Real-Time Approximate Subsurface Scattering

Robert Patro

Abstract— Subsurface scattering is a complex physical process, which, in many cases, significantly affects the appearance of certain materials. In the pursuit of generating ever more realistic scenes, it is a phenomenon which must be incorporated into rendering frameworks. However, the complexity of the physical process which causes subsurface scattering has a tremendous effect on rendering time. Though such rendering costs are acceptable in many settings, they are far to high for any process which requires interactive visualization.
In this paper, we propose a very simple approximation of subsurface scattering which requires a pre-computation step but incurs absolutely no runtime overhead. We illustrate the basic model used to derive our results, as well as a spectral mesh processing framework which generalizes our model while producing images of higher visual fidelity. This makes it possible to incorporate subsurface scattering effects into real-time visualizations where performance is the primary goal. We hypothesize that the complexity of the underlying physical process of subsurface scattering results in both highly apparent and subtle effects. Accepting the assumption that the realism of our subsurface scattering effect is secondary in importance to the speed at which it can be displayed, we make sacrifices in the physical realism of our approximation for the sake of speed. The results, however, maintain the overall landmark effects of subsurface scattering and provide the ability to produce visually compelling results with no runtime overhead.

Index Terms—Subsurface scattering, real-time, spectral filtering, graph Laplacian

1 Introduction

There is a fundamental dichotomy in computer graphics between physical accuracy and interactivity. The greater the degree to which reality is approximated in the process of rendering, the more computation that rendering necessarily requires. Even the generality of the rendering equation makes the simplification of geometric optics, sacrificing some expressibility for the sake of simplicity. The ways in which this dichotomy has most frequently been addressed are through preprocessing and approximation. Sometimes, it is possible to capture complex physically based rendering effects through a pre-processing phase; the results of which can be evaluated, displayed, and possibly modified in real-time. For example, precomputed radiance transfer allows for complex lighting effects to be evaluated at runtime, assuming that the geometry and lighting environment, and the manner in which they may change, are known a priori. Thus, realistic lighting effects which would be impossible to compute at runtime can still be displayed at interactive rates. Another facet of precomputed radiance transfer, however, is that of approximation. In order to allow the precomputed results to be evaluated and displayed in real-time, an appropriate representation for these results, such as spherical harmonics, must be utilized. The spherical harmonics basis allows for the efficient storage and evaluation of many precomputed radiance transfer results. However, this efficency is achieved at the cost of its low-frequency approximation of the original lighting signal. The decision as to which methods to use to simulate complex physical effects in the process of rendering is often reduced to an application dependent cost-benefit analysis, where the designer of the application must weigh the level of accuracy necessary against the level of interactivity required for the application.

One such computationally expensive rendering effect is subsurface scattering. Subsurface scattering is a physical property of the interaction of light with different materials. It occurs when light is incident upon a material with some degree of translucency. Instead of strictly reflecting off the surface of the material, some amount of light will actually enter the material, scattering and diffusing as it moves, and will then exit at points distinct from where it entered. Subsurface scattering produces visual effects which are significant enough to warrent implementation in many different visualization applications. When physical accuracy and realism are the foremost goals, many high fidelity approximations to subsurface scattering may be used to achieve highly convincing results, such as [3]. However, even the fastest existing methods of approximating subsurface scattering result in a decrease in rendering speeds by a factor of two or more.

Main Results: In this paper, we present an approximation to subsurface scattering which allows results generated in a pre-processing phase to be expressed using the standard local illumination model [1]. Our method computes the effects of subsurface scattering at a given surface point by incorporating information about the incident light at neighboring surface points. We present both a naïve implementation, using a weighted sum of neighboring normals, and a more refined, spectral mesh processing implementation, which produces higher quality results by eliminating certain descretization artifacts. Further, our spectral processing approach allows for the filtering kernel being used to create the subsurface scattering effect to be changed at a near interactive rate. Both approaches produce a set of filtered normals, which need only be used in place of the original normals of the mesh to display the subsurface scattering effect. The fact that the effect is achieved by using these pre-filtered normals during the rendering phase, means that no overhead, beyond that of the standard local illumination model, is incurred. Thus, runtime framerates are not at all affected, and a coarse, but "free", approximation of subsurface scattering is provided.

2 Background and Related Work

2.1 The Rendering Equation, BRDFs and BSSRDFs

The rendering equation is an equation based on the physical properties of the interaction between light and matter. It is an analytic expression which represents the *ideal* lighting model, and is the standard against which realistic lighting models are compared. The rendering equation contains a term describing the BRDF (Bidirectional Reflectance Distribution Function) at each surface point for which it is evaluated. The BRDF relates the outgoing light at a point on the surface to the light incident at the point and the direction of that incoming light. The BRDF allows the exitant radiance at a given surface point x to be expressed as:

$$dL_o(x,\vec{\omega}_o) = F(x,\vec{\omega}_i \rightarrow \vec{\omega}_o)d\phi_i(x,\vec{\omega}_i) \quad (1)$$

Where $L_o(x,\vec{\omega}_o)$ is the exitant radiance at a given surface point x, in the direction $\vec{\omega}_o$, $\phi_i(x,\vec{\omega}_i)$ is the incident flux at point x in direction

• *Robert Patro is with the University of Maryland, College Park, E-mail: rob@cs.umd.edu.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 2 November 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

$\vec{\omega}_i$, and F is the BRDF which relates incoming light direction to outgoing light direction. In the rendering equation, the BRDF is integrated along with a number of other terms over all incoming light directions, and the results can produce very realistic renderings of a wide variety of materials. However, despite its generality, the BRDF is unable to properly model a number of different materials due to the restrictions inherent in the basic assumption that the exitant light at a surface point is dependent only upon light incident at that point. Though this is the case for perfectly opaque materials, it is not true of translucent materials.

In translucent materials, the exitant light at a given surface point is dependent upon the light incident at that point, its direction, and the light incident on the surrounding area. The BSSRDF (Bidirectional Subsurface Scattering Reflectance Distribution Function) takes all of these parameters into consideration when computing the radiance transfer at a surface point, and is thus capable of modeling the appearance of materials for which the BRDF is insufficient. The BSSRDF is commonly fomulated as follows [3]:

$$dL_o(x,\vec{\omega}_o) = S(x_i,\vec{\omega}_i;x,\vec{\omega}_o)d\phi_i(x_i,\vec{\omega}_i) \tag{2}$$

Where the variables that appear are the same as in equation 1, except that now, F is replaced with S, the BSSRDF, which relates the exitant radiance at a surface point x to the incident radiance at points x_i distinct from x. When the BSSRDF is used in the rendering equation, in addition to being integrated over all incoming light directions like the BRDF, it is also integrated over a given area, so that the contribution of the incident light at an area around the evaluation point is considered.

Clearly, the evaluation of a double integral at each surface point that would be required to compute the exitant radiance with the above equations is not a real-time friendly operation. Hence, a number of approximations have been proposed to speed up the computation of subsurface scattering effects. For example, in [3] Jensen et. al. propose an approximation of subsurface scattering effects based on a dipole diffusion model. This results in a tremendous increase in speed over the evaluation of the BSSRDF versus Monte Carlo integration (about two orders of magnitude) with little percieveable loss in quality. Going even further, in [4] Lensch et. al.decopuled the subsurface scattering effects into local and global impulse responses which are precomputed and combined at runtime to achieve almost interactive frame rates. Hao et. al., in [1], introduce a simpler subsurface scattering model based on the observation that the visual results of subsurface scattering are dominated by local effects due to the exponential falloff of light as it is scattered in the modeled material. They use a precomputation step which calculates the multiple scattering effects between a surface point and it's neighbors (using the dipole diffusion approximation). They then combine this infomation with the transmission and reflection at each surface point at runtime and achieve frame rates with subsurface scattering as high as 30-50% of that without subsurface scattering. This work was the first to make subsurface scattering effects tenable in realtime, but the extra information carried along with each vertex in a mesh was significant. Hao et. al. address this storage overhead in [2], while maintaining the same level of performance. Carr et. al. [6] use a hierarchical texture atlas of surface patches along with precomputed radiosity and scattered irradiance maps to allow for evaluation of subsurface scattering effects on the GPU. They achieve real-time rendering speeds for small meshes, but do not mention the effect their three pass approach has on actual rendering speed.

3 Our Approach

Inspired by the method of Hao et al. [2], we wish to approximate subsurface scattering effects in a local illumination model. However, we wish to simplify the subsurface scattering approximation even more, possibly sacrificing some degree of visual realism to achieve significantly faster frame rates. We first present the simple approximation we use, and explain how a naïve implementation might proceed. Then we present how a spectral mesh processing framework may be used to achieve higher fidelity results and allow for modification of the subsurface scattering properties at near interactive rates.

3.1 Weighted Neighbor Contributions

The most basic feature of subsurface scattering, and that which represents the majority of its apparent effect, is that some portion of the light incident at a surface point will actually enter the material, scattering and diffusing as it travels. As a result, the illumination exiting from a given surface point will be dependent upon the light which is incident upon, and diffuses from, its neighboring points. The absorbtion and diffusion of the light as it passes through the material means that the active neighborhood which contributes to the exitant radiance from a surface point will be relatively small. Particularly, we note that most methods which exists for calculating subsurface scattering effects use the dipole diffusion equation to model light falloff. This equation is dominated by a term which induces exponential falloff as a function of distance.

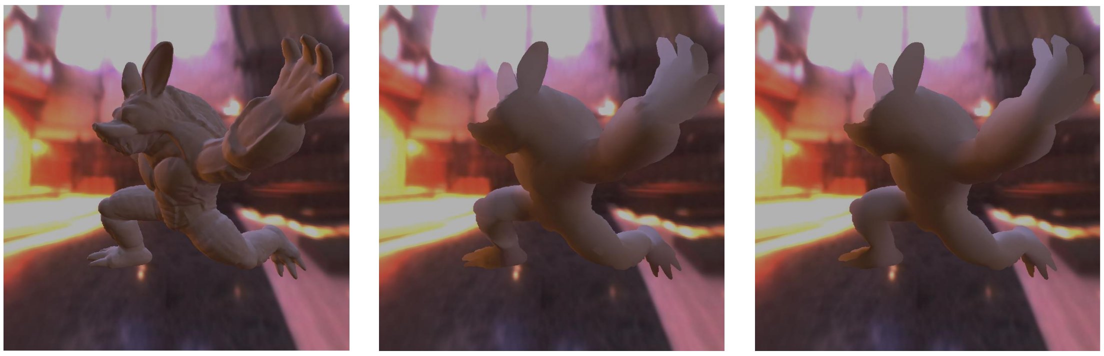
In a simple local illumination model, the lighting at a surface point is trivially related to the surface normal at that point. If we wish to crudely approximate subsurface scattering in such a local illumination model, we must express the dependency of the lighting at a surface point on the incident lighting at surrounding surface points, which is, in turn, directly related to the surface normals of those surrounding points. Thus, in such a model, we can express the relationship between the illumination of a surface point and the incident radiance at the points in its *effective scattering neighborhood* as a weighted sum of the normals of those points, if we choose the weights in an intelligent fashion. Assume we wish to modify the normal at a point, v_i, to account for the incident light at its neighboring points. Let the *effective scattering neighborhood* of v_i be $N(v_i) = \{v_j | dist(v_i,v_j) < d\}$, for some small distance d. Further, let n_{v_i} denote the (original) surface normal at v_i. Then, the subsurface scattering normal at v_i, denoted ssn_{v_i}, is computed as:

$$ssn_{v_i} = \alpha n_i + \sum_{n_j \in N(v_i)} n_j * e^{-(\sigma_{tr} * dist(v_i,v_j))} \tag{3}$$

Where $\alpha < 1.0$ is some value which designates how much of the lighting at v_i is determined by n_{v_i} versus the neighboring normals, and σ_{tr} is the effective transport coefficient, which represents the degree to which light is impeeded as it moves through the material. Both the choice of α and the choice of σ_{tr} affect the amount of subsurface scattering.

Now, when rendering the model, if the normal of a given surface point (n_{v_i}) is replaced by the subsurface scattering normal (ssn_{v_i}) as calculated in equation 3, then the illumination displayed at v_i will be dependent upon the light incident in its *effective scattering neighborhood*. The subsurface scattered normal which is computed, is effective both in standard local illumination models such as Blinn-Phong using point light sources (as used in OpenGL), as well as when approximating more complicated lighting environments through the use of methods such as irradiance maps [7].

Though this approximation is very simple, it captures the most prevelant features of subsurface scattering. Considering that no runtime overhead is incurred, the effects it produces are relatively convincing and such an approximation could prove useful in application domains where speed is essential (such as games). While this method is fairly effective, the computations involved produce some “descretization” artifacts as seen in figure 2. These artifacts occur when the mesh is coarsely or unevenly sampled (as are many meshes used in real-time applications), and result from triangles where the normals vary significantly among the vertices. Further, the only physically interpretable parameter in our approximation is σ_{tr}, the effective transport coefficient. Thus, to achieve the desired appearence, the user must re-run the preprocessing step, modifying α, σ_{tr}, and d to suit their specific needs. To overcome these issues, we present a spectral processing framework that provides an alternate method of computing these subsurface scattering normals, which does not suffer from these shortcomings.



(a) Lit by a diffuse irradiance map

(b) Subsurface scattering normals generated using the geometric space filter

(c) Subsurface scattering normals generated using the frequency space filter

Fig. 1. The Stanford Armadillo in a modified pose

3.2 Spectral Processing Framework and Low-Pass Frequency Filter

If we view the normals associated with each vertex in our mesh as a signal on the surface of the mesh, and we have an acceptable set of basis functions on which to represent this signal, then we can use techniques from signal processing to achieve an effect very similar to that detailed above. To achieve this goal, we first need to determine an appropriate set of basis functions onto which we will project our normals. We chose the eigenfunctions of the graph Laplacian as defined by the Laplace-Beltrami operator as presented in [5]. These eigenfunctions provide an ideal basis because they combine information which is inherent in both the geometry and topology of the mesh, and, given the formulation below, they form and orthogonal basis, so projection of the signal onto this basis becomes trivial.

First, we compute the graph Laplacian matrix, $\mathscr{L}$, defined by:

$$\mathscr{L}_{ij} = \begin{cases} \frac{(\frac{1}{2}(\cot(\beta)+\cot(\beta')))}{(\frac{(|\Omega_i|+|\Omega_j|)}{2})}, & \text{if } (i,j) \text{ is an edge and } i \neq j \\ -\sum_{\forall j} \mathscr{L}_{ij}, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Where β and β' refer to the angles opposite the edge shared by v_i and v_j, and Ω_i, is $\frac{1}{3}$ of the sum of the areas of the faces incident at vertex v_i and similarly for Ω_j. The resultant matrix will be both sparse and symmetric. Further, since $\mathscr{L}$ will be a positive semi-definite matrix, its eigenvectors will be orthogonal. Next, we compute some number, m, of the eigenvalues and eigenvectors of $\mathscr{L}$ with $m << n$ (where n is the number of vertices in our target mesh). Though there are n total eigenpairs, only signals on the mesh upto a certain frequency, ω_{cutoff}, are represented in the spectral transform. We approximate this ω_{cutoff} with ω_m, ten times the average edge length, as suggested in [8]. Thus, we compute only the eigenpairs, $(\lambda_k, \vec{v_k})$ where $\lambda_k < \omega_m^2$ (as an eigenvalue, λ_k corresponds to the frequency $\omega_k = \sqrt{(\lambda_k)}$). We then use these m eigenvectors as the basis functions $\phi_1, \phi_2, \ldots, \phi_m$ onto which we will project our normal vectors. The projection of the normal $n_j = (n_j^x, n_j^y, n_j^z)$ onto each of our m basis functions is simply computed as follows:

$$\eta_i^x = \sum_{j=1}^{m} n_j^x \phi_i(j) \tag{5}$$

$$\eta_i^y = \sum_{j=1}^{m} n_j^y \phi_i(j) \tag{6}$$

$$\eta_i^z = \sum_{j=1}^{m} n_j^z \phi_i(j) \tag{7}$$

This projections takes our normals from the standard geometric space to the spectral, or harmonic, space. Once we have obtained our spectral coefficients $\eta_1^x, \eta_1^x, \ldots, \eta_m^x$ and similarly for y and z, we can apply our low pass spectral filter. Our filter $F(\omega)$ assigns a weight to each frequency ω in the spectral domain. For a low pass filter the value of $F(\omega)$ will be close to 1 when ω is low and $F(\omega)$ will fall off to 0 as ω increases. In our experiments we used the normal distribution as our $F(\omega)$, varying the standard deviation to effect the rate of falloff. Since each of the basis functions onto which we have projected the normals is orthognal, we can apply the filter when reconstructing our normals, transforming them back from frequency to geometric space as follows:

$$n'^x_j = \sum_i F(\omega_i) \eta_i^x \phi_i(j) \tag{8}$$

$$n'^y_j = \sum_i F(\omega_i) \eta_i^y \phi_i(j) \tag{9}$$

$$n'^z_j = \sum_i F(\omega_i) \eta_i^z \phi_i(j) \tag{10}$$

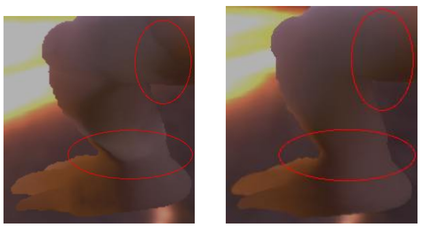
Where, for a given η_i, the corresponding frequency, ω_i is given by $\frac{\sqrt{(\lambda_i)}-\sqrt{(\lambda_1)}}{\sqrt{(\lambda_m)}-\sqrt{(\lambda_1)}}$, the normalized eigenvalue of the eigenfunction ϕ_i. Our, reconstructed normal is now $n' = (n'_x, n'_y, n'_z)$ and it is analogous to our subsurface scattered normal from the previous section. The frequency space convolution of the filtering kernel $F(\omega)$ with the projected normal signal mimics the filtering of the intensity of the lighting signal in geometric space.

It should be noted that the very high detail components of the normal signal, those which have corresponding frequencies greater than ω_m, are not considered above in the creation of the subsurface scattered normal. It is, however, possible to apply the filter, $F(\omega)$, to the high frequency signals as described in [8]. To do this, we explicitlly store the difference of each each surface normal from the unfiltered reconstruction of that normal. That is, assume we reconstruct the normal, n', as given above from equations 8 9 and 10, and we use the constant frequency space filter $F(\omega) = 1$. Such a formulation represents the most exact reconstruction possible. Therefore, the difference of the reconstructed normal from the original normal is given by $\Delta n = n - n'$. This difference, Δn, is precisely the detail of the normal which was not represented in the spectral transform. We may now apply our convolution filter $F(\omega)$ to Δn. This high detail signal exists in the frequency space, occupying the bands from ω_m to ω_M (where ω_M is the Nyquist frequency of the mesh). Thus, though we cannot futher separate this Δn into multiple frequency space bands, we can filter the signal as a whole by the average value of $F(\omega)$ on the interval $[\omega_m, \omega_M]$. This average, γ, is given by the following equation:

$$\gamma = \frac{1}{\omega_M - \omega_m} \int_{\omega_m}^{\omega_M} F(\omega) d\omega \tag{11}$$

Then, instead of simply $n' = (n'_x, n'_y, n'_z)$, our reconstructed normal becomes $n'' = n' + (\gamma * \Delta n)$.

By using this framework to compute our subsurface scattered normals, we avoid the pitfalls of the previous method. First, the "descretization" artifacts, highlighted figure 2, which are introduced by the previous method are avoided when using this method. This is due to the fact that the projection of the normal signal onto our chosen basis functions induces an implicit resampling of the signal. This results in the higher overall visual fidelity of the rendering and a more convincing subsurface scattering effect. Second, since the reconstruction of the normal from the frequency space coefficients is simply a sum of products, it is fast to compute, and is likely amenable to a GPU based implementation. Thus, if the frequency space coefficients are stored, then a new filtering kernel may be applied at runtime, and the resultant subsurface scattering normals can be computed and displayed. This allows the degree of the subsurface scattering effect to be adjusted to the user's taste.



(a) Normals generated by geometry space filter

(b) Normals generated by frequency space filter

Fig. 2. Artifacts introduced into the subsurface scattering approximation by the naïve implementation are avoided using the spectral processing method

4 Conclusion and Future Work

We have presented a simple approximation which simulates the most basic features of subsurface scattering, and which represents the majority of its apparent effect. Our method relies on the results from a preprocessing stage, but once these results are available, it can be rendered without incurring any runtime overhead. We have described a naïve approach which can be implemented in an extremely simple and straighforward manner, but which suffers from two main setbacks. First, the naïve approach produces "descretization" artifacts during rendering which are the result of high variability in sampling density across the mesh. Second, since our approximation has only a single physically interpretable parameter, the costly preprocessing step may need to be run many times before the other, non physically interpretable parameters can be set in a fashion that produces the level of subsurface scattering desired by the user. To address these shortcomings, we have presented a spectral processing framework based on [5] and [8] which allows for a more general and expressive method of computing the subsurface scattered normals. This framework does not suffer from the "descretization" artifacts of the naïve approach. Further, for the simple cost of explicitly storing the spectral coefficients of the normals, the user can test the effects of different frequency space filters quickly at runtime. This allows the user to much more quickly converge upon the appearence they want for their model, because they need not run the expensive preprocessing phase each time they want to approximate a different degree of subsurface scattering on the mesh. Because this method of approximating subsurface scattering effects has no associated overhead during runtime, we forsee its use in applications where subsurface scattering is desired as a visual effect, but where performance is the primary goal (applications such as video games). Despite the simplicity of our approximation, we believe it produces relatively convincing results and is suitable for many real-time visualization needs.

Currently, our method suffers from an inability to bleed subsurface scattering effects across shadow boundaries. We envision a few possible solutions, and intend to address this issue in our future work. Further, as mentioned in 3.2, although application of the frequency space filter is already fairly fast, we believe it would be sped up significantly if implemented on the GPU instead of on the CPU as is currently done. This would allow the filter to be modified at interactive rates, even for extremely large models.

Acknowledgements

We would like to thank the Aim @ Shape project for providing an excellent repository of "clean" models, many of which can be obtained at multiple resolutions. We would also like to thank Professor Bruno Levy for his advice in helping us re-implement the results of [5]. Finally, we would like to thank Professor Amitabh Varshney for his discussions and suggestions which have contributed to the design and presentation of this work.

References

[1] X. Hao, T. Baby, and A. Varshney. Interactive subsurface scattering for translucent meshes. In *ACM Symposium on Interactive 3D Graphics*, pages 75 – 82, April 28 – 30, 2003.

[2] X. Hao and A. Varshney. Real-time rendering of translucent meshes. *ACM Transactions on Graphics*, 23, No. 2:120 – 142, 2004.

[3] M. L. Henrik Wann Jensen, Steve Marschner and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH*, pages 511–518, August 2001.

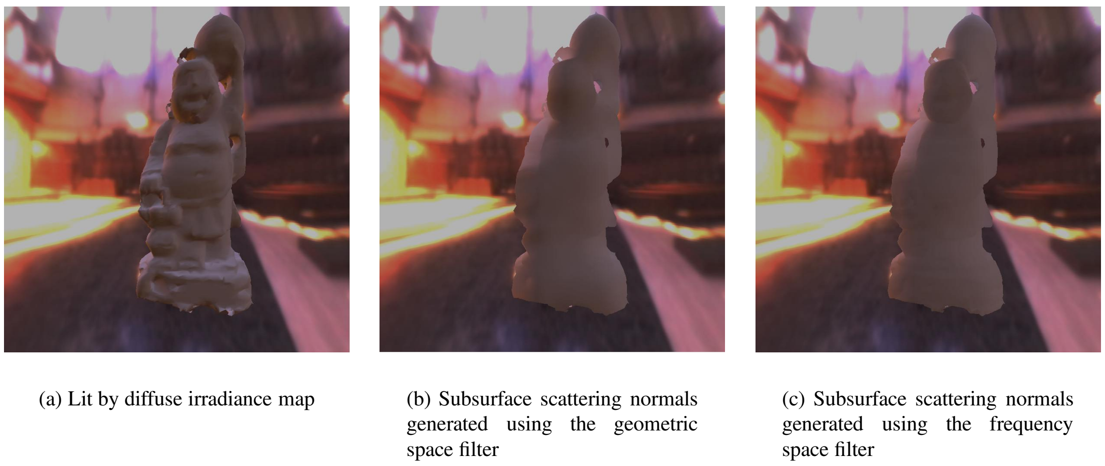
[4] G. M. B. P. K. J. M. M. L. J. Lensch, H. and H.-P. Seidel. Interactive rendering of translucent objects. In *Proceedings of Pacific Graphics*, pages 214–224, 2002.

[5] B. Levy. Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications*, 2006.

[6] J. C. H. Nathan A. Carr, Jesse D. Hall. Gpu algorithms for radiosity and subsurface scattering. In *Proc. Graphics Hardware*, July 2003.

[7] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of SIGGRAPH*, pages 497–500, August 2001.

[8] B. Vallet and B. Levy. Spectral geometry processing with manifold harmonics. In *Tech Report*, April 2007.



(a) Lit by diffuse irradiance map

(b) Subsurface scattering normals generated using the geometric space filter

(c) Subsurface scattering normals generated using the frequency space filter

Fig. 3. A mesh of a monk