EUROGRAPHICS 2017 / L. Barthe and B. Benes
(Guest Editors)

Volume 36 (2017), Number 2

Spatial Adjacency Maps for Translucency Simulation under General Illumination

S. Maisch† and T. Ropinski‡

Visual Computing Group, Ulm University, Germany

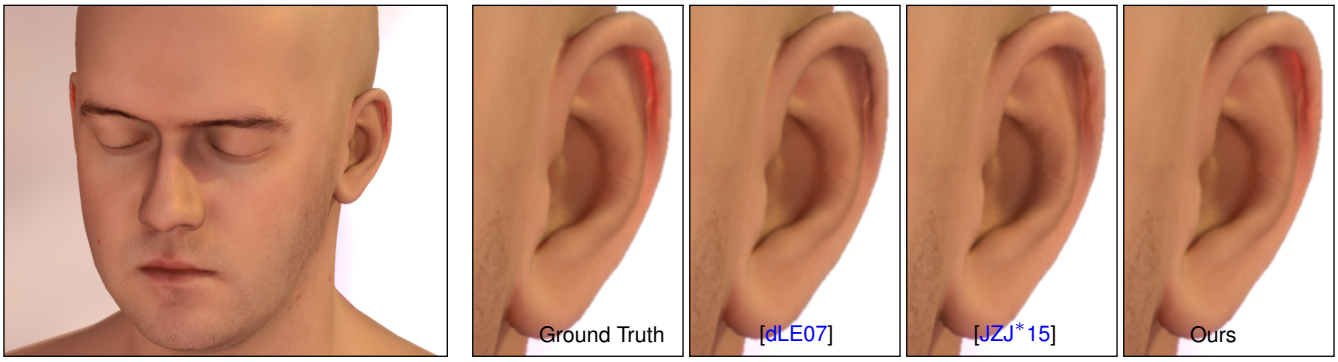

Figure 1: *Our novel translucency technique allows for rendering of realistic translucency effects in real time. This effect can be seen on thin surfaces of translucent materials such as the ears of the human head in the leftmost picture. The other pictures show a close-up view of the ear rendered with different existing real-time techniques compared to the ground truth.*

Abstract

Rendering translucent materials in real time is usually done by using surface diffusion and/or (translucent) shadow maps. The downsides of these approaches are, that surface diffusion cannot handle translucency effects that show up when rendering thin objects, and that translucent shadow maps are only available for point light sources. Furthermore, translucent shadow maps introduce limitations to shadow mapping techniques exploiting the same maps. In this paper we present a novel approach for rendering translucent materials at interactive frame rates. Our approach allows for an efficient calculation of translucency with native support for general illumination conditions, especially area and environment lighting, at high accuracy. The proposed technique's only parameter is the used diffusion profile, and thus it works out of the box without any parameter tuning. Furthermore, it can be used in combination with any existing surface diffusion techniques to add translucency effects. Our approach introduces Spatial Adjacency Maps that depend on precalculations to be done for fixed meshes. We show that these maps can be updated in real time to also handle deforming meshes and that our results are of superior quality as compared to other well known real-time techniques for rendering translucency.

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

1. Introduction

Translucent objects or participating media are commonly encountered in natural environments. All dielectric materials show translucency to some degree which often results in a smooth appearance due to light scattering inside these materials. Depending on how much light is absorbed when passing through a medium and the depth of an object, translucency effects can also appear when light shines through an object. The process leading to these effects is called subsurface scattering. Subsurface scattering is very important for the appearance of materials like skin, marble or candle wax.

† sebastian.maisch@uni-ulm.de
‡ timo.ropinski@uni-ulm.de

However the appearance of non-solid materials, like milk, sea water, smoke, and fog, is also influenced by this process.

Simulating light transport in translucent objects is especially difficult in real time, since non-local contributions need to be taken into account. As we will show, the effect can be split in two parts, a surface and the translucency part. While the surface part can be simulated by exploiting a local neighborhood in texture or screen space, the translucency part cannot be simulated using a 2D local neighborhood and is thus much harder to handle. To investigate the possibilities of interactive subsurface rendering approaches, it is mandatory to refer to the related models described in literature. The optical appearance of dielectric materials is well described by Radiative Transport Theory, as covered in depth by Chandrasekhar [Cha60]. First approaches for rendering subsurface scattering effects were proposed by Blinn [Bli82] for simulating light interaction of clouds and dusty surfaces. Hanrahan and Krueger [HK93] presented a model that included single scattering in homogeneous participating media. While single scattering is an adequate solution for optically thin media, it is not sufficient to render optically thick materials where multiple scattering has a more prominent effect. For these materials, Stam [Sta95] approximates multiple scattering as a diffusion process. This approach assumes a high amount of scattering events happening inside the material so the scattering can be assumed to be isotropic. Using the diffusion process Jensen et al. [JMLH01] created a BSSRDF approximation based on dipoles assuming semi-infinite heterogeneous media.

Based on the technique by Jensen et al. two approaches dealing with the two optical effects of subsurface scattering were developed. Hao et al. [HBV03] as well as Borshukov and Lewis [BL03] rendered the smooth appearance of translucent objects by using texture space blurring. The translucency effect can be rendered in real time using Translucent Shadow Maps (TSMs) introduced by Dachsbacher and Stamminger [DS03]. TSMs use information about the object as seen from a light source to calculate the translucency. All current real-time algorithms combine modified versions of these techniques to render translucent objects, although in particular TSMs have several downsides. They restrict the types of lights used in a scene to point or directional lights. As generalized illumination, especially environment and area lighting is used more frequently in real-time rendering, this becomes a major disadvantage. Another drawback is the number of render targets needed to store the information representing translucency, as well as for rendering high-quality shadows. While the depth component can be used for shadow mapping as well as translucency without any shadow map filtering, if filtering is applied the depth component needs to be stored unfiltered as well, to be able to provide correct translucency information. This is particularly important with shadow mapping techniques which require multiple components in the shadow map. When for instance using Variance Shadow Maps [DL06] (2 components) or Moment Shadow Maps [PK15] (at least 4 components) the combination with information needed for translucency would require a vast amount of render targets. And on top of this, TSMs will still generate inaccurate results.

We see current surface diffusion techniques as an excellent solution for the smoothing effect resulting from subsurface scattering. Unfortunately, the algorithms for rendering translucency are only crude approximations of the real effect. Therefore, within this paper, we introduce Spatial Adjacency Maps (SAMs) which eliminate this approximation, while being combinable with any surface diffusion technique to account for all subsurface scattering. Furthermore, SAMs do not impose any constraints regarding the types of light sources, and do not need TSMs to generate a translucency effect while at the same time creating physically-based results that are more accurate than current interactive techniques. To achieve these goals, a SAM computes and stores connections between vertices that contribute to each other strongly in terms of light transport. Furthermore, SAMs are created in a way that they can be evaluated quickly during rendering to create a realistic, physically-based approximation. When dealing with deforming objects, we exploit spatial and temporal coherence of a mesh between two subsequent animation steps to update the corresponding SAM on the fly.

2. Previous Work

Previous work directly related to our presented technique can be separated into several fields. We will briefly describe general techniques to render participating media. We focus on known methods for dealing with translucency in real-time applications and then mention different diffusion profiles and material parameters required by our method.

2.1. Offline Rendering of Subsurface Scattering

Besides Monte Carlo methods, subsurface scattering can be simulated using multiple techniques, for example photon mapping [DEJ*99] or scattering equations [PH00]. While these methods all provide high quality results, they are not suitable for fast image generation when using highly scattering materials. When dealing with such materials, the dipole diffusion approximation by Jensen et al. [JMLH01] with its various improvements (see Sec. 2.3) can provide fast and accurate results. Rendering speed of this method is further increased by Jensen and Buhler [JB02] but this approach is still not sufficient for real-time applications.

2.2. Translucency

The smoothing effect due to subsurface scattering is addressed by different approaches. D'Eon et al. [dLE07, dL07] use a sum of Gaussians to perform fast texture space blurring in real time for their skin rendering. They fit the Gaussians to closely match any chosen diffusion profile. For the translucency effect, they use a modification of TSMs but reduce the number of components used to just two texture coordinates. These are used to fetch all important properties needed for the translucency calculation. This approach was modified by Jimenez et al. [JSG09] to a screen space solution and finally altered to have only two 1D blurring passes [JZJ*15]. The screen space approaches are faster compared to texture space blurring, as only visible surfaces are taken into account, but introduce artifacts, as blurring around object edges into invisible areas is not possible. For translucency, these techniques further simplify TSMs to store only the depth by making several assumptions about the object's back side. While reducing the problems of TSMs regarding the amount of information stored, these modifications still suffer from incorrect translucency as the samples taken from the TSM might be too far away from the corresponding points on the

rendered side to have any real effect, while closer samples are not considered.

Lensch et al. [LGB*03] also use a vertex-based approach for rendering subsurface scattering at interactive speeds. They discriminate between a local response that is calculated by texture filtering and a vertex-based global response. They also exploit a precalculation step on a triangle fan level, but in contrast to our approach they do not discriminate between vertices that lie on different sides of an object. This way they need to consider all vertices within a certain distance to each other and need to do a special blending between these responses to avoid a doubled contribution. In contrast, our approach will take into account spatial configurations and orientations between vertices to separate surface blurring and vertex to vertex throughput. Furthermore, as already mentioned by Mertens et al. [MKB*03], Lensch et al. use a matrix multiplication to compute the global response which might not be fast enough for real-time applications in scenarios with high vertex counts. Mertens et al. use a low resolution mesh to calculate form factors [MKB*03] similar to ours, but they do not discriminate between translucency and surface diffusion. Therefore, their results do not show smoothed details as do techniques that use a surface diffusion approach. They are also restricted to diffusion profiles generated using the classical dipole approximation and do not profit from more physically accurate models.

Spherical harmonics as described by Sloan at al. [SKS02] can also be used to precompute subsurface scattering. The general technique can not be trivially extended to deforming objects without major recalculation of the coefficients. Several techniques, for example by Sloan et al. [SLS05], are available to handle this downside. Although, subsurface scattering produces low frequency light, effects like self shadowing in thin regions are hard to encode properly with only few a spherical harmonics coefficients. Using more coefficients may result in vast memory usage and performance issues as hardware support for matrices with a large number of columns/rows is limited. Subsurface scattering using spherical harmonics is also hard to combine with surface blurring methods, which are a widely accepted practical solution for the smoothing effect.

Nalbach et al. [NRS14] propose another method for subsurface scattering. They use surfels to represent a scene's geometry and can thus access information in screen space that would be occluded otherwise. They use these surfels for several global illumination effects by applying their illumination contribution to all pixels in the framebuffer covered by them. Their technique can be applied to dynamic scenes as well, and it provides good results for several effects at the same time. As the authors state themselves the effect is computed in a similar way as suggested earlier by Jensen and Buhler [JB02]. Just like other methods they do not discriminate between translucency and surface diffusion with similar downsides as the techniques by Mertens et al. [MKB*03] or Sloan et al. [SKS02, SLS05].

A recently released paper by Shinya et al. [SDS*16] offers an efficient solution especially for optically thin participating media. Unfortunately, this technique is not feasible for current real-time applications. It is also not clear how their technique performs in case of optically dense media which is the focus of our approach.

2.3. Diffusion Profiles

Several similar techniques are known today as an enhancement to the dipole approximation first introduced by Jensen et al. [JMLH01]. All of these models are based on diffusion theory as an approximation to the radiative transport equation. The original dipole model is an exact solution to the diffusion model for semi-infinite homogeneous media and introduces errors especially for thin slabs. Donner and Jensen [DJ05] present a multipole technique that describes light transport through multi-layer materials, which are especially applicable for rendering human skin. This multipole approximation can also provide correct results for slab geometries where the original dipole model failed. D'Eon and Irving [DI11] propose a quantized diffusion model that takes into account single scattering effects also for multi-layer materials, resulting in even higher image quality. All these techniques produce radial-symmetric diffusion profiles that can be used directly and – approximated as a sum of Gaussians – applied even to real-time applications. Our method supports arbitrary diffusion profiles and can thus benefit from the superior quality of all these profiles.

Dipole approximations resulting in non-radial symmetric diffusion profiles, i. e. models assuming a normally incident light, as for example proposed by Donner et al. [DLR*09] or Frisvad et al. [FHK14], can in theory also be used with our technique. However, the surface blurring then needs to be done with a 2-dimensional filter kernel, which is not fast enough in practice for real-time applications.

Material parameters for rendering of participating media were measured by Jensen et al. [JMLH01] and Narasimhan et al. [NGD*06]. Especially for skin rendering Weyrich at al. [WMP*06] fitted a BRDF and BSSRDF to their measured lighting. For the same purpose, Iglesias-Guitian et al. [IGAJG15] created a biophysically based model for generating diffusion profiles from parameters such as age, gender, or skin type.

In this paper we use diffusion profiles generated by Jimenez et al. [JZJ*15] based on the values measured by Jensen et al. [JMLH01] for the milk and marble materials. For the skin material we use the diffusion profiles proposed by D'Eon and Irving [dLE07] and the reflectance parameters measured by Weyrich et al. [WMP*06].

3. Vertex Based Translucency

In this section we will discuss the known equation of subsurface light transport in Sec. 3.1 and rearrange it to be suitable for our simplifications detailed in Sec. 3.2. We will then describe how exactly we discriminate contributions of surface diffusion and translucency in Sec. 3.3 before we describe in Sec. 3.4 how we handle deformable meshes.

3.1. Radiative Transport Derivation

For an exact simulation of subsurface scattering in an object, an integral over the object's surface needs to be solved as described by the BSSRDF model by Nicodemus et al. [NRH77]. Solving this integral is usually done by using Monte Carlo integration. The downside of this method is the number of samples required for a noise

free solution which makes this method unsuitable for real-time applications. In our approach we use the spatial information provided by the existing triangle and vertex structure of a triangle mesh to approximate the surface integral used in the BSSRDF model. This allows us to find a reasonable approximation using information generally available in real-time rendering. The derivations presented in this subsection are inspired by the derivations made by Mertens et al. [MKB*03].

Using the BSSRDF model the outgoing radiance L_o, due to subsurface scattering at exit point $\mathbf{x}_o$ and exit direction $\vec{\omega}_o$, is defined by the surface integral over the object's surface A of the incident radiance L_i at entry points $\mathbf{x}_i$ and directions $\vec{\omega}_i$.

$$L_o\left(\mathbf{x}_o,\vec{\omega}_o\right)=\int_A\int_{2\pi}S\left(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o,\vec{\omega}_o\right)L_i\left(\mathbf{x}_i,\vec{\omega}_i\right)\left(\vec{n}\cdot\vec{\omega}_i\right)\mathrm{d}\vec{\omega}_i\,\mathrm{d}\mathbf{x}_i \tag{1}$$

The BSSRDF S is originally defined by a diffusion profile R_d (see Sec. 2.3) and the Fresnel transmission F_t at entry and exit points of the light.

$$S\left(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o,\vec{\omega}_o\right)=\frac{1}{\pi}F_t\left(\mathbf{x}_i,\vec{\omega}_i\right)R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)F_t\left(\mathbf{x}_o,\vec{\omega}_o\right) \tag{2}$$

Donner and Jensen [DJ05] adjusted this model to diffuse surfaces using a diffuse reflection factor ρ_{dt} to replace the Fresnel transmission terms. The diffuse reflection factor is defined by the light not reflected in any direction by the surface BRDF.

$$\rho_{dt}\left(\mathbf{x},\vec{\omega}_o\right)=1-\int_{2\pi}f_r\left(\mathbf{x},\vec{\omega}_o,\vec{\omega}_i\right)\left(\vec{\omega}_i\cdot\vec{n}\right)\mathrm{d}\vec{\omega}_i \tag{3}$$

We will use these equations to combine all light at each entry point that is subject to the diffusion process as L_d.

$$L_d\left(\mathbf{x}_i\right)=\int_{2\pi}L_i\left(\mathbf{x}_i,\vec{\omega}_i\right)\rho_{dt}\left(\mathbf{x}_i,\vec{\omega}_i\right)\left(\vec{\omega}_i\cdot\vec{n}\right)\mathrm{d}\vec{\omega}_i \tag{4}$$

This allows us to simplify Equation 1 by removing constant parts from the integral leaving it with only information locally available at $\mathbf{x}_i$ and the distance to the $\mathbf{x}_o$:

$$L_o\left(\mathbf{x}_o,\vec{\omega}_o\right)=\frac{1}{\pi}\rho_{dt}\left(\mathbf{x}_o,\vec{\omega}_o\right)\int_A R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)L_d\left(\mathbf{x}_i\right)\mathrm{d}\mathbf{x}_i \tag{5}$$

3.2. Form Factor Calculation

Although Mertens et al. also use form factors to calculate subsurface scattering, our approach differs from theirs as they store light transport from triangles to vertices while we use a local area around vertices for light transport calculation. This allows us to directly use available vertex information later in rendering. One of the goals of our technique is to separate the light contributions due to the surface diffusion from those due to translucency. We do this by defining a 'front' and 'back' side of an object at each surface point (see Sec. 3.3). Contributions from the front will be handled by a surface diffusion technique whereas our translucency technique will be applied to the contribution from the back. Therefore, we split the surface area A into a front surface A_+ and a back surface A_- where $A=A_+\cup A_-$ and $A_+\cap A_-=\emptyset$. With this definition we can split the integral from Eqn. 5 to one that evaluates the front side (L_{o+}) and one that evaluates the back side (L_{o-}).

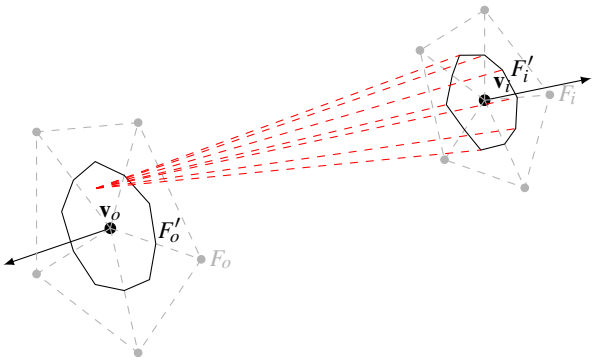

Figure 2: *Exemplary front and back side triangle fans (*dashed grey*) around $\mathbf{v}_o$ and $\mathbf{v}_i$ and their reduced fans (*black*) used for our light transport calculations. We calculate the light transport (*dashed red*) between the reduced triangle fan at the back side to each point inside the reduced fan area at the front side. The average of this value over the front side area is used as the form factor.*

$$L_{o+}\left(\mathbf{x}_o\right)=\int_{A+}R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)L_d\left(\mathbf{x}_i\right)\mathrm{d}\mathbf{x}_i \tag{6}$$

$$L_{o-}\left(\mathbf{x}_o\right)=\int_{A-}R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)L_d\left(\mathbf{x}_i\right)\mathrm{d}\mathbf{x}_i \tag{7}$$

The final radiance can be calculated using the terms from Eqn. 6 and 7 as $L_o\left(\mathbf{x}_o,\vec{\omega}_o\right)=\frac{1}{\pi}\rho_{dt}\left(\mathbf{x}_o,\vec{\omega}_o\right)\left(L_{o+}\left(\mathbf{x}_o\right)+L_{o-}\left(\mathbf{x}_o\right)\right)$, whereby L_{o+} can be evaluated using a surface blur as described by several techniques mentioned in Sec. 2.

However, as the computation of the translucency effect is more challenging, in the remainder we focus on term $L_{o-}\left(\mathbf{x}_o,\vec{\omega}_o\right)$. We assume the surface A to be a triangle mesh with vertices $\mathbf{v}\in V\subset A$. We split the set of vertices V in the same way we split the surface A, to get two subsets $V_+\subset A_+$ at the front side and $V_-\subset A_-$ at the back side for each surface point. Each vertex $\mathbf{v}$ of these subsets is surrounded by a triangle fan F. We can divide each triangle of F into three parts of equal surface area that are associated with each vertex of this triangle. We now define F' as the reduced triangle fan that consists only of the parts that are associated with $\mathbf{v}$. The area of F' will be $\hat{F}'$. With this we convert the integral over the back surface in Eqn. 7 into a sum of integrals over reduced triangle fans (see also Fig. 2).

$$L_{o-}\left(\mathbf{x}_o\right)=\sum_{\mathbf{v}_i\in V_-}\int_{F'_i}R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)L_d\left(\mathbf{x}_i\right)\mathrm{d}\mathbf{x}_i \tag{8}$$

We use barycentric coordinates to easily find all points on a triangle that are closer to one vertex than to any of the others to create this reduced triangle fan. Due to the very low frequency light transmitted through an object and assuming a reasonably high tessellation we can approximate this by a constant irradiance L_d on each reduced triangle fan F'_i.

$$L_{o-}\left(\mathbf{x}_o\right)\approx\sum_{\mathbf{v}_i\in V_-}L_d\left(\mathbf{v}_i\right)\int_{F'_i}R_d\left(\|\mathbf{x}_i-\mathbf{x}_0\|\right)\mathrm{d}\mathbf{x}_i \tag{9}$$

Our goal is now to find a good estimation for this integral at each

vertex $\mathbf{v}_o$ instead of at any point $\mathbf{x}_o$, whereby each vertex is a representative for all points on the associated F_o'. To get more accurate results, we average all contributions from a back side fan over the reduced fan's area $\hat{F}_o'$ around the vertex $\mathbf{v}_o$ as shown in Figure 2, where you can see the triangle fans around two exemplary vertices and the associated F' we use in our calculations. Light transport from the associated F_i' around $\mathbf{v}_i$ to a single point of the associated F_o' around $\mathbf{v}_o$ is also outlined. We introduce form factors K_{oi} that describe the light transport between the reduced fan around $\mathbf{v}_i$ and the reduced fan around $\mathbf{v}_o$.

$$\bar{L}_{o-}(\mathbf{v}_o) = \sum_{\mathbf{v}_i \in V_-} L_d(\mathbf{v}_i) K_{oi} \tag{10}$$

$$K_{oi} = \frac{1}{\hat{F}_o'} \int_{F_o'} \int_{F_i'} R_d\left(\|\mathbf{x}_i - \mathbf{x}_o\|\right) \mathrm{d}\mathbf{x}_i \, \mathrm{d}\mathbf{x}_o \tag{11}$$

We approximate this sum by associating to each vertex $\mathbf{v}_o$ n vertices $\mathbf{v}_i[\mathbf{v}_o;k] \in V_-$ $(k \in [0 \dots n-1])$ on the object's back side with respect to $\mathbf{v}_o$. We chose the vertices $\mathbf{v}_i[\mathbf{v}_o;k]$ so that the terms K_{oi} are the greatest n among all back side vertices. This way we find the vertices that contribute most to the translucency effect we want to achieve. In practice we can choose a rather small n as we can restrict the contribution to few vertices since diffusion profiles show an exponential falloff in distance and thus most vertices will have no actual contribution to the effect. The final light contribution for translucency is then calculated by using only the three form factors K_{ok} associated with the vertices $\mathbf{v}_i[\mathbf{v}_o;k]$. As the form factors only depend on the meshes' geometry and the diffusion profile and not on the incoming light, we can precalculate them using Monte Carlo integration.

$$\tilde{L}_{o-}(\mathbf{v}_o) = \sum_{k=0}^{n} L_d\left(\mathbf{v}_i[\mathbf{v}_o;k]\right) K_{ok} \tag{12}$$

Thus, during rendering we evaluate Eqn. 12 to approximate the translucency using the form factors while evaluating L_d directly.

3.3. Mesh Surface Splitting

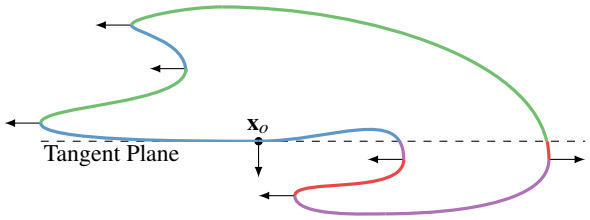

Figure 3: *The back and front sides of an object for point $\mathbf{x}_o$ in a 2-dimensional case. The areas marked with* blue *color are considered front side because they do not pass the criterion defined by Eqn. 14. The areas marked with* red *color are considered front side because they do not pass the criterion defined by Eqn. 13. Areas with* purple *color do not pass any of those conditions. The areas marked with* green *color are considered back side. The dashed line is the tangential hyperplane defined by the normal at $\mathbf{x}_o$; other normals are drawn to show the positions where the general orientation changes in respect to $\mathbf{x}_o$.*

As mentioned above we split an object's surface into front and back surface to be able to split the contributions due to surface diffusion and translucency. Therefore, we define some criteria that describe surfaces suitable to contribute to the translucency effect. These surfaces highly depend on the point at which we want to simulate the effect $\mathbf{x}_o$, so the distinction we make also needs to depend on this point. As a first criterion we need all suitable points to be on the negative side of the tangential plane at $\mathbf{x}_o$ defined by its normal vector $\vec{n}_o$ (see Eqn. 13). We also need to ensure that normal vectors for points considered on the back side are pointing in a generally opposing direction (see Eqn. 14).

$$(\mathbf{x}_i - \mathbf{x}_o) \cdot \vec{n}_o < 0 \qquad \text{and} \tag{13}$$

$$\vec{n}_i \cdot \vec{n}_o < 0 \tag{14}$$

Thus, a point $\mathbf{x}_i$ is considered to be on the back side of the object in respect to $\mathbf{x}_o$ if the two criteria in Eqn. 13 and 14 apply. The front side is then defined by all points that do not pass these criteria. Fig. 3 shows a 2 dimensional object with all different configurations of these criteria.

3.4. Mesh Deformation

The approach described above has severe limitations when used with deformable meshes as the precalculated form factors would have to change over time. There are two reasons for this that need to be addressed separately. The first reason is the deformation of the reduced fans F_o' and F_i' and the change in their relative positions. The second reason is due to the fact that we only use the n back side vertices with the highest contribution and their form factors. The choice for these vertices might also change when deforming a mesh, so new vertices need to be found.

Our solution to both of these problems is to use vertex adjacency information to be able to update precalculated data on the fly. We cannot do the required Monte Carlo integration in real time but we can use previous results to approximate a solution. To do this we compute constants that we can use to approximate the form factors described in Eqn. 11 by removing information that will change with deformation. We do this by introducing another approximation. When assuming the distance between each two points on the reduced triangle fans F_o' and F_i' being constant Eqn. 11 is reduced to $\tilde{K}_{oi} = \hat{F}_i' R_d\left(\|\mathbf{v}_i - \mathbf{v}_o\|\right)$ which can be calculated during rendering. We calculate the constants to approximate our form factors by storing only the relation to the approximated form factors:

$$K_{oi}' = \frac{K_{oi}}{\hat{F}_i' R_d\left(\|\mathbf{v}_i - \mathbf{v}_o\|\right)} \quad . \tag{15}$$

With these constants K_{oi}' we need to recalculate the reduced fan area and the diffusion profile for each frame to get the approximated form factors, which can be done trivially by using vertex adjacency. Information about the back side vertices is available directly through our SAMs which will be described in Sec. 4.1. Using this information the R_d term can be evaluated. For the reduced fan area $\hat{F}_i'$ we also need information of all adjacent vertices $\mathbf{v}_{ij}$ of each vertex $\mathbf{v}_i$.

$$\hat{F}_i' = \sum_{j=0}^{n} \frac{\left\|\left(\mathbf{v}_{ij-1} - \mathbf{v}_i\right) \times \left(\mathbf{v}_{ij} - \mathbf{v}_i\right)\right\|}{6} \tag{16}$$

To update the back side vertices used to calculate translucency,

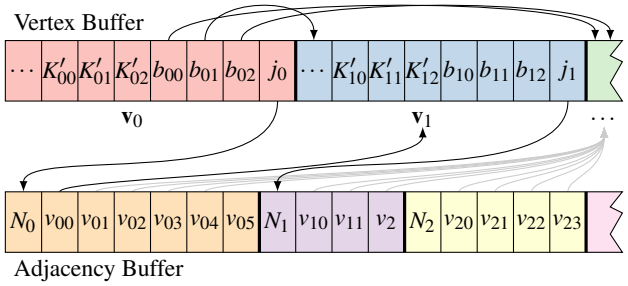

Figure 4: *Schematic representation of a Spatial Adjacency Map. The thick lines are drawn as delimiters of different vertices. The buffer at the top (Vertex Buffer) contains regular vertex attributes (...) as well as the constants for form factor calculation (K'_{ik}) and indices (b_{ik}) to three back side vertices for each vertex $\mathbf{v}_i$. It also contains an index (j_i) to the buffer at the bottom (Adjacency Buffer) containing the number of adjacent vertices (N_i) for each vertex and indices to those vertices in the Vertex Buffer (v_{il}). The arrows show the destination of these indices.*

we exploit the fact that changes in the meshes' geometry are only very small between two subsequent frames. Because of this we can merely check the adjacent vertices of the vertices that previously had the highest contribution for better ones.

3.5. Limitations of Approximations

The assumptions we use for calculating our form factors are correct as long as there is a constant irradiance across the reduced triangle fan around a vertex. The smaller a triangle fan is, the more likely this assumption will become true. On the other hand we try to approximate the sum in Eqn. 12 with as few vertices on the back side (samples) as possible. To get a correct overall result we need to cover the complete back side area contributing a relevant part to the translucency effect with the reduced triangle fans. This can be achieved with either a higher number of samples or larger triangles relative to the materials mean free path. To put it in another way, with constant material parameters, triangulation and, sample count the correct result also depends on the spatial size of the model.

Our approximations regarding mesh deformations are a bit heavier. When calculating the approximated form factors we effectively scale the precalculated ones linearly by changes in the diffusion profile and reduced fan area. This is not correct for major changes in the fan area or the angle at the fan's center. Also due to the fact that we use adjacency information of the triangle mesh we would need to update this information if the topology of the mesh changes. Because of this we only consider animation which does not affect the mesh's topology.

4. Technical Realization

In the previous section we described our technique in theory. We will now go into detail how we can realize our real-time goals using this information, whereby we exploit the capabilities of modern GPUs.

4.1. Spatial Adjacency Map Structure

To be able to access all relevant information for rendering and animation on the graphics card, we use a new data structure that we call Spatial Adjacency Map (SAM). A SAM is an extension of a regular vertex buffer with indices to back side vertices and adjacency information. We save for each vertex ($\mathbf{v}_i$) indices to the n vertices with the highest contribution (b_{ik}) and the corresponding constants for form factor calculation (K'_{ik}) together with an index to an adjacency buffer (j_i). This buffer contains for each vertex the number of adjacency vertices (N_i) and indices to access them in the first buffer (v_{il}, $l \in [0,\dots,N_i-1]$). We found that a reasonable number for n is to use 3 vertices. This provides us with high quality results in most cases (see Sec. 4.4) while still keeping the computation times low. We show the layout of our Spatial Adjacency Map in Figure 4.

4.2. Spatial Adjacency Map Generation

Creating the adjacency information is straightforward. We just iterate the indices used to render a triangle mesh to find all adjacent vertices for each vertex. This information then needs to be concatenated to create the final adjacency buffer.

Finding the three vertices with the highest contribution is also simple to implement. Unfortunately, we need to process all vertices in V_- to find the best ones which can take a lot of time. A naïve implementation will be $\mathcal{O}(m^2)$ for m being the number of vertices in a mesh. We can speed this up by using a spatial tree (R-Tree in our implementation) to consider only back side vertices that can actually contribute to the translucency (based on their Euclidean distances and the material parameters).

4.3. Real-Time Rendering

The general rendering algorithm requires four passes which is the same number needed by Separable Subsurface Scattering [JZJ*15]. In general the number of required passes mainly depends on the used surface diffusion technique. Using the indices stored in our SAMs we can access all information needed to do a complete light calculation at the three back side samples for each vertex. With this we can calculate the L_d term from Eqn. 10 for point or directional lights directly. Global illumination and area lighting can be added using any suitable real-time algorithm. We use a preintegrated environment map in our implementation. The result is then multiplied with the stored form factors. To evaluate ρ_{dt} we use a lookup texture. The translucency is added to the irradiance also calculated in that pass to be blurred and combined with directly reflected light in a final pass.

When rendering the translucency we encountered some flickering that occurred when single vertices at the mesh's back side switched from a lit region to a shadowed region. This problem could be reduced by using more than three samples at the mesh's back side. To limit the number of samples needed we use another approach to this problem. We do the calculation of the translucency on a per vertex basis as a pre-pass in a compute shader and perform a linear interpolation between the final results from the previous

frame $\tilde{L}_{o-}^{(n-1)}$ and the currently calculated translucency $\tilde{L}_{o-}$:

$$\tilde{L}_{o-}^{(n)} = (1-\alpha)\,\tilde{L}_{o-}^{(n-1)} + \alpha \tilde{L}_{o-} \tag{17}$$

$$\alpha = 1 - \left(10^{-10}\right)^{\Delta t} \tag{18}$$

We use the duration for rendering a single frame Δt to normalize this interpolation in time. We found the constant 10^{-10} used in Eqn. 18 to be a good compromise that strongly reduces the flickering while not adding visible temporal artifacts.

4.4. Practical Limitations

To be able to store the information described in Sec. 4.1 efficiently we restrict the number of samples at the meshes back side to $n = 3$. The size of the vertex buffer will grow by 4 indices, 3 for the back side vertices and another one for the adjacency buffer, and 4 color values, 3 constants for form factor calculation and a fourth color value to store translucency results from a compute shader (see Sec. 4.3). The area of the reduced fan around a vertex is stored in the alpha component of one of the three constants. This adds in total 80 bytes for each vertex. When using position, normals, tangents, binormals and a single set of texture coordinates this will about double the size of the original vertex buffer. We believe this is still reasonable as the vertex data is most likely still a lot smaller than the texture data used for rendering. The size of the adjacency buffer depends on the triangulation of the mesh but will most likely not add more than another 40 bytes for each vertex.

In our evaluation we found that depending on the mesh and material data, 3 samples might not always be enough to produce correct results. Especially a configuration with a small model compared to the mean free path and at the same time a high triangulation will need more samples for correct results. We go into this problem further in Sec. 5.2 where we use a modification of the SAMs that is able to use arbitrary numbers of samples.

Another limitation of our technique is the precomputation. Using a spatial tree, this can be done quite fast for most meshes but will take a lot of time for some special cases (see Sec. 5.3 for actual timings). Especially for highly tessellated meshes that are small in comparison to the materials mean free path the precalculation is not practical. As these cases also have problems with image quality a direct application of our algorithm is not feasible. To tackle these problems we create a proof of concept that works on a lower triangulation for back side information. We describe this adjustment and show that it will help with the downsides described here in the Appendix.

5. Evaluation

To demonstrate the image quality of our approach we have compared it to four other techniques differing in the way translucency is handled. To obtain reference images, we have implemented an offline ground truth for translucency as described further in Sec. 5.1. We also use the approach proposed by D'Eon et al. [dLE07] which can unfortunately not be used with all of our meshes as it relies heavily on a continuous texture space parametrization to produce visually correct results. The other real-time technique we compare against is Separable Subsurface Scattering by Jimenez et al. [JZJ*15], which is one of the fastest real-time techniques available for subsurface scattering. To create a translucency effect the authors use a crude approximation that still produces visually acceptable results. For the surface diffusion effect we use Separable Subsurface Scattering for our technique as well as for the ground truth. This has, as all screen space techniques, downsides which are explained by Jimenez et al. [JSG09]. We can ignore these for the comparison as our translucency approach can be combined with both screen space and texture space algorithms depending on the quality and performance requirements and also the available mesh data. For image comparison we have chosen six different models. We use the *Head* model already shown in Fig. 1 to demonstrate our technique's performance for skin rendering which is one of the most important use cases for subsurface scattering. Our *Milk* model is an example geometry generated by fluid simulation with several small blobs disconnected from each other. The *Happy Buddha* model is an example for quite complicated connected geometry while the *Chinese Dragon* is used as an example of the limitations of our algorithm. The discussion of the *Chinese Dragon* model can be found the Appendix. We use the *Heptorioid* as an example for very thin structures containing lots of self shadowing. Finally we use a model we call *Animated* which is a simple model that is deformed at run time to show our method's applicability to scenarios involving deforming meshes. We use different fitting materials for rendering our models. These are *skin*, *milk* and *marble*. In all renderings we use environment as well as direct lighting.

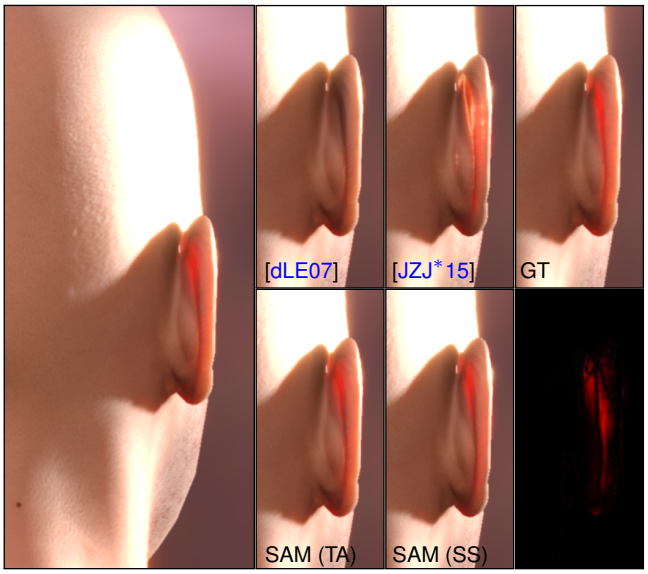

Figure 5: *Rendering of the* Head *model with stronger front lighting. Our technique is shown with screen space diffusion (SS) and texture space diffusion (TA). The difference between these approaches can be seen in the lower part of the ear where the texture space technique shows a smoother effect. The image in the lower right shows the difference between our technique with texture space diffusion and the ground truth enhanced by factor 4.*

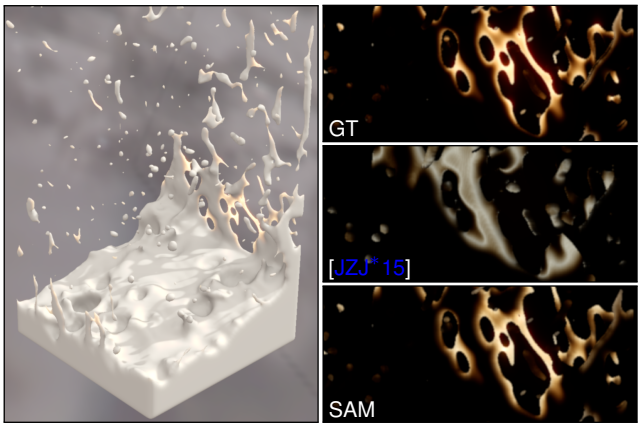

Figure 6: *The images show the* Milk *dataset with a single white light and environment map illumination. The detail images show only the translucency effect to demonstrate the high similarities between our technique and the ground truth. The technique by Jimenez et al. produces incorrectly colored translucency.*

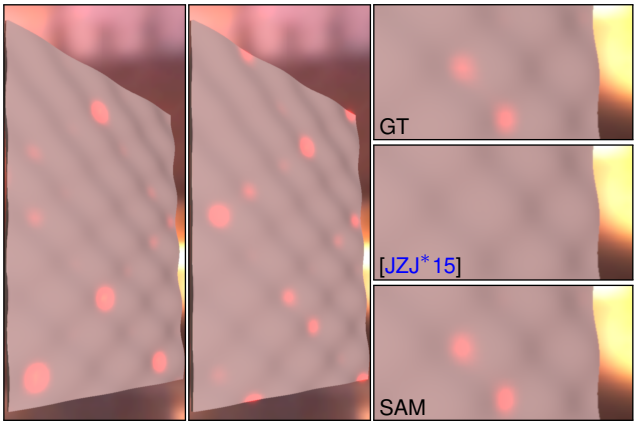

Figure 7: *Two animation steps of the* Animated *dataset are shown. The detail images show high similarities to the ground truth. The technique by Jimenez et al. does not show any translucency in these images.*

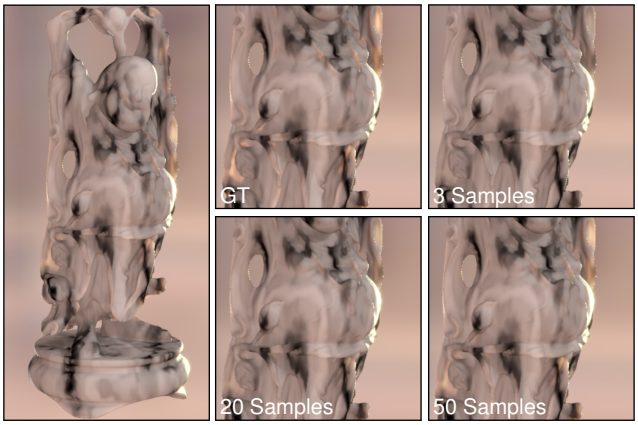

Figure 8: *The* Happy Buddha *model shows our method's downsides with high tessellated geometry. When rendering with only 3 samples our result shows no translucency. Using more samples our technique converges towards the ground truth (GT). Detail pictures for 3, 20 and 50 samples are shown.*

5.1. Ground Truth

We create our ground truth by using the same splitting of front and back sides as proposed in Eqn. 6 and 7. This splitting allows us to create the surface diffusion effect in the same way as it is done in all other compared techniques so the differences will only be in the translucency effect. We perform the surface diffusion in texture space if a continuous parametrization is available and fall back to screen space if not. The translucency is calculated by accumulating the term inside the integral of Eqn. 7 for all pixels in texture space that fit our back side criteria. This sum is weighted by the surface area the pixel will have in world space. The weighted sum is the solution for $L_{o-}(\mathbf{x}_o)$ and results in a very well sampled translucency contribution that we can parametrize using texture coordinates and use when rendering an image. Using the solution for $L_{o+}(\mathbf{x}_o)$ obtained by the surface diffusion technique, we can calculate all contributions due to subsurface scattering.

5.2. Image Comparison

For a qualitative evaluation we have compared images rendered with the algorithms mentioned above in different configurations. The *Head* model is the only one with a (nearly) continuous parametrization so we can include the algorithm by D'Eon et al. [dLE07] in our comparison, as it needs to calculate surface diffusion in texture space. Due to the model's shape, translucency will be visible mostly in thinner regions, especially the ears. In Fig. 1 we show four zoomed in renderings of the left ear created using all four compared techniques. The *ground truth* (GT) shows clearly visible translucency effects in the upper part of the ear, while the compared rendering techniques show only a low contribution which is nearly invisible due to the bright environment lighting. Our technique shows a visible contribution that is similarly shaped and only slightly lower than the *ground truth*. Fig. 5 shows another perspective of the *Head* model with stronger front lighting. The translucency effect is seen very clearly in this scenario. While our technique shows only little difference to the *ground truth*, the differences to other techniques are very noticeable. The technique by D'Eon et al. again shows nearly no translucency while the technique by Jimenez et al. [JZJ*15] shows issues with their TSM adaption and an incorrect coloring of the translucency effect. Differences between texture space diffusion and screen space diffusion are also noticeable in these images. While the results generated using texture space diffusion (our texture space technique, the technique by D'Eon et al., and the ground truth) show a more smooth light distribution in the lower part of the ear, our screen space technique and Separable Subsurface Scattering (Jimenez et al.) does not have that effect. We rendered our *Milk* model w.r.t. different lighting conditions to show the stability of our algorithm

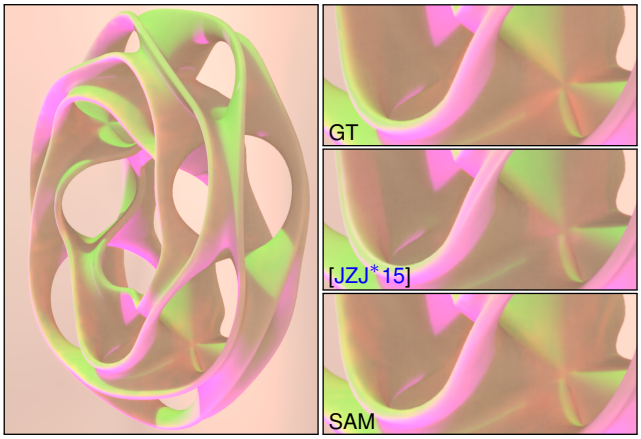

Figure 9: *The* Heptoroid *model is another example of more complex geometry. It is lit by green and pink light sources and the environment. Translucent parts appear in red due to the material properties. Our method captures translucency in several regions that the technique by Jimenez et al. does not cover as can be seen in the detail images.*

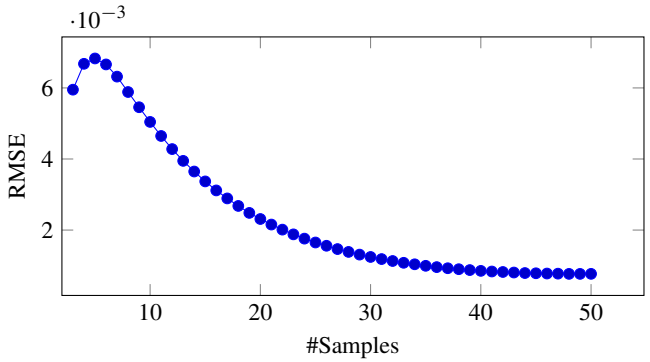

Figure 10: *Convergence of our technique using multiple samples. We used the* Happy Buddha *model with different sample numbers (3-50) and show the root mean squared error (RMSE) of the result images compared to the* ground truth.

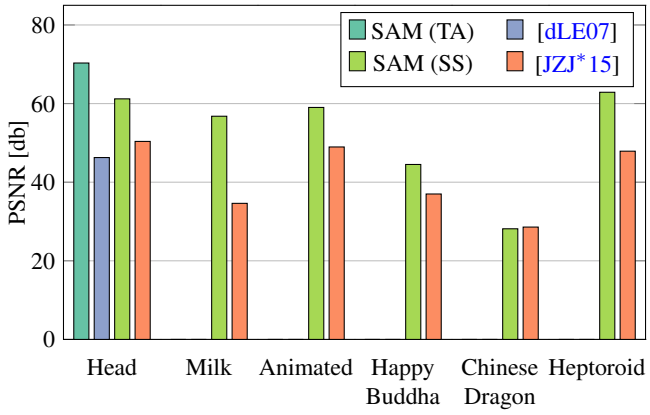

Figure 11: *PSNR values for renderings presented in this paper (*Head*: Fig. 5,* Milk*: Fig. 6,* Animated*: Fig. 7,* Happy Buddha*: Fig. 8,* Chinese Dragon*: Appendix, Fig. 2,* Heptoroid*: Fig. 9) compared. In all cases but the* Chinese Dragon *our technique outperforms the other tested techniques clearly. For the* Chinese Dragon *we added further evaluations for multiple sample counts and lower resolution meshes for these samples in the Appendix.*

with different scenarios. The model does not have a continuous texture parametrization so we use screen space surface diffusion for all algorithms (including the ground truth). Fig. 6 shows images with only the resulting translucency for one scenario. Images with the complete illumination and another scenario are shown in the Appendix, Fig. 1. As with other scenarios, the technique by Jimenez et al. produces an incorrectly colored translucency effect. Our technique matches the ground truth closely.

The *Animated* model shows that our technique can be used with deforming geometry. The images we have produced match the ground truth closely as demonstrated in Fig. 7. In some areas our technique overestimates the translucency but compared to the other techniques we are still very close.

The results of our renderings with the *Happy Buddha* model are not as impressive as the previous ones. When using 3 samples for our technique a translucency effect can unfortunately not be seen. This is due to the high tessellation of this model. Using more samples does show a clearly visible effect that converges towards the *ground truth*. Images showing this can be seen in Fig. 8. The convergence can also be seen in Fig. 10. In the Appendix we show results where we use a mesh with lower tessellation for the vertices on the back side that are very close to the ground truth. We have another scene with multiple light source where we use the *Heptoroid* model. The model itself has a lot of self occlusion where we expected TSMs to miss some regions in respect to translucency as the direct path to a light source might be occluded by other geometry. Fig. 9 shows several regions where the technique by Jimenez et al. misses translucency or introduces hard edges whereas our method produces low frequency results similar to the ones shown in the ground truth.

For all images shown we calculated the peak signal-to-noise ratio (PSNR) for our technique and the compared ones in respect to the ground truth. These values are plotted in Fig. 11 which shows our technique is superior to the other tested techniques in all tested scenarios. We calculated these values only for the pixels that did actually differ from the ground truth to make sure larger areas in the image that are not subject to a translucency effect do not distort the PSNR. This explains relatively low values in cases where the pictures compared are still very similar.

5.3. Performance

We rendered all our examples at a resolution of 1920×1080 pixels with an Intel i7-4790 CPU (3.60GHz), 16GB RAM on a NVidia Geforce GTX 980 graphics card. The results depend on two factors. As the translucency computation is done per vertex in a com-

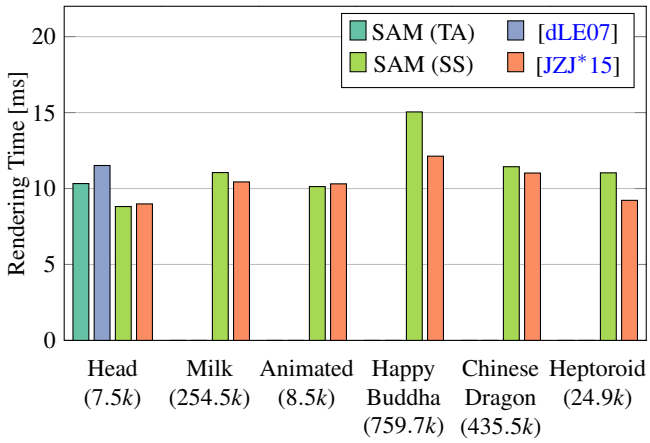

Figure 12: *Average rendering times for our example models (number of vertices for each model written in parentheses) shown in comparison. We measured all passes needed to render a final image. The rendering times mainly depend on the surface diffusion technique used as can be seen clearly in the results for the* Head *model. For the texture space diffusion (TA) we use a texture atlas of* 1024^2 *pixels. The final images where rendered with* 1920×1080 *pixels but only a part of the screen is covered by our model.*

pute pass before rendering, performance is lower for meshes with a higher number of vertices. While this is true for all techniques, our technique needs to process all vertices at least twice (twice for screen space diffusion, three times for texture space diffusion) this dependency is higher than with other techniques. The computation times also depend on the surface diffusion technique used. For texture space diffusion we used a 1024^2 texture atlas. The texture space techniques are slower in general because more render passes are needed. Also most of the texture area is filled with information that needs to be blurred while in screen space bigger areas can be skipped. The results presented in Fig. 12 are the times needed to render the final image including post processing. We can see that our technique is comparable in speed to all compared real-time techniques. Performance of the *Animated* model also contains the update process for our SAM which needs to be done for each frame as described in Sec. 3.4. Still, the results differ only slightly for smaller meshes (*Head*, *Milk*, and *Animated*) and are still reasonable fast for higher resolution meshes (*Happy Buddha* and *Chinese Dragon*).

Besides rendering, SAM generation also needs to be considered. As mentioned in Sec. 4 the preprocessing time has a squared dependency on the number of vertices in a mesh. By using a spatial tree this dependency can be reduced but it is still very noticeable if the real world size of the model is very small compared to the mean free path (which is the case with the *Chinese Dragon* model). The preprocessing times of the models we used are 1.152s for the *Head* model, 138.5s for the *Milk* model, 0.165s for the *Animated* model, 20.16min for the *Happy Buddha* model, 23.83h for the *Chinese Dragon* model and 13.17s for the *Heptoroid* model. The vertex count of each model can be seen in Fig. 12.

6. Conclusions and Future Work

We have presented a technique to generate a translucency effect in real-time applications without relying on shadow mapping techniques. The vertex based approach is independent of the used light source type and can be used in general lighting conditions. Our approach needs a precomputation step to calculate form factors but is still applicable for deformable meshes, as updating the form factors can be done interactively using our SAM data structure. The results produced with the presented technique provide higher quality images compared to all other tested techniques while still having a comparable rendering time. Our algorithm works well with different materials relying only on their diffusion profiles without any parameter tuning. We introduced a method for dividing a mesh's surface area into different sides to be able to correctly use a surface diffusion technique for one side combined with a translucency technique for regions that can not be covered using a surface effect only.

There is still some room for improvement as can be seen in our renderings of the *Happy Buddha* and *Chinese Dragon* models. Using more vertex samples will reduce the problem of incorrect translucency but there is an impact on rendering times. As we show in the Appendix using a lower tessellated mesh for finding the back side vertices can improve the visual quality and will also lower the precomputation times that are especially bad for these examples. It is still interesting to also find methods that can generate these samples in real time.

Our algorithm can be integrated in current real-time rendering pipelines for subsurface scattering removing the dependency on TSMs. It will work with any materials without parameter tuning and will produce convincing results in many cases which makes it an interesting alternative to current approaches for rendering translucency effects.

As an especially interesting use case for our data structure we see the efficient usage of multipole diffusion profiles. Multipole profiles outperform dipole profiles as they produce better results for slab geometries. In these cases a profile that would not only depend on the distance to a sample but also on the slab thickness could be used to calculate the effect efficiently. While finding the slab thickness would be difficult to achieve using for example TSMs, our data structure stores this distance (or at least a good approximation) implicitly as the distance to the nearest back side sample.

Acknowledgments

The authors want to thank Morgan McGuire's Computer Graphics Archive [McG11] for the *Chinese Dragon*, *Happy Buddha*, and *Head* models and Paul Debevec's Light Probe Image Gallery [Deb99] for the environment maps used. The *Heptoroid* model used is a remeshed version of the 'Heptoroid' model by Brent Collins and Carlo Sèquin and is courtesy of UC Berkeley.

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, pp. 1–1. 2

[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 - SIGGRAPH '82* (New York, New York, USA, 1982), vol. 16, ACM Press, pp. 21–29. 2

[Cha60] CHANDRASEKHAR S.: *Radiative Transfer*. Dover Publications, Inc., New York, 1960. 2

[Deb99] DEBEVEC P.: Light Probe Image Gallery, 1999. `http://www.pauldebevec.com/Probes/`. 10

[DEJ*99] DORSEY J., EDELMAN A., JENSEN H. W., LEGAKIS J., PEDERSEN H. K.: Modeling and rendering of weathered stone. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM, pp. 225–234. 2

[DI11] D'EON E., IRVING G.: A Quantized-diffusion Model for Rendering Translucent Materials. In *ACM SIGGRAPH 2011 Papers* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 56:1–56:14. 3

[DJ05] DONNER C., JENSEN H. W.: Light Diffusion in Multi-layered Translucent Materials. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 1032–1039. 3, 4

[DL06] DONNELLY W., LAURITZEN A.: Variance shadow maps. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2006), I3D '06, ACM, pp. 161–165. 2

[dL07] D'EON E., LUEBKE D.: *GPU Gems 3*. Addison-Wesley Professional, 2007, ch. Advanced Techniques for Realistic Real-Time Skin Rendering, pp. 293–347. 2

[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, Switzerland, 2007), EGSR'07, Eurographics Association, pp. 147–157. 1, 2, 3, 7, 8, 9, 10

[DLR*09] DONNER C., LAWRENCE J., RAMAMOORTHI R., HACHISUKA T., JENSEN H. W., NAYAR S.: An Empirical BSSRDF Model. In *ACM SIGGRAPH 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 30:1–30:10. 3

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent Shadow Maps. In *Proceedings of the 14th Eurographics Workshop on Rendering* (2003), EGRW '03, Eurographics Association, Eurographics Association, pp. 197–201. 2

[FHK14] FRISVAD J. R., HACHISUKA T., KJELDSEN T. K.: Directional Dipole Model for Subsurface Scattering. *ACM Trans. Graph. 34*, 1 (Dec. 2014), 5:1–5:12. 3

[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. 2

[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. 2

[IGAJG15] IGLESIAS-GUITIAN J. A., ALIAGA C., JARABO A., GUTIERREZ D.: A Biophysically-Based Model of the Optical Properties of Skin Aging. *CGF* (2015). 3

[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. 2, 3

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. *Siggraph* (2001), 511–518. 2, 3

[JSG09] JIMENEZ J., SUNDSTEDT V., GUTIERREZ D.: Screen-space perceptual rendering of human skin. *ACM Trans. Appl. Percept. 6*, 4 (Oct. 2009), 23:1–23:15. 2, 7

[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. 1, 2, 3, 6, 7, 8, 9, 10

[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. 3

[McG11] MCGUIRE M.: Computer Graphics Archive, August 2011. `http://graphics.cs.williams.edu/data`. 10

[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, Switzerland, 2003), EGRW '03, Eurographics Association, pp. 130–140. 3, 4

[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), SIGGRAPH '06, ACM, pp. 1003–1012. 3

[NRH77] NICODEMUS F. E., RICHMOND J. C., HSIA J. J.: *Geometrical considerations and nomenclature for reflectance*. National Bureau of Standards (US), 1977. 3

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

[PH00] PHARR M., HANRAHAN P.: Monte carlo evaluation of nonlinear scattering equations for subsurface reflection. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM, pp. 75–84. 2

[PK15] PETERS C., KLEIN R.: Moment shadow mapping. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2015), i3D '15, ACM, pp. 7–14. 2

[SDS*16] SHINYA M., DOBASHI Y., SHIRAISHI M., KAWASHIMA M., NISHITA T.: Multiple Scattering Approximation in Heterogeneous Media by Narrow Beam Distributions. *CGF* (2016). 3

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 527–536. 3

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, Deformable Precomputed Radiance Transfer. In *ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), SIGGRAPH '05, ACM, pp. 1216–1224. 3

[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. 2

[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. 3