Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Technical Report

RR-02-12 July 2012

Separable Subsurface Scattering

Jorge Jimenez Adrian Jarabo Diego Gutierrez

Universidad de Zaragoza

Graphics and Imaging Lab

July 2012
Dpto. Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Separable Subsurface Scattering

Jorge Jimenez, Adrian Jarabo and Diego Gutierrez

Universidad de Zaragoza

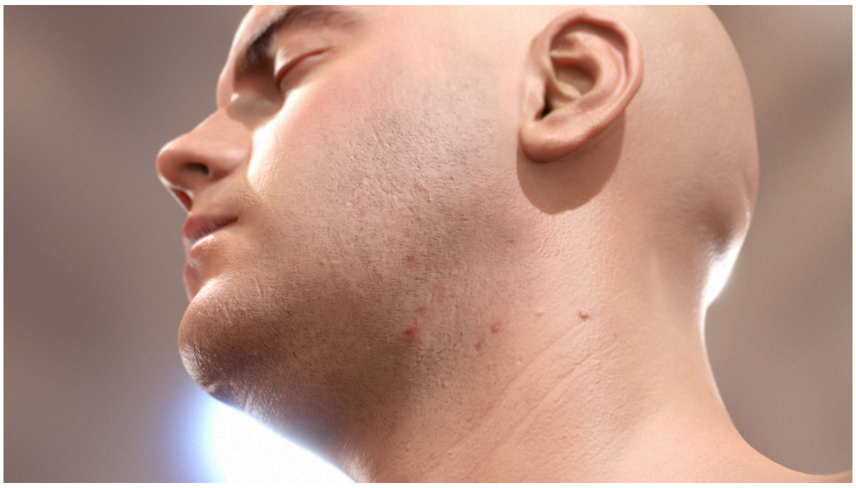

Figure 1: Rendering of a human face with our real-time separable subsurface scattering shader running at 112.5 frames per second on a GeForce GTX580 at 1080p, with multiple lights, depth of field, state-of-the-art morphological antialiasing and high-quality textures and geometry. Our separable approximation takes only 1.05 ms to simulate subsurface scattering, while getting high-quality results. This makes the technique practical for challenging game scenarios.

Abstract

Graphics in games are continuously improving, due to the parallel development of both hardware and software. Still, for a 60 fps game, everything contained in a frame must be computed in just about 16 milliseconds. Given this tight time budget, certain effects cannot be computed and are simply not simulated, sacrificing realism to reallocate resources to other aspects of the game. We propose a technique to simulate subsurface scattering for human skin that runs in just over 1 millisecond per frame, making it a practical option for even the most challenging game scenarios. Previous real-time approaches simulate it by approximating the non-separable diffusion kernel using a sum of Gaussians, which required several (usually six) 1D convolutions. In this work we decompose the exact 2D diffusion kernel with only two 1D functions. This allows to render subsurface scattering with only two convolutions, reducing both time and memory without a decrease in quality. Our 1D functions are defined in an intuitive way with just three parameters, allowing for easy edits.

1. Introduction

Accurate depiction of skin is very important in film and game industries to display realistic humans. However, rendering realistic skin with subsurface scattering implies simulating how light travels and scatters through a multi-layered complex material, which turns out to be an expensive process. While offline rendering scenarios (such as movies) can afford long computation times, real-time applications such as video games impose very severe time constraints, which go well beyond the definition of real-time. For a game running at 60 fps, *all* the computations necessary to produce in a frame, not only including the rendering pipeline, but other aspects of the game like physics simulation or AI, must be done in just about 16 ms. This gives each desired rendering effect (e.g. subsurface scattering, lighting, anti-aliasing) a very limited time budget; any millisecond saved can be used to improve or include other tasks to help improve the game. Thus, in game scenarios not only real-time, but *practical real-time* rendering is required, where every millisecond spent counts. This means that the common solution in games with respect to subsurface scattering is to simply ignore it. The result is an obvious loss of quality in how virtual humans are depicted.

One of the most common approaches exploits the fact that subsurface scattering blurs high-frequency details and illumination. This means that simulating subsurface scattering can be approximated as a convolution with a diffusion kernel. The exact diffusion kernel is non-separable, though, which in principle requires an expensive two-dimensional convolution. D'Eon and colleages [dLE07] showed that this kernel can be approximated by a sum of Gaussians, which are separable functions. This transforms the 2D convolution into a set of cheaper 1D passes, which allows high-quality skin rendering in real-time. This approach has been applied in both texture and [dLE07] and screen-space, modulating the variance of the kernel according to per-pixel depth information [JSG09].

However, in order to get good results, several Gaussians need to be used to approximate the diffusion profile, which translates into several convolutions per frame. This does not fit well in some hardware (e.g. Xbox 360 or PS3) and is usually still too expensive for games. Specially on the Xbox 360, given the fact that multi-pass shaders require to move the frame buffer from eDRAM to main memory for each pass by means of a memory resolve. In this work we present an practical approach for rendering skin with subsurface scattering effects in just two passes, suitable for game scenarios. It is based on the observation that, although exact diffusion kernels might be mathematically non-separable, they can be approximated with a separable kernel without a perceived decrease in quality. This allows to reduce the number of convolutions, saving precious milliseconds while getting high-quality results. Figure 1 shows an example of skin rendered with our separable approximation, where subsurface scattering is calculated in just 1.06 ms on a GTX 580; in contrast, a six-pass sum-of-Gaussian approach performed in screen-space takes 3.07 ms. This effectively means that we can add for instance a state-of-the-art morphological antialiasing technique using the free extra two milliseconds.

We formulate the separable kernel from a base 1D diffusion profile, which is defined by three parameters. These parameters define the global level of subsurface scattering, the width of the gradients and the amount of light that gets into the skin, on a per-channel basis. We optimize these three parameters to find the kernel that fits best for the exact diffusion kernel. Our formulation allows intuitive editing and tweaking of the appearance of the skin while preserving realism.

Using only two convolutions results into a significant boost of performance, especially in close-up renders where the effect of subsurface scattering counts the most. The lower preset of our shader is implemented using a regular 9-sample separable convolution, making of it a very cheap solution even on very low-end hardware. Additionally, fixed costs (i.e. scenes with no visible scattering) are neglectable: on a 580 GTX, our implementation imposes only a 0.1 ms overhead, compared with the 0.71 ms needed by the screen-space technique (this difference increases on slower cards). Last, our approach is much simpler to implement, without the need for complex alpha blending pipelines or Gaussian levels-of-detail [JG10]. We believe our technique is very attractive for in-game rendering pipelines, beyond game cinematics.

2. Previous work

Many recent papers have addressed scattering and translucency [GJJD09]. We provide here a brief discussion of existing works that deal specifically with subsurface scattering in skin rendering. The first work to introduce a practical formulation of subsurface scattering is the work of Jensen and colleagues [JMLH01, JB02]. Their techniques capture the soft appearance of the materials due to translucency, by using a dipole approximation of light diffusion. Donner and Jensen [DJ05] extend the dipole into a multipole model that allows to model multi-layered translucent materials, such as skin. The same authors later introduce a spectral BSSRDF for human skin [DJ06] and the photon diffusion technique, which combines photon tracing and the diffusion approximation [DJ07]. These works yield impressive results, although the computation times are in the order of seconds per image, so they cannot be used in real-time applications or games.

Borshukov and Lewis [BL03] blur a 2D diffuse irradiance texture with a Gaussian filter. While this technique is efficient and maps well on the GPU, it fails to approximate some of the subtle details of subsurface scattering. This idea is later extended by D'Eon and colleagues [dLE07, dL07] to develop a high-quality real-time skin shader. They approximate the multipole model with a sum of Gaussians, and use them to blur the irradiance values in texture space. Since Gaussians are separable functions, this allows to transform the expensive 2D diffusion convolutions into a cheaper set of 1D convolutions. This translates into real-time frame rates, while getting results comparable with offline simulations. Follow-up work optimize the global pipeline proposed by d'Eon et al. by reintroducing back-face culling and view frustum clipping [JG08]. An additional optimization is concurrently introduced, based on computing a single 2D convolution at thirteen jittered sample points, which account for direct reflection and two levels of scattering [HBH09]. The sum of Gaussians approach is also an integral part of a recent sophisticated model by Donner et al. [DWd*08]. It takes into account the heterogeneous spectral reflectance of skin (due to veins, hemoglobin or even tattoos), which the authors acquire physically and model with five parameters.

Although these latest techniques provide real-time frame rates, they scale poorly with the number of translucent objects in the scene, since the subsurface scattering simulation is performed in texture space for each object. This is a huge limitation for some applications such as games, where several characters might be on screen at the same time. To overcome this problem, Jimenez et al. [JSG09, JG10] propose to translate the simulation to screen space. In their technique, the diffuse reflection of the translucent object is blurred as a post-process using the sum of Gaussians formulation [dLE07], and simulating subsurface scattering only in the visible parts of the objects. However, the effect of light transmittance through thin slabs (e.g. ears or nostrils) is lost, since there is no information of light from behind the slab. This is solved in later work by pre-integrating the transmitted radiance, making some simplifying assumptions about the model shape [JWSG10]. This screen-space approach has also been used in a recent dynamic model based on in-vivo measurements, where skin color takes into account the subtle changes in appearance due to hemoglobin variations [JSB*10]. Other techniques operating in screen space include Mertens et al. [MKB*05], using importance sampling of the BSSRDF, Shah et al. [SKP09], who use a splatting process for computing integration instead of a gathering process, or Mikkelsen's work, who shows how convolution by a Gaussian can be expressed as a cross bilateral filter [Mik10]. Recently, Penner and Borshukov [PB11] pre-integrate the illumination effects of subsurface scattering due to curvature and shadowing into textures. Additionally, the normal map is pre-blurred using the diffusion profile. This technique gives very good results, but it assumes that normals can be pre-blurred, which is not always the case. It also relies on soft shadows, which may not be available due to their high cost. In contrast to these works, our technique does not require soft shadows, is simpler to implement and to customize, and yields frame rates high enough to be used in challenging game scenarios with very limited time budgets for each effect.

3. Separable subsurface scattering

We focus on subsurface scattering inside skin, for which the exact diffusion kernel is non-separable. Thus, to convolve the irradiance it is necessary in principle to perform a two-dimensional convolution. This is a expensive process, which is not suitable for real-time applications.

Our key idea to accelerate this process is very simple: inspired by recent work in the context of ambient occlusion [HBR*11] and mesh filtering [VB11], we decompose the exact, mathematically non-separable 2D diffusion kernel into two 1D kernels that avoid the costly sum of Gaussians (for skin, this required about six convolutions for good results), while being easy to tweak for different types of skin.

In the following, we first describe the formulation adopted to model the separable approximation of the diffusion kernel, and then the optimization process performed to get the best fit to the actual kernel.

3.1. Separable approximation

Separability is a desired property for multidimensional filters. For two-dimensional filters it means that a filter $F[x,y]$ can be decomposed into two one-dimensional signals, a vertical and a horizontal kernel, such that $F[x,y] = v[x] \times h[y]$. This transforms the two-dimensional convolution into two cheaper one-dimensional passes.

Although the diffusion kernel is non-separable, its matrix form $S[x,y]$ is low rank, with the first eigen value storing most of the total energy ($\sim 80\%$) (see Figure 2). This property, together with the circular symmetry of the diffusion profile, suggests that it can be approximated with a separable kernel defined by a one-dimensional filter $s[x]$ such that $S[x,y] \approx s[x] \times s^T[y]$, where $s^T[y]$ is the transpose of $s[x]$.

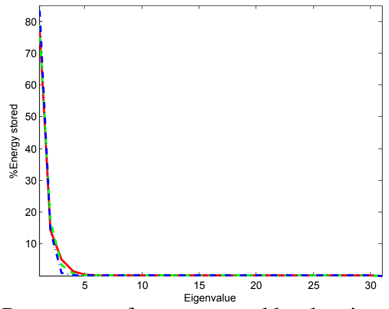

Figure 2: Percentage of energy stored by the eigen values of the diffusion kernel used to simulate subsurface scattering in skin. Each curve corresponds to channels red, green and blue, respectively.

We formulate $s[x]$ by using an initial 1D diffusion profile $p[x]$ which is modified using a small set of transformations. The parameters of these transformation are adjusted to obtain the kernel that best fits the target diffusion profile. These parameters are: *width*, *strength* and *falloff*. *Width* specifies the global level of subsurface scattering, i.e. the width of the filter; it is the same for the red, green and blue channels. *Strength* specifies how much diffuse light penetrates the skin, per channel, and thus contributes to subsurface scattering. Finally, *falloff* defines the per-channel falloff of the gradients. The kernel $s[x]$ is defined as a function of these three parameters and the intiial diffusion profile $p[x]$:

$$s(w,t,f)[x] = p[\frac{x * w}{0.001 + f}] * t + \delta(x) * (1 - t) \qquad (1)$$

Parameter	R	G	B
Width	0.014		
Strength	0.7800	0.7000	0.7500
Falloff	0.5700	0.1300	0.0800

Table 1: Parameters obtained by optimizing our model to minimize the error with the actual diffusion profile of the skin.

where w, t and f are the parameters *weight*, *strength* and *falloff* respectively, and $\delta(x)$ is a delta function that returns 1 if $x = 0$ and 0 otherwise.

The initial 1D diffusion profile $p[x]$ used in this work is the red channel of the original skin profile defined in [dLE07]. We make the observation that it can also be used for green and blue channels (scaled using the *falloff* parameter) without introducing noticeable differences and allowing for total control over the profile. The effects of each of these parameters are illustrated in Figure 6.

3.2. Optimization

We search the space defined by the parameters described above to find the separable kernel $\tilde{S}[x,y] = s[x] \times s^T[y]$ that best approximates the exact diffusion kernel $S[x,y]$. The parameters are obtained by applying a constrained nonlinear optimization over the function:

$$f(w,T,F) = \int_A \sum_{c\in C} (S_c[x] - \tilde{S}_c(w,T_c,F_c)[x])^2 dx \qquad (2)$$

where A is the area around the center of the kernel, $C = \{r,g,b\}$ is the set of color channels, T and F are the vectors containing the parameters *strength* and *falloff* for each channel respectively (remember that w remains a scalar) and T_c and F_c are individual components of such vectors.

In order to reduce the cost of the optimization, we only evaluate the function over a limited discretized space. This space is centered in 0 and is more densely sampled in the areas close to the center, since the diffusion profile has more importance in these areas. Thus Equation 2 is approximated by:

$$f(w,T,F) \approx f_s(w,T,F) = \sum_{a\in A_s} \sum_{c\in C} (S_c[a] - \tilde{S}_c(w,T_c,F_c)[a])^2 \qquad (3)$$

where A_s is the discrete set of samples in area A. We optimize Equation 3 using the function *fmincon* from the Matlab Optimization Toolbox [Mat]). The parameters obtained after optimizing Equation 3 are sumarized in Table 1.

4. Rendering

We simulate subsurface scattering in screen-space using the separable approximation of the diffusion profile explained in Section 3. First, the irradiance over the visible geometry is obtained and stored in the frame buffer. Then, this irradiance is blurred by convolving it with the diffusion profile: first, the horizontal convolution is saved into an intermediate buffer; then, the vertical pass convolves this intermediate buffer and stores the final result into the back buffer. Similar to D'Eon et al. [dLE07], we use 16-bits render targets to store irradiance in linear space, since all our subsurface scattering computations must be performed in linear space before tone-mapping. Our algorithm is implemented as a short HLSL post-process shader. This makes it easy to implement and integrate in existing render engines. The full shader is listed in the Appendix.

Working in screen-space imposes the limitation of being unable to simulate translucency through thin slabs, since we have no information about back-face irradiance. To include translucency in our implementation we use the technique by Jimenez et al. [JWSG10]. This technique simulates translucency by using an heuristic derived from a set of observations, which is used to approximate the amount of light traveling through the medium. This technique is performed in the initial pass (when irradiance is calculated), before the subsurface scattering computations, so it imposes no additional cost. In this initial pass, we make use of multiple render targets of modern graphics hardware to decouple diffuse and specular components, which is common in deferred engines. This is needed to avoid including specular irradiance in the subsurface scattering simulation. The specular irradiance is calculated using the Kelemen/Szirmay-Kalos model, as suggested by d'Eon and Luebke [dLE07].

Since we are working in screen-space, the size of the kernel depends on the projected surface area in the pixel, which depends on depth and surface orientation. This area is specified in world-space units, instead of pixels, making the definition of the kernel size more intuitive for artists. To deal with large depth variations, the effect of the surface orientation is modulated using a technique similar to previous work on ambient occlusion [FM08]: if the depth variation between the sample and the center of the convolution is larger than the diffusion profile, the contribution of the sample is linearly interpolated with the original color.

To avoid blurring pixels that display non-translucent objects, a matte mask can be used. If multisampling is not used, this mask is defined with the stencil buffer, to optimize computations. If multisampling is used, we use a different approach: the first pass is guided by the matte mask stored in the alpha channel of the frame buffer, and initializes the stencil buffer. Then, the stencil buffer masks which pixels should be convolved in the second pass. Additionally, if multisampling is used, pixel depths are calculated by averaging the depth of the subsamples, to avoid artifacts in the silhouette of the object.

As previous screen-space approaches [JSG09], the scale of the subsurface scattering can be modulated on a per-pixel basis. The per-pixel scale is stored in an additional texture.

It allows using variable kernel sizes, and removing the effect of subsurface scattering in eyebrows or facial hair. In the geometry rendering pass, this scale is stored in the alpha channel of the render buffer, so it can be accessed in subsequent passes. Last, we have used the recent SMAA technique [JESG12] in T2x mode for anti-aliasing.

5. Results

Figure 3 shows some results obtained using our subsurface scattering technique on a scanned head model (please see also the accompanying video). We compare our results with a six-pass screen-space subsurface scattering technique based on the sum-of-Gaussians approximation [JG10], in a range of GPUs. Table 2 shows that our separable approximation is much faster than the sum-of-Gaussians approach. In close views, where the head almost fills the screen, the cost is around a millisecond using high-end hardware. This type of shot occurs typically in game cinematics, which are more forgiving in terms of resources used; for medium shots, where subsurface scattering has to be calculated in less pixels, the two-pass algorithm is almost free on high-end computers. As argued in the introduction, the fixed costs of the technique are very low, which makes this technique practical for both game cinematics and in-game rendering. As shown in Figure 4, the error from approximating the exact diffusion kernel with our separable kernel is very low, and translates into no visible differences in the final images.

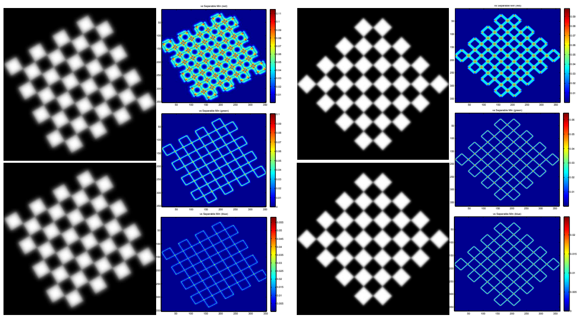
(a) (b) (c) (d)

Figure 4: The error introduced by our separable approach does not produce visible differences in the images. (a) and (c), top, images rendered with the exact kernel; bottom, our separable approximation. (b) and (d) show the error per channel due to approximating the kernel, for (a) and (c) respectively (from top to bottom, red, green and blue).

The quality of the simulation can be adjusted by setting the number of samples in the kernel, which gives a trade off between quality and cost. Unless it is explicitly stated, all our results use 17 samples for each pass. Figure 5 shows quality differences in close-up renderings varying number of samples. Good results can be achieved with as few as nine samples under certain lighting conditions.

	GTX580		GTX470		GT130M	
	close	mid	close	mid	close	mid
Ours	1.05	0.1	4.6	0.13	13.66	0.81
[JG10]	3.07	0.71	10.38	0.89	40.02	3.94

Table 2: Comparison of timings (in milliseconds) between our technique and the screen-space subsurface scattering by Jimenez et al. [JG10], for close and mid shots.

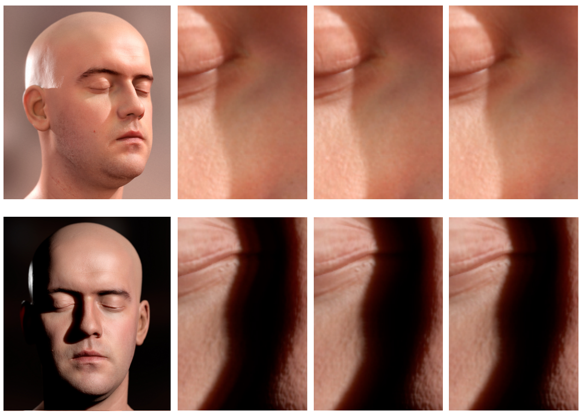
Figure 5: The effect of number of samples on the quality of subsurface scattering (from left to right: head rendered with 17 samples, detail rendered with 9, 17 and 33 samples respectively). With predominant ambient lighting (top), the differences are practically indistinguishable. In sharper illumination conditions (bottom) banding starts to be visible with 9 samples (note the effect on the gradient in the shadow). The timings are 1.02, 1.77 and 3.15 ms for 9, 17 and 33 samples respectively on a GTX470 at 1080p.

Editing. Figure 6 shows examples of subsurface scattering customization by modifying the parameters used to model our separable kernel (see Section 3.1). The formulation used (based on the parameters *width, strength* and *falloff*) allows to intuitively modify the subsurface scattering to match the desired look, while keeping a physically realistic appearance.

6. Conclusions

We have presented a practical method to simulate subsurface scattering for skin rendering. It yields state-of-the-art results in just over one millisecond per frame, which makes it affordable in game environments where every desired effect has a few-milliseconds budget, and real-time is just not enough. It is based on approximating the exact diffusion profile by a separable kernel. This allows to perform only two passes, as opposed to the six passes needed by the established sum-of-Gaussians approximation, saving significant time and memory resources. Since the algorithm works as a post-process, it is very easy to integrate in existing game engines, suitable for the current generation of consoles and

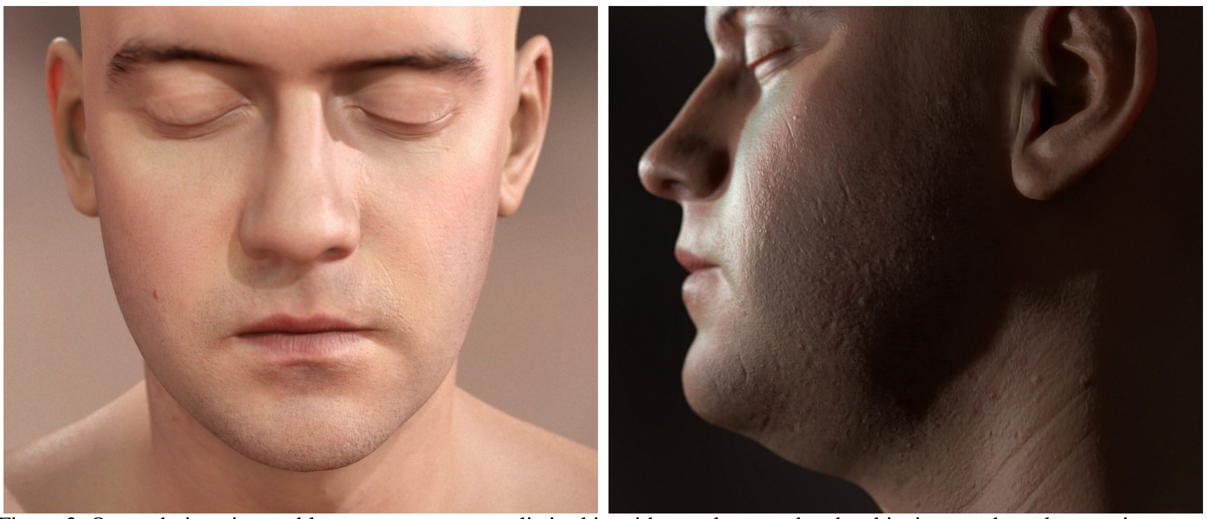

Figure 3: Our technique is capable to generate very realistic skin with very low overhead, achieving good results even in scenes with low ambient light (right).

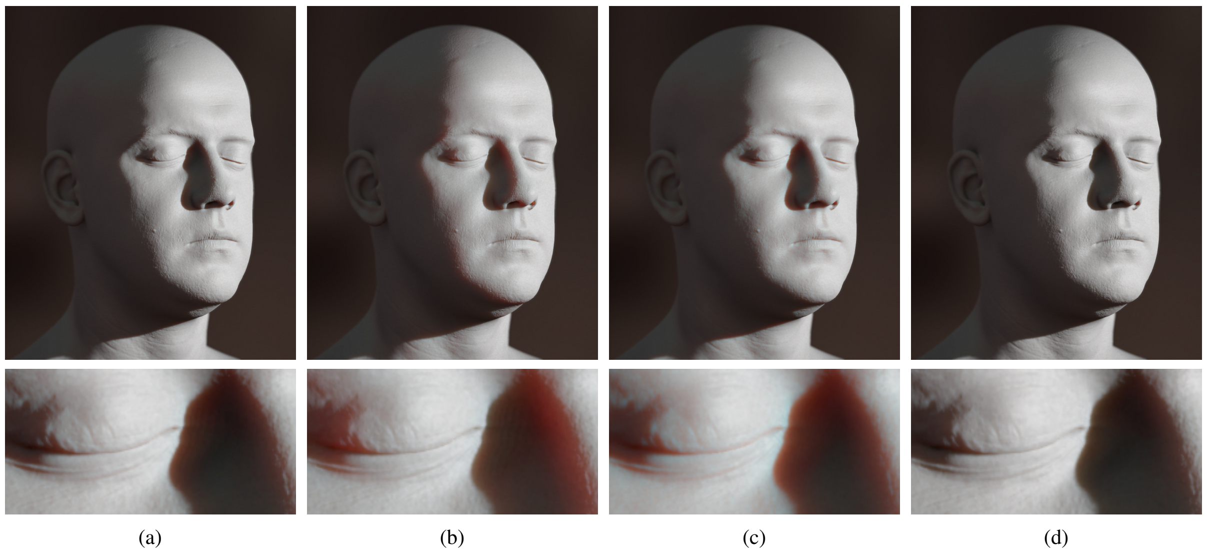

Figure 6: Examples of editing the appearance of subsurface scattering: from left to right, (6a) the original kernel; (6b) *width* increased up to .025, allowing the light to travel further inside the skin; (6c) *strength* increased up to $strength = \{1, 0.71, 0.51\}$, note that the skin appears softer; (6d) the *falloff* of the red channel has been reduced to 0.5, note that the red gradient is more narrow in the shadow produced by the nose. The texture of the model has been removed to better depiction of the effect of subsurface scattering.

practically free in modern graphics hardware. Additionally, the proposed formulation opens up intuitive editing capabilities, tweaking the appearance of the skin while keeping visually realistic results.

For skin rendering, mathematically non-separable filters can be approximated with separable kernels and produce good visual results. We hope that this observation can be extended to accelerate other computer graphics and image processing algorithms. As a future work, we would like to generalize our separable subsurface scattering approximation to a wider range of translucent materials.

Appendix A: HLSL shader

```
float4 SSSSBlurPS(float2 texcoord, SSSSTexture2D colorTex,
                  SSSSTexture2D depthTex, float sssWidth,
                  float2 dir, bool initStencil) {

    // Fetch color of current pixel:
    float4 colorM = SSSSSamplePoint(colorTex, texcoord);

    // Initialize the stencil buffer in case it was not
    // already available:
    if (initStencil)
        if (SSSS_STREGTH_SOURCE == 0.0) discard;

    // Fetch linear depth of current pixel:
    float depthM = SSSSSamplePoint(depthTex, texcoord).r;

    // Calculate the sssWidth scale (1.0 for a unit plane
        // sitting on the projection window):
    float distToProjWindow = 1.0 /
                             tan(0.5 * radians(SSSS_FOVY));
    float scale = distToProjWindow / depthM;

    // Calculate the final step to fetch the surrounding
    // pixels:
    float2 finalStep = sssWidth * scale * dir;

    // Modulate it using the alpha channel:
    finalStep *= SSSS_STREGTH_SOURCE;

    // Divide by 3 as the kernels range from -3 to 3:
    finalStep *= 1.0 / 3.0;

    // Accumulate the center sample:
    float4 colorBlurred = colorM;
    colorBlurred.rgb *= kernel[0].rgb;

    // Accumulate the other samples:
    SSSS_UNROLL
    for (int i = 1; i < SSSS_N_SAMPLES; i++) {
        // Fetch color and depth for current sample:
        float2 offset = texcoord + kernel[i].a * finalStep;
        float4 color = SSSSSample(colorTex, offset);

        #if SSSS_FOLLOW_SURFACE == 1
        // If the difference in depth is huge, we lerp
        // color back to "colorM":
        float depth = SSSSSample(depthTex, offset).r;
        float s = 300.0f * distToProjWindow *
                  sssWidth * abs(depthM - depth);
                  s = SSSSSaturate(s);
        color.rgb = SSSSLerp(color.rgb, colorM.rgb, s);
        #endif

        // Accumulate:
        colorBlurred.rgb += kernel[i].rgb * color.rgb;
    }
    return colorBlurred;
}
```

References

[BL03] BORSHUKOV G., LEWIS J. P.: Realistic human face rendering for 'The Matrix Reloaded'. In *ACM SIGGRAPH 2003 Sketches & Applications* (2003). 2

[DJ05] DONNER C., JENSEN H. W.: Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics 24*, 3 (2005), 1032–1039. 2

[DJ06] DONNER C., JENSEN H. W.: A spectral BSSRDF for shading human skin. In *Proceedings of Eurographics Symposium on Rendering* (2006), pp. 409–417. 2

[DJ07] DONNER C., JENSEN H. W.: Rendering translucent materials using photon diffusion. In *Proceedings of the Eurographics Symposium on Rendering* (2007), pp. 243–252. 2

[dL07] D'EON E., LUEBKE D.: Advanced techniques for realistic real-time skin rendering. In *GPU Gems 3*, Nguyen H., (Ed.). Addison Wesley, 2007, ch. 14, pp. 293–347. 2

[dLE07] D'EON E., LUEBKE D., ENDERTON E.: Efficient rendering of human skin. In *Proceedings of Eurographics Symposium on Rendering* (2007), pp. 147–157. 2, 3, 4

[DWd*08] DONNER C., WEYRICH T., D'EON E., RAMAMOORTHI R., RUSINKIEWICZ S.: A layered, heterogeneous reflectance model for acquiring and rendering human skin. *ACM Transactions on Graphics 27*, 5 (2008). 3

[FM08] FILION D., MCNAUGHTON R.: Effects & techniques. In *ACM SIGGRAPH 2008 classes* (New York, NY, USA, 2008), SIGGRAPH '08, ACM, pp. 133–164. 4

[GJJD09] GUTIERREZ D., JENSEN H. W., JAROSZ W., DONNER C.: Scattering. In *ACM SIGGRAPH ASIA 2009 Courses* (New York, NY, USA, 2009), SIGGRAPH ASIA '09, ACM, pp. 15:1–15:620. 2

[HBH09] HABLE J., BORSHUKOV G., HEJL J.: Fast skin shading. In *Shader X7*, Engel W., (Ed.). Charles River Media, 2009, ch. 2.4, pp. 161–173. 3

[HBR*11] HUANG J., BOUBEKEUR T., RITSCHEL T., HOLLÄNDER M., EISEMANN E.: Separable approximation of ambient occlusion. In *Eurographics 2011-Short papers* (2011). 3

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics 21*, 3 (2002), 576–581. 2

[JESG12] JIMENEZ J., ECHEVARRIA J. I., SOUSA T., GUTIERREZ D.: Smaa: Enhanced morphological antialiasing. *Computer Graphics Forum (Proc. EUROGRAPHICS 2012) 31*, 2 (2012). 5

[JG08] JIMENEZ J., GUTIERREZ D.: Faster rendering of human skin. In *CEIG* (2008), pp. 21–28. 2

[JG10] JIMENEZ J., GUTIERREZ D.: *GPU Pro: Advanced Rendering Techniques*. AK Peters Ltd., 2010, ch. Screen-Space Subsurface Scattering, pp. 335–351. 2, 3, 5

[JMLH01] JENSEN H., MARSCHNER S., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 511–518. 2

[JSB*10] JIMENEZ J., SCULLY T., BARBOSA N., DONNER C., ALVAREZ X., VIEIRA T., MATTS P., ORVALHO V., GUTIERREZ D., WEYRICH T.: A practical appearance model for dynamic facial color. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 29*, 6 (2010), 141:1–141:10. 3

[JSG09] JIMENEZ J., SUNDSTEDT V., GUTIERREZ D.: Screen-space perceptual rendering of human skin. *ACM Transactions on Applied Perception 6*, 4 (2009), 1–15. 2, 3, 4

[JWSG10] JIMENEZ J., WHELAN D., SUNDSTEDT V., GUTIERREZ D.: Real-time realistic skin translucency. *IEEE Computer Graphics and Applications 30*, 4 (2010), 32–41. 3, 4

[Mat] MATHWORKS: Matlab optimization toolbox user's guide. http://www.mathworks.com/help/pdf_doc/optim/optim_tb.pdf. Last accessed 2-March-2012. 4

[Mik10] MIKKELSEN M.: *Skin Rendering by Pseudo-Separable Cross Bilateral Filtering*. Tech. rep., Naughty Dog Inc., August 2010. URL: http://jbit.net/~sparky/subsurf/cbf_skin.pdf. 3

[MKB*05] MERTENS T., KAUTZ J., BEKAERT P., REETH F. V., SEIDEL H. P.: Efficient rendering of local subsurface scattering. *Computer Graphics Forum 24*, 1 (2005), 41–50. 3

[PB11] PENNER E., BORSHUKOV G.: *GPU Pro 2*. AK Peters Ltd., 2011, ch. Pre-Integrated Skin Shading, pp. 41–55. 3

[SKP09] SHAH M. A., KONTTINEN J., PATTANAIK S.: Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE Computer Graphics and Applications 29* (2009), 66–78. 3

[VB11] VIALANEIX G., BOUBEKEUR T.: SBL mesh filter: A fast separable approximation of bilateral mesh filtering. In *Vision, Modeling and Visualization (VMV) 2011* (2011). 3