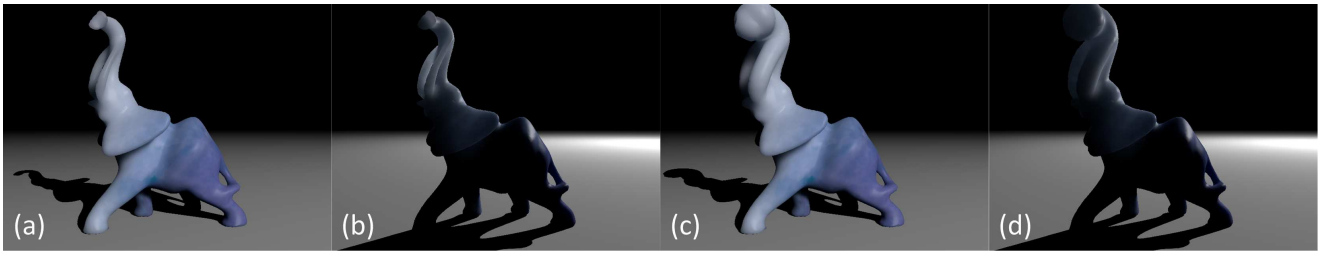

Fig. 11 Measured blue wax [18] mapped onto a deformable elephant model (**(a),(b)** before deformation, **(c),(d)** after deformation of the trunk. Note how the appearance due to subsurface scattering changes with shape-deformation.