DOI: 10.1111/cgf.14086

COMPUTER GRAPHICS *forum*
Volume 00 (2020), number 00 pp. 1–15

Interactive Subsurface Scattering for Materials With High Scattering Distances

Sebastian Maisch[1] and Timo Ropinski[1,2]

[1]Ulm University, Germany
{sebastian.maisch, timo.ropinski}@uni-ulm.de
[2]Linköping University, Sweden

Abstract

Existing algorithms for rendering subsurface scattering in real time cannot deal well with scattering over longer distances. Kernels for image space algorithms become very large in these circumstances and separation does not work anymore, while geometry-based algorithms cannot preserve details very well. We present a novel approach that deals with all these downsides. While for lower scattering distances, the advantages of geometry-based methods are small, this is not the case anymore for high scattering distances (as we will show). Our proposed method takes advantage of the highly detailed results of image space algorithms and combines it with a geometry-based method to add the essential scattering from sources not included in image space. Our algorithm does not require pre-computation based on the scene's geometry, it can be applied to static and animated objects directly. Our method is able to provide results that come close to ray-traced images which we will show in direct comparisons with images generated by PBRT. We will compare our results to state of the art techniques that are applicable in these scenarios and will show that we provide superior image quality while maintaining interactive rendering times.

Keywords: rendering, global illumination, real-time rendering

ACM CCS: • Computing methodologies → Rasterization; Reflectance modelling

1. Introduction

Subsurface scattering is a common effect in natural materials. All non-metal (dielectric) materials show this effect with differing scattering distances. The subsurface scattering effect occurs if light penetrates a materials surface, scatters at least once inside the material and exits it at a different location. It appears as a low-frequency effect on the material's surface especially at shadow edges where light bleeding can be noticed. The scattering distance determines the strength and the distance of the (visible) translucency. For very low scattering distances the effect is not visible anymore to the human eye. Materials with high scattering distances entail very prominent subsurface scattering. In physical quantities the scattering distance is determined by the scattering and absorption coefficients (σ_s and σ_a) of a material. These quantities are introduced in radiative transfer theory that is covered in depth by Chandrasekhar [Cha60]. Their sum describes the inverse value of the mean free path until a scattering or absorption event occurs in a material.

In real-time computer graphics, subsurface scattering algorithms can be divided into two categories according to their strengths and weaknesses. The first category directly implements the surface integral in the subsurface scattering function as a surface filter, the second category focuses on the three-dimensional nature of the object. Especially for rendering of human skin (which is one of the most prominent applications of subsurface scattering) texture filtering approaches are often used. This was first proposed by Lensch *et al*. [LGB*03] and Borshukov and Lewis [BL03]. The irradiance at the surface of the object (and possibly position and normals) is rendered into a texture which is then filtered by a 2D scattering kernel. Usually this is done either in a texture atlas or in screen space. Images rendered with this method usually provide high degree of geometry and texture details but they miss some scattering, either because some positions are close in the 3D world space but far away in the texture atlas representation or because they are simply not rendered (due to the screen space approach). Using a texture atlas also adds some restrictions to the mesh parameterization because 2D filter kernels do not work well over gaps or seams. In addition,

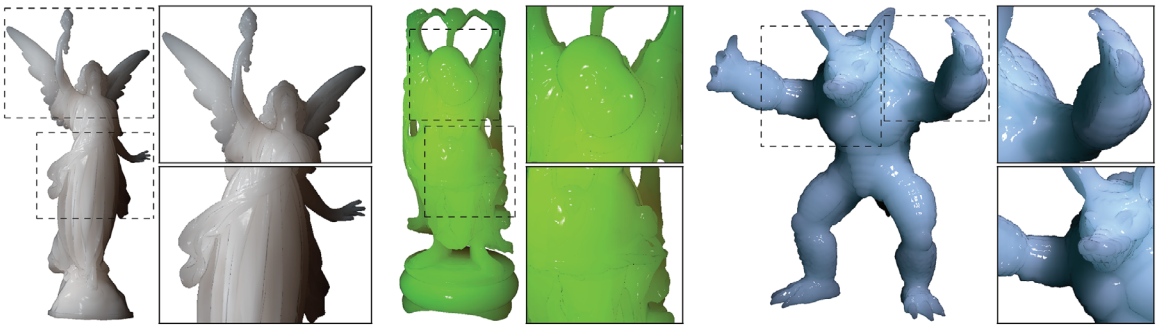


Figure 1: *Three different applications of our novel algorithm for subsurface scattering of materials with high scattering distances. We added a dielectric boundary to these images to emphasize geometric variations. Each scenario has two close-up regions that show areas of interest in detail. All models show intricate geometric details and we ensured to put focus on the crossover between shadowed and lit regions.*

scattering distances that are very large when projected to screen space lead to huge filter kernels that are rather inefficient to compute. This can either be due to the physical material parameters, but also due to the size of the object. Small objects rendered as a close-up or zoom will have the same problems as large physical scattering distances on bigger objects.

The second category better exploits the 3D nature of the rendered objects, but often lacks precision in preserving high-frequency local details. Algorithms in this category either use the vertex representation of the object, for example the initial work by Hao *et al*. [HBV03], or a (modified) translucent shadow map (TSM) first introduced by Dachsbacher and Stamminger [DS03].

Combinations of both categories also exist and provide high-quality results in some cases. However, for materials with high scattering distances these approaches often fail due to the huge size of the filtering kernel. Either they produce visual artefacts caused by the low sampling of the kernel (which is sufficient for lower scattering distances but not for higher), or the performance is extremely poor because of the huge number of samples needed in the kernel.

Our goal is to provide high quality for small-scale details while including all scattering effects that occur. We want our algorithm to be able to render arbitrary meshes and thus not require them to be nicely parameterized (no seams, area and angle preserving) since this can be impossible to do even as an approximation. We exclude using geometry-based pre-calculations to enable rendering of animated objects without complicated adjustments. Since there are many material models for subsurface scattering with different strengths (e.g. physical correctness, artist friendliness) we want to be able to use any model we choose with our algorithm, assuming normal incidence of the light, which we think is still necessary for real-time rendering today. In this paper we will use the physically based BSSRDF (bidirectional subsurface scattering distribution function) model proposed by Habel *et al*. [HCJ13b] as it is applied in Physically Based Rendering: From Theory To Implementation (PBRT) [PJH16] (i. e. assuming normal incidence of the light and combining single- and multi-scattering) for comparison reasons, but other models can also be used without changing the algorithm. Finally, our algorithm should be applicable with all kinds of types of light sources (even area and environment lights) and scale well with higher numbers of light sources. The latter goal can of course only be achieved to a limited degree as shadowing techniques always apply some kind of overhead per light source to the rendering time. They also are hard to achieve correctly for area and environment lights in real time. Disregarding these general problems with illumination this goal can be achieved.

From these goals and restrictions we derive an algorithm with a similar general structure as that presented by Maisch and Ropinski [MR17]. We split the contributions into a high detailed texture filtering approach and a vertex-based approach for scattering that is not included in the texture filtering. To avoid any problems with mesh parameterization we perform the texture filtering in screen space. The filtering presented by Jimenez *et al*. [JZJ*15] that was also used by Maisch and Ropinski [MR17] cannot be applied to these scattering distances. This is because they use a world space kernel that is projected to screen space based on the camera distance at the kernel centre. For smaller scattering distances (and thus smaller kernels) differences in camera distance can be compensated for with a simple interpolation. However, this solution is impossible for larger kernels. To prevent this issue, we provide a novel filtering pattern that works well with large scattering distances and apply that in screen space. We use the splatting method proposed by Nalbach *et al*. [NRS14] to handle the lit parts of the object that are not visible on screen. This prevents the pre-calculations that were needed by the Spatial Adjacency Maps (SAM) proposed by Maisch and Ropinski [MR17] while still being able to handle all kinds of light sources as they do. Finally, to be able to compare our algorithm to a physically based path tracer we ensure the introduced approximations are all physically based and subsequent errors are minimal. We found that approximating the average of the scattering kernel across a certain area by a single sample at the centre of this area, which is used by several other approaches, introduces a lot of errors. Thus, we propose a novel kernel pre-integration that allows us to calculate this average over circular areas of arbitrary size.

To summarize our contributions

- we combine a novel screen space with a geometry-based approach that can correctly render animated meshes,
- our novel kernel pre-integration allows for renderings close to a path traced ground truth generated by PBRT [PJH16],
- we introduce a novel screen space sampling pattern applicable for large scattering distances.

We compare out technique to a ground truth and other existing techniques to show ours can provide a superior image quality as shown in Figure 1.

2. Previous Work

We divide the previous work relevant to this paper in two sections. First, we discuss offline rendering solutions and BSSRDF models, and second we review real-time rendering techniques. We cover different types of BSSRDF models as well as acquisition of material parameters. As to the real-time techniques, we consider texture-based techniques, and discuss TSMs and related techniques as well as texture filtering. We also discuss geometry-based techniques, while discriminating between gathering and splatting approaches.

2.1. Offline rendering and BSSRDF models

Subsurface scattering effects were first used by Blinn [Bli82] for simulating light interaction of clouds and dusty surfaces. Later Hanrahan and Krueger [HK93] presented a more sophisticated model for scattering that included single scattering in homogeneous participating media. Single scattering is a prominent effect in optically thin media but when rendering optically thick materials multi scattering gets more dominant. Stam [Sta95] presented a solution for multi-scattering by approximating it as a diffusion process. This proposition is sufficient if many scattering events happen inside the material, so the actual direction of incoming light becomes less important and scattering itself becomes more and more isotropic. Using this approximation Jensen *et al*. [JMLH01] provided a BSSRDF model assuming a flat, semi-infinite geometry. In turn, this model is only applicable in cases of semi-infinite geometry, but is used on other geometries as well. The solution required the incident light to be perpendicular to the surface. The authors fixed this requirement by multiplying fresnel terms that added a directional component. For offline rendering purposes the rendering speed of the method for using the BSSRDF model by Jensen *et al*. [JMLH01] was further improved by Jensen and Buhler [JB02]. They used a two pass technique to first compute the irradiance for translucent objects and then evaluated the BSSRDF hierarchically but even this improvement could not speed up the computing process enough for real-time rendering purposes. A different approach for rendering subsurface scattering was used by Munoz *et al*. [MESG11]. They use convolutions in Fourier space to solve the scattering integral and divide the image in multiple layers to ensure covering all irradiance. This results in a fast approximation of the effect for a lower number of layers but even with a higher number their results still differ visually from the ground truth by Jensen and Buhler [JB02].

Using this solution as a basis, Donner and Jensen [DJ05] generalized the BSSRDF model to support slabs (an infinitely large plain but with a finite thickness) and multi-layered materials. In another paper [DJ07] the same authors proposed a photon diffusion model for the BSSRDF that directly contained directional information from the incident light. They also proposed a model for geometries with 90 degrees edges. While it still is the basis for current on- and offline rendering methods for subsurface scattering, the basic diffusion model has been improved by d'Eon [d'E12] and also Habel *et al*. [HCJ13a]. Both proposed using a Green's function, which has lesser errors for high absorption materials, for the diffusion equation in BSSRDF models. D'Eon and Irving [dI11] used these new solutions for a *Quantized Diffusion* that quantized the light response at discrete time steps to create a BSSRDF model consisting of a sum of Gaussians. Due to the nature of this model it can be applied to slab geometries and multi-layered materials much easier than previous models. Habel *et al*. [HCJ13b] combined the new diffusion solutions with the photon diffusion model to create *Photon Beam Diffusion* that directly supports arbitrary directions of incident light. All models with a directional component mentioned have the disadvantage that they comprise an integral that needs to be solved numerically. The method proposed by Frisvad *et al*. [FHK14] overcame this by incorporating the directional component directly in the solution of the diffusion equation. An approximate BSSRDF model was proposed by Christensen [Chr15] that can directly transfer to diffuse illumination for far away objects. Frederickx *et al*. [FD17] proposed a model for forward scattering materials. Due to the higher number of parameters it is rather hard to tabulate and thus not suitable for real-time applications. All of the previously mentioned models are only valid for specific geometries (either semi-infinite or slab geometries). Vicini *et al*. [VKW19] used a learning-based approach to adjust a subsurface scattering model to the local geometries to overcome constraints due to the geometries assumed by most models. Currently their approach is not feasible for real-time rendering.

Material parameters for subsurface scattering were measured by Jensen *et al*. [JMLH01] and fitted to their model. Other measurements were done by Narasimhan *et al*. [NGD*06] for diluted materials, Gkioulekas *et al*. [GZB*13] for several common materials including soap and mustard, and Weyrich *et al*. [WMP*06] for human skin. For the latter Iglesias-Guitian *et al*. [IGAJG15] developed a model for skin aging that also provides scattering parameters.

2.2. Real-time rendering

A first real-time solution for subsurface scattering was presented by Dachsbacher and Stamminger [DS03] who used TSMs to capture the irradiance from a single light source on the scene. By reprojecting the TSM onto the geometry from a viewer's perspective and applying a sampling pattern subsurface light transport is calculated. Lensch *et al*. [LGB*03] used a texture atlas for a similar purpose to calculate what they call 'local' scattering. These techniques are adapted and combined with texture filtering techniques by d'Eon *et al*. [dLE07, dL07]. They fit a sum of Gaussians to match the subsurface scattering kernel and could thus apply multiple separable kernels to approximate the effect using a texture atlas that required a parameterization with only a few seams. Jimenez *et al*. [JSG09] modified this to a screen space solution which was finally modified to use only a single separable kernel [JZJ*15]. Both relied on a world space texture filtering kernel that is projected to screen space to find the correct sampling positions. This avoided a lot of problems regarding the need for a proper kernel pre-integration to create results that are stable with regard to changing the camera distance. For larger kernels this projection becomes inaccurate, though. For smaller kernels especially the latter technique was very fast and thus perfectly suitable for real-time applications. The texture filtering techniques provided highly detailed results preserving small local texture and geometrical features. Because of reprojection and

shadow map resolution this is not the case for TSMs. The downside of the texture filtering techniques is, that the translucency effect from parts of the scene that are not visible in camera space were ignored. Jimenez *et al*. [JWSG10] used an estimation for the missing scattering which was not based on the actual illumination at the objects' back side and only relied on a single sample. While this approximation could create plausible results in some cases, it could not reproduce the actual scattering generally. All mentioned techniques involved some kind of texture filtering which provided convincing results for materials with lower scattering distances. For higher scattering distances it could not be applied though because the texture filtering was not optimized for that purpose. Chang *et al*. [CLH*08] proposed a texture-based approach based on importance sampling. They render the irradiance into texture atlas and use MipMapping for importance sampling. Because of this their technique is not affected by the issues caused by parameterization but also lack the detailed results of texture filtering. Elek *et al*. [ERS13] presented a texture-based technique for larger scattering distances for scenes with objects placed in a scattering medium. A representation similar to TSMs is also used by Shah *et al*. [SKP09] and Chen *et al*. [CPZT12] as a source for generating splats. We will fully expose this further in the next paragraph where splatting algorithms are discussed. Existing image space techniques based on shadow maps lack quality for high-frequency details due to shadow mapping projection. Screen space techniques on the other hand can miss the contributions to the translucent effect of the material. Our technique relies on the positive effects of screen space techniques while reducing their downsides by combining it with a geometry based translucency.

Another group of real-time techniques takes into account the objects' whole 3D structure. Lensch *et al*. [LGB*03] used a vertex-based approach for rendering their 'global' subsurface scattering. They pre-calculated a transport matrix on triangle level and applied that by matrix multiplication during rendering. This multiplication might not be suitable for real-time applications with higher vertex counts. Similarly Mertens *et al*. [MKB*03] introduced a low resolution mesh to calculate form factors used for subsurface scattering. Both techniques rely on a pre-calculation that might not be directly applicable to animated meshes. They are gathering techniques that calculate scattering for each vertex and interpolate it over a triangle which reduces quality. While Shah *et al*. [SKP09] and Chen *et al*. [CPZT12] created a TSM first, their principle was to render splats on the geometry buffer not to directly calculate scattering. Both, splats and geometry buffer, were used in different resolutions to reduce the overhead that splatting creates. Their technique works well with animated meshes and also covers all scattering even with large scattering kernels. As a downside their approach was restricted to light sources that allow for creating shadow maps and the quality of the results suffers from shadow map resolution and reprojection. Nalbach *et al*. [NRS14] proposed a *Deep Screen Space* (DSS). They created splats directly in multiple resolutions with the tessellation shader. To reduce the rendering time they also used a multi-resolution approach. Since all splats were created separately for each image, animated meshes will naturally work. While both algorithms, in general, are quite promising, their speed does not scale well with screen resolution. The technique by Chen *et al*. [CPZT12] had to find a balance between the resolution of their TSM that determines image quality and rendering speed. Nalbach *et al*. [NRS14] had a similar trade-off with mesh tessellation. Preserving high details will result in poor rendering speeds with both algorithms as the splats create a lot of overdraw.

Maisch and Ropinski [MR17] proposed a combination of *Separable Subsurface Scattering* (SSSS) by Jimenez *et al*. [JZJ*15] and the vertex-based approach by Mertens *et al*. [MKB*03] to gain high details as well as low-frequency contributions previously unobtainable with texture space algorithms. They adjust their pre-calculations such that they are applicable to animated meshes as well. For higher scattering distances their approach shows the same downsides as SSSS. Our algorithm similarly splits contributions between a screen space technique to ensure high-frequency details and a geometry-based algorithm to add missing translucency. Since the geometric approach is only applied if irradiance is not covered by the faster screen space technique computation time is reduced.

Another approach to interactive subsurface scattering was proposed by Dal Corso *et al*. [DCFMB17] who used the directional dipole model proposed by Frisvad *et al*. [FHK14]. Their technique also comprised the scattered light to further illuminate the scene. While their technique provided promising results for larger scattering distances it has similar problems as the original TSM approach. For changing light sources it also gets very slow which is insufficient for our real-time use case. Another feature of the article by Chen *et al*. [CPZT12] is their ability to render heterogeneous materials. To do this they apply a non–physically based combination of BSSRDFs from materials at the light entry and exit point. While we did not design our algorithm with this feature in mind, our algorithm can be easily extended to also handle these cases.

3. Our Approach

As mentioned before, the scenario we address with our work restricts the design of our algorithm. In this section we want to derive these restrictions and give an overview on how we address them.

3.1. Derivation

We start deriving our solution with the same ideas that are presented by Maisch and Ropinski [MR17]. The equation for subsurface scattering describes radiance L leaving a single point $\mathbf{x_o}$ in direction $\vec{\omega_o}$:

$$L(\mathbf{x_o}, \vec{\omega_o}) = \int_A \int_{2\pi} S(\mathbf{x_i}, \vec{\omega_i}, \mathbf{x_o}, \vec{\omega_o}) L_i(\mathbf{x_i}, \vec{\omega_i}) (\vec{n_i} \cdot \vec{\omega_i}) \mathrm{d}\vec{\omega_i} \mathrm{d}\mathbf{x_i} \,, \quad (1)$$

where A is the surface area of the object to visualize, L_i is the incident light at that surface and S is the subsurface scattering BSSRDF. The BSSRDF can be split into directional factors (F_t), in this case Fresnel terms, that describe the light entering the material at $\mathbf{x_i}$ and exiting at $\mathbf{x_o}$, a factor based on Fresnel moments (C_Φ), and a distance factor (R_d) that covers light transport between these points as described by Habel *et al*. [HCJ13b]:

$$S_d(\mathbf{x_i}, \vec{\omega_i}, \mathbf{x_o}, \vec{\omega_o}) = \frac{1}{\pi} F_t(\mathbf{x_i}, \vec{\omega_i}) R_d(\|\mathbf{x_i} - \mathbf{x_o}\|) \frac{F_t(\mathbf{x_o}, \vec{\omega_o})}{4 C_\Phi \left(1/\eta\right)},$$

$$c_o = \frac{F_t(\mathbf{x_o}, \vec{\omega_o})}{4\pi C_\Phi \left(1/\eta\right)}.$$

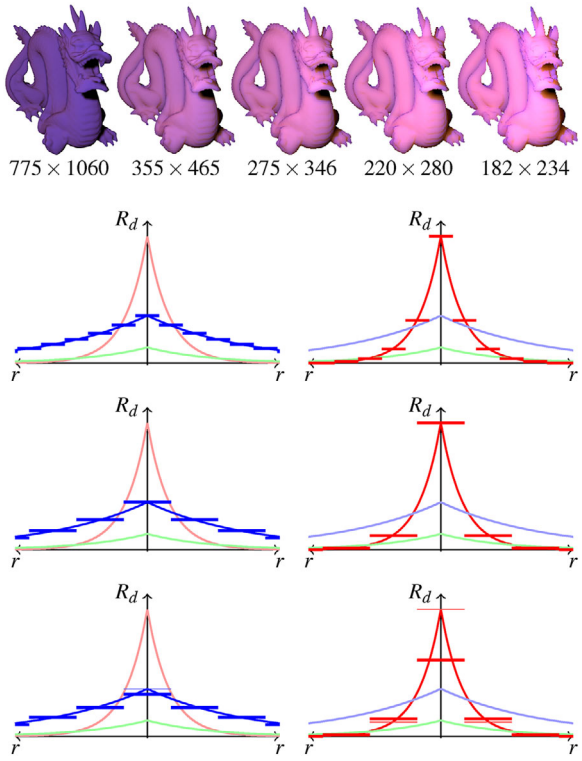


Figure 2: *Illustration of the effects of using incorrectly integrated subsurface scattering profiles multiple colour channels. The first row shows how this effect on the* Chinese Dragon *model rendered with a naive screen space approach at different camera distances. The leftmost image has the lowest camera distance while the one on the right has a high camera distance. All images are then normalized to show the models at the same size (but different resolutions stated in the labels below each image). Due to more surface area being covered by a single pixel the zoomed out versions get brighter and also the colour changes from a dark violet to a pink. The plots in the rows below that explain how this effect comes to be. We use the red and blue colour channels as an example. In the second row the area covered by a pixel is small (high resolution) and the profile functions are represented well with the samples. Using a lower resolution leads to a higher surface area (third row) and the profile, especially the red one which has a highest peak gets less well represented with the chosen samples. A correct integration of the profile adjusts the sample values to always represent the profile best regardless of the areas of the samples (last row). The thin lines shown there represent the naive values for this sampling area and shows once more this effect is stronger with colour channels which profiles contain higher peaks.*

As a means to simplification we only consider a single incoming light direction here, so that we can ignore the second integral. Multiple light sources can be used by just adding up their effects independently as presented by Maisch and Ropinski [MR17]. We also introduce c_o describing constant terms pulled out of the integral to simplify the following equations. The key idea for our approach is to split the surface area integral into two disjoint parts where one part is visible in screen space and the other is not.

To combine the surface effect with the translucency effect Maisch and Ropinski [MR17] use surface splitting criteria to define where each effect is applied. These criteria define which points ($\mathbf{x_i}$ with normals $\vec{n_i}$) belong to the 'front side' (where they apply the screen space effect) and which belong to the 'back side' (where they apply their translucency), for each point $\mathbf{x_o}$ on the surface of the mesh. Since they needed splitting that could be used in their pre-calculations they used the surface normal $\vec{n_o}$ at point $\mathbf{x_o}$ to define the criteria as follows:

$$(\mathbf{x_i} - \mathbf{x_o}) \cdot \vec{n_o} < 0 \qquad \text{and}$$
$$\vec{n_i} \cdot \vec{n_o} < 0.$$

If both inequalities are true a point will be on the 'back side' and used with the translucency effect. We adjusted these criteria to use the view direction $\vec{v}$ instead of the normal at $\mathbf{x_o}$:

$$(\mathbf{x_i} - \mathbf{x_o}) \cdot \vec{v} < 0 \qquad \text{and} \tag{2}$$
$$\vec{n_i} \cdot \vec{v} < 0. \tag{3}$$

Since it was a pre-requisite to always use a screen space surface effect and discard any pre-calculations, we can use the non-constant view direction. This enables us to calculate the screen space effect without checking for the criteria explicitly.

3.2. Pre-integration of subsurface scattering kernels

Equation (1) needs to be split into a sum of integrals as described by Maisch and Ropinski [MR17] to be calculated in real-time. For the screen space calculation we sum over areas overlapped by pixels, in the geometric part we sum over triangles in our case:

$$L(\mathbf{x_o}, \vec{\omega_o}) = c_o \sum_{A_i \in A} \int_{A_i} F_t(\mathbf{x_i}) R_d(\|\mathbf{x_i} - \mathbf{x_o}\|) L_i(\mathbf{x_i}) \mathrm{d}\mathbf{x_i}. \tag{4}$$

While assuming $F_t L_i$ to be constant across the surface and removing them from the integral will only result in small errors, the same assumption does not work very well for R_d. Due to the strong peak of the function, simply assuming R_d to be constant over that area introduces a large error especially at small distances. This error is especially visible in the screen space effect because changing the camera zooming or distance will result in changes of the areas that are overlapped by a single pixel. This leads to effects of changing brightness and even colour as shown in Figure 2. There we visualize the visual effect as well as show that different sample areas can introduce large errors in the final integral. We also show that this error can be different for each colour channel depending on the peak of the profile function leading to a change in colour when zooming. This effect was not appearing in previous works because of the areas used for the integration were constant. In case of triangle areas which were used by Nalbach *et al.* [NRS14], it is obvious that the areas will not change by moving the camera. The areas of the splats used by Chen *et al.* [CPZT12] was also constant as long as the light source did not change. Finally the screen space kernel used by Jimenez *et al.* [JZJ*15] uses world space distances that are projected to screen space. Similar to the splatting cases the areas the kernel integrates over are constant in world space which enables zooming without any change in colour. Smaller errors that did occur

would only become visible in direct comparison with a path tracer. Jimenez *et al.* [JZJ*15] pre-integrate their kernel for a more correct image but only in 1D. While the kernel is parameterized as a 1D function, it is actually 2D radially symmetric. Approximating this integration in 1D is not sufficient for producing correct images in our use case. We discuss in Section 7.2 that each point in the 1D kernel needs to be weighted depending on its distance to the origin and also the position of the area to be integrated over.

While theoretically the shape of the area is also relevant, our results were sufficiently accurate when approximating the shape by a circle. This way we can tabulate a kernel with two parameters. They are distance to the centre ($\|\mathbf{x_i} - \mathbf{x_o}\|$) and the integrated area A in a 2D texture. For each distance and area we store the average kernel R'_d as follows:

$$R'_d(\|\mathbf{x_i} - \mathbf{x_o}\|, A) = \frac{1}{A}\int_A R_d(\|(\mathbf{x_i} - \mathbf{x}) - \mathbf{x_o}\|)\mathrm{d}\mathbf{x}. \tag{5}$$

We can thus simplify Equation (4) to

$$L\big(\mathbf{x_o}, \vec{\omega_o}\big) \approxeq c_o \sum_{A_i \in A} F_t(\mathbf{x_i}) L_i(\mathbf{x_i}) A_i R'_d(\|\mathbf{x_i} - \mathbf{x_o}\|, A_i), \tag{6}$$

which we can calculate in real-time. An explanation how we approximate this integral numerically is given in Section 7.2.

3.3. Overview

Our algorithm consists of seven render passes. The first pass is a deferred rendering pass where we store positions, normals, index of refraction and (optionally) texture colour in a G-Buffer. Additionally, we calculate a mask for the objects we want to apply the algorithm to and the projected pixel area and also store them in the G-Buffer. Further material properties not directly needed for our algorithm can also be stored here (e.g. glossiness).

In the second pass we render (simple) shadow maps that are used in the third pass when we calculate all (screen space) illumination. For our purposes this illumination consists of directly reflected radiance and transmitted irradiance. We then create a MipMaps for the positions and normals in the G-Buffer and the transmitted irradiance.

In a fourth pass we calculate our translucency effect (see Section 5) in a lower resolution which is upscaled in an additional pass. Finally we generate the screen space subsurface scattering (see Section 4) and combine it with translucency and direct illumination. The same pre-calculated kernel described in Section 3.2 is used in both passes that calculate a part of the subsurface scattering effect.

4. Screen Space Technique for High Scattering Distances

Existing screen space subsurface scattering algorithms cannot be applied if high scattering distances are desired. We already discussed the changing in brightness and colour while zooming using a naive direct approach to screen space subsurface scattering in Section 3.2. Jimenez *et al.* [JZJ*15] offered a solution for this problem for their approach, which is unfortunately not applicable to our situation, large scattering distances will most likely result in scattering kernels that cover the whole screen or at least a large portion of it. As a consequence applying this kernel naively will slow down the rendering significantly. One solution would be to separate the kernel but the assumption of an additively separable signal (in this case the irradiance) that Jimenez *et al.* [JZJ*15] rely on does not hold for large kernels. Rotating the kernel to improve on this problem is also impossible since the errors will lead to a very noisy image. Using a fixed world space kernel and projecting it to the screen is also not possible since this assumes the camera distance being constant inside the kernel. Jimenez *et al.* [JZJ*15] use interpolation to approximate small changes in camera distance across their kernel but with larger kernels the changes will become too big. We thus need samples with fixed positions in screen space and use world space distances as kernel parameter.

4.1. Sampling the kernel

Jimenez *et al.* [JZJ*15] sampled the centre of the kernel more densely than the exterior in their implementation. This makes sense since the kernel function will have the biggest changes in the centre. They do not use averaging for samples in less dense regions, because it is not really needed for lower scattering distances. It is also unclear how to correctly apply this to a separable approach. For larger kernel sizes another problem arises if the distances between two samples in the outer regions become too large, while their kernel contribution is still relevant. Due to this, sharp features in areas with a low sample density are reproduced as dimmer copies of these features at the position of the kernel centre in the resulting image. When using a dense kernel, neighbouring pixels will sample the same features, resulting in a blurred version of the original feature up to the point where the copy of the feature is not visible anymore. In less dense kernels these copies will be visible in the resulting image at distances to the original feature that corresponds to the distances of the samples in the kernel. Dachsbacher and Stamminger [DS03] sample from a downsampled buffer when using less dense samples to circumvent this problem. The downside of their pattern is that it is fixed and cannot be adjusted to the screen or kernel size. We propose a pattern that can be adjusted as required.

4.2. New sampling pattern

We introduce a new and adaptable sampling pattern that includes downsampling of the original irradiance image and G-Buffer. To minimize the number of samples necessary for the kernel, we sample the pixels near the kernel-centre more densely than the pixels further away. We assign a region to each sample that is determined by the sample's distance to other samples. To avoid creating visible copies of the image, we sample the average of the region the sample covers by means of downsampling. We ensure the presence of high-frequency details by having an inner region of our kernel sampled in full resolution. The remaining kernel is layered. Each layer is a shell around the previous one with the first one being the full resolution inner region. Each layer has a width of n_l samples and can be represented by a regular grid. Only the outermost grid cells are used. Thus, the layer has $4(n_l - 1)$ samples in total. The diameter of each sample is $\frac{d_{i-1}}{n_l - 2}$, where d_i is the diameter of layer i, and d_0 is the width of the inner kernel. For an inner kernel with a width of

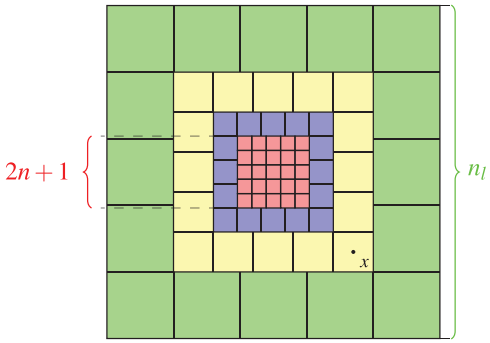


Figure 3: *An example for our sampling pattern that has an inner kernel (red) with 25 samples ($n = 2$) and three outer layers (blue, yellow and green) with a width of five samples each ($n_l = 5$). We also marked an exemplary sampling position $x = p_2(4)$.*

$2n + 1$ samples, the positions of all grid points of the i-th layer on a single axis are

$$p_i(s) = \frac{1}{2}(2n+1)\left(\frac{n_l}{n_l - 2}\right)^{i-1}\left(\frac{2s-1}{n_l - 2} - 1\right), \tag{7}$$

where $s \in 0, \ldots, n_l - 1$. Figure 3 shows an example for such a pattern with an inner kernel with 25 samples and three outer layers with a width of five samples each. We defined our sampling pattern in that way to be more flexible and to choose varying numbers of samples. We do not only choose our sampling positions, but also can deliberately choose the MipMap level to look up the sample and area for the kernel integration according to the samples size in the pattern. We found that the image quality will be best in cases of $n = 2$ and $n_l = 5$ with 10 layers. The number of layers might require further adjustment to the screen size if the sampling of the whole screen is necessary.

5. Translucency Effect

Other than the screen space technique we have to specify how to include scattering that is not visible on screen. We do not use the 'Spatial Adjacency Maps' proposed by Maisch and Ropinski [MR17] because of the necessary pre-calculation and the resulting problems with applying the approach to animated objects. Instead, we create our translucency effect by using a modified version of 'Deep Screen Space' [NRS14]. DSS is a technique to render several global illumination effects using splatting. We focus on the subsurface scattering technique. It is based on hardware tessellation to create splats from the triangles in a triangle mesh. The tessellation level is chosen in accordance to the camera distance and triangle area. The position, normal and irradiance at the centre of a triangle will be assigned to the splat as well as the triangles area. The size of the splat is determined by the scattering distance. These splats are rendered into a G-Buffer storing, at least, positions and normals (other parameters such as texture colour or varying material properties can be added) where we calculate the light transport between the splat's position and the position in the G-Buffer. We need to add all contributions of all splats covering a pixel, so we cannot use the depth buffer to reduce the overdraw. The GPU's blending unit has to be adjusted to add all results. This leads to a considerable amount of fragment shader evaluations per pixel, especially in case of subsurface scattering as the splats sizes are determined by the scattering distance. To reduce the rendering time for the splatting, the original DSS uses a hierarchical G-Buffer to render different distances of the splat in different resolutions. Pixels near the splats centre thus receive a more detailed effect than pixels that are further away.

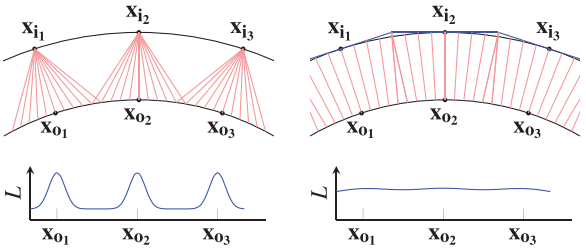
For our modification we simplify this multi-resolution approach. Since we already have a high-quality effect that is applied at small distances we can use the DSS effect directly in a lower resolution. Contributions of the model's back side will be of lower frequency anyway. Nalbach *et al.* [NRS14] suggested that rendering in half the original resolution is still slightly slower than their multi-resolution technique. We choose a quarter of the original resolution for our approach. As also suggested by Nalbach *et al.* [NRS14], we recommend using a high resolution model for the actual rendering and a lower resolution model for the splatting. In contrast to their approach we do not further tessellate the triangles though as this is not necessary for our low-frequency effect. They also cull splats in the geometry shader based on their irradiance and we do the same but cull even more splats due to our splitting criteria. For this we perform the test on those criteria in the geometry shader and choose $\mathbf{x}_o$ as the position stored at the splats centre in the G-Buffer. This is not perfectly correct but it drastically reduces the number of splats actually drawn which entails major advances in performance.

6. Adjustments to the Model for the Discrete Case

The theoretic model proposed above involves several points where we discretize a continuous model of an object's surface into either pixels or triangles. While this is not a big issue for smaller pixel or triangle areas, it can introduce some artefacts when these areas become too big. In this section, we describe how to handle such cases.

6.1. Sampling pattern

As described in Section 4.2 we use MipMapping to get average positions, normals and irradiance for the subsurface scattering calculation. These MipMap levels may get quite large so that the original image is reduced to merely a few pixels. The pre-integrated kernel resolves some issues that can occur due to the subsurface scattering kernel itself but cannot account for issues that occur due to the scene geometry. Our kernel pre-integration assumes a uniform distribution of positions in the covered area which is usually not the case in the actual geometry. Especially for samples that cover a larger surface area, this assumption is not valid. As smaller distances to the kernel centre have an exponentially larger influence than bigger distances, samples will appear dimmer the larger the area they cover. To compensate for this, we use a 'brightening term' based on the index of the outer sampling layer (i), as this index also determines the sample size of that layer (see Secion 4.2). The term that provided the best results was $\sqrt{i}$ which we multiplied with each contribution by a sample from an outer layer to obtain the final contribution of the sample.



(a) Naive splatting. (b) Our adjusted splatting.

Figure 4: *Example of a thin slab-like geometry in which subsurface scattering is simulated by splatting. The splat's positions are* $\mathbf{x}_{i_1}$, $\mathbf{x}_{i_2}$ *and* $\mathbf{x}_{i_3}$ *and the visible surface goes through three exemplary points* $\mathbf{x}_{o_1}$, $\mathbf{x}_{o_2}$ *and* $\mathbf{x}_{o_3}$. *Figure 4(a) shows the naive approach in which light transport (shown in red) is observed between the slabs' positions to the opposite surface. The plot shows the expected light distribution. In Figure 4(b) we use evaluation:textures' our adjusted positions on a disk (shown in blue) around the slabs centres for the light transport resulting in a more even light distribution.*

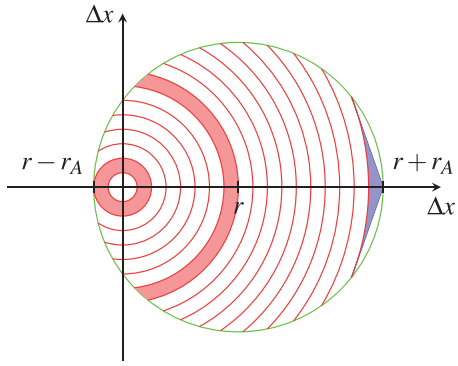


Figure 5: *Illustration of our method to integrate* R_d *over a specific area. The area is given in green as a circle with centre at r and radius* r_A. *Pictured in red are isocurves of* R_d *with pre-calculated values. In light red two examples of arc segments, the first is a complete ring, and the second is an arc segment described by Equation (11). The blue area shows an example in which our integration deviates strongly from the real circle area.*

6.2. Model combination

In Section 4 we explained how we can split the subsurface scattering integral into two distinct parts that we can calculate separately. Our final result is the sum of these parts. The parts, we split this integral into, are the screen space part and the geometric part, and the criterion to split these is explained in Section 5. Triangles with normal vectors that barely need to be included in the screen space technique (according to Equation (3)) have a very steep angle towards the view direction. This can lead to numerical problems when calculating the area of a projected pixel for these triangles because this area would become infinite. We detect these problems and set these areas to zero thereby effectively removing their contribution from the screen space part. We thus account for this by introducing them in the geometric part of the integral by adjusting the criterion from Equation (3):

$$\vec{n_i} \cdot \vec{v} < 0.2. \tag{8}$$

To allow a small overlap of the regions we also allow angles slightly smaller than 90° by using 0.2 as a constraint instead. Due to the differences in geometric resolution between the meshes used for splatting and the ones used for the screen space part this will produce some overlap. This overlap violates the distinct splitting but only to a small proportion that is not expected to affect image quality.

6.3. Low resolution splatting

Using lower resolution models will cause the splats to have a bigger distance between each other. As the function we apply using the splats will have a strong peak for low distances, thin geometries can cause this method to produce bright spots on the opposite side of the geometry with respect to the splat's position. In most cases this should produce a smooth bright surface instead of spots. To correct this error, we adjust the distance used to calculate the subsurface scattering. We do not consider the distance between the splat's position $\mathbf{x_i}$ and the pixel's position from the G-Buffer $\mathbf{x_o}$ but create a disk around the splat that is oriented along the splat's normal $\vec{n_i}$. The radius of the disk r_{disk} is determined by setting the disk's area equal to the area of the triangle that produces the splat. The adjusted position $\mathbf{x}'_\mathbf{i}$ is then

$$\mathbf{x}'_\mathbf{i} = \mathbf{x_i} + \frac{\bar{\mathbf{x}}_\mathbf{i} - \mathbf{x_i}}{\|\bar{\mathbf{x}}_\mathbf{i} - \mathbf{x_i}\|} \min\left(r_{\text{disk}}, \|\bar{\mathbf{x}}_\mathbf{i} - \mathbf{x_i}\|\right), \tag{9}$$

where $\bar{\mathbf{x}}_\mathbf{i} = \mathbf{x_o} + \vec{n_i}((\mathbf{x_o} - \mathbf{x_i}) \cdot \vec{n_i})$ is the point on the splat's plane nearest to $\mathbf{x_o}$. The subsurface scattering is then calculated based on the distance between $\mathbf{x}'_\mathbf{i}$ and $\mathbf{x_o}$ as illustrated in Figure 4. The figure also illustrates the effects of not adjusting the splats positions. The resulting translucency will still show a small variation in brightness but the effect is much more evenly distributed.

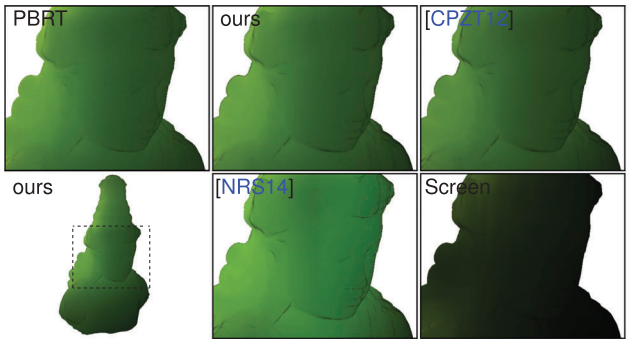
As we render the splatting into a render target with a lower resolution than the final result, an upsampling strategy is required to ensure the image quality. We tried several strategies to up-sample the lower resolution splatting results. We found that bicubic sampling (as described by Sigg and Hadwiger [SH05]) would give us the best results.

7. Implementation

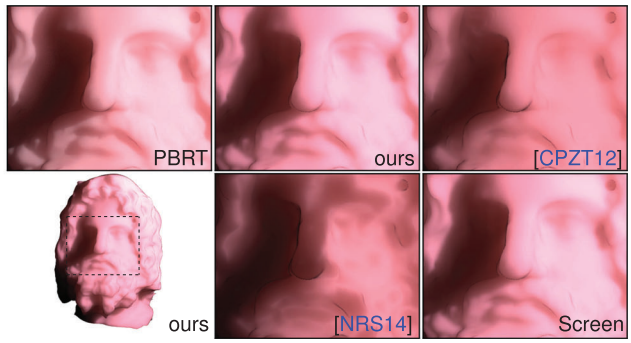
After the theoretic description of our technique we will explain implementation details. We discuss some parameters that were used to generate the results and give a more detailed description of some algorithms.

7.1. Generating kernel textures

To generate the kernel textures we first need to choose the texture size as well as minimum and maximum radii and areas stored. We use the *Photon Beam Diffusion* BSSRDF by Habel *et al.* [HCJ13b] in the same way as PBRT [PJH16] does. Notably they include the single scattering term that was separated in the original BSSRDF into their pre-calculated, tabulated BSSRDF. Similarly, the original model depends on the direction of incidence. In contrast,



(a) A Comparison of different rendering techniques in case of the *Indonesian Statue* illuminated from the back. While all algorithms using splatting ('ours', [NRS14], and [CPZT12]) do not show much differences compared to the ground truth ('PBRT'), the screen space algorithm ('Screen') generates a dark image due to the omitted direct illumination in the image. Rendering resolution: 1920×1080.



(b) The *Serapis* model with mostly direct illumination and several areas with shadow edges. This scenario shows that our technique ('ours') can preserve shadow edges and surface details very well. The screen space method ('Screen') does similarly well in this scenario. The technique by Chen et al. [CPZT12] provides a plausible image due to the high number of splats that are directly lit. DSS by Nalbach et al. [NRS14] can not preserve these details very well. Note that with this material we experience a slight shift in color for all our algorithms compared to the ground truth ('PBRT'). Rendering resolution: 1920×1080.

Figure 6: *Comparisons highlighting the necessity to combine geometry-based and image-based subsurface scattering to generate realistic images in all scenarios. For objects lit from the back (as in Figure 6(a)), the screen space only method will not show much of the effect because the directly illuminated part will not be seen. Geometry-based methods on the other hand will not be able to create the detailed scattering effect needed for a realistic image in case of directly illuminated objects (as in Figure 6(b)).*

PBRT [PJH16] as we do, assumes that light is incident along the normal direction. The chosen 2D textures have $r = \|\mathbf{x_i} - \mathbf{x_o}\|$ and A as parameters. The minimum of these values is 0. We determine the maximum value of r such that we obtain an acceptable cutoff value for the BSSRDF's source function $Q(r) = \alpha' \sigma_t' e^{-\sigma_t' r}$. The source function is a good choice because it can easily be inverted in contrary to the actual BSSRDF but is still a reasonable estimate for the falloff in this case. After trying different values, we cut off the BSSRDF at $r_{\max}$, where $Q(r_{\max}) \leq 10^{-7}$. The maximum value for A is then derived from $r_{\max}$: $A_{\max} = r_{\max}^2 \pi$. After some tests we found that a lookup texture resolution of 512×512 provides the best results in terms of image quality while still having a reasonably low memory footprint.

7.2. Fast numerical kernel pre-integration

As discussed in Section 3.2 we need to pre-integrate the kernel for different surface areas. The analytic formula is given in Equation (5). We pre-calculate R_d at fixed locations Δx_i that we have chosen based on the maximum distance between $\mathbf{x_i}$ and $\mathbf{x_o}$ and the size of the lookup texture. We then assume that the area we want to integrate over has a circular shape with radius $r_A = \sqrt{\frac{A}{\pi}}$, and its centre is located at distance r from the centre of the coordinate system. The function R_d is radial symmetric at the centre of the coordinate system. We can exploit this to approximate the integral over a ring centred with respect to the coordinate system cut with the circle area. We thus iterate over all $\Delta x_i \in \max(0, r - r_A), \ldots, r + r_A$. We assume R_d can be regarded linear between two sequential Δx_i, which is why we approximate the integral over the arc area by:

$$(b_i R_d(\Delta x_i) + b_{i+1} R_d(\Delta x_{i+1}))(\Delta x_{i+1} - \Delta x_i). \tag{10}$$

b_i being the arc length of the arc centred with respect to the coordinate system with radius Δx_i, cut with the circle area:

$$b_i = 2\Delta x_i \arccos\left(\frac{{\Delta x_i}^2 - r_A^2 + r^2}{2r\Delta x_i}\right). \tag{11}$$

If $r \pm r_A$ is not already part of our pre-calculated values, we linearly interpolate to get the proper values to start and end the integration. Figure 5 illustrates the basic idea: the circular area A is shown in green and different arc segments are shown in red. If $\Delta x_i < |r - r_A|$, the inverse cosine is undefined and the arc length is then $2\pi\,\Delta x_i$. As you can see in Figure 5, while most arc segments are nicely approximating the full circle, the last arc segment (drawn in blue) does not. In other configurations this can also occur for the first segment. However, adjustments of our method (by super-sampling the last arc segment) to obtain a better representation of the circle did not result in differences in the resulting image.

8. Evaluation

To demonstrate the outcome of our algorithm we compare its image quality and rendering speed against other approaches. As a ground truth we use images rendered with *PBRT* [PJH16] where we use the 'subsurface' material. This material implements the same BSSRDF model, and is therefore a perfect match for our comparison. Unfortunately PBRT handles textures in a different way that we cannot directly emulate in real time. Therefore, comparisons with PBRT will all be without textures. We do additionally provide textured examples in the supplementary material (Appendix 1.2) though, to show our algorithm can work in these scenarios as well. In Appendix 1.4 we also give detailed information about rendering times of specific

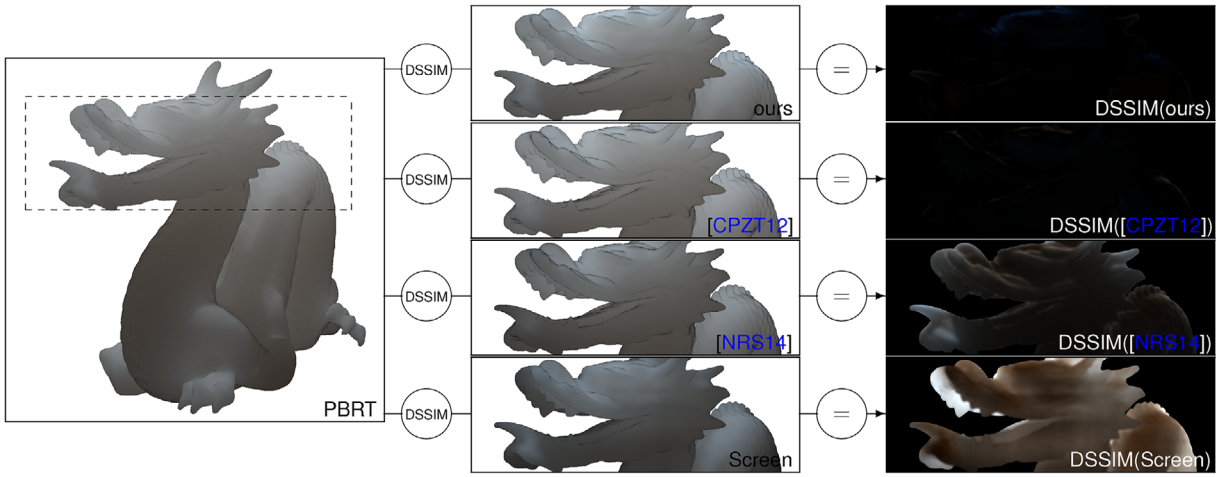


Figure 7: *The* Chinese Dragon *model rendered with PBRT (big image) and different real-time techniques (middle column). In the right column the per-pixel and colour channel DSSIM values are visualized for each technique. This scenario provides some directly illuminated regions (top of head, top of the body), some shadow edges (visible at the top of the body) and also regions without direct illumination (inside the mouth, horns). It also combines parts with smaller geometric features with larger parts. Our technique provides the images closest to the ground truth (black pixels mean no perceptual difference) by numbers, but the approach by Chen* et al. *[CPZT12] is also very close and DSS still provides reasonable image quality. The results generated by the screen space approach ('Screen') do not fit the ground truth that well. Rendering resolution:* 1920×1080.

render passes, GPU memory needed, objects sizes and material parameters the renderings in this paper.

Besides PBRT we also compare our algorithm to other real-time algorithms. We use an implementation of DSS as an example for geometry-based subsurface scattering techniques [NRS14]. As suggested by Nalbach *et al*. [NRS14] we used a lower resolution version of the model where it made sense to improve the performance of the algorithm. This is the same lower resolution model we used for our algorithm. The number of multi-resolution layers $l_{\max}$ is 3 in our examples and we set $\epsilon = 0.1$. We also compare against an implementation of the algorithm presented by Chen *et al*. [CPZT12]. We used a resolution of 2048×2048 (when using a lower resolution we encountered serious visual artefacts) for the irradiance buffer and 3 MipMap levels as suggested in the original paper. Although, both algorithms are based on splatting we included the algorithm by Chen *et al*. [CPZT12] to provide a comparison to an algorithm that we did not use as part of our own.

Another interesting point for comparison is an approach that only relies on screen space filtering. These algorithms tend to be very fast and thus a natural choice for real-time applications. Because existing screen space approaches do not work well with large scattering distances as described in Section 4, we compare against the screen space technique described in this section. Although this is by far a perfect candidate for a complete evaluation, the method will show the general shortcomings of screen space only methods very well.

In case we have a PBRT generated ground truth we generated comparison images using structural dissimilarity (DSSIM) [LMCB06].

8.1. Image quality

We put forward four different scenarios that are interesting to compare. In the first scenario we show the necessity for geometry-based subsurface scattering, especially for the materials we want to render. To make this point, we display a scene with a mesh rendered from a perspective with no or little direct illumination. In the second scenario, we demonstrate the opposite case: a mesh rendered from a perspective with full direct illumination. This will prove the necessity of image-based subsurface scattering to create a more detailed image. The third comparison presents the dissimilarities of an exemplary rendering using the compared techniques. Finally, we compare the impact of different tessellation levels on the image quality in cases in which meshes have uneven tessellations. We also provide average DSSIM values in these cases.

Back Lighting Scenario: For materials with lower scattering distances (where image-based methods work well) the effect of geometry-based subsurface scattering is very small and may be neglected for performance reasons. But this is not the case for the materials we focus on. Due to large scattering distances, light may travel through an object completely. Even areas without direct illumination will appear to be glowing. Image-based (especially screen space) methods will have trouble creating this effect either because the geodesic distance between a lit point and a glowing point on the object's surface can be quite large or because the lit points are simply not present in the image. This effect can be seen in Figure 6(a) where the *Indonesian Statue* is lit from behind. While all approaches using splatting (ours, DSS, and the technique by Chen *et al*. [CPZT12]) generate results that match the ground truth equally good, the screen space method cannot correctly catch the appearance of the statue.

In turn, the similarities between the splatting approaches (including ours) originate from the fact that the appearance is completely dominated by the translucency effect in that scenario.

Front Lighting Scenario:. For the directly opposing scenario the results differ substantially. The subsurface scattering algorithm mostly needs to preserve the shadow edges and geometry details of the *Serapis Bust* as seen in Figure 6(b). The view chosen here contains many directly lit areas as, a prominent shadow of the nose, and several smaller shadows of the hair and mouth. Here, the screen space technique can provide for high-quality results very close to the ground truth. Our technique will not use many samples from the geometric approach, because they are facing away from the camera, and mainly rely on the screen space technique. Subsequently, the results will look just as the ones generated by the screen space algorithm. This scenario perfectly visualizes the downsides of geometric approaches. The technique by Chen *et al*. [CPZT12] does comparatively well since the splats are generated from pixels in a shadow map. However, differences in colour of the final image underline the importance of kernel pre-integration as described in Section 3.2. Because of the exponential falloff in the BSSRDF, using the value at the centre of the area as the average results in an underestimation of the actual result. In case of a different falloff for different colour-channels, this effect not only affects the brightness but also the actual colour. Due to the relatively coarse sampling of DSS, shadow edges are not preserved very well with that technique. It also does not provide much of the geometric details present in the ground truth.

Combined Scenario: Both scenarios were very extreme examples. In general, the geometric approach will perform better than the screen space approaches when dealing with materials with large scattering distances. Thus, we provided the per pixel DSSIM values of the rendering of the *Chinese Dragon* in Figure 7. This perspective demonstrates some directly lit regions, regions in shadow, shadow edges, and regions with smaller geometric features. We compare the ground truth image ('PBRT') with images generated by our technique and its competitors. In the right column we show the DSSIM values for the respective method. The perceptive differences of our method compared to the ground truth are nearly invisible (black regions mean no difference). The image rendered with the technique by Chen *et al*. [CPZT12] has some noticeable errors at some contours in the DSSIM image. Our technique on the other hand has an area at the top of the head that shows a small difference. DSS by Nalbach *et al*. [NRS14] provides good results too. The screen space method cannot provide a similar visual quality simply because several surface regions of the dragon that are not visible on screen, contribute substantially to the subsurface scattering. Overall the total dissimilarity for our results is slightly better than all of our competitors as we discuss in Section 8.2.

Effect of Tessellation:. In the last scenario, we want to discuss here, we focus on the effect of tessellation. This effect only applies to our technique because the other techniques either do not use the meshes' triangles for splatting or use an adaptive tessellation that prevents visual artefacts. As a consequence the other techniques suffer from a severe impact on performance which we will discuss in Section 8.2. We will also show examples of the other techniques showing similar artefacts in Appendix. 1.1. There we also explore the problems that occur with the techniques of Chen *et al*. [CPZT12] and Nalbach *et al*. [NRS14] without a sufficiently small area for their splats. We chose the *Monkey Trefoil* model as an exemplar for triangles that have one edge which is shorter than the others. Despite all efforts to reduce the problem, visual artefacts as described in Section 6.3 may occur. The effect can be observed in the image of low tessellation (Figure 8). It shows as bands of a different colour in the image. Due to the technique described in Section 6.3 and the kernel pre-integration we can minimize the visibility of these bands. They are still visible though and without changing the mesh itself we were only able to remove them by using a higher tessellation (which we achieved by creating 6 smaller triangles in the tessellation shader). The high tessellation image in Figure 8 illustrates the results without visible artefacts.

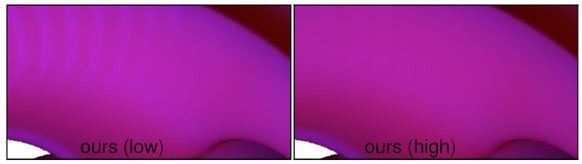


Figure 8: *Influence of tessellation of unevenly distributed triangles on banding artefacts as described in Section 6.3 shown using the* Monkey Trefoil *model. The left column shows our technique using the models original resolution, while the right column shows the results where each triangle was split into six smaller triangles. While in the left column the banding artefacts are clearly visible, in the right column they are not. Rendering resolution:* 1920×1080.

Comparison to Other State of the Art Techniques:. Up to now we only compared our approach to existing splatting approaches and a screen space technique that we created ourself. Since other screen space techniques exist, most notable SSSS [JZJ*15] and SAM [MR17], we want to show an example why we did not do an in-depth comparison with those. First, we want to mention that while both techniques do very well in the parameter ranges they were created for (about the scattering distance of human skin) the images we show here are created with materials outside of this range. We explained before why these techniques are not very suitable for our tasks and will provide some images in Figure 9 supporting this claim. There our result of rendering the *Armadillo* is compared to only the translucency part of SAM [MR17] and also SSSS [JZJ*15]. We also added a combination of the two methods as it is described in SAM [MR17]. It is very obvious that due to the low number of translucency samples SAM [MR17] does not include all translucency. SSSS [JSG09] results show a noisy image due to the breaking of the separation criterion and using a random rotation on the separated kernel.

As an overview of all scenarios we present their average perceptual dissimilarities in all of these as a plot in Figure 10. The figure summarizes all of our previous findings. The results of the *Indonesian Statue* do not have any problems with colours not matching the ground truth. Our technique provides very good results in that case. For the *Serapis Bust* model our results are marginally worse than the screen space results but this difference remains almost unnoticeable.

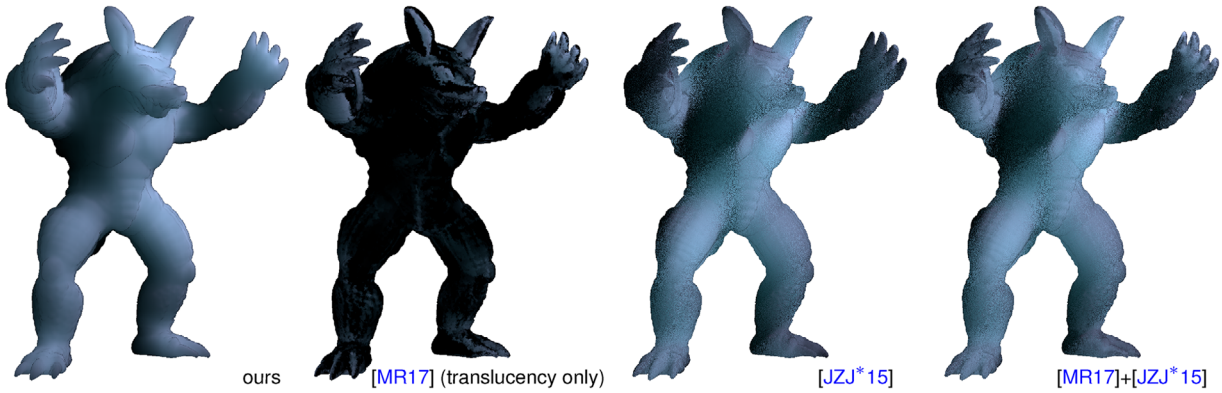


Figure 9: *Comparison of our technique against other state of the art techniques that were not designed for high scattering distances. The second image shows the translucency technique of SAM [MR17]. It is noticeable that there is a strict cutoff that occurs due to the low number of samples used for the effect which is not enough for high scattering distances. In the third image SSSS [JZJ*15] is presented which has some noisy artefacts due to the breaking of the separation criterion. The last image combines the latter as suggested in SAM [MR17].*

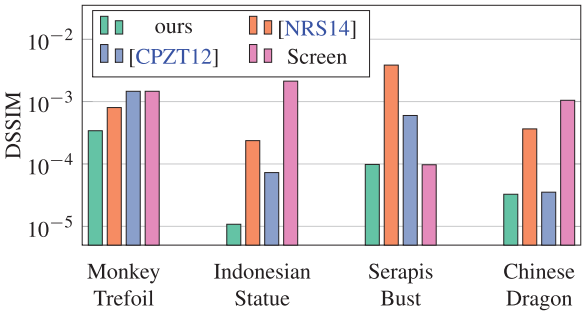


Figure 10: *Average perceptual dissimilarity values (DSSIM) of all scenes compared (small values imply better results). In most cases as the* Monkey Trefoil *(high triangulation),* Indonesian Statue*, and* Chinese Dragon *scenes our technique has the best image quality compared to our competitors. For the* Serapis Bust *Models our technique is slightly worse than one of our competitors but the differences there are marginal as can be seen in the images presented in Figure 6. The technique by Chen* et al. *[CPZT12] always provides convincing results but in a direct comparison with a path traced image errors can be seen that are not present in rendering generated with our approach. In sum, our technique appears to be best choice for visualizing arbitrary scenarios.*

In that case, the problem with colour differences become visible in the DSSIM values very well. Our renderings of the *Chinese Dragon* provide the best image quality of all compared techniques but the technique by Chen *et al*. [CPZT12] provides similarly good results. Finally, for the *Monkey Trefoil* case, our technique also outperforms the competitors.

Other realistic applications of our technique are displayed in Appendix 1.2 where the *Indonesian Statue* and the *Serapis Bust* models are rendered with textures and specular reflections. Figure 1 provides more examples of our technique with specular reflections. In Appendix 1.3 we show an example of environment lighting. In general, our algorithm will provide an overall superior image quality by combining the positive side of all competitors. Additionally all of our competitors have cases where they cannot produce convincing images as illustrated in Figure 6(b).

8.2. Performance evaluation

Up to this point we did not explicitly test for the rendering times to create the images. For a comparison we rendered all of our examples at 1920 × 1080 pixels with an Intel i7-4790 CPU (3.6GHz), 16GB RAM on an NVidia Geforce RTX 2080 Ti graphics card. Due to the nature of the technique, we expect the splatting approaches to be considerably slower than our approach. Both of them will perform well within interactive limits if the rendering resolution is kept quite low for current standards. Chen *et al*. [CPZT12] and Nalbach *et al*. [NRS14] both used a resolution of 800 × 600 for their images. Higher resolutions affect splatting algorithms negatively for two reasons. First, each splat covers the same relative screen area which means more pixels are covered. In addition to that, the areas that generate the splats need to be smaller to avoid visual artefacts similar to the ones shown in Figure 8. This effectively means that more splats are required for the same effect.

The rendering time is presented in Figure 11. Obviously, the screen space technique ('Screen') is always faster than our approach because of the additional time required for the geometry-based part. In comparison with DSS, our technique is always several times faster due to the lower resolution and also because we could remove all samples that face the camera. This drastically reduces the number of fragments we need to calculate the subsurface scattering for. The technique by Chen *et al*. [CPZT12] is faster than DSS because of the more sophisticated way to determine the size of the splats. These numbers are only valid for a single light source though. Our technique and DSS are only influenced marginally by the number of light sources because this will not change the number of splats involved or the screen space filtering. The technique by Chen *et al*. [CPZT12] will use about double the amount of splats for two light sources compared to a single one.

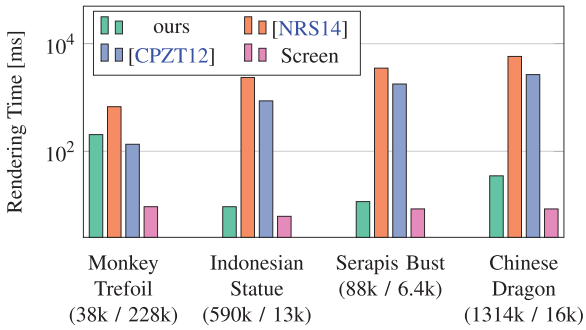


Figure 11: *Comparison of rendering speeds of our algorithm. Each model has the number of triangles of the full resolution model and the version used for the geometry-based subsurface scattering below their names. Usually the last number should be lower than the first one, but for the* Monkey Trefoil *mesh we had to use a higher number as explained in Section 8.1. While the rendering speed of the approach by Chen* et al. *[CPZT12] is the fastest of all competitor splatting techniques, in general our approach is faster due to the fewer number of splats rendered. The results show that the screen space part of our algorithm is not affected much by the number of triangles of the model, and also that we could drastically reduce the impact of DSS for our technique.*

Figure 11 also contains the number of triangles and the number of triangles, that the (usually lower resolution) models have, we used to generate the samples for the geometry-based part of our algorithm. Despite the *Monkey Trefoil* mesh where we needed a very high number of triangles to remove the banding effect seen in Figure 8 both numbers did not have a big effect on the performance. The more important impact comes from the number of pixels covered on the screen which is higher in case of the *Chinese Dragon* compared to the *Indonesian Statue*. Using a huge amount of triangles for the geometry part as in the *Monkey Trefoil* example on the other hand does have a huge impact on the performance and should thus be avoided. In that extreme case the technique by Chen *et al*. [CPZT12] outperforms ours marginally.

Summary: In general we could show that our approach outperforms our competitors in terms of performance and image quality.

9. Conclusions and Future Work

We have presented a technique to render materials with high scattering distances in real time. Our technique does not rely on any precalculations based on scene geometry and thus can be used with animations or deformations. For performance reasons a simplification of the geometry used for the translucency effect can be necessary though. Pre-calculations are necessary for the subsurface scattering kernel that we do pre-integrate as described in Section 7.2. As material changes do usually not occur in real scenes, and the precalculation times are also fast (less than 5 s) this does neither limit the applicability of our algorithm nor affect the loading times of a program much (when calculated at program start). We have shown that our algorithm provides for quality close to that of ray-traced images, and we think the shortcomings of our algorithm in that case can be explained by the approximations that we made in Section 4 in which we average irradiance over a possibly larger area. Keeping these areas small would improve the correctness of our assumptions but will at the same time impact frame times negatively.

While being faster than our strongest competitors (the technique by Chen *et al*. [CPZT12] and DSS), a small downside of our algorithm is that it can not compete in performance with existing subsurface scattering algorithms that work on material configurations with shorter scattering distances like the one presented by Jimenez *et al*. [JZJ*15]. We currently do not see any way to improve the speed other than using models with an even lower triangle count for the geometric approach. Finding a way to separate subsurface scattering kernels of the sizes we need in screen space would also have a strong positive impact on the performance of our technique.

Even though we only used a single BSSRDF model in our approach, our technique is model-agnostic and can be applied to other non-directional models. The choice of of BSSRDF models is only restricted to those that use the same parameters as ours. Directional diffusion models for example can only be used when restricting the direction of incidence of light (usually to the normal). Our model also works well with arbitrary light sources because it does not rely on shadow mapping to work. Overall our technique represents a compromise between image quality, speed and versatility for materials with large scattering distances.

Currently this paper did not present any comparisons with results produced by volume raytracing. Due to general restrictions when using BSSRDF models based on semi-infinite geometry assumptions, we think that there will still be large differences in appearance to images generated with algorithms that do not rely on these assumptions. Finding better BSSRDF models (e.g. multipole-based models that assume a slab-based geometry) and applying them to real-time rendering is still a challenging task and we aim to improve our algorithm in that direction in the future.

Acknowledgements

The authors want to thank Morgan McGuire's Computer Graphics Archive [McG17] for the *Indonesian Statue*, *Serapis Bust*, *Happy Buddha* and *Chinese Dragon* models. We used the *Lucy* and *Armadillo* model provided by the Stanford University Computer Graphics Laboratory. The *Monkey Trefoil* model was created by Carlo Sèquin and is courtesy of UC Berkeley. Open access funding enabled and organized by Projekt DEAL. [Correction added on 6 August 2020, after first online publication: Projekt Deal funding statement has been added]

References

[BL03] Borshukov G., Lewis J. P.: Realistic human face rendering for "The Matrix Reloaded". In *ACM SIGGRAPH 2003 Sketches & Applications* (New York, NY, USA, 2003), SIGGRAPH '03, ACM.

[Bli82] Blinn J. F.: Light reflection functions for simulation of clouds and dusty surfaces. In *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques - SIG-*

GRAPH '82 (New York, New York, USA, 1982), vol. *16*, ACM Press, pp. 21–29.

[Cha60] Chandrasekhar S.: *Radiative Transfer*. Dover Publications, Inc., New York, 1960.

[Chr15] Christensen P. H.: An approximate reflectance profile for efficient subsurface scattering. In *ACM SIGGRAPH 2015 Talks* (New York, NY, USA, 2015), SIGGRAPH '15, ACM, pp. 25:1–25:1. https://doi.org/10.1145/2775280.2792555.

[CLH*08] Chang C.-W., Lin W., Ho T., Huang T., Chuang J.: Real-time translucent rendering using GPU-based texture space importance sampling. *CGF 27*, 2 (1 2008), 517–526. https://doi.org/10.1111/j.1467-8659.2008.01149.x.

[CPZT12] Chen G., Peers P., Zhang J., Tong X.: Real-time rendering of deformable heterogeneous translucent objects using multiresolution splatting. *The Visual Computer 28*, 6 (Jun 2012), 701–711. https://doi.org/10.1007/s00371-012-0704-1.

[DCFMB17] Dal Corso A., Frisvad J. R., Mosegaard J., Bærentzen J. A.: Interactive directional subsurface scattering and transport of emergent light. *The Visual Computer 33*, 3 (Mar. 2017), 371–383. https://doi.org/10.1007/s00371-016-1207-2.

[d'E12] d'Eon, Eugene: *A Better Dipole*. Tech. rep., Nov 2012.

[Deb99] Debevec P.: Light Probe Image Gallery, 1999. http://www.pauldebevec.com/Probes/. Accessed July 2020.

[dI11] D'Eon E., Irving G.: A quantized-diffusion model for rendering translucent materials. *Journal of the ACM 30*, 56:1–56:14. https://doi.org/10.1145/2010324.1964951.

[DJ05] Donner C., Jensen H. W.: Light diffusion in multi-layered translucent materials. *Journal of the ACM 24*, 1032–1039. https://doi.org/10.1145/1073204.1073308.

[DJ07] Donner C., Jensen H. W.: Rendering translucent materials using photon diffusion. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 243–251. https://doi.org/10.2312/EGWR/EGSR07/243-251.

[dL07] d'Eon E., Luebke D.: Advanced techniques for realistic real-time skin rendering. In *GPU Gems 3*, Nguyen H., (Ed.), Addison-Wesley Professional, Upper Saddle River, NJ, 2007, pp. 293–347.

[dLE07] d'Eon E., Luebke D., Enderton E.: Efficient rendering of human skin. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (Aire-la-Ville, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 147–157.

[DS03] Dachsbacher C., Stamminger M.: Translucent shadow maps. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), EGRW '03, Eurographics Association, pp. 197–201.

[ERS13] Elek O., Ritschel T., Seidel H.: Real-time screen-space scattering in homogeneous environments. *IEEE CG&A 33*, 3 (May 2013), 53–65. https://doi.org/10.1109/MCG.2013.17.

[FD17] Frederickx R., Dutré Ph.: A forward scattering dipole model from a functional integral approximation. *ACM TOG 36*, 4 (July 2017). https://doi.org/10.1145/3072959.3073681.

[FHK14] Frisvad J. R., Hachisuka T., Kjeldsen T. K.: Directional dipole model for subsurface scattering. *ACM TOG 34*, 1 (Dec. 2014), 5:1–5:12.

[GZB*13] Gkioulekas I., Zhao S., Bala K., Zickler T., Levin A.: Inverse volume rendering with material dictionaries. *ACM TOG 32*, 6 (Nov. 2013). https://doi.org/10.1145/2508363.2508377.

[HBV03] Hao X., Baby T., Varshney A.: Interactive subsurface scattering for translucent meshes. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2003), I3D '03, ACM, pp. 75–82.

[HCJ13a] Habel R., Christensen P. H., Jarosz W.: *Classical and Improved Diffusion Theory for Subsurface Scattering*. Tech. rep., Disney Research Zürich, June 2013.

[HCJ13b] Habel R., Christensen P. H., Jarosz W.: Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. *CGF 32*, 4 (2013), 27–37. https://doi.org/10.1111/cgf.12148.

[HK93] Hanrahan P., Krueger W.: Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '93* (New York, New York, USA, 1993), ACM Press, pp. 165–174.

[IGAJG15] Iglesias-Guitian J. A., Aliaga C., Jarabo A., Gutierrez D.: A biophysically-based model of the optical properties of skin aging. *CGF 34*, (2015), 45–55.

[JB02] Jensen H. W., Buhler J.: A rapid hierarchical rendering technique for translucent materials. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 576–581.

[JMLH01] Jensen H. W., Marschner S. R., Levoy M., Hanrahan P.: A practical model for subsurface light transport. *SIGGRAPH* (2001), 511–518.

[JSG09] Jimenez J., Sundstedt V., Gutierrez D.: Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception 6*, 4 (Oct. 2009), 23:1–23:15.

[JWSG10] Jimenez J., Whelan D., Sundstedt V., Gutierrez D.: Real-time realistic skin translucency. *IEEE CG&A 30*, 4 (2010), 32–41.

[JZJ*15] Jimenez J., Zsolnai K., Jarabo A., Freude C., Auzinger T., Wu X.-C., von der Pahlen J., Wimmer M., Gutierrez D.: Separable subsurface scattering. *CGF 34*, 6 (Sept. 2015), 188–197.

[LGB*03] Lensch H. P., Goesele M., Bekaert P., Kautz J., Magnor M. A., Lang J., Seidel H.-P.: Interactive rendering of translucent objects. *CGF 22*, 2 (2003), 195–205.

[LMCB06] Loza A., Mihaylova L., Canagarajah N., Bull D.: Structural similarity-based object tracking in video sequences. In *2006 9th International Conference on Information Fusion* (July 2006), pp. 1–6. https://doi.org/10.1109/ICIF.2006.301574.

[McG17] McGuire M.: Computer graphics archive, July 2017. https://casual-effects.com/data.

[MESG11] Munoz A., Echevarria J. I., Seron F. J., Gutierrez D.: Convolution-based simulation of homogeneous subsurface scattering. *CGF 30*, 8 (2011), 2279–2287. https://doi.org/10.1111/j.1467-8659.2011.02034.x.

[MKB*03] Mertens T., Kautz J., Bekaert P., Seidelz H.-P., Van Reeth F.: Interactive rendering of translucent deformable objects. In *Proceedings of the 14th Eurographics Workshop on Rendering* (Aire-la-Ville, Switzerland, 2003), EGRW '03, Eurographics Association, pp. 130–140.

[MR17] Maisch S., Ropinski T.: Spatial adjacency maps for translucency simulation under general illumination. *CGF 36*, 2 (2017), 443–453. https://doi.org/10.1111/cgf.13139.

[NGD*06] Narasimhan S. G., Gupta M., Donner C., Ramamoorthi R., Nayar S. K., Jensen H. W.: Acquiring scattering properties of participating media by dilution. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 1003–1012.

[NRS14] Nalbach O., Ritschel T., Seidel H.-P.: Deep screen space. In *I3D '14: Symposium on Interactive 3D Graphics and Games* (2014), ACM.

[PJH16] Pharr M., Jakob W., Humphreys G.: *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Oct. 2016.

[SH05] Sigg C., Hadwiger M.: Fast third-order texture filtering. *GPU Gems 2* (2005), 313–329.

[SKP09] Shah M. A., Konttinen J., Pattanaik S.: Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE CG&A 29*, 1 (Jan 2009), 66–78. https://doi.org/10.1109/MCG.2009.11.

[Sta95] Stam J.: Multiple scattering as a diffusion process. In *Rendering Techniques '95*, Hanrahan P. M., Purgathofer W., (Eds.), Eurographics. Springer Vienna, Vienna, June 1995, pp. 41–50.

[VKW19] Vicini D., Koltun V., Wenzel J.: A learned shape-adaptive subsurface scattering model. *ACM TOG 38*, 4 (Aug. 2019).

[WMP*06] Weyrich T., Matusik W., Pfister H., Bickel B., Donner C., Tu C., McAndless J., Lee J., Ngan A., Jensen H. W., Gross M.: Analysis of human faces using a measurement-based skin reflectance model. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 1013–1024.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Data S1

Figure 1: Influence of tessellation of unevenly distributed triangles on banding artifacts as described in Sec. 6.3 shown using the *Monkey Trefoil* model.

Figure 2: Examples for textured meshes rendered with our technique.

Figure 3: Examples of our technique using only an environment map for illumination.

Table 1: DSSIM values for different scenes and techniques.

Table 2: Rendering times for different scenes and techniques.

Table 3: Rendering times of each pass of our algorithm. All images are rendered in 1920×1080 and the number of triangles in each mesh is given in the parentheses after the mesh name.

Table 4: Overhead of computational cost for our method.

Table 5: Overview over all scenes used in our paper with exact triangle numbers and physical quantities.