# Hybrid techniques for interactive photorealistic rendering

Alessandro Dal Corso

Kongens Lyngby 2018

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

七転び八起き.
*Fall down seven times, stand up eight.*
- Japanese proverb

# Summary (English)

The goal of the thesis is to ...

# Summary (Danish)

Målet for denne afhandling er at ...

# Preface

The work from this Ph.D. thesis was carried under the Section for Image Analysis and Computer Graphics, at the Department for Applied Mathematics and Computer Science (DTU Compute) at the Technical University of Denmark (DTU). This thesis was prepared in fullfillment of the requirements to obtain a doctor of philosophy (Ph.D.) in computer science. The work presented was funded on a DTU Compute internal scholarship.

This thesis deals with the topic of bringing techniques from photorealistic offline rendering into the interactive and real time domain. To achieve this goal, a set of publications was produced over the course of the studies. These publications that are summarized at page xi. The publications are available in Appendices I-VII. From these, a total of five peer reviewed publications have been included in this thesis (I-V), while two (VI-VII) have been left out as not relevant to the overall thesis topic. Moreover, during the course of the Ph.D. some notes were produced. These notes are grouped as non peer reviewed material and attached in Appendices VIII-XI.

The work has been carried under the main supervision of Associate Professor Jeppe Eliot Revall Frisvad, with the co-supervision of Associate Professor Andreas Bærentzen. The research activies have mostly been conducted at DTU, with the exception of two external research stay periods. The first research stay was an internship at NVIDIA Corporation, under managers Aaron Lefohn and David Luebke, in the real time rendering research team based in Redmond, Washington, USA. The second external research stay was under the supervision of Professor Toshiya Hachisuka, Department of Creative Informatics, The University of Tokyo.

Lyngby, 31-March-2018

Alessandro Dal Corso

# Acknowledgements

First and foremost, I would like to thank my supervisors Jeppe Revall Frisvad and Andreas Bærentzen. In particular, I would like to thank Jeppe for the support during the PhD, the deep technical discussions and a push to strive for perfection. I would like to thank Andreas for the deep and interesting discussion on how to push my work towards more and more challenging directions. I would also like to thank my managers and mentors during my internship at the real-time rendering research group in NVIDIA: Craig Kolb, Marco Salvi, Aaron Lefohn and David Luebke, for sharing some of their deep knowledge about graphics. Finally, I would like to thank Toshiya Hachisuka and its group for hosting me at Tokyo University, for the their hospitality and the interesting discussions.

I would like to thank DTU Compute for financing my scholarship and giving me the opportunity to pursue a PhD in the realm of computer graphics.

A great thank you of course goes to my colleagues, collaborators, and office mates at the section for Image Analysis and Computer Graphics. Thank you for the continuous support, the coffee calls, and the general silliness that makes everyday a fun day to come to the office. It has been a great pleasure to be part of this section, sharing thoughts and ideas, and having interesting discussion over a coffee.

Last but not least, I would like to thank my loving family and friends. Without your continuous support and help, I would not have been able to complete this endeavor.

# Contents

**Summary (English)** **i**

**Summary (Danish)** **iii**

**Preface** **v**

**Acknowledgements** **vii**

**1 Introduction** **1**
1.1 Scope . . . 1
1.2 Motivation . . . 2
1.2.1 3D printing . . . 4
1.2.2 Architectural visualization . . . 6
1.2.3 Computer games . . . 6
1.3 Outcome . . . 7
1.4 Outline . . . 8

**2 Background** **9**
2.1 Radiometry . . . 9
2.2 Scattering Media . . . 12
2.2.1 The radiative transfer equation . . . 12
2.2.2 The BSSRDF . . . 14
2.2.3 Connecting BSSRDFs and the radiative transfer equation 15
2.2.4 Global solution to the BSSRDF . . . 20
2.2.5 Rendering scattering media . . . 21
2.2.6 Analytical BSSRDF Models . . . 24
2.3 Non-scattering media . . . 27
2.3.1 The BRDF . . . 28
2.3.2 Empirical BRDFs . . . 29
2.4 Rendering techniques . . . 30
2.4.1 Rasterization . . . 31
2.4.2 Ray tracing . . . 33

**3 Related work** **35**

3.1 Approximate physically based rendering . . . . . 36
3.2 Interactive rendering techniques . . . . . 37
3.2.1 Caching . . . . . 38
3.2.2 Pre-computation . . . . . 39
3.2.3 Filtering . . . . . 40

**4 Contributions 43**
4.1 Outline and motivation . . . . . 43
4.2 Defining photorealistic rendering . . . . . 44
4.3 Interactive rendering of scattering media . . . . . 46
4.4 Interactive stable ray tracing . . . . . 48
4.5 Applying interactive photorealistic techniques . . . . . 50
4.6 Discussion . . . . . 50

**5 Conclusions 53**

**I Interactive Appearance Prediction for Cloudy Beverages 57**

**II Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering 63**

**III Interactive Directional Subsurface Scattering and Transport of Emergent Light 77**

**IV Interactive Stable Ray Tracing 91**

**V Virtual Reality inspection and painting with measured BRDFs 103**

**VI VirtualTable: a projection augmented reality game 107**

**VII Our 3D Vision Data-Sets in the Making 109**

**VIII Derivation of standard and directional dipole quantities 115**

**IX Point cloud method for rendering BSSRDFs 129**

**X On BSSRDF estimation 133**

**XI Camera space BSSRDF importance sampling 139**

**Bibliography 143**

# List of contributions

## Peer rewiewed

- **Alessandro Dal Corso**, Jeppe Revall Frisvad, Jesper Mosegaard, and J. Andreas Bærentzen. Interactive directional subsurface scattering and transport of emergent light. *The Visual Computer*, 33(3):371–383, Mar 2017. [DCFMB17]

- Jonathan Dyssel Stets, **Alessandro Dal Corso**, Jannik Boll Nielsen, Rasmus Ahrenkiel Lyngby, Sebastian Hoppe Nesgaard Jensen, Jakob Wilm, Mads Brix Doest, Carsten Gundlach, Eythor Runar Eiriksson, Knut Conradsen, Anders Bjorholm Dahl, Jakob Andreas Bærentzen, Jeppe Revall Frisvad, and Henrik Aanæs. Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering. *Applied Optics*, 56(27):7679–7690, Sep 2017. [SDN+17]

- **Alessandro Dal Corso**, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. Interactive stable ray tracing. In *Proceedings of High Performance Graphics*, HPG '17, pages 1:1–1:10, New York, NY, USA, 2017. ACM. [DCSK+17]

- **Alessandro Dal Corso**, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. Virtual reality inspection and painting with measured brdfs. In *SIGGRAPH Asia 2017 VR Showcase*, SA '17, pages 15:1–15:2, New York, NY, USA, 2017. ACM. [DCSL+17]

- **Alessandro Dal Corso**, Jeppe Revall Frisvad, Thomas Kim Kjeldsen, and Jakob Andreas Bærentzen. Interactive Appearance Prediction for

Cloudy Beverages. In Reinhard Klein and Holly Rushmeier, editors, *Workshop on Material Appearance Modeling*. The Eurographics Association, 2016. [DCFKB16]

## Non included

- Henrik Aanæs, Knut Conradsen, **Alessandro Dal Corso**, Anders Bjorholm Dahl, Alessio Del Bue, Mads Emil Brix Doest, Jeppe Revall Frisvad, Sebastian Hoppe Nesgaard Jensen, Jannik Boll Nielsen, Jonathan Dyssel Stets, et al. Our 3d vision datasets in the making. In CVPR Workshop: The Future of Datasets in Vision 2015, 2015. [ACDC+15]
- **Alessandro Dal Corso**, Mikkel Olsen, Kasper Hornbak Steenstrup, Jakob Wilm, Sebastian Jensen, Rasmus Reinhold Paulsen, Eythor Eiríksson, Jannik Boll Nielsen, Jeppe Revall Frisvad, Gudmundur Einarsson, and Hans Martin Kjer. Virtualtable: A projection augmented reality game. In *SIGGRAPH Asia 2015 Posters*, SA '15, pages 40:1–40:1, New York, NY, USA, 2015. ACM. [DCOS+15]

## Unpublished material

- **Alessandro Dal Corso, Jeppe Revall Frisvad, Thomas Kim Kjeldsen**. *Derivation of standard and directional dipole quantities.* Unpublished note. Text in Appendix VIII.
- **Alessandro Dal Corso, Jeppe Revall Frisvad**. *On measuring BSSRDFs.* Unpublished note. Text in Appendix X.
- **Alessandro Dal Corso, Jeppe Revall Frisvad**. *Point cloud method for rendering BSSRDFs.* Unpublished note. Text in Appendix IX.
- **Alessandro Dal Corso**. *Note on efficient color calibration.* Unpublished note. Text in Appendix **??**.
- **Alessandro Dal Corso**. *Note on BSSRDF importance sampling from the camera, derivation from Mertens et al..* Unpublished note. Text in Appendix XI.

Chapter 1

# Introduction

**FIXME:** Add Jeppe's overall comments

## 1.1 Scope

The main goal of this thesis is to describe new techniques to bring movie production quality rendering into interactive applications. The goal is to achieve photorealistic appearance as close as possible to the real world, while retaining the possibility for the user to interact with the simulation. In this thesis, we name these techniques "hybrid", as they combine techniques from the interactive world of rendering techniques with accurate appearance models from physics and movie production pipelines.

Interactive realistic rendering applications are relevant in many fields, including

- 3D digital modeling and 3D printing,
- Product development and visualization
- Digital prototyping

**Figure 1.1:** Some examples of photorealistic rendering. Left, rendering of a forest obtained in Maya (courtesy Greg Zdunek). Right, real time rendering in *Far Cry 4* (courtesy Ubisoft).

- Computer games and animation

In recent years, many advances have been made to achieve photorealistic quality in the interactive applications listed above, often achieving results remarkably close to slower photorealistic rendering techniques (see Figure 1.1). This thesis contributes with various techniques, exploring the area between interactive and real time rendering techniques (e.g. rasterization and interactive ray tracing) and photorealistic rendering techniques (e.g. path tracing, hierarchical point cloud methods), with the goal of applying photorealistic appearance models (BRDFs and BSSRDFs, both analytical and measured). As most modern rendering techniques, we exploit the high throughput of graphics processing units (GPUs). Our contributions exploit both the long standing GPU hardware rasterization pipeline, but also focus on applications of the currently expanding field of interactive ray tracing.

## 1.2 Motivation

From the beginning of computer graphics, researchers have been striving to synthesize more and more realistic images. Nowadays, for still images and carefully crafted scenes, it is almost impossible to distinguish a synthetic image from an

**Figure 1.2:** Comparing different preview techniques. From left to right: converged image, noisy unconverged image, downsampled image, different physical model, diffuse appearance. All images took 120 seconds to render on a GPU ray tracer. Converged result took 4 hours.

actual photograph. To achieve these photorealistic images, a great amount of hours must be spent in modeling or acquiring the 3D geometry, and choosing or measuring the correct materials. Finally, some time must be spent rendering the image, that can take up to several hours. In the rendering part, the more advanced the physical model used, the higher the rendering time. This high rendering time can cause production bottlenecks, e.g. when some artistic tweaking is needed to achieve a specific style or artistic constraint. Once the tweaks are performed, the whole rendering needs to be redone in order to see the influence of the tweak on the final result, wasting precious production time.

To mitigate this problem, many applications display a preview of the final rendering result. Depending on the application the technique can be different (see Figure 1.2), like displaying a noisy result, a downsampled one, or simply switching to a simpler rendering model. All these techniques come with different trade offs, advantages and disadvantages. In general, these preview techniques introduce some sort of bias (i.e. the image is no longer physically correct) in exchange to getting rid of noise.

Given these problems, when immediate feedback is important, there is a need in the industry for clever techniques that exploit the main two rendering constraints, namely time and accuracy. We classify and visualize various applications according to these constraints in Figure 1.3. The first constraint, allowed rendering time, ranges from a few milliseconds per frame in a real-time environment (like games or virtual reality), to some fractions of a second (3D printing

**Figure 1.3:** Visualizing the time-accuracy graph, showing the different application fields.

preview) up to some minutes or hours (frame in movie production). The second constraint we mentioned is accuracy, i.e. the accuracy in modeling the physical process behind the rendering. In some applications, the accuracy is limited. For example, in a video game, it is more important to obtain a fast result that feels physically correct rather than a physically accurate result. In some other applications, the requirements are stricter: think architectural light simulation, where architects need to know precisely how the light reacts to different surfaces to see the overall illumination of the environment.

We now discuss three more detailed use cases in which there is a demand for improvements in photorealistic rendering techniques, visualized in Figure 1.4. These cases are not meant to be comprehensive of all applications in which interactive photorealistic rendering techniques are needed. However, they provide insights on possible industrial applications of the contributions in this thesis.

### 1.2.1 3D printing

The first case we discuss applications of interactive photorealistic rendering is additive manufacturing, in particular 3D printing. In this application, we want to preview the appearance of the printed object. Many challenges are required

**Figure 1.4:** Applications of photorealistic interactive rendering. Left to right: 3d printing preview, architectural visualization (courtesy *Lumion*), virtual reality games (courtesy *The Climb* by *Crytek*).

to faithfully represent a 3D printed object, depending on the production process used. This process greatly influences the shape and the appearance of the final object. Moreover, the materials used for 3D printing exhibit some particular radiometric properties, that makes them even more challenging to render accurately.

In this application, the role of photorealistic rendering is dual. Apart from previewing the final result of the 3D printing process, we can actively use photorealistic rendering to predict and measure the radiometric properties of the object. This requires a specific setup and calibrated scene, in which a picture of the object is taken. Then, lots of different renderings can be made, tweaking the radiometric properties to predict the final appearance of the object. In both these applications, we require a high accuracy in our rendering (see Figure 1.3). In this case, a noisier or downsampled result can be acceptable, at the price of a certain inaccuracy in measuring the parameters.

All these innovations go towards a more strict integration of 3D printing processes and digital image synthesis as a whole (the digital-physical ecosystem). Photorealistic rendering helps reduce the number of iterations necessary to achieve a satisfying printed piece. This translates to reduced printing times for the same quality, and it avoids wasting potentially expensive printing material. Moreover, in a production environment, the validation through rendering enables us to perform quality assurance on the finished pieces, by comparing them with images of previously printed versions of the same piece.

### 1.2.2 Architectural visualization

Our second case is architectural visualization. In this particular field, architects use interactive visualization tools to show how light propagates through a building at different times of the day. This allows to create buildings that best use natural illumination, minimizing electricity consumption for the building. In this particular application, an important focus is placed on accurately represent materials. Materials used in construction like frosted glass, marble or cloth, are radiometrically complex and hard to render fast and accurately. These materials can be either measured using some sort of apparatus, or matched by an artist so that they represent the appearance and light interaction properties of the original material. Given the complexity of the models and the generally needed artistic tweaking time, we found this field an interesting application field for interactive photorealistic techniques.

Recent efforts have been trying to offer a more interactive experience, in order to show the finished product to project stakeholders. So, architectural visualization is moving towards interactive or real-time visualization, in particular in a virtual reality environment. Virtual reality pipelines need to maintain the same standard of physically based accuracy as the original simulation, while fitting into extreme time constraints, a handful of milliseconds per frame. In this case, fast techniques that exploit current hardware to achieve photorealistic visualization are even more important.

### 1.2.3 Computer games

The final application of fast interactive photorealisitic techniques we discuss is in computer games. Computer games have the strictest time constraints of all the applications so long presented. For a standard frame in a game, the industry standard is around 16 milliseconds (60 frames per second). In recent year, virtual reality games are starting to become prominent, lowering the standard to 7-11 milliseconds total (90-120 frames per second). Generically, games use triangle based rasterization, so that they can leverage custom hardware on GPUs.

As games vary for gameplay, environment and style, photorealistic rendering can or cannot be employed in games. Nowadays, a good number of games strive to achieve a photorealistic or quasi-photorealistic look. In games, the focus is on performance and immersiveness, so the simulation does not need to be accurate from a physical standpoint. Actually, sometimes the rendering algorithms are made non realistic on purpose to achieve a specific artistic look. Given the hard time constraints, specific techniques need to be developed to achieve a

**Figure 1.5:** The main contributions from this thesis in the time-accuracy graph. The roman number on each contribution links to the original published text.

photorealistic look. These techniques are often tailored to a specific game, and no overall solution for all games exists. This makes it an interesting field to contribute with general scalable physically based techniques, with the ultimate goal to reduce production costs.

## 1.3 Outcome

During the three years period, we achieved a number of academic publications, reported at page xi. In these publication, we managed to achieve a number of insights in the realm of computer graphics and material appearance. In particular, we created a range of techniques that can be readily applied to existing interactive applications. We believe we pushed the boundary on the degree of realism that can be achieved by modern GPUs on the same strict time constraints.

In particular, papers III and IV achieve this by providing widely applicable caching in the realm of interactive subsurface scattering and global illumination. Some examples of the results achieved in these papers are reported in Figure 1.5.

## 1.4 Outline

A list of all the contributions created during the course of the PhD studies at page xi, including some unpublished notes, that we will refer mostly from Chapter 2 to provide additional details. All the various publications are available as appendices.

The thesis into four main chapters. In Chapter 2, we first cover some background on some of the necessary foundations in radiometry, photorealistic appearance models and interactive rendering techniques. We will then cover in Chapter 3 an overarching literature review on the current efforts of the rendering community to achieve photorealistic interactive rendering, referring to the individual contributions for more detailed related work. In Chapter 4, most importantly, we will go into more detail about the individual contributions and how each one of them contributes to reaching the outcome of the thesis. We will finally summarize our conclusions in Chapter 5.

Chapter 2

# Background

In this section, we will introduce the theorectical elements necessary to complete the related work and the contributions illustrated in Chapters 3 and 4. Our contributions range a wide span of physically based materials, including both scattering and non scattering media. We will describe both in this chapter (Sections 2.2 and 2.3 respectively), with a brief introduction to basic radiometric concept (Section 2.1). We will finally conclude (Section 2.4) with a more detailed description of the two main rendering paradigms we used to achieve our results, rasterization and ray tracing.

## 2.1 Radiometry

Radiometry is a branch of physics that describes and measures electromagnetic radiation. In this section, we will define some basic radiometric quantities. In particular, we want to give a definition of radiance, the most useful quantity in describing light transport along rays.

First, let us consider an ideal point source of photons in space. This source emits a certain amount of energy $U$, measured in Joules $[J]$. The first quantity we derive is radiant flux or radiant power $\Phi$, defined as the amount of energy

**Figure 2.1:** Configuration to define radiance.

per second emitted by the light:

$$\Phi = \frac{dU}{dt} \quad [\mathrm{W}].$$

The total flux represents the power the light emits in all directions. We usually want to be more descriptive on how a light or a surface is emitting light, since not all the sources we consider are ideal. First, we are interested on how light emission changes across directions. We define as *radiant intensity* $I$ the amount of flux the light emits towards a specific direction $\vec{\omega}$:

$$I(\vec{\omega}) = \frac{d\Phi}{d\omega} \quad [\mathrm{W} \cdot \mathrm{sr}^{-1}],$$

where $d\omega$ is an infinitesimal solid angle centered on direction $\vec{\omega}$. An isotropic point light, by definition, has constant intensity across all directions.

We now consider an infinitesimal surface area element $dA$ that receives light, with center point $\mathbf{x}$. We now define *irradiance* as the amount of incoming flux received per unit area:

$$E(\mathbf{x}) = \frac{d\Phi}{dA} \quad [\mathrm{W} \cdot \mathrm{m}^{-2}].$$

If the flux is emitted by the surface, we use the name *radiosity* and define it with the symbol $B$. The most simple irradiance is the one emitted by an isotropic point light. In this case, the irradiance at a distance $R$ from the point light is

$$E = \frac{\Phi}{4\pi R^2} \quad [\mathrm{W} \cdot \mathrm{m}^{-2}].$$

So, the farther we go from a point light, the less irradiance we receive, because the power $\Phi$ is spread across a larger area.

Finally, we combine the definitions of intensity and irradiance above into one, to define our last important basic radiometric quantity, *radiance*:

$$L(\mathbf{x}, \vec{\omega}) = \frac{d^2\Phi}{dA \cos\theta d\omega} \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}],$$

**Figure 2.2:** Configuration to prove the equality of radiance across a ray.

where $\cos\theta = \vec{n} \cdot \vec{\omega}$ is the cosine of the angle between the normal at $\mathbf{x}$ and the direction of evaluation. $dA \cos\theta$ is called the *projected area element*. As irradiance, radiance can be incoming or outgoing from a specific point. We indicate these quantities with $L_i$ and $L_o$, respectively. In graphics, radiance is usually the most useful quantity for two main reasons. First, the other quantities can be easily computed from radiance through radiometric integrals:

$$\begin{aligned}
\Phi &= \int_{\Omega^+} \int_A L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ dA\ d\omega, \\
I(\vec{\omega}) &= \int_A L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ dA, \\
E(\mathbf{x}) &= \int_{\Omega^+} L(\mathbf{x}, \vec{\omega})(\vec{n} \cdot \vec{\omega})\ d\omega,
\end{aligned}$$

where $\Omega^+$ and $A$ are the hemisphere around $\vec{n}$ and the total surface, respectively. Second, radiance carried by a ray *in vacuo* is constant. We can prove this quite easily, with the aid of Figure 2.2. Note that for point $\mathbf{x}_o$, we have by definition $d\omega_o = \frac{dA_i \cos\theta_i}{r^2}$, and similarly for $d\omega_i$. Then:

$$\begin{aligned}
L_i(\mathbf{x}_i, \vec{\omega}_i) = \frac{d^2\Phi}{dA_i \cos\theta_i d\omega_i} &= \frac{d^2\Phi}{dA_i \cos\theta_i \frac{dA_o \cos\theta_o}{r^2}} \\
&= \frac{d^2\Phi}{\frac{dA_i \cos\theta_i}{r^2} dA_o \cos\theta_o} = \frac{d^2\Phi}{dA_o \cos\theta_o d\omega_o} = L_o(\mathbf{x}_o, \vec{\omega}_o).
\end{aligned}$$

Note that we assume that the flux does not vary across the path, which is generally true if no objects are in the way and there is no medium in between the two points causing scattering events.

**Figure 2.3:** Individual processes in the local form of the radiative transfer equation: absorption, emission, in-scattering and out-scattering. The directional derivative $\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega})$ corresponds on the variation over an infinitesimal length element $\mathrm{d}r$ in direction $\vec{\omega}$.

## 2.2 Scattering Media

The first group of physically based materials we tackle is scattering media. Scattering is the process photons go through once they hit an optically dense medium. Photons are deflected from their current path by electromagnetic forces, due to atom nuclei and atomic bonds. It is possible to model this process mathematically. We start by discussing one of such models, the radiative transfer equation, which is a local solution of the scattering process for a point in the medium. Then, we will provide a more global formulation of the scattering process as a function of light propagating between two surface points, the BSSRDF. We will then proceed to prove that the BSSRDF indeed respects the radiative transfer equation, and then describe two ways to render materials: one using the radiative transfer equation, one using BSSRDF. Finally, we will introduce three BSSRDF analytical models that can be used in rendering.

### 2.2.1 The radiative transfer equation

The radiative transfer equation [Cha50] describes light propagation in scattering media. It is a particular form of the Boltzmann transport equation from thermodynamic system. In this section, we describe its integro-differential form, that describes how radiance $L(\mathbf{x}, \vec{\omega})$ varies at a point $\mathbf{x}$ in a scattering medium towards direction $\vec{\omega}$. For the sake of simplicity, from now on we assume a scattering media that respects linear optics (excluding i.e. fluorescent materials), and in the stationary state (i.e. the radiance $L$ is not changing in time).

Traditionally, light traveling a medium is subject to four processes: absorption, emission, in-scattering and out-scattering. Each of these processes can be expressed in terms the directional derivative $\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega})$ $[\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}]$. All effects are illustrated in Figure 2.3.

**Emission** increases the overall radiance of the ray by a term $q(\mathbf{x}, \vec{\omega})$:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = q(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Emission describes how some form of energy (like heat) is becoming radiant energy.

**Absorption** is the dual effect of emission, where photons are absorbed by the material, and generally being transformed from radiant energy into heat energy. On the opposite hand of emission, the loss due to absorption is proportional to the radiance at the point $L$. The *absorption coefficient* $\sigma_a(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$ describes the amount of photons absorbed per unit traveled within the medium. In formulas,

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_a(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

**Out-scattering** describes photons that are deflected from their original path $\vec{\omega}$ due to interaction with the atoms of the material. The effect is similar to absorption, with a different coefficient named *scattering coefficient* $\sigma_s(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$. This gives a loss of radiance that is similar to absorption:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_s(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Absorption and out-scattering, given their similarities, are often combined into one effect, *attenuation*, described by a value called the extinction coefficient $\sigma_t(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}, \vec{\omega}) + \sigma_a(\mathbf{x}, \vec{\omega})[\mathrm{m}^{-1}]$.

**In-scattering**, finally, describes the radiance aligning towards $\vec{\omega}$ from other scattering events. Let us consider another direction $\vec{\omega}'$ from $\mathbf{x}$. We define a probability distribution for a photon to scatter from $\vec{\omega}'$ towards an infinitesimal solid angle $d\omega$ around $\vec{\omega}$. This probability distribution is called the *phase function* $p(\mathbf{x}, \vec{\omega}', \vec{\omega})[\mathrm{sr}^{-1}]$. With the phase function, and integrating over all directions, we get the effect of in-scattering.

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega' \quad [\mathrm{W} \cdot \mathrm{m}^{-3} \cdot \mathrm{sr}^{-1}].$$

Note the multiplication by $\sigma_s$so that we account for the right amount of scattering as we move along the ray. From the phase function, we can calculate the dimensionless radiometric property $g(\mathbf{x}, \vec{\omega})$, describing the mean cosine of the scattering angle between $\vec{\omega}$ and $\vec{\omega}'$ as

$$g(\mathbf{x}, \vec{\omega}) = \int_{4\pi} (\vec{\omega}' \cdot \vec{\omega}) p(\mathbf{x}, \vec{\omega}', \omega) d\omega' \quad [-].$$

**Figure 2.4:** Configuration in which we define the BSSRDF for a generic surface.

As we will see, most analytical models describe a scattering medium in terms of the three coefficients $\sigma_s$, $\sigma_a$ and $g$, plus the ratio of the index of refraction inside the medium and the one outside the medium $\eta$.

We can now combine all terms to obtain the complete integro-differential form of the radiative transfer equation:

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = q(\mathbf{x}, \vec{\omega}) - \sigma_t(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) + \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega'. \tag{2.1}$$

### 2.2.2 The BSSRDF

In Section 2.1, we have defined radiometric quantities as either incoming or outgoing. We will now describe how radiometric quantities change between incoming and outgoing. This gives a way to describe light interaction due to a surface.

Let us first consider a surface $A$ illuminated by a light, as in Figure 2.9. Let us consider the element of flux $d\Phi_i$ arriving from direction $\vec{\omega}_i$ on a surface element $dA_i$ centered at a point $\mathbf{x}_i$. Due to surface interaction, part of the incoming light will emerge at a point $\mathbf{x}_o$, in direction $\vec{\omega}_o$. We consider the proportionality factor between the emitted radiance and the incoming flux:

$$dL(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) d\Phi_i(\mathbf{x}_i, \vec{\omega}_i) \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.2}$$

The factor $S$, dependent on the surface, is called *bidirectional scattering-surface reflectance distribution function*, or BSSRDF for short [NRH+77]. The units for the BSSRDF are $[\mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]$. From the definition of BSSRDF and flux, we

obtain right away the reflected radiance equation [JMLH01]:

$$L(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_{\Omega^+} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n}_i \cdot \vec{\omega}_i) dA_i d\omega_i \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.3}$$

Note that we did not assume anything about the material, deriving the BSSRDF from purely radiometric quantities. The BSSRDF can be seen as a global formulation of scattering processes within the material.

Note that Equation 2.3 does not give the full outgoing radiance distribution $L_o$, since we need to add the radiance emitted by the body:

$$L_o(\mathbf{x}, \vec{\omega}) = L_e(\mathbf{x}, \vec{\omega}) + L(\mathbf{x}, \vec{\omega}).$$

This equation gives the so called extended form of the rendering equation [JMLH01]. Given the physical nature of the BSSRDF, we can impose conservation of energy, so that the reflected light never increases:

$$0 \leq \int_A \int_{\Omega} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{n}_i \cdot \vec{\omega}_i) dA_i d\vec{\omega}_i \leq 1 \quad [-]. \tag{2.4}$$

Moreover, the BSSRDF respects the Stokes-Helmholtz reciprocity principle [Sto09, Cha58]:

$$S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S(\mathbf{x}_o, \vec{\omega}_o, \mathbf{x}_i, \vec{\omega}_i) \quad [\mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.5}$$

### 2.2.3 Connecting BSSRDFs and the radiative transfer equation

Given the above definition of BSSRDF (Equation 2.2), one may wonder how the BSSRDF is related to any scattering process at all, since it simply describes a relationship between an incoming and an outgoing radiometric quantity. In this section, starting from the definition of the BSSRDF, we will derive the local integro-differential form of the radiative transfer Equation 2.1, to show how the BSSRDF fits nicely as a mathematical description of an underlying scattering process. Most of this derivation comes from Preisendorfer [Pre65, Pre76], where the BSSRDF is called *scattering function*.

To show this relationship, we to define some mathematical operators. A operator in our context maps a function into another function, as does this $\mathbf{Q}$ to integrate over an hemisphere:

$$\mathbf{Q} = \int_{\Omega} [\quad] d\omega.$$

**Figure 2.5:** Cylinder configuration to prove the radiance solution.

So, using the operator left associativity, $E = L\mathbf{Q}$ denotes the equation

$$E(\mathbf{x}) = \int_{\Omega} L(\mathbf{x}, \vec{\omega}) d\omega.$$

Note that in this case operator $\mathbf{Q}$ maps the functional $L$ into functional $E$. Where obvious, we drop the dependencies on $\mathbf{x}$ and $\omega$ from the operator for the sake of clarity. We can now define a new functional operator $\mathscr{S}$, also called the *standard operator*:

$$\mathscr{S}(a, b) = \int_a \int_{\Omega_i^+} [\ \ ] S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{n}_i \cdot \vec{\omega}_i) d\omega_i dA_i.$$

Where $S$ is the BSSRDF, $a$ and $b$ are parts of the surface of the medium containing $\mathbf{x}_i$ and $\mathbf{x}_o$, respectively, and $\Omega_i^+$ is the hemisphere oriented towards $\vec{n}_i$. This allows us to write $L = L_i \mathscr{S}(a, b)$ for equation 2.3, greatly simplifying notation. We want now to describe how radiance varies across a path. Let us consider the configuration of Figure 2.5. In this figure, we have a path $P_r(\mathbf{x}_i, \vec{\omega})$ starting at $\mathbf{x}_i$, with direction $\vec{\omega}_o$ for $r$ units, terminating in $\mathbf{x}_o$ (so that $\mathbf{x}_o = \mathbf{x}_i + r\vec{\omega}_o$). Let us now consider a cylindrical medium $C$ with the same index of refraction as the surroundings, composed of three parts: a top circle $a$ on $\mathbf{x}_i$, a bottom circle $b$ on $\mathbf{x}_o$ and a flank surface $c$. We orient the cylinder so that the top $a$ points towards $-\vec{\omega}_o$, and the center of the cylinder is aligned with $P_r(\mathbf{x}_i, \vec{\omega}_o)$. This defines a direction for the hemispheres, e.g. $\Omega_i^+$ and $\Omega_i^-$ are the hemispheres centered in $\mathbf{x}_i$ and oriented towards or against $\vec{n}_i$, respectively. As a final bit of notation, we indicate with $L^+(a)$ some radiance $L(\mathbf{x}, \vec{\omega})$ where $\mathbf{x} \in a$ and $\vec{\omega} \in \Omega^+$.

At the steady state and by conservation of energy, the total radiance going out of the cylinder through the surface $b$ is:

$$L^-(b) = L^+(b)\mathscr{S}(b, b) + L^-(a)\mathscr{S}(a, b) + L^-(c)\mathscr{S}(c, b). \tag{2.6}$$

We want now to find the behavior of the system at equilibrium, when the cylinder becomes thinner and thinner ($C \to P_r(\mathbf{x}_i, \vec{\omega}_o)$). We will analyze each one of the terms in Equation 2.6 independently.

The first term on the right hand side of Equation 2.6 describes the part of $L^-(b)$ that stays inside due to reflection on $b$. This can be shown to tend to zero as the cylinder shrinks. This comes from the fact that for a transparent plane, all the radiance contribution passes through so none is reflected back. In formulas,

$$\lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^+(b)\mathscr{S}(b,b)](\mathbf{x}_o, \vec{\omega}_o) = 0. \tag{2.7}$$

The second term defines the radiance transmitted inbetween $a$ and $b$. As the cylinder shrink, the photons have less and less room to scatter within the cylinder, so only the photons staying directly on $\vec{\omega}$ (even if they scatter back and forth) will be considered at the end. Let us consider the limit:

$$L_r^0(\mathbf{x}_o, \vec{\omega}_o) = \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^-(a)\mathscr{S}(a,b)](\mathbf{x}_o, \vec{\omega}_o).$$

At the limit $a \to \mathbf{x}_i$, we have $L^-(a) = L^0(\mathbf{x}_i, \vec{\omega}_o)$, that can be brought out:

$$L_r^0(\mathbf{x}_o, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [\mathscr{S}(a,b)](\mathbf{x}_o, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o), \tag{2.8}$$

where the last term is called *beam transmittance*, i.e. the amount of light blocked on the direct path. We define a dual term $1 - T_r(\mathbf{x}_i, \vec{\omega})$, called *beam attenuation*. The rate of change of beam attenuation per unit length at $\mathbf{x}_i$ on $\vec{\omega}$ is the extinction coefficient defined in Section 2.2.1:

$$\sigma_t(\mathbf{x}_i, \vec{\omega}) = \lim_{r \to 0} \frac{1 - T_r(\mathbf{x}_i, \vec{\omega})}{r},$$

which leads to a natural definition for the beam transmittance

$$T_r(\mathbf{x}_i, \vec{\omega}) = \exp\left( - \int_0^r \sigma_t(\mathbf{x}_i + r'\vec{\omega}, \vec{\omega}) dr' \right).$$

We now need to calculate the last term in the summation, which we call $L_r^*$:

$$L_r^*(\mathbf{x}_o, \vec{\omega}_o) = \lim_{C \to P_r(\mathbf{x}_i, \vec{\omega}_o)} [L^-(c)\mathscr{S}(c,b)](\mathbf{x}_o, \vec{\omega}).$$

We will not include the full derivation, but by subdividing the cylinder into infinitesimally thin slices, each with its own transmittance, and then taking the limit, to obtain a integral form for $L_r^*$:

$$L_r^*(\mathbf{x}_o, \vec{\omega}_o) = \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr', \tag{2.9}$$

where $\mathbf{x}' = \mathbf{x}_i + r'\vec{\omega}_o$ and where $L_*$ is the radiance per unit length

$$L_*(\mathbf{x}', \vec{\omega}_o) = \lim_{r \to 0} \frac{L_r^*(\mathbf{x}_o, \vec{\omega}_o)}{r}.$$

**Figure 2.6:** Sphere configuration to prove the radiance solution.

By putting together 2.7, 2.8 and 2.9 into 2.6, we obtain the following form:

$$L_r(\mathbf{x}_o, \vec{\omega}_o) = L_r^0(\mathbf{x}_o, \vec{\omega}_o) + L_r^*(\mathbf{x}_o, \vec{\omega}_o), \tag{2.10}$$

$$L_r(\mathbf{x}, \vec{\omega}_o) = L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) + \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr'.$$

We only need now to define $L_*$ in terms of $L$. For this derivation, let us still consider the configuration of Figure 2.6, where we have a point $\mathbf{x}'$ and vector $\vec{\omega}'$ on the surface of a sphere $d$ surrounding a point $\mathbf{x}$ on path $P_r(\mathbf{x}_i, r)$ with direction $\vec{\omega} = \vec{\omega}_s$. It is possible to derive an approximation of $S$ for small $s$ as

$$S(\mathbf{x}', \vec{\omega}', \mathbf{x}, \vec{\omega}) = \sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) \frac{s}{A_i'} + o(s).$$

$\mathbf{x}', \mathbf{x}$ are two points on the sphere, $s$ is the sphere radius, and $A_i'$ is the area of the sphere that would be lit by a light shined from direction $-\vec{\omega}'$. The term $\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) = \sigma_s(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega})$ is the non-normalized phase function. $o(s)$ is an error such as $\lim_{s \to 0} \frac{o(s)}{s} = 0$. We can now recalculate $L_s^*$ for the sphere configuration using the approximation:

$$\begin{aligned} L_s^*(\mathbf{x}, \vec{\omega}) &= \int_{A_i'} \int_{\Omega_i^-} L(\mathbf{x}', \vec{\omega}') [\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) \frac{s}{A_i'} + o(s)] (\vec{n}' \cdot \vec{\omega}') d\omega' dA_i' \\ &= \int_{4\pi} L(\mathbf{x}', \vec{\omega}') [\sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) s + A_i' o(s)] (\vec{n}' \cdot \vec{\omega}') d\omega' \\ &= s \int_{4\pi} L(\mathbf{x}', \vec{\omega}') \sigma_s(\mathbf{x}', \vec{\omega}', \vec{\omega}) d\omega' + o(s) \int_{4\pi} L(\mathbf{x}', \vec{\omega}') A_i' d\omega'. \end{aligned} \tag{2.11}$$

As the sphere gets smaller $s \to 0$, the second term disappears with the $o(s)$, so that we finally obtain $L_*$ as

$$L_*(\mathbf{x}, \vec{\omega}) = \lim_{s \to 0} \frac{L_s^*(\mathbf{x}, \vec{\omega})}{s} = \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\omega'. \tag{2.12}$$

Putting it all together, we obtain the so called *integral form* of the radiative transfer equation:

$$\begin{aligned} L_r(\mathbf{x}_o, \vec{\omega}_o) &= L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) + \\ &+ \int_0^r \sigma_s(\mathbf{x}', \vec{\omega}_o) \int_{4\pi} L(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega}_o) d\omega' T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr'. \end{aligned} \tag{2.13}$$

This form is described along a the extent of a path rather than a single point as in the integro differential form in Equation 2.1. We now derive back that form by deriving the above across $r$, which is the same as a directional derivative $\vec{\omega} \cdot \nabla$. Of the two terms on the right hand side of the equation above, the first becomes:

$$\begin{aligned} \frac{d}{dr} L^0(\mathbf{x}_i, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o) &= L^0(\mathbf{x}_i, \vec{\omega}_o) \frac{d}{dr} T_r(\mathbf{x}_i, \vec{\omega}_o) \\ &= L^0(\mathbf{x}_i, \vec{\omega}_o)(-\sigma_t(\mathbf{x}_o, \vec{\omega}_o) T_r(\mathbf{x}_i, \vec{\omega}_o)) = -\sigma_t(\mathbf{x}_o, \vec{\omega}_o) L_r^0(\mathbf{x}_o, \vec{\omega}_o), \end{aligned}$$

where the last step comes from Equation 2.8. As for the second term, we use the form from Equation 2.12:

$$\begin{aligned} \frac{d}{dr} L_r^*(\mathbf{x}_o, \vec{\omega}_o) &= \frac{d}{dr} \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' \\ &= \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) \frac{d}{dr} T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}, \vec{\omega}_o) \int_0^r L_*(\mathbf{x}', \vec{\omega}_o) T_{r-r'}(\mathbf{x}', \vec{\omega}_o) dr' + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}_o, \vec{\omega}_o) L_r^*(\mathbf{x}_o, \vec{\omega}_o) + L_*(\mathbf{x}_o, \vec{\omega}_o). \end{aligned}$$

By putting it all together:

$$\begin{aligned} \frac{d}{dr} L_r(\mathbf{x}_o, \vec{\omega}_o) &= -\sigma_t(\mathbf{x}_o, \vec{\omega}_o)[L_r^0(\mathbf{x}, \vec{\omega}_o) + L_r^*(\mathbf{x}_o, \vec{\omega}_o)] + L_*(\mathbf{x}_o, \vec{\omega}_o) \\ &= -\sigma_t(\mathbf{x}_o, \vec{\omega_o}) L_r(\mathbf{x}, \vec{\omega}_o) + L_*(\mathbf{x}_o, \vec{\omega}_o). \end{aligned}$$

By dropping the dependency on $r$ and the $o$ subscript, and introducing the definition for $L_*$ (Equation 2.12) and the directional derivative symbol, we obtain the integro-differential form of the radiative transfer equation

$$\vec{\omega} \cdot \nabla L(\mathbf{x}, \vec{\omega}) = -\sigma_t(\mathbf{x}, \vec{\omega}) L(\mathbf{x}, \vec{\omega}) + \sigma_s(\mathbf{x}, \vec{\omega}) \int_{4\pi} L(\mathbf{x}, \vec{\omega}') p(\mathbf{x}, \vec{\omega}', \vec{\omega}) d\vec{\omega}'.$$

We can then add an additional term to account for emission, obtaining the form in Equation 2.1.

### 2.2.4 Global solution to the BSSRDF

Now, we proved that the BSSRDF and the RTE are indeed connected through scattering theory. In this section, we will show an equivalent of the global solution 2.3 using the radiative transfer equation. To achieve this, we define the operator $\mathbf{S}^1$:

$$\mathbf{S}^1 = \int_0^r \sigma_s(\mathbf{x}', \vec{\omega}) \int_{4\pi} [\ \ ] p(\mathbf{x}', \vec{\omega}', \vec{\omega})(\vec{n} \cdot \vec{\omega}') d\omega' T_{r-r'}(\mathbf{x}', \vec{\omega}) dr'.$$

Note on how this operator is similar to the second term of Equation 2.13. Now, let us consider a bounded medium, with a starting radiance distribution on the surface $L_0(\mathbf{x}_o, \vec{\omega})$ at a boundary point $\mathbf{x}_o$, where $\vec{\omega}_o$ points towards the inside of the medium. We can extend this initial radiance to any point $\mathbf{x} = \mathbf{x}_o + r\vec{\omega}$ of the medium:

$$L^0(\mathbf{x}, \vec{\omega}) = L^0(\mathbf{x}_o, \vec{\omega}) T_r(\mathbf{x}_o, \vec{\omega}).$$

After defining $L^0$, we can define an $n$-ary radiance function $L^{n+1}$ recursively using operator $\mathbf{S}^1$:

$$L^{n+1} = L^n \mathbf{S}^1.$$

The $n$-ary radiance can easily be calculated using a continuous application of the $S$ operator. If we define a new operator $\mathbf{S}^{n+1} = \mathbf{S}^1 \mathbf{S}^n$, one can show in a straightforward way that

$$L^n = L^0 \mathbf{S}^n$$

for every scattering order $n$. We can bring this process to infinity by defining the two quantities:

$$L = \sum_{j=0}^{\infty} L^j, \qquad \mathbf{S} = \sum_{j=0}^{\infty} \mathbf{S}^j.$$

In this particular case, we define $\mathbf{S}^0 = \mathbf{I}$, where $\mathbf{I}$, the identity operator, is a functional for which $f\mathbf{I} = f$ for every choice of $f$. That leads to the formulation:

$$L = \sum_{j=0}^{\infty} L^j = \sum_{j=0}^{\infty} L^0 \mathbf{S}^j = L^0 \mathbf{S}. \tag{2.14}$$

It is simple to prove that this formulation satisfies the integral form of the radiative transfer equation:

$$
\begin{aligned}
L &= L^0\mathbf{S} \\
&= L^0\left(\mathbf{I} + \mathbf{S}^1 + \sum_{j=2}^{\infty}\mathbf{S}^j\right) \\
&= L^0\left(\mathbf{I} + \mathbf{S}^1 + \sum_{l=1}^{\infty}\mathbf{S}^{l+1}\right) \\
&= L^0\left(\mathbf{I} + \left(\mathbf{I} + \sum_{l=1}^{\infty}\mathbf{S}^{l}\right)\mathbf{S}^1\right) \\
&= L^0\left(\mathbf{I} + \mathbf{S}\mathbf{S}^1\right) \\
&= L^0 + (L^0\mathbf{S})\mathbf{S}^1 \\
L &= L^0 + L\mathbf{S}^1,
\end{aligned}
$$

where the last equation corresponds to the integral form of the radiative transfer equation in Equation 2.13.

Note that, since the solution of the radiance must be unique, the above calculated $L$ in functional form corresponds to the radiance $L$ calculated from Equation 2.3. This tells us that the calculating the outgoing radiance through the BSSRDF is the same as integrating the radiance across all the possible paths within the material. This fundamental connections justifies comparing BSSRDF renderings, an approximation of equation 2.3, with volume path tracing, in itself an approximation of Equation 2.14.

### 2.2.5 Rendering scattering media

Now, we want to use our formulas to derive a way to render scattering materials. The usual technique used in rendering to solve the rendering equation is Monte Carlo integration with importance sampling [KW08]. Let us assume we need to solve the integral

$$I = \int_a^b f(x)dx.$$

If we draw samples uniformly samples in $[a, b[$, we can define a $N$-th estimator for $I$:

$$\hat{I}_N = \frac{b-a}{N}\sum_{i=1}^{N} f(x_i).$$

From the law of large numbers, $\hat{I}_N \to I$ for $N \to \infty$. To improve the convergence rate, we use importance sampling. If we know a distribution $\text{pdf}(x)$ that is somewhat close in shape to $f(x)$ and from which it is easy to draw samples $x_i \in [a, b]$. We can rewrite the integral $I$ as :

$$I = \int_X \frac{f(x)}{\text{pdf}(x)} \text{pdf}(x) dx,$$

which we can evaluate again using Monte Carlo as:

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{\text{pdf}(x_i)}.$$

Where the $x_i$ follow the distribution $\text{pdf}(x)$. As before, $\hat{I}_N \to I$ for $N \to \infty$.

#### 2.2.5.1 Volume path tracing

Now let us apply importance sampling to obtain a first technique to render scattering materials, namely *volume path tracing* [Rus88]. We start by using the integral form of the radiative transfer equation 2.13. For the sake of this example, we assume a non emissive homogeneous medium, so that the coefficients do not depend on position and direction within the medium. We also assume index matched media, so we can simplify the interaction at the boundary.

The algorithm, traces a ray from the camera direction $-\vec{\omega}$. The ray hits the surface at some point $\mathbf{x}_o$. We need to evaluate $L_r(\mathbf{x}_o, \vec{\omega})$. We first evaluate $r$ as the distance to the other side of the medium, hitting a point $\mathbf{x}_t = \mathbf{x}_o - r\vec{\omega}$. Because of the assumption of homogenous medium, the transmittance is easily evaluated:

$$T_r(\mathbf{x}_t, \vec{\omega}) = \exp\left(-\int_0^r \sigma_t dr'\right) = e^{-\sigma_t r}.$$

We evaluate $L^0(\mathbf{x}_t, \vec{\omega})$ by continuing the path outside the medium in direction $-\vec{\omega}$. Now, we want to evaluate the second part of Equation 2.13, which is $L_r^*(\mathbf{x}_o, \vec{\omega})$ from Equation 2.10. For this, we use Monte Carlo integration with importance sampling. We first reparameterize the integral by imposing $s = r - r'$:

$$L_r^*(\mathbf{x}_o, \vec{\omega}) = \int_0^r \sigma_s \int_{4\pi} L(\mathbf{x}', \vec{\omega}') p(\mathbf{x}', \vec{\omega}', \vec{\omega}) d\omega' T_s(\mathbf{x}', -\vec{\omega}) ds.$$

Note that $\mathbf{x}' = \mathbf{x}_o - s\vec{\omega} = \mathbf{x}_t + r'\vec{\omega}$. Now we can write the Monte Carlo estimator sampling both integrals at the same time:

$$\hat{L}_r^*(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \frac{\sigma_s L(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}_q') p(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}_q', \vec{\omega}) \exp(-\sigma_t s_p)}{\text{pdf}(s_p)\text{pdf}(\vec{\omega}_q')}. \quad (2.15)$$

**Figure 2.7:** Volumetric path tracing, with various paths going through the medium from camera $\mathbf{x}_c$ to light $\mathbf{x}_l$. Path (a) represents direct transmission to calculate $L^0(\mathbf{x}_t, \vec{\omega})$. Path (b) represents a path killed through absorption, while path (c) successfully exits the volume and connects to the light.

For now, we assume that $s_p < r$, i.e. we are always sampling a point within the medium. Now, we need to choose the pdfs. We can sample $s_p$ according to the formula:

$$s_p = \frac{-\ln(1-\xi)}{\sigma_t},$$

for $\xi \in [0, 1)$, that corresponds to a $\text{pdf}(s_p) = \sigma_t \exp(-s_p \sigma_t)$. Once we have chosen $s_p$, we can move onto sampling $\vec{\omega}'_q$. For standard phase functions such as Henyey-Greenstein [HG40], it is possible to analytically find a way to importance sample a vector $\vec{\omega}'_q$. So, if the vector is carefully chosen, we have $\text{pdf}(\vec{\omega}'_p) = p(\mathbf{x}_o - s_p\vec{\omega}, \vec{\omega}'_q, \vec{\omega})$.

Plugging it in 2.15 we get the form

$$\hat{L}^*_r(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \alpha L(\mathbf{x}_o - s_p, \vec{\omega}, \vec{\omega}'_q),$$

where $\alpha = \sigma_s/\sigma_t = \sigma_s/(\sigma_s + \sigma_a)$ is called the *single scattering albedo*. Note that $0 < \alpha < 1$. Finally, we can introduce an additional importance sampling. We introduce a scattering probability $\text{pdf}_s = \alpha$. We decide then if we want to either scatter of absorb the photon, using russian roulette [AK90]. If we decide to absorb the photon, we simply terminate the path. With this procedure in

place , we get our final formulation

$$\hat{L}_r^*(\mathbf{x}_o, \vec{\omega}) = \sum_{p=1}^{N} \sum_{q=1}^{M} \alpha \frac{L(\mathbf{x}_o - s_p, \vec{\omega}'_q)}{\mathrm{pdf}_s} = \sum_{p=1}^{N} \sum_{q=1}^{M} L(\mathbf{x}_o - s_p, \vec{\omega}'_q).$$

From this result, we see that we just need to recursively evaluate radiance to get the final result. At the next recursive step, we will evaluate a new $s_p$ and $\vec{\omega}'_q$, continue the process with probability $\alpha$ and so on. This creates a random walk within the medium, that does not terminate for $s_p < r$. If $s_p > r$, it means that we are exiting the medium. In this case, we continue path tracing from the exit point in the last evaluated direction $\vec{\omega}'_q$.

#### 2.2.5.2 The extended rendering equation

Instead of using the various forms of the radiative transfer equation, we can use the form of the rendering equation in Equation 2.3. Jensen and Buhler [JB02] proposed one such technique, namely hierarchical integration. In general, we want to be able to sample points and directions on the surface of the scattering medium, so that we can solve the rendering equation via Monte Carlo integration using

$$\hat{L}(\mathbf{x}_o, \vec{\omega}_o) \approx \sum_{p=1}^{N} \sum_{q=1}^{M} \frac{S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q})(\vec{n}_{i,p} \cdot \vec{\omega}_{i,q})}{\mathrm{pdf}(\vec{\omega}_{i,q})\mathrm{pdf}(\mathbf{x}_{i,p})}.$$

We refer to our note in Appendix IX for a rendering technique that uses this estimator for the extended rendering equation.

### 2.2.6 Analytical BSSRDF Models

So far, we have defined the BSSRDF as a proportionality constant between radiometric quantities. Given some assumptions about light propagation in materials, it is possible to derive analytical formulas for a BSSRDF for practical use in rendering. In the case of analytical models, Jensen et al. [JMLH01] first provided an analyitical BSSRDF function based on the dipole solution in the work of Farrell et al. [FPW92] (the standard dipole). In this section, we outline the derivation of three analytical models, the standard dipole [JMLH01], the better dipole [d'E12] and the directional dipole [FHK14]. Our note in Appendix VIII provides additional derivation details for the interested reader.

Generically, the light exiting from a scattering medium at a point $\mathbf{x}_o$ at a direction $\vec{\omega}_o$ can be thought as coming from three contributions: direct transmission,

single and multiple scattering. We define $\vec{\omega}_{21}$ as the refraction of $\vec{\omega}_o$ into the medium oriented towards the surface. Direct transmission is the contribution of the light coming from direction $-\vec{\omega}_{21}$ that does not scatter along the path (see Equation 2.8). The single scattering term comes from light that enters at any point of the medium, scatters once, then aligns with $\vec{\omega}_{21}$ to refract out in direction $\vec{\omega}_o$. The multiple scattering term is similar, but it includes light that has scattered more than once before exiting the medium.

Single and multiple scattering are usually separated due to their different characteristics. Single scattering is highly peaked and directional, while multiple scattering can be handled as a stochastic process. The three analytical models we describe model the scattering of light as a diffusion process [Sta95]. As such, the standard and better dipole model multiple scattering only, while the directional dipole models also part of the single scattering contribution.

As before, we define operators to simplify notation:

$$\mathbf{G} = \int_{4\pi} [\ \ ] d\omega, \quad \vec{\mathbf{G}} = \int_{4\pi} [\ \ ] \vec{\omega} d\omega.$$

In radiative transport, we use spherical harmonics to simplify radiance [EVNT78]:

$$L(\mathbf{x}, \vec{\omega}) = \frac{1}{4\pi}\phi(\mathbf{x}) + \frac{3}{4\pi}\vec{E}(\mathbf{x}),$$

where $\phi = L\mathbf{G}$ and $\vec{E} = L\vec{\mathbf{G}}$. Once we combine the radiative transfer equation and the diffusion approximation, we get the form:

$$(D\nabla^2 - \sigma_a)\phi(\mathbf{x}) = -Q_0(\mathbf{x}) + 3D\nabla \cdot \vec{Q}_1(\mathbf{x}, \vec{\omega}), \tag{2.16}$$

where $Q_0 = q\mathbf{G}$ and $\vec{Q} = q\vec{\mathbf{G}}$ are the integrals of the emission term in the radiative transfer equation $q(\mathbf{x}, \vec{\omega})$ (see Equation 2.1). The choice of $q$, or *source term* leads to different approximations. Standard and better dipole use a point source term, while the directional dipole uses a ray source. Equation 2.16 is a particular case of screened Poisson equation, that can be explicitly integrated [FWP03]. For a point source term, we obtain:

$$\phi(\mathbf{x}) = \frac{\alpha' \Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r}. \tag{2.17}$$

And for the ray source term, we get:

$$\phi(\mathbf{x}) = \frac{\alpha' \Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \left(1 + 3D \frac{1 + \sigma_{tr} r}{r} \cos\theta\right). \tag{2.18}$$

Now that we have a formulation for the fluence, we need to connect it to the BSSRDF. First, we consider only the part of the BSSRDF depending on the

diffusion approximation $S_d$. From the definition of BSSRDF in Equation 2.2 and assuming a pure Fresnel boundary we get to this formulation in terms of the diffusive radiant exitance $M_d$

$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) = \frac{1}{4\pi T_{12} C_\phi(1/\eta)} \frac{dM_d(\mathbf{x}_o)}{d\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}.$$

Through the diffusion approximation, we can approximate the diffusive radiant exitance $M_d$ as

$$M_d(\mathbf{x}_o) = C_\phi(\eta)\phi(\mathbf{x}_o) - C_E(\eta) D \nabla\phi(\mathbf{x}_o) \cdot \vec{n}_o,$$

where $T_{12}$ is the incoming Fresnel coefficient, $C_\phi$ amnd $C_E$ are the first two orders of Fresnel integrals, which can be approximated by analytical formulas [dI11]. The value $\eta$ is the index of refraction of the medium. By plugging in our solutions for the fluence in Equations 2.17 and 2.18, we get the analytical formula for a BSSRDF if we disregard the fact that light does not scatter outside the medium. In the case of a point source term, we obtain

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{1}{4\pi C_\phi(1/\eta)} \frac{\alpha'}{4\pi^2} \frac{e^{-\sigma_{tr} r}}{r^3} \left[ C_\phi(\eta) \frac{r^2}{D} + C_E(\eta)(1 + \sigma_{tr} r)\mathbf{x} \cdot \vec{n}_o \right],$$

where $\mathbf{x} = \mathbf{x}_o - \mathbf{x}_i$ and $r = \|x\|$. This is the term used for the better dipole [d'E12]. We can further simplify if we take the limit of the Fresnel integrals, obtaining $C_\phi(\eta) = 0$, $C_E(\eta) = 1$ and $C_\phi(1/\eta) = 1/4$. This leads to

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{\alpha'}{4\pi^2} \frac{e^{-\sigma_{tr} r}}{r^3} (1 + \sigma_{\mathrm{tr}} r)\mathbf{x} \cdot \vec{n}_o,$$

which is used for standard dipole [JMLH01]. Finally, if we use the ray source solution, we obtain

$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) = \frac{1}{4\pi C_\phi(1/\eta)} \frac{1}{4\pi^2} \frac{e^{-\sigma_{\mathrm{tr}} r}}{r^3} \Bigg[ C_\phi(\eta)(\frac{r^2}{D} + 3(1 + \sigma_{\mathrm{tr}} r)\mathbf{x} \cdot \vec{\omega}_{12})$$
$$- C_E(\eta) \left[ 3D(1 + \sigma_{\mathrm{tr}} r)\vec{\omega}_{12} \cdot \vec{n}_o - \left( (1 + \sigma_{\mathrm{tr}} r) + 3D \frac{3(1 + \sigma_{\mathrm{tr}} r) + (\sigma_{\mathrm{tr}} r)^2}{r^2} \mathbf{x} \cdot \vec{\omega}_{12} \right) \mathbf{x} \cdot \vec{n}_o \right] \Bigg],$$

which is used for he directional dipole [FHK14]. As we mentioned, all these solutions are suitable only when light is allowed to scatter within an infinite medium with coefficients $\sigma_s$, $\sigma_a$ and $g$. To get a full BSSRDF, we need to introduce a boundary. We assume a refractive boundary with relative index of refraction $\eta$ as the ratio of the index of refraction of the incoming medium to the index of the refraction of the scattering medium. For the vast majority of analytical BSSRDFs, we derive a half-space solution based on a plane surface, then correct the formulas for non planar geometry.

Standard and better dipole Directional dipole

**Figure 2.8:** Placement of virtual (red) and real (blue) sources for the dipoles described in the text.

To achieve a solution for a point $\mathbf{x}_o$ on a semi infinite plane with normal $\vec{n}_o$, we let the fluence $\phi$ vanish at the boundary. This leads for the following boundary condition for the diffusion approximation:

$$\phi(\mathbf{x}_o) - 2Ad_e(\vec{n}_o \cdot \nabla)\phi(\mathbf{x}_o) = 0$$

Where $d_e$ is called extrapolation distance, $A(\eta)$ is a term that accounts for mismatched index of refraction at the boundaries. The idea here is that the diffusion approximation matches better the exact solution if we let the fluence vanish not at the boundary, but at an *extrapolated boundary* at distance $2Ad_e$ along the normal direction. This boundary condition can be satisfied by introducing two light sources, a positive *real* source and a negative *virtual* source, that together form a *dipole configuration*. Standard and better dipole use two point lights in the configuration to the left of Figure 2.8, with the real source (blue) below and the virtual source (red) above the surface. The directional dipole uses a real ray source on $\mathbf{x}_i$, and a virtual ray source above the surface (see right of Figure 2.8). The solutions for infinite media reported above are then evaluated for the real and the virtual source, and the results are then combined to obtain the final BSSRDF value for rendering.

## 2.3 Non-scattering media

In this section, we will introduce some simplifications to the scattering theory introduced in Section 2.2. We introduce the description of the BRDF, a reflectance function far more widespread in real time application than the BSSRDF.

**Figure 2.9:** Configuration in which we define the BRDF for a generic surface.

### 2.3.1 The BRDF

The methods that include scattering media we described in Section 2.2.5 are usually quite complex to render, requiring a high number of samples to converge. For many materials such as plastic or metal, the scattering process is quite restricted around the point of emergence $\mathbf{x}_o$. For these particular materials, we can introduce some simplifications to obtain more tractable functions. We assume that the BSSRDF is limited across a small area around the emergence point $\mathbf{x}_o$, and zero everywhere else. In this configuration, we can assume that the radiance is constant across the plane ($L_i(\mathbf{x}_i, \vec{\omega}_i) \approx L_i(\vec{\omega}_i)$). We also need to assume that the material is locally isotropic, i.e. its properties do not change across the surface. In this case, we can approximate the outgoing radiance as:

$$dL(\mathbf{x}_o, \vec{\omega}_o) \approx \int_A dL(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) d\Phi_i(\mathbf{x}_i, \vec{\omega}_i).$$

By the definitions of flux and irradiance and the assumption of constant radiance outlined above, we obtain

$$\begin{aligned} dL(\mathbf{x}_o, \vec{\omega}_o) &= \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) dA_i d\omega_i \\ &\approx L_i(\vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) dA_i \\ &= dE_i(\vec{\omega}_i) \int_A S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) dA_i. \end{aligned}$$

The last integral eliminates the dependency on $\mathbf{x}_i$, as it is a function purely dependent on the point of emergence $\mathbf{x}_o$ the two angular vectors $\vec{\omega}_i$ and $\vec{\omega}_o$. This proportionality constant between incoming irradiance and outgoing radiance $f_r$

$$dL(\mathbf{x}_o, \vec{\omega}_o) = f_r(\mathbf{x}_o, \vec{\omega}_i, \vec{\omega}_o) dE_i(\mathbf{x}_o, \vec{\omega}_i), \tag{2.19}$$

is called the *bidirectional reflectance distribution function* (BRDF) [NRH+77]. The BRDF is measured in [$\mathrm{sr}^{-1}$]. As before, we can obtain the reflected radiance

Polar representation
$f_r(\theta_i, \phi_i, \theta_o, \phi_o)$

Rusinkiewicz representation
$f_r(\theta_h, \phi_h, \theta_d, \phi_d)$

**Figure 2.10:** Different representations for BRDFs. To the left, standard representation using polar coordinates. To the right, the representation from [Rus98].

from the definition above:

$$L(\mathbf{x}, \vec{\omega}) = \int_{\Omega} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) L_i(\mathbf{x}, \vec{\omega}_i)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \quad [\mathrm{W} \cdot \mathrm{m}^{-2} \cdot \mathrm{sr}^{-1}]. \tag{2.20}$$

This is called the reflected radiance equation. As before, we can add the emitted radiance $L_e$ to obtain the traditional form of the rendering equation [Kaj86]. We can easily extend the two properties of the BSSRDF in Equations 2.4 and 2.5. Conservation of energy imposes

$$0 \leq \int_{\Omega} f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o)(\vec{n} \cdot \vec{\omega}_i) d\omega_i \leq 1.$$

On the other hand, Helmholtz reciprocity gives

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\mathbf{x}, \vec{\omega}_o, \vec{\omega}_i).$$

From the reflectance equation above, we can see why BRDFs are usually employed in real time applications, since they require one less integral to solve (compare Equations 2.20 and 2.3). Moreover, in applications such as games we may have a finite number of point and directional lights, which simplifies the integral in Equation 2.20 to a sum over all the lights.

### 2.3.2 Empirical BRDFs

Given the low dimensionality of the BRDF, it is possible to actually measure and tabulate it, so that it can be used in future renderings. If we assume a non-spatially varying BRDF (dropping the dependency from $\mathbf{x}_o$ in Equation 2.19)

and expressing each vector $\vec{\omega}$ in terms of its polar angles $\theta, \phi$, we obtain a four dimensional function:

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\theta_i, \phi_i, \theta_o, \phi_o).$$

Moreover, from now on we assume the BRDF to be *isotropic*, so that it depends only on the relative angle $\phi_{\text{diff}} = \phi_o - \phi_i$:

$$f_r^{\text{iso}}(\theta_i, \phi_i, \theta_o, \phi_o) = f_r^{\text{iso}}(\theta_i, \theta_o, \phi_{\text{diff}}).$$

Anisotropic materials (e.g. brushed metal) are rarer in nature and are usually created using some sort of manufacturing process. This naïve representation, presented in Figure 2.10, can be used to store BRDFs in a compact way, though with some issues. Generally, BRDFs tend to have a highly peaked region around the half vector:

$$\vec{\omega}_h = \frac{\vec{\omega}_i + \vec{\omega}_o}{\|\vec{\omega}_i + \vec{\omega}_o\|},$$

where there is usually a strong peak due to reflection, or *specular highlight*. If we use the naïve parametrization, we would encounter some discretization issues around $\vec{\omega}_h$. To avoid this loss of precision, it is important to represent the BRDF more accurately around the half vector. So for measured datasets such as the MERL dataset [MPBM03], the Rusinkiewicz half-difference parameterization is used (see Figure 2.10). This representation measures angles relative to the half vector instead of the surface normal:

$$f_r(\mathbf{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\theta_h, \phi_h, \theta_d, \phi_d),$$

where $\theta_h, \phi_h$ are the spherical coordinates of $\vec{\omega}_h$, and where $\theta_d, \phi_d$ are the spherical coordinates of $\vec{\omega}_i$ rotated so that $\vec{\omega}_h$ matches the reference frame of $\vec{n}$. In this case, isotropic materials can be described by dropping the dependency on $\phi_d$, i.e. the overall rotation of the system around the normal.

After we have chosen our representation, we can pack our isotropic BRDF in a 3D data structure. Note that similar packing representations exist also for BSSRDFs [DLR+09], but the high dimensionality of the BSSRDF (5 up to 14 dimensions) renders these representations somewhat less practical.

## 2.4 Rendering techniques

In the previous sections, we introduced theory and algorithms for rendering physically based materials, without going into details on how the different algorithms are implemented. To conclude our background section, we introduce two of the main paradigms used to render physically based materials on GPUs, namely rasterization and ray tracing.

**Figure 2.11:** The rasterization pipeline. Following the arrow: bottom left, vertices in local coordinates are fetched from memory; top left, vertices are disposed by the vertex shader, then the triangle is assembled; top right, fragments (gray) are generated; bottom right, framents are shaded; bottom, fragments are combined by a depth test to form the final image.

### 2.4.1 Rasterization

The first of the two techniques is rasterization. In this technique, some rendering primitives, triangles, are scanned one by one and then drawn on the screen. One example of algorithms used to transform triangles into pixel-size elements (or fragments) is scanline rendering [WREE67], though it depends on the GPU vendor. By its nature, rasterization is highly parallelizable, since every primitive can be drawn in parallel. On modern graphics cards, the rasterization process is performed in hardware as part of the *graphics pipeline*, a highly fine tuned sequence of steps, both hardware and software, to render triangles efficiently. A simplified version of the pipeline is illustrated in Figure 2.11. The pipeline is

composed of programmable parts (called *shaders*) and hardware parts, that are only controllable through state flags. The core rasterization process, transforming primitives into fragments, is executed in parallel by a specified hardware unit.

Let us describe the life of a single triangle through the pipeline. Once the CPU issues the command to draw some triangles, a *draw call* (Figure 2.11, a), the pipeline starts on the GPU. For each of the three vertices composing a triangle, we independently execute a programmable part, called the *vertex shader*, that allows operation such as model and perspective transformation (Figure 2.11, b). Once the vertices have been processed, the primitive is assembled and then processed by the hardware rasterizer, generating a certain number of fragments (Figure 2.11, c). For each fragment, we execute another programmable shader, the *fragment shader*. This stage needs to output the final color of the fragment, so light interaction is usually added at this stage (Figure 2.11, d). Multiple attributes can be passed in between vertex and fragment shader, which will be interpolated using the triangle's barycentric coordinates. This allows interpolation of attributes, such as normal and texture coordinates, across the triangle, leading to a more accurate appearance.

Once the fragments are generated, they need to be stored on the image plane. Note that multiple fragments can land on the same pixel, and that the order of landing of the fragments is not defined, since the full pipeline is entirely asynchronous. So, modern GPUs use another hardware unit, the Render OutPut unit (ROP, Figure 2.11, e) to determine how to store the fragments in the final image. This unit usually can perform depth test (via a Z-buffer) or blend the fragments, allowing us to obtain a consistent result across invocations.

This is only a simplified view of the full graphics pipeline, that, allows tessellation (domain and hull shaders), geometry manipulation (geometry shaders) or writing back into a vertex stream (transform feedback). There are also shaders that are completely detached from the graphics pipeline (compute shaders), allowing to perform tasks in parallel on the GPU that are not directly related to a graphics task.

Rasterization is extensively used in game development and real time applications, given its high speed and predictability. However, it is not particularly well suited for propagating light across a scene, making it difficult to achieve complex optical effects, why these often require ad-hoc solutions.

**Figure 2.12:** Ray tracing four triangles in two dimensions. The thicker line corresponds to the ray (a). At each step, we consider a different bounding box (thick red border). We illustrate different cases progressively: hitting bounding box without hitting primitive (b), hitting primitive (c), hitting bounding box but not hitting primitive because triangle is behind tmax (d), updating tmax to closest primitive hit (d). The last figure (f) shows all four any hit invocation (black dots) plus the closest hit (black dot with tmax).

### 2.4.2 Ray tracing

We start this section by describing *ray casting*, that can be considered the dual technique to rasterization. In rasterization, we match each triangle to its final position on the screen. In ray casting, we do the opposite: for each pixel on the screen, we find the corresponding triangles that land within that pixel. This equates to *shooting* a ray through the pixel, and intersecting it though the scene geometry. Once the hit point is found, attributes can be generated and the point shaded as in a fragment shader.

We note that, once the data structure is established, we can trace rays from any

point to another in the scene. So ray tracing describes this process of querying a data structure to generate intersection points. Depending on the algorithm used, ray tracing can be used as a base of multiple rendering algorithms, such as the volume path tracing we illustrated in Section 2.2.5. One of the first ray tracing algorithms was proposed by Whitted [Whi80].

For the purpose of this thesis, we focus on GPU ray tracers. Modern GPU ray tracers build a tree data structure to efficiently lookup intersections. The most commonly used data structure is a *bounding volume hierarchy*, or BVH. In a BVH, each of the primitives (usually triangles, but custom primitives such as spheres, cubes or quadrilaterals are possible) is enclosed in a bounding box. The various bounding boxes are then arranged in a tree data structure for fast lookup. Once a ray is traced, it is first inexpensively tested against the bounding boxes. For the bounding boxes that are hit, an expensive intersection test is performed to check for a hit. Depending on the shading algorithm, a callback can be issued at every encountered intersection (*any hit*), or at the intersection closest to the camera (*closest hit*). The process to trace a single ray in a two dimensional example is illustrated in Figure 2.12.

Many challenges need to be solved to obtain an efficient GPU ray tracer. The first challenge is recursion. In the case of an algorithm such as path tracing, a probabilistic recursive ray tracing call is performed at the closest hit. So, an efficient ray tracer needs to keep track of the state at each intersection, generically though a recursion stack (OptiX [PBD+10] is such an example, where the stack is built automatically at the instruction level). The second challenge is on how to manage the data structure if the geometry of the material changes, through deformation or animation. Simple rigid tranformations can be handled relatively inexpensively by the BVH, tough more complicated operations need to be done if changes are performed at the single primitive level. Modern ray tracing techniques such as the TRBVH [KA13] can rebuild part of the BVH on the fly without excessive memory consumption.

Given its state as a "natural" solution to light transport and its ability to model complex optical effects, ray tracing is commonly employed in the movie industry. Moreover, ray tracing scales better to scenes with a huge amount of triangles, such as in animated movies or virtual effects scenes. On the other hand, ray tracing is limited in the game developers community, due to smaller scenes and increased memory and lookup costs. Moreover, algorithms such as path tracing tend to give a noisier result, that is not acceptable in game applications.

Chapter 3

# Related work

**FIXME:** more connections to theory

**FIXME:** connections to following chapter more clear

**FIXME:** further explanations

**FIXME:** include references

In this section, we present some related work on the overall topic of the thesis, namely physically based techniques brought into the interactive domain. We will make use of the background theory we presented in Chapter 2, to better discuss our contributions in Chapter 4. This section is not meant to be a complete survey on such techniques, but we will point to the most important contributions, referring to a number of individual papers and surveys for a more complete overview. We also refer to the related work of the individual contributions, reported in the Appendix, for a more details in the context of the individual papers. As we mention in Chapter 1, interactive rendering requires a compromise between accuracy and speed. There are various examples in literature on the compromises that are needed to hit this trade-off. Since the rendering time constraints are usually set, compromises are usually done in physical accuracy. We will start by discussing some of the assumptions that are usually employed in interactive rendering environments.

## 3.1 Approximate physically based rendering

One of the natural approaches into achieving interactive photorealistic rendering is making some assumptions about our physical simulation. In principle, we can describe any light interaction by running a brute-force simulation of all the photons interacting with the particles of the material, then measuring the number of photons arriving at a set light sensor. Of course, running this sort of computation, even for extremely simple scenes, in reasonable times is not possible REF?. In general, Monte Carlo path tracing REF with extension to scattering media [Rus88] is considered a ground truth technique to generate reference images. Path tracing is an unbiased technique (i.e. converges always to the correct solution), but it gives extremely noisy results, especially for particular configurations of light and camera positions. To solve this problem, the offline rendering community proposed various improvements to reduce variance in path tracing, such as multiple importance sampling REF, Metropolis sampling REF, Bidirectional path tracing, MCMC methods, etc. The book by Pharr et al. REF gives a good overview of different path tracing techniques. We described one of these importance sampling techniques in Section TODO, where we derived a formulation for interactive volume path tracing.

So when we need interactivity, different approximations can be used. Once we know the underlying physical process, different choices can be made on a per material basis. For example, in the case of most metals, the scattering of light is fairly limited around the point of incidence, so the BRDF approximation described in section TODO can be used. In the case of scattering materials, we can use the analytical BSSRDFs described in Section TODO instead of a fully path traced simulation. Depending on the directionality of the scattering effects, different models can be used, at the price of increased rendering times. In applications such as games or virtual reality, often it is not even possible to use BSSRDF models, using an approximate BRDF instead. This gives acceptable results, but since scattering effects are important, this result in a "waxy" effect. This is particular noticeable in the case of skin.

Another approach, instead of simplifying the physical model, real measured data can be used instead. This is the approach for example used by the discretized BRDF described in section TODO. The drawback of measured materials is that they often require large storage spaces. This is often a problem, since most rendering pipelines are memory-bound. Discretized models often have discretization issues, and require particular care in deciding a proper storage space, such as the Runsinkewitz parameterization for measured BRDF. Finally, measured data are limited to the setup used to generate them. In the simulated discretized BSSRDF by REF, many assumptions, including planarity of the configuration, are required to reduce the 14 dimensions of the BSSRDF to a

more tractable 5.

So far we have discussed material simplification. In general, the light hierarchy of a scene can also be object of simplification. Generically, a light is an object like any other in the scene, but that also emits light in the visible spectrum. This can be handled by adding the $L_e$ term in equations TODO and TODO, so that lights can be easily added to any path tracing framework. However, since the position of the lights is generally known in advance, multiple techniques can be used to importance sample the lights. Further simplifications are possible: in games and real time applications, lights often do not have an area extent. This is the case of directional, point and spot lights REF. This allows to represent lights as delta functions and replace the integral over $\Omega^+$ in Equations TODO and TODO with a sum over all the lights in the scene. Recently, real-time polygonal lights have been introduced by REF.

One final simplification we discuss in interactive applications is geometry simplification. Tough GPUs are able to push more and more triangles out, sometimes the scene needs to be simplified in order to maintain acceptable frame rate. This is particularly true of massive scenes, where techniques as occlusion culling and LODs need to be employed to achieve interactive framerates.

## 3.2 Interactive rendering techniques

Once we have made our choice of physical model, as we discussed in the previous section, some other choices need to be done on the implementation side. These implementation choices account for the fact that our algorithm runs on a discrete system with finite memory and processing power. In this thesis, we focus on GPU techniques, that thus usually exploit the massive parallelism offered by the streamed multiprocessing units on GPUs. In literature, we identified three common approaches the various techniques employ, namely caching, precomputation and filtering. Each of these approaches leverages some approximations or introduces some limitations in order to work. In caching, we use some intermediate data structures to accelerate the per-frame rendering time. Caching also implies using data structure to efficiently reuse data, either spatially (to exploit cache coherence) or temporally, to amortize computation across frames. Precomputation, the second approach, allows to move some of the computation before the program actually executes, given some assumptions about the materials or geometry. Finally, filtering, the final approach, reconstructs missing information based on a sparse sampling of a target function. Note that techniques often fall into multiple approaches: each technique is usually a combination of caching, precomputation and filtering. Bearing this in mind, we will

now proceed to present relevant theory for each approach.

### 3.2.1 Caching

In many cases in rendering, we need knowledge of both the local geometry around a geometric point (e.g. to estimate the occlusion of a point) in the scene and the overall global geometry (e.g. for global lighting effects). A number of techniques provide efficient data structures to efficiently retrieve both the local and global geometry of a point. The most simple and widely used of these techniques is deferred shading REF, that rasterizes geometry, depth and positions into a highly optimized screen space data structure. This structure allows sampling of local geometry, and it can be used to implement various screen space techniques REF. For global effects, a common approach is to create a traversal structure to efficiently implement ray tracing, such as the bounding volume hierarchies described in section TODO. Data structures for interactive ray tracing is a field of itself, so we point out to a relevant survey by REF for further reading REF. Once a traversal structure for ray tracing is in place, various classical ray tracing algorithms can be efficiently implemented, such as recursive ray tracing, path tracing, or volumetric path tracing. Other data structures can be used, such as octrees REF, or trace rays directly in the G-buffer, if available REF.

Other techniques involve some form of light caching. One of the simplest light caching techniques, the radiance cache REF, can be done in screen space. This techniques stores the radiance from a frame and reprojects it to the next, then filling holes created through visibility mismatches. Screen space techniques can be used to efficiently render also scattering media, by solving the extended rendering equation REF by sampling geometry in a local neighborhood. Other light caching techniques propagate illumination in the scene, store it, then render the scene again from the camera. This allows to achieve global effects. Photon mapping REF is a seminal paper in this regard. In this technique, photons are cast from light sources, bounced around the scene though ray tracing, then arranged into photon maps. Density estimation is then use to derive the final illumination from the maps. The overall efficiency can be enhanced by building a data structure for fast gathering of nearby photons. Instant radiosity techniques REF use a similar approach, but use gathering instead of density estimation in order to estimate the final radiance value at the exit point. These techniques are often named VPL (virtual point light) techniques, since each photon stored in the scene is treated as a small point light. A lot of literature deals with eliminating VPL singularities REF, adding visibility REF or generalizing them into virtual area lights REF.

Mixed techniques that combine geometry and light approximation are also possible. One example is radiosity, where we approximate the scene as a series of geometric patches, then precompute the light transport in between patches. After this, the overall light transport problem can be described as solving a linear system. Another approach that employs both geometry simplification and light caching is point based global illumination, where the scene is represented as a series of surface elements (surfels). In this technique, we first build a hierarchy of surfels. Surfels are then shaded. Finally, for each pixel in the final rendering, the relevant surfels falling within that pixel are rendered, obtaining the final result.

Other techniques use efficient data structures to do a light transport simulation. These techniques are particular suitable to render participating media, but they can also be used to render traditional diffuse and glossy illumination. In the case of participating media, these techniques often use finite elements to solve the radiative transfer equation on a discretized grid. Various improvements can be applied, such as using a adaptive octree data structure to propagate lighting. This particular techniques are used to obtain real time results, such as in the case of light propagation volumes REF or voxel cone tracing REF.

### 3.2.2 Pre-computation

In this section, we deal with the aspect of precomputation. By introducing some limitation in our scene, we can precompute some data for efficient rendering. Limitations include static or mostly static geometry, static light, fixed materials or fixed cameras.

Relighting techniques REF are a first example in which we require camera and geometry to be fixed. In this case, we precompute and change lighting in the scene. These techniques are suitable in a preview environment, or to perform artistic tweaks in the final rendering. REF is an example of this technique used in movie rendering pipelines.

Precomputed radiance transfer techniques REF assume static geometry only (light and view can freely change). This family of techniques involve precomputing the radiance transfer at the surface for infinitely distant lights. Basis function are used to approximate lighting transfer at the surface, allowing slow preprocessing times but a fast evaluation via a dot product. The quality of the rendering depends on the number of basis functions used, though the memory and performance requirements then increase as well. Spherical harmonics are the most commonly used basis functions REF, but more complex ones are possible REF. Precomputed radiance transfer gives good result representing low

frequency lighting changes across the scene. Still in the case of techniques that require static geometry, radiosity discretizes the scene as a series of patches, then calculates the transfer function between each patch. This allows fast illumination with changing lights and viewpoints, at the price of a quadratic memory increase depending on the number of patches.

if we assume static light and geometry, the whole incoming illumination at a point can be cached, then used to re-light objects moving through the scene. Moreover, static illumination can be stored as an additional texture map (baking), with direct illumination added on top.

### 3.2.3 Filtering

The last branch of techniques we discuss is filtering. Filtering approaches regard reconstructing the final appearance based on a limited set of samples. As in previous sections, we start by describing screen space techniques. A big area of research includes anti-aliasing techniques, that involve improving appearance of undersampled features in the scene. Temporal antialiasing techniques involve recycling color information from the previous frame via motion vectors, and combining with the current color distribution, to achieve antialiasing across time. More recently, new techniques have been developed to filter noisy one sample Monte Carlo simulations to achieve a smooth temporally stable result REF. Work in this area include advanced edge-aware bilateral filtering REF, temporal variance averaging REF, and machine-learning based filtering REF. Screen space techniques can be effectively used also in the case of scattering media, approximating the scattering process as a series of Gaussian filters. This effectively approximates the diffusion process using the standard dipole describe in Section TODO REF, leading to plausible realistic results.

Moving into more advanced structures, irradiance caching techniques store the illumination at a limited set of points in the scene, or pixels, then interpolating illumination across the points to achieve overall results. The technique can be further enhanced with various heuristics in order to avoid light leaking though objects. Another technique in global space is probe sampling, where precomputed radiance transfer is stored at fixed points in the scene, then the illumination is interpolated across the probes to allow lighting of both static and dynamic geometries. Interpolation can be done in many ways, including Voronoi 3D grids to allow arbitrary placements of probes.

As we have seen in this section, the space of interactive techniques that try to deliver a physically accurate result is huge. Most of these techniques rely on simplifications, assumptions and discretizations that allow fast rendering, often

sacrificing physical accuracy in the process. In the next chapter, we will show how we improved upon current techniques to achieve a more physically accurate result.

Chapter 4

# Contributions

**FIXME:** All subsections need to be expanded and further clarified.

**FIXME:** Time of text: past or present?

In this section we will tie together the different publications made during the course of the PhD studies. We will focus on the general message of the publications in relation to the goal of the PhD. We remand to the full text of the individual publications in Appendices I-V for the full technical details.

## 4.1 Outline and motivation

**FIXME:** (Figure: Fields in which we need instant feedback on appearance: 3d printing, artist feedback, quality control, meat)

As we discussed in Chapters 2 and 3, many efforts have been put forward by the community towards interactive physically based rendering. We barely scratched the surface on reporting some of these techniques in Chapter 3. These techniques are becoming widely popular in recent years, mainly due to increased GPU computational power.

We argue that there is a need in the industry for photorealistic accurate interactive rendering. In many fields, people need immediate feedback on on the aspect of the final product. Some examples include visual inspection of produced parts, preview of 3D printed objects, artistic iterations for movie scenes, and prediction of the outcomes of an industrial process. In all these applications, different requirements are set on how fast the rendering should be. For example, in quality inspection of a produced part, minutes can be acceptable. For artistic feedback in movie production, this may go down to a few seconds, depending on the number of necessary iterations. In the real-time rendering of a game or a virtual reality application, this can go as low as an handful of milliseconds. Our contributions over the course of the PhD studies contribute in various parts of this wide spectrum of needs.

Regarding our contributions, Contribution I provides a first example on why we need both fast and accurate rendering, in the form predicting the appearance of cloudy apple juice after production. In Contribution II we discuss on the definition of photorealistic rendering, and how good a rendering should be to allow accurate parameter estimation. Afterwards, we will discuss Contribution III an interactive method expanding current techniques to improve physical accuracy for translucent materials. We will conclude with two applications at the other end of the spectrum, one (IV) leveraging recent innovations in interactive ray tracing to solve a widespread problem in real time graphics, temporal stability, and one (V) leveraging physically based materials in a hard real time physically based application.

## 4.2 Defining photorealistic rendering

Our first discussion point is in what is photorealistic rendering, and the distinction between *physically based* and *photorealistic* rendering. In the former, the emphasis is onto creating rendering models and techniques that are based on physical radiometric processes. In the latter, we put emphasis on *validating* the models with real images (see Figure 4.1). Our first contributions II and I set out to test the limits of physically based rendering. In our work, we initially set to test how much a rendering can get close to real images.

These two investigations led to a number of interesting insight on physically based rendering. A first insight is that renderings and pictures can be compared, and actual physical quantities can be measured from them (See Figure 4.2). In our first contribution I, we created a new appearance model to predict the appearance of cloudy apple juice. The models uses data from various sources to create the rendering of a glass of apple juice. In our proof of concept model,

rendering photograph

**Figure 4.1:** Cloudy apple juice photographed and rendered the appearance model from Contribution I. In the model, we inferred apple particle concentration (0.8 g/l) and apple storage period (4 days) to match the photograph.

we have the concentration of apple juice and the number of days the apples were stored as inputs. By comparing pictures with renderings made using this parameters, we can estimate them for any glass of cloudy apple juice. In our endeavor, we identified a number of different challenges in comparing pictures and renderings, mostly related to the scene. Subtle changes in the scene can lead to big differences in direct comparisons, especially when the material influences its surroundings, such and in the case of cloudy apple juice. See the light caustic next to the glass in Figure 4.1. In this initial proof of concept, we simply compared a selected patch to get a rough estimate of the parameters.

In our main contribution II, we strive to improve upon the previous results of comparing rendering with images, for a different application. In this case, we are comparing images of glass objects, using a full pipeline to accurately estimate the scene, scan glass objects with CT scanners and placing them in the scene for final rendering. Our main contribution of this work is actually the pipeline, that allows researchers to compare pictures and renderings of glass objects, getting a quantitative comparison at the end. This will allow researchers to experiment with each step of the pipeline, improving upon the state of the art techniques used. As we can see from Figure 4.2 left, we are able to quanti-

**Figure 4.2:** To the left, comparing quantitavely renderings (top row) with real images (mid row). Error is shown in the bottom row. On the right, estimating the absorption parameter $\sigma_a$ for the glass bowl. Each coefficient was estimated independently. Each dot in the graph corresponds to a rendered image.

tatively compare images of glass objects with pictures, something that as never been done beforehand. At a first glance, the pictures and rendering are almost indistinguishable. This ability of qualitatively compare images and rendering would allow in the future to further improve existing techniques in acquisition, rendering and reconstruction. If the quality of the reconstruction becomes good enough, even path tracing could be compared with the original image.

As in the previous contribution, absorbing glass is a material that greatly influences its surroundings, so material property estimation is not trivial. As in contribution I, we use the comparisons to measure material properties, in this case the index of refraction and the spectral absorption coefficient $\sigma_a$ of the glass objects, as we can see in Figure 4.2, right. To estimate the material properties we need to generate many of them. Moreover, since we are using path tracing, we need low levels of noise to avoid uncertainties in the final parameter estimation. So, we use GPU-accelerated path tracing to achieve fast and accurate rendering of these images.

## 4.3 Interactive rendering of scattering media

After discussing photorealistic rendering in the previous section, in this section we start discussing how to bring photorealistic rendering into the interactive domain, namely in the specific case of rendering translucent materials. Our contribution is a caching scheme to improve efficiency of existing techniques.

**Figure 4.3:** Equal time comparison (left column) of our method with the reference method and qualitative comparison with diffuse subsurface scattering (upper right) and the converged reference solution (lower right). The scene is lit by a point light in a white grapefruit candle holder.

Contribution III describes an interactive technique for commodity GPUs to render translucent materials. The most important contribution in this technique is that allows rendering using *directional* BSSRDFs like the one proposed by TODO. The directional dipole is part of recent analytical models where the BSSRDF depends on $\vec{\omega}_i$, the direction of the incoming light. This allows more subtle scattering effects to be computed, accounting partially for single scattering, that in previous techniques needed to be added separately. See Figure 4.3, top row, for a comparison between the seminary analytical BSSRDF, the standard dipole REF and the directional dipole. Most of the interactive and real time techniques for rendering BSSRDFs assume that the BSSRDF can be represented as a function of only one variable (i.e. the distance between the point of incidence and emergence). This can be exploited for different optimizations, like filtering, precomputation or tabulation, that are not feasible anymore when it comes to using a directional technique.

Our technique leverages the strengths of rasterization, storing progressive maps of scattered radiosity rendered from different directions around the object. We can now account for the directionality of the light in the computation, and in addition progressively store the intermediate result as soon as the light and the

object do not change. We contribute with a fully interactive technique, that does not require neither precomputation nor texture parameterization. Given this features, we can apply this technique to procedural deformable objects, something that is generally quite difficult to achieve with precomputation techniques.

In our technique, we contribute with an improved sampling scheme, that via our maps can sample radiosity always close to the light source, allowing light to propagate through objects and around sharp corners. This can be seen in particular in Figure 4.3, in the top left image, where a point light source is places inside a candleholder made of a scattering material.

As another contribution of our technique, we leverage our scattered radiosity maps to place virtual point lights on the surface of the objects. This allows to transfer emergent light onto other surfaces. An example can be seen in Figure 4.3, where we transport the light from a point light inside an object on the outside, illuminating the bunny on the tabletop. In the same figure, we can see on how we successfully compare to a fully path traced simulation (bottom right) that in the same time we render our solution cannot achieve the same results, looking unconverged and noisy (bottom left).

## 4.4 Interactive stable ray tracing

After dealing with subsurface scattering, we continue exploring the spectrum of interactive photorealistic techniques in Contribution IV. As in the previous section, we use caching as a technique to improve existing physically based techniques. In this contribution, in particular, we discuss the problem of achieving renderings that are temporally stable, a known problem in the real time rendering community. In our case, we discuss a possible solution applicable to interactive/real time ray tracing.

When going to a interactive or real time ray tracing technique, the number of paths we can shoot per pixel becomes extremely limited, in the order or one or two. Also, the shading locations change every frame, causing a noisy image, both spatially and temporally. Existing techniques, namely temporal anti aliasing, can mitigate the problem, tough they generally introduce blur. in our technique, we recycle shading locations across frames. Shading always the same points, we improve temporal stability, while retaining sharpness. We can see the results in Figure 4.4. In the result we can see how stable ray tracing yields a sharper result than temporal antialiasing. The tehnique contributes as a rendering system to improve temporal stability: other existing techniques can

| Supersampling, 2 spp | Supersampling, 2 spp + temporal antialiasing | Supersampling, 32 spp |
|---|---|---|
| sharpness: 0.7924 | sharpness: 0.5348 | sharpness: 0.6771 |
| Stable RT, 1 spp | Stable RT, 1 spp + temporal integration | Supersampling, 32 spp |
| sharpness: 0.7085 | sharpness: 0.6060 | sharpness: 0.6771 |

**Figure 4.4:** Different techniques applied to a frame in the Sponza video, equal time comparison. Sharpness is also shown (higher is sharper). Stable ray tracing offers a less noisy result (first column) compared to reference (third column). When temporal techniques are applied (second column) Stable ray tracing yields the sharper result.

be further applies on top to further improve shading quality.

Another contribution of this technique is that it shows how we can leverage the strength of one interactive technique, namely ray tracing, against the most commonly used rasterization. Current hardware does not allow shading locations to be arbitrarily chosen within a pixel, relying on fixed patterns (such as in MSAA). In our technique, we allow shading locations to vary in screen space per pixel, while staying the same in world space.

Finally, another advantage of this technique, in particular in a photorealistic rendering context, is that it allows to store shading information across frames, e.g. to store indirect illumination. This will allow in the future to extend the

**Figure 4.5:** Pictures illustrating our VR demo application, with an in-game screenshot (left) and a picture of the setup (right).

technique with other sort of data, to further improve the technique.

## 4.5 Applying interactive photorealistic techniques

In contribution V, we push phisically based rendering even further, applying it to a hard real time environment, virtual reality. In this context, applications need to consistently perform at 90 frames per second or faster, to avoid issues for the users, e.g. dizziness or motion sickness. In our application, we modify the rendering pipeline of the Unity game engine to include phisically based BRDFs, in the discretized form of the MERL database. This proof of concept was born as an inspecting tool to debug physically based materials, aided by the provided in game controllers (see Figure 4.5). The application gives a glimpse into the future, showing how photorealistic materials augment the immersion of the application by allowing a virtual world that is more similar to the real one.

## 4.6 Discussion

Main talking points:

- Image comparison
- Parameter estimation
- Estimation of other objects - scattering parameters

- New techniques for interactive photorealistic rendering
- Extension to ray tracing and experimenting with hybrid techniques
- Challenging rendering models
- Storage of data in the point cloud
- Data driven BSSRDF models
- A look to the future, VR and which techniques can be ported

Chapter 5

# Conclusions

In this thesis, we have presented the results of the past three year of PhD studies, that started with the goal of developing new techniques for interactive rendering, with an eye onto interactive photorealism. The exploration of different techniques during the PhD did not followed a predefined scheme. The individual contributions have been developed as consequence of the findings along the way. We are happy that we managed to keep our contribution over a common theme, bringing photorealistic rendering into the interactive domain.

During the PhD we contributed with a number of results and publications. These results are relevant in many fields, including product visualization, architectural rendering, interactive previews, previews of rendering results, and video game production. We have presented a range of publications that contribute with techniques that can be employed in these fields, addressing various important challenges in each area.

In recent years, the real time rendering community is pushing more and more towards physically based models. Given this, we can use clever techniques to introduce additional photorealism in existing techniques. We saw a need for more accurate predictive rendering in an industrial domain in contribution TODO, where we tied industrial parameters to the rendering of physically based apple juice. We first investigated the need for photorealism to validate existing reconstruction and acquisition techniques in Contribution TODO. Moreover,

we proved the need of accuurate photorealistic rendering to measure unknown radiometric parameters.

After this excursion in hyper photorealistic rendering, we moved into developing techniques that improve upon existing algorithms to further enhance physically based models. In our contribution TODO, we contributed with the first technique able to render directional BSSRDF models on deformable objects, allowing also to transport emergent scattered light across the scene. Inspired by our industrial lookout, we set on solving the problem of temporally stable global illumination, in the growing field of interactive ray tracing. We contributed with a interactive technique that allows temporally stable sharp global illumination, and that can be easily added on top of many existing algorithms. Finally, in paper TODO we contributed with a proof of concept to the growing field of virtual reality rendering, introducing physically based measured BRDFs in a real time environment with hard constraints.

All the contributions listed above contributed to achieve new insights in the individual areas of interest. We can summarize the highlights of the single contributions in this thesis as the following:

- Investigated the challenges in comparing images with photorealistic renderings (Contributions I, II).
- Proposed a reconstruction and assembly pipeline that allows to compare images to renderings of the same scene (Contribution II).
- Created a new dataset of transparent objects scene and CT scan data (Contribution II).
- Proposed new techniques to estimate material parameters (Contributions I, II).
- Explored the challenges of fast interactive photorealistic rendering for rendering and quality assurance (Contributions I,III).
- Demonstrated the first interactive application of light-directional subsurface scattering (Contribution III).
- Developed a new interactive ray tracing technique to improve temporal stability without sacrificing sharpness (Contribution IV).
- Created new interactive rendering technques on to improve photorealistic light transport (Contributions III, IV).
- Created a novel Virtual Reality application to showcase phisically based materials in a hard real time context (Contribution V).

Based on these contributions, we conclude that the goal of developing new interactive techniques that push the boundaries of photorealistic rendering in the interactive and real time domain has been achieved.

Appendix I

# Interactive Appearance Prediction for Cloudy Beverages

Workshop on Material Appearance Modeling (2016)
H. Rushmeier and R. Klein (Editors)

# Interactive Appearance Prediction for Cloudy Beverages

Alessandro Dal Corso[1], Jeppe Revall Frisvad[1], Thomas Kim Kjeldsen[2], and J. Andreas Bærentzen[1]

[1]Technical University of Denmark
[2]Alexandra Institute, Denmark

**Abstract**

*Juice appearance is important to consumers, so digital juice with a slider that varies a production parameter or changes juice content is useful. It is however challenging to render juice with scattering particles quickly and accurately. As a case study, we create an appearance model that provides the optical properties needed for rendering of unfiltered apple juice. This is a scattering medium that requires volume path tracing as the scattering is too much for single scattering techniques and too little for subsurface scattering techniques. We investigate techniques to provide a progressive interactive appearance prediction tool for this type of medium. Our renderings are validated by qualitative and quantitative comparison with photographs. Visual comparisons using our interactive tool enable us to estimate the apple particle concentration of a photographed apple juice.*

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

## 1. Introduction

Quoting a food science article: "the visual appearance of a cloudy drink is a decisive factor for consumer acceptance" [Bev02]. We therefore believe that appearance prediction is highly relevant in the beverage industry of the future. It is important to predict the visual effect of a production parameter being modified, so that production parameters may be optimized without negative impact on the visual quality of the product. An appearance model also potentially enables analysis of product properties using camera sensors.

rendering photograph

**Figure 1:** *Cloudy apple juice photographed and rendered using our appearance model. In the model, apple particle concentration (1.0 g/L) and days the apples were stored (4) were selected to match the photograph.*

The cloudy part of a beverage typically consists of oil droplets or fruit flesh. We use Lorenz-Mie theory to go from a particle size distribution of fruit flesh particles, for example, to the scattering properties that we would use in a volume rendering [FCJ07]. Once the scattering properties are available, there are many ways to render the medium. However, in the case of a cloudy beverage, the scattering is neither low enough for single scattering techniques nor high enough for subsurface scattering techniques. We thus use full volume path tracing [Rus88], which quite accurately predicts the appearance of milk [FCJ12].

The complex refractive indices ($n = n' + in''$) of host liquid and particle inclusions are required as input for the Lorenz-Mie theory. In many cases, these parameters depend to some degree on production parameters. In addition, the concentration of particles is often a production parameter. This enables us to build appearance models that are parameterized by production parameters. As a case study, we build such an appearance model for apple juice.

© 2016 The Author(s)

The ability to quickly test the visual influence of different parameters is one of the key advantages of material appearance prediction. This advantage is however hampered by the very long rendering times that we typically experience with full path tracing of multiple scattering in a volume. To have fast visual feedback and shorter rendering times, we use progressive path tracing implemented on the GPU using the OptiX framework [PBD*10]. We shortly cover the steps that we took to obtain an interactive appearance prediction tool based on this framework.

We validate our appearance model by qualitative comparison of rendered results with a photograph of a commercial unfiltered apple juice (Figure 1). As the visual similarity of rendered and photographed juice is better with some parameter settings than others, we can use our appearance model to roughly estimate what the parameters might have been during production of the photographed juice.

## 2. Appearance Model

The model we present for apple juice is intended for unfiltered press juices. Consequently, we model it simply as apple particles in clear apple juice. To obtain complex refractive indices for these constituents, we exploit that the imaginary part is directly related to the absorption of the material [FCJ07]. As we shall see in the following, the absorption of the apple particles and juice both depend on how the apples are handled, and on whether anything is done to prevent the enzymatic browning that naturally takes place when apples are peeled, bruised, or pressed in an oxidative environment.

**Host.** Fruit juices have a significant amount of dissolved solids (mostly sugars [HRC09]). The total concentration of soluble solids $X$ influences the real part of the refractive index of the host. This concentration is typically measured in degrees Brix (°Bx), which is the weight percentage of dissolved solids, and it depends on the ripeness of the harvested fruits. More ripe fruits have a higher sugar content. We use two measurements from Genovese and Lozano [GL06] to find a correction for the real part of the refractive index of water. After correction, the real part of the host refractive index is

$$n'_{\text{host}}(\lambda) = n'_{\text{water}}(\lambda) + 0.0016X \ ,$$

where $\lambda$ is the wavelength of the light (*in vacuo*). For the imaginary part of the refractive index, we use the different absorbance spectra of browned clarified apple juice reported by Beveridge et al. [BFH86, Fig. 4A, curve 3]. As the wavelength increases, the absorbance values become smaller and more uncertain. Since they should not decrease below the absorption of water, we convert them to absorption coefficients and gradually blend them with the absorption spectrum of water in the range of wavelengths from 400 nm to 700 nm. The spectra we get for $n''_{\text{host}}$ as a result are in Table 1.

| $\lambda$ [nm] | $n''_{\text{host}}$ 4 days | $n''_{\text{host}}$ 9.5 days peeled | $n''_{\text{host}}$ 9.5 days | $n''_{\text{host}}$ 27 days | $n''_{part}$ |
|---|---|---|---|---|---|
| 375 | $1.58 \cdot 10^{-6}$ | $1.03 \cdot 10^{-6}$ | $2.20 \cdot 10^{-6}$ | $3.20 \cdot 10^{-6}$ | $1.14 \cdot 10^{-5}$ |
| 400 | $1.69 \cdot 10^{-6}$ | $1.10 \cdot 10^{-6}$ | $2.35 \cdot 10^{-6}$ | $3.41 \cdot 10^{-6}$ | $1.02 \cdot 10^{-5}$ |
| 425 | $6.78 \cdot 10^{-7}$ | $8.92 \cdot 10^{-7}$ | $1.39 \cdot 10^{-6}$ | $2.78 \cdot 10^{-6}$ | $1.08 \cdot 10^{-5}$ |
| 450 | $4.47 \cdot 10^{-7}$ | $8.25 \cdot 10^{-7}$ | $1.17 \cdot 10^{-6}$ | $2.68 \cdot 10^{-6}$ | $1.15 \cdot 10^{-5}$ |
| 475 | $4.08 \cdot 10^{-7}$ | $7.18 \cdot 10^{-7}$ | $9.79 \cdot 10^{-7}$ | $2.19 \cdot 10^{-6}$ | $9.83 \cdot 10^{-6}$ |
| 500 | $3.97 \cdot 10^{-7}$ | $5.50 \cdot 10^{-7}$ | $7.33 \cdot 10^{-7}$ | $1.44 \cdot 10^{-6}$ | $8.36 \cdot 10^{-6}$ |
| 525 | $3.51 \cdot 10^{-7}$ | $3.65 \cdot 10^{-7}$ | $5.06 \cdot 10^{-7}$ | $8.71 \cdot 10^{-7}$ | $3.59 \cdot 10^{-6}$ |
| 550 | $2.53 \cdot 10^{-7}$ | $2.43 \cdot 10^{-7}$ | $3.04 \cdot 10^{-7}$ | $5.05 \cdot 10^{-7}$ | $2.54 \cdot 10^{-6}$ |
| 575 | $1.67 \cdot 10^{-7}$ | $1.67 \cdot 10^{-7}$ | $1.84 \cdot 10^{-7}$ | $3.09 \cdot 10^{-7}$ | $1.92 \cdot 10^{-6}$ |
| 600 | $1.06 \cdot 10^{-7}$ | $1.13 \cdot 10^{-7}$ | $1.13 \cdot 10^{-7}$ | $1.90 \cdot 10^{-7}$ | $1.52 \cdot 10^{-6}$ |
| 625 | $6.78 \cdot 10^{-8}$ | $8.22 \cdot 10^{-8}$ | $7.64 \cdot 10^{-8}$ | $1.18 \cdot 10^{-7}$ | $1.41 \cdot 10^{-6}$ |
| 650 | $4.74 \cdot 10^{-8}$ | $6.23 \cdot 10^{-8}$ | $5.24 \cdot 10^{-8}$ | $7.22 \cdot 10^{-8}$ | $1.47 \cdot 10^{-6}$ |
| 675 | $3.70 \cdot 10^{-8}$ | $4.63 \cdot 10^{-8}$ | $3.91 \cdot 10^{-8}$ | $4.58 \cdot 10^{-8}$ | $1.78 \cdot 10^{-6}$ |
| 700 | $3.48 \cdot 10^{-8}$ | $3.48 \cdot 10^{-8}$ | $3.48 \cdot 10^{-8}$ | $3.48 \cdot 10^{-8}$ | $6.13 \cdot 10^{-7}$ |
| 725 | $8.59 \cdot 10^{-8}$ | $8.59 \cdot 10^{-8}$ | $8.59 \cdot 10^{-8}$ | $8.59 \cdot 10^{-8}$ | $2.08 \cdot 10^{-7}$ |
| 750 | $1.47 \cdot 10^{-7}$ | $1.47 \cdot 10^{-7}$ | $1.47 \cdot 10^{-7}$ | $1.47 \cdot 10^{-7}$ | $1.78 \cdot 10^{-7}$ |
| 775 | $1.49 \cdot 10^{-7}$ | $1.49 \cdot 10^{-7}$ | $1.49 \cdot 10^{-7}$ | $1.49 \cdot 10^{-7}$ | $1.58 \cdot 10^{-7}$ |

**Table 1:** *Absorption of clarified apple juice (host) and apple flesh (particles) reported in the form of the imaginary part of the index of refraction.*

**Particles.** We model the apple particles as browned apple flesh. The real part of the refractive index of the apple particles was estimated by Benitez et al. [BGL07] to be $n'_{\text{part}}(\lambda) = 1.487$. For the imaginary part, we use the absorption spectrum measured by Lu et al. [LCHA10] for bruised apple tissue (two days after bruising). These measurements are in the range of wavelengths from 500 nm and up. For the shorter wavelengths, we use the absorption spectrum measured for apple flesh by Saeys et al. [SVRT*08]. The two curves fit fairly well together, so we assume that the bruising mostly affects absorption at wavelengths from 500 nm and up. The combined spectrum that we use is in Table 1. The size distribution of the apple particles is typically bimodal [Bev02]. Smaller particles (diameter of less than around 0.65 $\mu$m) will stay suspended, while larger particles will sediment during storage and will require the juice to be shaken to be resuspended. We use a bimodal particle size distribution measured for a centrifuged apple juice [ZPDG94, Bev02].

## 3. Rendering Method

Our outset is unidirectional path tracing [Kaj86, PH10]. We use Russian roulette to terminate paths probabilistically (based on material absorption) and to choose between reflection and refraction in the case of transparent materials [AK90]. Our scene is composed of four different kinds of materials: emissive, diffuse, glass, and liquid. For accurate rendering, we assign liquid/air, glass/air, and liquid/glass interfaces to the specular surfaces in our scene.

Assuming that the specular surfaces are perfectly smooth, we use the laws of reflection and refraction for their path tracing with Fresnel reflectance as the probability of reflection [PH10]. We also account for glass absorption using an RGB absorption coefficient for crown glass $\sigma_{a,\text{glass}} = (0.234, 0.233, 0.376)$ that we calculated from the imaginary part of the refractive index of Schott N-K5 glass. The

© 2016 The Author(s)

**Figure 2:** *The scattering process using shadow rays. Each shadow ray has a sampled direction (sampled with the Henyey-Greenstein phase function) and length (from an exponential distribution of average $\sigma_t$). In (a), the random walk exits the material and continues path tracing. In (b), the last ray is absorbed and the random path discarded.*

average over the color bands of the beam transmittance $T_r = e^{-\sigma_{a,\text{glass}} s}$, where $s$ is the distance traveled through the glass medium, is then the probability that the path continues through the glass without being terminated.

The cloudy beverage is also considered a specular material, but it also contains light scattering particles. When a ray passes through this medium (either by refracting into it or by internal reflection), we perform a volumetric scattering process as illustrated in Figure 2. This is done by a stochastic walk inside the medium based on the scattering properties (scattering and extinction coefficients, $\sigma_s$ and $\sigma_t$, and asymmetry parameter, $g$) that we obtain from the Lorenz-Mie theory. At each step of the walk, we use the scattering albedo $\sigma_s/\sigma_t$ in a Russian roulette as the probability of the path surviving without being absorbed. We discard the whole walk if the ray is absorbed. If the ray is not absorbed, we sample the distance to the next scattering event using $s = \ln(-\xi)/\sigma_t$, where $\xi \in [0,1)$ is a uniform random variable. If $s$ is beyond the surface of the medium, the next scattering event is interaction with the surface in the usual way. If not, we sample a new scattering direction using the Henyey-Greenstein phase function [PH10] (which is a function of $g$). To improve efficiency, we randomly pick one RGB components when performing the scattering process, and all tracings inside the medium are done with shadow rays using the distance to the next scattering event as the maximum trace distance.

## 4. Materials

To capture a reference photograph, we set up a scene consisting of a drinking glass with unfiltered apple juice placed on a neutral white surface with a neutral white background. The glass was illuminated by a large diffuse light source at a 45° angle. We used a standard digital single-lens reflex (DSLR) camera with a 50 mm lens. The light source was a Bowens BW3370 100W Unilite, which is a compact fluorescent light source with a correlated color temperature (CCT) of 6400 K. This is fairly close to the equal energy point (E with $x = y = 1/3$) in the chromaticity diagram, which is also the reference white point of the CIE RGB color space. We thus model our light source as being purely white, and convert the spectral optical properties obtained from our appearance model to CIE RGB using the RGB color matching functions reported by Stockman and Sharpe [SS00].

| | 4 days | 9.5 days peeled | 9.5 days | 27 days |
|---|---|---|---|---|
| 0.0 g/L | 0.1333 | 0.1337 | 0.1508 | 0.2015 |
| 0.1 g/L | 0.0480 | 0.0649 | 0.1025 | 0.1741 |
| 0.2 g/L | 0.0411 | 0.0657 | 0.1048 | 0.1753 |
| 0.5 g/L | 0.0368 | 0.0689 | 0.1072 | 0.1755 |
| 1.0 g/L | **0.0363** | 0.0687 | 0.1055 | 0.1731 |
| 2.0 g/L | 0.0416 | 0.0563 | 0.0858 | 0.1519 |

**Table 2:** *RMSE for a patch of color in the lower part of the glass. We note a minimum error around the 0.5-1.0 g/L concentration and 4 days storage time.*

The apple juice is an unfiltered press juice from Orskov Foods with a sugar content of 10 g per 100 ml. We estimated $X = 11.25$ as the juice also contains pectin, organic acids, and salts in addition to the sugar [GL06], and with this $X$ we get $n'_{\text{host}}$ equal to one previously measured for regular pressed cloudy apple juice [BGL07].

## 5. Results

We performed two sets of comparisons, one qualitative in Figures 1 and 3 and one quantitative in Table 2. All comparisons show the converged result after 5000 accumulated frames. On a qualitative visual comparison, the photograph matches most closely the rendering with concentration in the 0.5-1.0 g/L range and a storage time of 4 days. For the quantitative comparison, we compare a patch of color on the lower part of the glass. As a metric, we use the RMSE of the color bands on the patch. The quantitative result places the concentration between 0.5 g/L and 1.0 g/L, slightly leaning towards the latter and confirming our qualitative comparison. Strong highlights in the glass are due to rays getting trapped due to total internal reflection.

Our implementation of the method, based on OptiX [PBD*10], runs progressively on the GPU, with an average frame rate of 12 frames per second on an NVIDIA GeForce 780 Ti card. This enables us to get nearly converged results in less than 10 minutes. A visual comparison of the main liquid is possible within a minute (~600 frames), while caustics from the juice take much longer to converge.

## 6. Discussion

By combining existing techniques in our framework, we enable interactive testing of the influence of different production parameters on the appearance of a cloudy beverage. Our case study enables approximate simulation of the appearance

© 2016 The Author(s)

**Figure 3:** *Visual comparison of our renderings varying two parameters: particle concentration and apple storage time.*

of cloudy apple juice during production. However, we cannot match the photograph perfectly, and can thus only give a rough estimate of the parameters. As a possible future step, we want to improve precision by better estimating scene geometry, camera parameters, and lighting environment.

**Acknowledgements.** Thanks to Anders Wang Kristensen for setting up a digital scene to match the photographed scene, and to him and Marek Krzysztof Misztal for their help with respect to capture of the reference images.

## References

[AK90] ARVO J., KIRK D.: Particle transport and image synthesis. *Computer Graphics (Proceedings of SIGGRAPH '90) 24*, 4 (August 1990), 63–66. 2

[Bev02] BEVERIDGE T.: Opalescent and cloudy fruit juices: Formation and particle stability. *Critical Reviews in Food Science and Nutrition 42*, 4 (2002), 317–337. 1, 2

[BFH86] BEVERIDGE T., FRANZ K., HARRISON J. E.: Clarified natural apple juice: Production and storage stability of juice and concentrate. *Journal of Food Science 51*, 2 (March 1986), 411–433. 2

[BGL07] BENITEZ E. I., GENOVESE D. B., LOZANO J. E.: Scattering efficiency of a cloudy apple juice: Effect of particles characteristics and serum composition. *Food Research International 40*, 7 (August 2007), 915–922. 2, 3

[FCJ07] FRISVAD J. R., CHRISTENSEN N. J., JENSEN H. W.: Computing the scattering properties of participating media using Lorenz-Mie theory. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (July 2007), 60:1–60:10. 1, 2

[FCJ12] FRISVAD J. R., CHRISTENSEN N. J., JENSEN H. W.: Predicting the appearance of materials using Lorenz-Mie theory. In *The Mie Theory: Basics and Applications*, Hergert W., Wriedt T., (Eds.), vol. 169 of *Springer Series in Optical Sciences*. July 2012, ch. 4, pp. 101–133. 1

[GL06] GENOVESE D. B., LOZANO J. E.: Contribution of colloidal forces to the viscosity and stability of cloudy apple juice. *Food Hydrocolloids 20*, 6 (August 2006), 767–773. 2, 3

[HRC09] HUANG Y., RASCO B. A., CAVINATO A. G.: Fruit juices. In *Infrared Spectroscopy for Food Quality Analysis and Control*, Sun D.-W., (Ed.). Academic Press/Elsevier, 2009, ch. 13, pp. 355–375. 2

[Kaj86] KAJIYA J. T.: The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4 (August 1986), 143–150. 2

[LCHA10] LU R., CEN H., HUANG M., ARIANA D. P.: Spectral absorption and scattering properties of normal and bruised apple tissue. *Transactions of the American Society of Agricultural and Biological Engineers 51*, 1 (2010), 263–269. 2

[PBD*10] PARKER S. G., BIGLER J., DIETRICH A., FRIEDRICH H., HOBEROCK J., LUEBKE D., MCALLISTER D., MCGUIRE M., MORLEY K., ROBISON A., STICH M.: OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010) 29*, 4 (July 2010), 66:1–66:13. 2, 3

[PH10] PHARR M., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation*, second ed. Morgan Kaufmann/Elsevier, 2010. 2, 3

[Rus88] RUSHMEIER H. E.: *Realistic image synthesis for scenes with radiatively participating media*. PhD thesis, Cornell University, Ithaca, NY, USA, 1988. 1

[SS00] STOCKMAN A., SHARPE L. T.: The spectral sensitivities of the middle- and long-wavelength-sensitive cones derived from measurements in observers of known genotype. *Vision Research 40*, 13 (2000), 1711–1737. 3

[SVRT*08] SAEYS W., VELAZCO-ROA M. A., THENNADIL S. N., RAMON H., NICOLAI B. M.: Optical properties of apple skin and flesh in the wavelength range from 350 to 2200 nm. *Applied Optics 47*, 7 (March 2008), 908–919. 2

[ZPDG94] ZIMMER E., PECORONI S., DIETRICH H., GIERSCHNER K.: Process technological and chemical basis of the production of cloudy apple juice, also considering continuous processes. I. Contributions to the chemical analysis of natural cloudy apple juices including physical parameters. Part II. *Die industrielle Obst- und Gemüseverwertung 79*, 12 (1994), 426–434. 2

© 2016 The Author(s)

Appendix II

# Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering

Research Article Vol. 56, No. 27 / September 20 2017 / Applied Optics 7679

applied optics

# Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering

Jonathan Dyssel Stets,[1,†] Alessandro Dal Corso,[1,†] Jannik Boll Nielsen,[1] Rasmus Ahrenkiel Lyngby,[1] Sebastian Hoppe Nesgaard Jensen,[1] Jakob Wilm,[1] Mads Brix Doest,[1] Carsten Gundlach,[2] Eythor Runar Eiriksson,[1] Knut Conradsen,[1] Anders Bjorholm Dahl,[1] Jakob Andreas Bærentzen,[1] Jeppe Revall Frisvad,[1,*] and Henrik Aanæs[1]

[1]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, 2800 Kongens Lyngby, Denmark
[2]Department of Physics, Technical University of Denmark, Fysikvej, 2800 Kongens Lyngby, Denmark
*Corresponding author: jerf@dtu.dk

Received 23 May 2017; revised 15 August 2017; accepted 15 August 2017; posted 16 August 2017 (Doc. ID 295986); published 19 September 2017

**Transparent objects require acquisition modalities that are very different from the ones used for objects with more diffuse reflectance properties. Digitizing a scene where objects must be acquired with different modalities requires scene reassembly after reconstruction of the object surfaces. This reassembly of a scene that was picked apart for scanning seems unexplored. We contribute with a multimodal digitization pipeline for scenes that require this step of reassembly. Our pipeline includes measurement of bidirectional reflectance distribution functions and high dynamic range imaging of the lighting environment. This enables pixelwise comparison of photographs of the real scene with renderings of the digital version of the scene. Such quantitative evaluation is useful for verifying acquired material appearance and reconstructed surface geometry, which is an important aspect of digital content creation. It is also useful for identifying and improving issues in the different steps of the pipeline. In this work, we use it to improve reconstruction, apply analysis by synthesis to estimate optical properties, and to develop our method for scene reassembly.** © 2017 Optical Society of America

***OCIS codes:*** (150.4232) Multisensor methods; (150.6910) Three-dimensional sensing; (150.1488) Calibration; (160.4760) Optical properties; (290.1483) BSDF, BRDF, and BTDF; (330.1690) Color.

https://doi.org/10.1364/AO.56.007679

## 1. INTRODUCTION

Several research communities work on techniques for optical acquisition of physical objects and their appearance parameters [1–5]. Thus, we are now able to acquire nearly any type of object and perform a computer graphics rendering of nearly any type of scene. The range of applications is broad and includes movie production [2], cultural heritage preservation [3], 3D printing [4], and industrial inspection [5]. A gap left by these multiple endeavors is a coherent scheme for acquiring a scene consisting of several objects that have very different appearance parameters, together with the reassembly of a digital replica of such a scene. Our objective is to fill this gap for the combination of transparent and opaque objects, as many real-world scenarios exhibit this combination. An example is a living room, like the one rendered in Fig. 1 (right). We propose a pipeline for acquiring and reassembling digital scenes from this type of heterogeneous real-world scene. In addition, our pipeline closes the loop by rendering calibrated images of the digital scene that are commensurable with photographs of the original physical scene (see Fig. 1, left). This allows for validation and fine-tuning of appearance parameters. The quantitative evaluation we get from pixelwise comparison of rendered images with photographs is a great improvement with respect to validation of the acquired digital representation of the physical objects.

When addressing the problem of acquiring a heterogeneous scene, there is an infinite variety of scenes and object types to choose from. So, to make our task feasible, we focus on scenes that combine glassware and non-transparent materials, more specifically, a white tablecloth and cardboard with a checkerboard pattern. We made these choices, as glass requires a different acquisition modality, the tablecloth bidirectional reflectance distribution function (BRDF) is spatially uniform but not necessarily simple, and the cardboard has simple two-color variation. The latter is particularly useful for observing how light

**Fig. 1.** To the left, we compare rendered images (top) with photographs (bottom). More views are available in Appendix A. The scenes to the left were digitized using our pipeline and include both glass objects and non-transparent objects (tablecloth and backdrop). To the right, we exemplify the use of our pipeline for virtual product placement using our digitized glass objects, with estimated optical properties and artifact-reduced removal of markers.

refracts through the glass. The chosen case is also of particular interest, since glass is present in many intended applications of optical 3D acquisition. Considering the highly multidisciplinary nature of our work, we have released our dataset (http://eco3d.compute.dtu.dk/pages/transparency). This facilitates further investigation by other researchers of the different steps of our pipeline with the possibility of a quantitative feedback at the end of the process.

### A. Related Work and Contributions

Researchers occasionally compare renderings with photographs to provide a qualitative verification of a presented rendering technique. The work by Phong [6], Goral *et al.* [7], and Takagi *et al.* [8] are early examples of this trend. A procedure to bring a rendered image close to a photograph was first presented by Meyer *et al.* [9]. In this work, likeness of images was evaluated perceptually by human observers. Pixelwise comparison of photographs with rendered images is surprisingly uncommon. The few examples we have found are by Rushmeier *et al.* [10], Karner and Prantl [11], Pattanaik *et al.* [12], and Jones and Reinhart [13,14]. These examples build on the rendering framework described by Greenberg *et al.* [15]. Employing such a framework for more complex scenes is a long and tedious process [16]. The key issue is that a scene specification is expected as an input.

Several problems arise as a result of not having correspondence between the physical and the digital scene. Misalignment due to inaccurate scene and viewing geometry and inaccurate orientation of the lighting environment are some of the essential problems identified in previous work [17,18]. One way to deal with this problem is to calculate error for image patches when evaluating results [13,19,20]. As opposed to this, our digitization pipeline (Fig. 2) provides both reference photographs and correspondingly calibrated scene and viewing geometry so that pixelwise comparison becomes meaningful.

Pixelwise comparison of rendered images with photographs is not only useful for quantifying the photorealism of a rendering in terms of error measurements. We find it particularly useful for improving the digitization pipeline. The fact that our pipeline enables quantitative evaluation led us to more specific contributions in its different steps. These contributions are mostly in the reassembly and are as follows. (a) A cross-modality marker-based placement approach, enabling accurate placement of objects scanned with one modality into scenes scanned with another modality. (b) A soft object deformation technique dealing with surface intersections after object placement, which is critical for scenes containing transparent or translucent objects. (c) A micropolygon labeling approach for assigning BRDFs to acquired geometry. (d) A color calibration scheme enabling use

**Fig. 2.** Overview of our digitization pipeline in four main stages: acquisition, reconstruction, reassembly, and rendering. A video presentation of our pipeline is available in supplementary Visualization 1. Colored arrows show the path through the pipeline of transparent objects (dotted blue) and non-transparent objects (dashed red).

of spectral optical properties for calculating reflectance, transmittance, and absorption. (e) Perspective unwrapping of mirror probe images to improve precision when the environment is not very distant. (f) Use of analysis by synthesis for fine-tuning physics-based optical properties.

Digitization is most often unimodal and tailored toward objects with a specific type of surface reflectance behavior [1]. While unimodal techniques are becoming more versatile [21–23], objects with a transparent material, such as glass, still pose challenging problems. Their reflectance behavior is so different that they require an entirely different modality, such as computed tomography (CT) [24]. The transparent object must then be removed from the scene to be scanned elsewhere. In the meantime, the surrounding scene can be scanned with a more common technique. However, as the transparent object takes most of its appearance from its surroundings, it must be repositioned in the surrounding scene (physically and digitally) if we are to take reference images for comparison with rendered images. The purpose of our scene reassembly is to address this type of issue.

Our digitization technique is multimodal. Currently, such techniques seem to exist only in the context of sensor fusion [25–27]. Here, the goal is to optimize reconstruction by fusing data from different sensor modalities with complementary characteristics. Even so, the different modalities see the same object and thus work for materials with a similar reflectance behavior. The challenge is then mostly in registration of the scans. In their final remarks and suggestions for future work, Weinmann and Klein [1] discuss possible ways of combining multiple techniques tailored to different types of surface reflectance. Our pipeline is a different way to take a step in this direction.

In summary, our work makes it possible to perform multimodal digitization and scene reassembly in such a way that rendered images of the reassembled scene can be quantitatively compared to photographs of the original. This enables us to provide the first empirically founded investigation of the appearance accuracy of objects digitized using a non-optical scanner.

## 2. DIGITIZATION PIPELINE

We divide our pipeline into four stages: (1) acquisition, (2) reconstruction, (3) reassembly, and (4) rendering. Figure 2 provides an overview. As illustrated, transparent objects (dotted blue arrows) and non-transparent objects (dashed red arrows) take different paths through the pipeline. The acquisition stage includes structured light scanning of non-transparent objects, CT scanning of transparent objects, gonioreflectometric reflectance measurements, and photographic capture of environment, color chart, and scene reference images. Figure 3 provides details of our workflow in these acquisition steps (except the simpler captures of environment and color chart). The second stage includes reconstruction of surface meshes, material BRDFs, and color space. The third stage is reassembly of the digital scene consisting of geometric objects, material appearance properties, and environment map. The fourth and final stage is rendering and comparison with reference images.

Our acquisition stage requires an elaborate hardware setup. We assemble the physical scene in a black light-proof enclosure. This has five LED light tubes for scene lighting, which we capture by high dynamic range (HDR) imaging of a light probe. To acquire non-transparent geometry inside this enclosure, we use a structured light scanner consisting of a toe-in stereo camera rig and a light projector mounted on a robotic arm [28,29]. We chose a converging camera configuration (toe-in) to increase the overlap of the fields of view so that we get a denser point cloud per stereo view. Together with an LED-based illumination arc, we also use this camera rig with exact control for measuring isotropic BRDFs. For transparent objects, we use a CT scanner. In the following subsections, we describe the individual steps of the pipeline with a focus on details required for reproducibility and on non-standard techniques that we introduce.

### A. Camera Calibration and Settings

The camera system is calibrated using a standard technique [30]. Our calibration board is an 11 by 12 black-and-white checkerboard. For the intrinsic calibration (Pass 1 of Fig. 3, left), we include a large variety of views to estimate good lens distortion coefficients. To facilitate stereo calibration, we also ensure that both cameras have the calibration board fully in view. For extrinsic calibration (Pass 2 of Fig. 3, left), we balance good coverage of the scene and good coverage of the calibration board. Since we cannot change the camera system while collecting data, we chose a small aperture to ensure that background

**Fig. 3.** Our workflow for scanning the geometry of non-transparent objects and collecting reference images (left), for scanning the geometry of transparent objects (middle), and for measuring material reflectance properties (right).

and projected structured light patterns are always in focus from all views. The full setup is in a dark room environment to eliminate external light, so we use a long shutter time (600 ms) to obtain sufficient exposure. A slight noise component is present in the images, but this is considered negligible. Finally, we use the estimated distortion coefficients to remove distortion from all images in the dataset so that subsequent algorithms may assume a pinhole camera model.

To avoid any compression or manipulation of the images by the camera software, in particular automatic color correction, we read the raw sensor data directly. We use bilinear interpolation to reconstruct RGB images from the raw Bayer pattern images. By doing this, we obtain a consistent RGB color space. Moreover, the raw sensor data are linear and correlate directly with radiometric quantities, which allows for better BRDF and environment map estimation in later stages of our pipeline.

We capture radiometrically relevant parts of our dataset in HDR by stacking multiple exposures [31]. More specifically, we stack 11 exposures at one-stop intervals ranging from 1 to 2048 ms. For the other parts of the dataset, we capture a single image at an exposure time of 600 ms.

### B. Surface Reconstruction from Structured Light

We use a standard Gray code structured light approach to generate raw point clouds for a scene [32,33]. With camera parameters from the calibration, we transform these point clouds into the same world coordinate system.

To reconstruct one connected triangle mesh from the point clouds, we merge them into a single point cloud and perform screened Poisson reconstruction with trimming and an octree depth of nine [34]. This technique requires point normals, so before the merging, we generate normals for each point cloud as follows. We resample the point cloud down to 100,000 vertices via Poisson disk sampling [35] and then compute normals via planar fitting to a nearest neighborhood of 500 points (~16 mm radius). We then reorient all the normals according to the location of one of the cameras and transfer them back onto the original point cloud. This procedure ensures smooth continuous normals, necessary for a good performance of the mesh reconstruction algorithm. As we rely on smoothing, we cannot reconstruct features in the mesh with the same physical size as the alignment error accumulated from structured light and calibration. The aim of the chosen constants was to preserve features by striking a balance between too noisy and too smooth. The operability of the pipeline is, however, not sensitive to the choice of these constants.

### C. Material BRDF Reconstruction

We assume that all non-transparent materials in the scene are opaque and isotropic, so we model their reflectance properties by BRDFs. To acquire a BRDF, we combine traditional canonical gonioreflectometric sampling [36] with a BRDF interpolation (reconstruction) technique [37]. We follow the workflow outlined in Fig. 3 (right). A light arc illuminates material samples from 11 unique inclinations, evenly distributed from 7.5° up to 90° with 7.5° steps. We place a flat material sample at the center of the circle partly traced by the light arc. Using the cameras mounted on the robot, we then measure radiance reflected by the sample across one octant of a sphere. The center of this sphere coincides with that of the light arc, while its radius is slightly larger to avoid collision between the robot and the arc. The robot moves in steps of 7.5° and captures 11 HDR images of the sample per step, one for each light direction. In total, this yields 2,783 HDR images per material. We avoid tangential and zenith viewing directions (90° and 0°, respectively). In the former case, no reflected radiance should be visible, while in the latter the light arc occludes the view of the sample.

The 2,783 observations are too few to faithfully represent the BRDF of a material in a photorealistic rendering. We need an interpolation scheme to fill the entire $(90 \times 90 \times 180)$ Mitsubishi Electric Research Laboratories (MERL) format BRDF look-up table [38]. The reconstruction method by Nielsen *et al.* [37] is our interpolation scheme. First, we use each of the 100 BRDFs in the MERL dataset [38] as sample points in a $90 \cdot 90 \cdot 180 = 1,458,000$ dimensional space. The nonlinear mapping of Nielsen *et al.* [37] is then applied to each of the samples. The mapped samples are ordered as rows of a matrix $\mathbf{X} \in \mathbb{R}^{m \times d}$ where $m$ is the number of BRDF samples, and $d$ is the dimension of the space. The zero-mean matrix is computed as $\mathbf{X} - \overline{\mathbf{x}}$, with $\overline{\mathbf{x}}$ being the sample mean. From this, the singular value decomposition $\mathbf{X} - \overline{\mathbf{x}} = \mathbf{U}\Sigma\mathbf{V}^T$ is used to compute the eigenvectors and eigenvalues of the covariance matrix of $\mathbf{X} - \overline{\mathbf{x}}$, which are given as the columns of $\mathbf{V}$ and the diagonal elements of $\Sigma$, respectively. This is effectively a principal component analysis (PCA), where the eigenvectors are the principal components. A matrix composed of the scaled principal components as columns are computed as $\mathbf{Q} = \mathbf{V}\Sigma$.

Now, the full BRDF can be reconstructed from this principal component space by projection. Let $x' \in \mathbb{R}^n$ be $n$ BRDF observations measured for a given material. Then, let $\overline{\mathbf{x}}' \in \mathbb{R}^n$ be the mean values and $\mathbf{Q}' \in \mathbb{R}^{n \times k}$ be the scaled eigenvectors corresponding to the direction pairs of those $n$ observations. A vector $\mathbf{c}$ that spans the full space can be constructed by finding the linear combinations of principal components that best approximate the $n$ observations. We do this by solving the linear least-squares optimization problem given by

$$\begin{aligned}\mathbf{c} &= \underset{\mathbf{c}}{\operatorname{argmin}} \|(\mathbf{x}' - \overline{\mathbf{x}})' - \mathbf{Q}'\mathbf{c}\|^2 + \eta\|\mathbf{c}\|^2 \\ &= (\mathbf{Q}'^T\mathbf{Q}' + \eta\mathbf{I})^{-1}\mathbf{Q}'^T(\mathbf{x}' - \overline{\mathbf{x}}').\end{aligned}$$

Note that by adding a penalty $\eta$ to the norm of $\mathbf{c}$, this effectively becomes a Tikhonov regularized least squares. Now, the full, mapped BRDF is reconstructed as $\mathbf{x} = \mathbf{Q}\mathbf{c} + \overline{\mathbf{x}}$. The inverse of the nonlinear mapping applied to $\mathbf{X}$ is applied to $\mathbf{x}$ to get the actual, unmapped BRDF of the material. The described approach is applied to every single non-transparent material in the scene in order to obtain models of their reflectance properties.

This approach assumes that the MERL database encompasses the class of materials present in the scene. Effectively, this is a practical compromise between dense, unbiased, canonical BRDF sampling and fast, inferred BRDF sampling. This enables us to obtain high confidence BRDFs in a matter of a few hours.

### D. Surface Reconstruction from CT

In our dataset, we have three glass objects: a sphere, a teapot (pot and lid), and a bowl (bowl and lid), for a total of five pieces. All objects have spherical plastic markers glued onto

**Fig. 4.** CT scans of the bowl (top row) and the teapot (bottom row) with markers glued onto them. In the left column, visualized using a 1D transfer function. Note the different density of the markers. In the right column, a slice scaled to display streak artifacts.

their outer surfaces. We CT scan each glass piece to obtain x-ray radiographs and use the CT PRO 3D reconstruction software from Nikon Metrology to obtain a volumetric image for each piece. The resolution of the reconstructed volume is up to $1000^3$ voxels. Due to beam hardening, high-density differences between materials lead to streak artifacts [39], especially around our markers and at the tops and bottoms of the objects (see Fig. 4). We account for these artifacts in the volumetric segmentation.

From a CT scan, we generate two triangular meshes with vertex normals: one for the glass object and one the plastic markers. Figure 5 provides an overview of our procedure. We start with the markers, which appear as elements of higher density in the scan. We preprocess the scan by clamping all the values under a certain threshold to zero and then create a mesh using dual contouring [40]. Generating the glass mesh is more cumbersome. We also use dual contouring in this case, but because of the streak artifacts (Fig. 4), it is not possible to isolate the glass mesh via a threshold. Instead, we use a lower threshold that removes only noise, then estimate the marker positions and use these to remove the markers from the glass mesh.

To estimate marker positions, we determine a series of center/radius pairs $(\mathbf{c}_i, r_i)$ by fitting a multi-sphere model to the marker mesh vertices using a tuned random sample consensus (RANSAC) algorithm [41]. We then carve a hole by excluding all the triangles that are inside a sphere with center $\mathbf{c}_i$ and radius $(1+\epsilon)r_i$, where $\varepsilon$ is usually in the 0.5–0.75 range. We store the marker positions $\mathbf{c}_i$ so that we can use them to transform from the local coordinate system of the glass object to the world coordinate system (see Section 2.F).

After removing the markers, the glass meshes still have aliasing artifacts. To deal with this issue, we first decimate the mesh down to 1% of the original vertices via quadric edge collapse. The holes are then easy to close by identifying the edge loops surrounding each hole and filling these with triangles. We then

**Fig. 5.** Reconstruction from CT with stages illustrated using Phong shading (top row) and wireframe shading (bottom row). After estimating the marker mesh (first column) and fitting spheres to the markers, we reconstruct the object mesh (second column). To eliminate noise, we first simplify the mesh (third column) and then close the holes and apply our subdivision-decimation loop to get the final object mesh (fourth column).

introduce a subdivision-decimation loop with alternating $\sqrt{3}$-subdivision [42] and decimation to 33% of the original vertices. We perform this subdivision-decimation operation four times to obtain a cleaned mesh. The decimation removes unwanted high-frequency features from the mesh. Thus, we generate smooth meshes at the cost of some geometric precision. We are again trying to strike a balance between reconstruction error and too much smoothing. In Section 4, we compare our method with a different cleaning procedure that better preserves geometry.

### E. Scene Reassembly for Non-transparent Objects

Two operations are necessary to prepare the background mesh for rendering: labeling and deformation. In the labeling, our objective is to identify BRDFs and label each face of the mesh with a BRDF. Assuming a scene with a small number of known BRDFs, we apply edge detection and watershed on the images of the scene to segment BRDF boundaries. Shadows, specular highlights, and different viewing angles of the scene complicate fully automatic BRDF identification. Our approach gets us most of the way, but we manually correct any residual misclassification. Figure 6 shows a label image produced by our labeling technique.

The label images can be used in multi-view projective texturing of the background mesh. However, we would like to precompute the view and label selection instead of doing

**Fig. 6.** Labeling of the image to the left results in the label image to the right. Each color in the label image represents a label that we assign a BRDF to. The black edges between labels indicate areas where we apply a nearest neighbor method.

**Fig. 7.** Subdividing the mesh dissolves unwanted boundary sawtooth artifacts that originate from the BRDF labeling.

it millions and millions of times while rendering. To avoid *uv*-unwrapping of the mesh for storing precomputed labels, we take an approach inspired by micropolygon rendering [43]. We project each vertex of a face onto the label images of the scene and select the face BRDF according to the image label that most of the face vertices were projected to. If a vertex projects to an unknown label, we resolve it by a nearest neighbor search. Since faces around material boundaries overlap multiple materials, we get sawtooth artifacts. We dissolve these by subdividing the mesh until the rendered triangles are smaller than the surface area observed in a pixel; see Fig. 7.

When applying physically based rendering, we observed intersections between background scene and glass meshes. This could be due to small errors in reconstruction and positioning, or perhaps the harder glass objects press down the tablecloth when placed for reference imaging. It causes significant visual artifacts, since the rendering exposes all surfaces of a transparent object. To eliminate these artifacts, we accommodate the hard object (glass) by deforming the soft object (tablecloth); see Fig. 8. To deform the soft object, we need a "down" direction in which to push the vertices. We first find contact vertices. These are vertices in each mesh that are close to any vertex of the other mesh. We consider vertices close if the distance between them is less than 7% of the bounding box diagonal of the hard object. Using least squares regression, we fit a contact plane to the contact vertices of the soft object. We set the sign of the contact plane normal so that the upper half-space contains the center of the hard object bounding box. Projection of a contact vertex to the normal of the contact plane then measures the height of the vertex. For each soft object contact vertex $\mathbf{x}$, we find the nearest hard object contact vertices and push $\mathbf{x}$ down below the lowest one of these.

### F. Scene Reassembly for Transparent Objects

To reposition the glass objects in the scene, we rigidly transform the meshes reconstructed from CT to the world coordinate system of the background mesh. We obtain this transformation by matching markers in the stereo images with the marker coordinates $\mathbf{c}_i$ computed during reconstruction from CT (see Section 2.D).

**Fig. 8.** Deformation of background mesh, where we push the background vertices down to avoid mesh intersection.

**Fig. 9.** Repositioning a CT scanned object in the background scene. We identify and match the markers in the stereo image pairs and calculate their corresponding 3D points. Pairing these with marker coordinates from the CT scans, we transform the CT scanned piece of an object into the world coordinate system.

To find the markers, we employ a size invariant circle Hough transform [44]. This works well for our dataset, where the markers show high contrast against their surroundings. We match markers in the left and the right images via Sampson distance [45]. Using this technique, markers on the same epipolar line lead to false positives, so we manually inspect the result. We also manually discard detected markers that are visible through the glass, as the refraction would lead to incorrect positioning. Markers in both stereo images with no match are discarded. The result is a set of matched markers in image coordinates, as seen in Fig. 9 (bottom left). We then triangulate the matched markers from the stereo views and gather them in clusters of 3D points. We remove outliers via their distances from the cluster centers, and for each cluster we select the point with the lowest reprojection error. An example of the points and clustering is shown in Fig. 9 (top middle).

We manually pair the 3D marker coordinates from the images with the marker coordinates $\mathbf{c}_i$ from the CT scans. We perform Procrustes analysis [46] on the two point sets, excluding reflection, since we assume a rigid transformation applied to each vertex of the mesh. The bowl and the teapot are composed of multiple pieces. For these objects, we compute the transformation individually for each piece. The result of the object transformed into the scene is shown in Fig. 9 (top right). We found that in order to have low error in the transformation, the chosen markers should sample the surface evenly and be visible from most views.

### G. Color Calibration

Images are quantitatively comparable only if they live in the same color space. Thus, we must ensure that our radiometry-dependent data, namely reference images, environment map, and BRDFs, are in the same color space. We do this by imaging a color chart of precisely known colors. More specifically, we use second-degree root-polynomial color correction [47] based on a 24-patch ColorChecker Classic from X-Rite. This provides a matrix that transforms from camera RGB to XYZ, where we assume illuminant D50 when specifying the XYZ values of the colorchecker. With the assumption of illuminant D50, we can transform colors to the CIE L∗a∗b∗ color

**Fig. 10.** Color calibration: raw images (left) and color corrected images (right). The camera sensor is particularly sensitive to green.

**Fig. 11.** Unwrapping of a spherical probe. We know the sphere radius $R$ from specification, the camera position $\mathbf{c}$ through calibration, and the sphere center $\mathbf{o}$ by triangulation. Radiance at $\mathbf{p}_{\text{proj}}$ in our image then corresponds to the environment map direction $\vec{l}$. The result for the robot enclosure is in the lower left corner in latitude–longitude panoramic format (here tone-mapped).

space and then compute color differences using the $\Delta E_{00}$ metric [48]. We use this to refine our result by minimizing $\Delta E_{00}$ using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [49]. The result is in Fig. 10. The average color difference is $\Delta E_{00} = 1.97 \pm 1.21$, which is larger than 1 JND (just noticeable difference) [50], but we find it acceptable.

Since we work with glass objects (and chrome; see Section 2.H), we need refractive indices to determine reflectance, transmittance, and absorption properties. Refractive indices can be found per wavelength in tables of research papers. To use such spectral optical properties together with our trichromatic image data, we integrate them to CIE RGB using the CIE RGB color-matching functions listed by Stockman and Sharpe [51]. It is important to normalize these functions [52] and to use RGB rather than XYZ [53]. This is because a refractive index is not a color, but rather a quantity that in trichromatic representation should resemble a sparse sampling of the spectrum. Thus, as recommended by other authors [54], we choose CIE RGB as our rendering color space. After transforming our image data from camera RGB to XYZ, we therefore convert them to CIE RGB [55]. As a final step, we apply Bradford chromatic adaptation [50], adapting to the originally assumed illuminant D50, so that renderings and reference images get closer to real-life appearance.

### H. Environment Lighting

To capture the lighting observed in the reference images, we use a method similar to the mirror probe technique [56]. However, we use a pinhole camera model for probe image unwrapping instead of the standard orthographic model. Our pipeline enables this, as we have a calibrated camera and know its position relative to the photographed mirror probe. With the pinhole model, we obtain a more precise estimate of the environment lighting. The environment map is generated from HDR images and stored in latitude–longitude panoramic format [50]. We use a polished grade G100 chrome bearing ball as mirror probe.

An environment map represents an infinite area light and maps a direction to a texture element (a texel). To do unwrapping, we map each texel direction $\vec{l}$ to the corresponding pixel position $\mathbf{p}_{\text{proj}}$ in a light probe image. Given the configuration illustrated in Fig. 11, we have

$$\vec{v} = \frac{\mathbf{c} - \mathbf{o}}{\|\mathbf{c} - \mathbf{o}\|}, \quad \vec{n} = \frac{\vec{v} + \vec{l}}{\|\vec{v} + \vec{l}\|}, \quad \mathbf{p} = \mathbf{o} + R\vec{n}, \quad \mathbf{p}_{\text{proj}} = \mathbf{M}[\mathbf{p}^T \quad 1]^T,$$

where camera matrix $\mathbf{M}$ and camera position $\mathbf{c}$ are available from our calibration. The radius of the sphere $R$ is available from the bearing ball specification, and we find the center of the sphere $\mathbf{o}$ by manually annotating the sphere and then triangulating it. We assume that the distance to the actual light along $\vec{l}$ is equal to the distance between camera and sphere $\|\mathbf{c} - \mathbf{o}\|$. This assumption works well in practice, leading to an error smaller than the uncertainty of $\mathbf{o}$ caused by the triangulation. With the original orthographic camera model, we can reconstruct the lighting for all directions except one ($-\vec{v}$). In our model, we cannot reconstruct the lighting for a set of directions ($\vec{n} \cdot \vec{v} \leq R/\|\mathbf{c} - \mathbf{o}\|$), so we set them to black. Since we do our unwrapping in world space, we can combine contributions from multiple camera views with no need to align them afterwards.

The environment map is color corrected according to Section 2.G, which enables us to correct for the angularly dependent reflectance of chrome. The correction is to divide by Fresnel reflectance, which we compute during unwrapping. As input for Fresnel's equations, we use the angle $\beta$ between $\mathbf{c} - \mathbf{p}$ and $\vec{n}$ and the complex refractive index of chrome [57] converted from spectrum to CIE RGB. The result is shown in the inset of Fig. 11.

### I. Rendering

We render images using progressive unidirectional path tracing [58,59] implemented in OptiX [60]. The captured HDR environment map is the sole light source in our scene [56]. When rendering non-specular materials, we importance-sample the environment map to get direct illumination and use sampling of a cosine-weighted hemisphere to get indirect illumination. From our labeling, we have one BRDF attached to each triangle in our scene. For non-transparent objects, we use our measured BRDFs tabulated in the MERL format [38]. To terminate paths probabilistically, we use Russian roulette based on the

bihemispherical reflectance of each measured BRDF. This reflectance is calculated in a preprocessing step using Monte Carlo integration. We deal with transparent objects in the usual way, setting reflectance and transmittance according to Fresnel's equations of reflection and Bouguer's law of exponential attenuation. Given their small surface, we were unable to estimate a BRDF for the markers. Instead, we render them as glass with all refracted rays being absorbed.

## 3. ANALYSIS BY SYNTHESIS

The ability to render images comparable to photographs enables us to use our pipeline for improving parameter estimates through analysis by synthesis. As an example, we need a scaling factor for our HDR environment map, as it measures relative radiance [31]. We estimate this factor by taking ratios of references and renderings with the background scene alone. Another example is estimating real and imaginary parts of glass refractive indices. As analysis by synthesis is fundamentally ill posed [61], we take our outset in physics-based initial guesses such as Schott K5 crown glass (sphere and teapot) and soda lime glass (bowl). Spectral refractive indices for these glasses were obtained from an online database (http://refractiveindex.info) and converted to CIE RGB. All parameters were estimated using views different from the ones in our comparisons of renderings with references.

As an example of our analysis by synthesis, we plot the evolution of the root-mean-squared error (RMSE) for different renderings of the glass bowl in Fig. 12. For each rendering, we vary a trichromatic component of the absorption coefficient (which directly relates to the imaginary part of the refractive index). We identify a distinct minimum in the error for each channel, with a slightly larger uncertainty in the red channel. The minimum values in this figure were used in our renderings of the glass bowl. We apply the same analysis to the teapot and the sphere.

Given an initial guess for a parameter, we can employ standard optimization algorithms, defining the RMSE between the reference and the rendering as a cost function to minimize. To reduce rendering times, the evaluation of the cost function can be calculated on a downsampled image or limited to a specific patch of the images. Various general optimization algorithms exist for minimizing expensive cost functions [62].

**Fig. 12.** Analysis by synthesis to estimate absorption of the glass bowl. We run renderings in low resolution and change the absorption in each color channel one at the time. In the case of the bowl, the blue channel is the most sensitive one.

## 4. RESULTS

Our scenes consist of a backdrop, a stand, and a glass object (with markers) placed on the stand. The backdrop is a 30 by 20 white-and-gray checkerboard print on 120 cm by 80 cm semi-matte cardboard, and the stand is a tabletop with a white cloth. An example scene is depicted in Fig. 13. We implemented our reconstruction and reassembly procedures as a modular software pipeline and computed all rendered images using our path tracer. As illustrated in Fig. 2 and mentioned in Section 2.G, we color correct both rendered images and reference images to have a meaningful perceptual comparison. Figure 14 compares markers in a reference image with rendered markers to validate our marker positioning. For the teapot, the average distance between the markers from stereo and the transformed markers from CT is 0.43 mm.

Figure 15 presents pixelwise comparisons of reference images and rendered images. The error images allow us to spot subtle differences not easily noticed in a perceptual comparison, such as the slight misalignments in geometry and highlights. As reference photographs were not captured in HDR, we clamp the renderings correspondingly. This means that areas of strong light intensity, such as highlights and intense caustics, appear black in the error images.

**Fig. 13.** Scene with checkerboard backdrop, lighting, glass teapot, and stand with table cloth observed by two cameras mounted on a 6-axis industrial robot arm.

**Fig. 14.** Markers rendered in blue and added to the reference image to validate marker positions by looking at pixel offsets.

**Fig. 15.** Pixelwise error for three rendering-reference pairs. Error is the $\ell^2$-norm of 32-bit per channel RGB images, visualized using a base 10 logarithmic scale.

**Fig. 17.** Zoom-in of Figs. 16(b) and 16(c) to emphasize the effect of our background deformation.

**Fig. 18.** Zoom-in of Fig. 16(c) and 16(d) to emphasize the effect of our perspective unwrapping of the environment map.

Figure 16 exemplifies the impact on error images of some of our contributions. In Fig. 16(a), we reposition only the glass object in the background scene and apply color correction (Sections 2.F and 2.G). This means that we use Lambertian materials (with bihemispherical reflectances from the measured BRDFs), an orthographic unwrapping model of the environment map, and no chrome reflectance correction or analysis by synthesis optimization. We compare to the reference image in Fig. 16(g), with error images as in Fig. 15. Figure 16(b) shows the impact of using measured BRDFs (Section 2.C), resulting in a more accurate representation of the folds of the cloth in the background scene (top image) and an overall reduction of the error (bottom image). In Fig. 16(c), we add deformation of the background mesh (Section 2.E), which ensures that the background mesh does not poke through the glass surface (see a close-up in Fig. 17). Additionally, we can see how this improves the error on the lid of the bowl, because of refraction of light in the glass. The next step, Fig. 16(d), shows the impact of our modified environment map unwrapping (Section 2.H) against the standard orthographic unwrapping rotated according to our camera parameters. A close-up is available in Fig. 18. Our modified unwrapping provides a better shape and alignment of highlights and caustics. Partially due to the assumption of infinitely distant environment light, some alignment artifacts persist. In Fig. 16(e), we show the effect of correcting for chrome reflectance in our environment map reconstruction. Quantitatively, this changes the distribution of the error (bottom image). On the cloth, the exposure increases, exposing the caustics misalignment. On the backdrop, the error reduces. Interestingly, the structural similarity index (SSIM) improves while the RMSE worsens. Finally, in Fig. 16(f), we use analysis by synthesis to adjust glass absorption. This improves the glass appearance, but it also leads to slight color changes in other parts of the scene due to indirect light paths. Because of this global influence, the analysis by synthesis introduces slightly too much absorption to compensate for the slightly too bright tablecloth.

As an example of how our pipeline can be used to validate existing algorithms, we investigate the case of glass object reconstruction. In Fig. 19, we compare two different reconstruction

**Fig. 16.** Qualitative (top) and quantitative (bottom) step-by-step evaluation of our reassembly techniques. The log error images have the same format as in Fig. 15 and the reference photograph is in the rightmost column (g). In each column, we provide root-mean-squared error and structural similarity index (RMSE/SSIM). Both measures attain their best scores in our final result (f).

**Fig. 19.** Trade-off in mesh reconstruction. If we smooth more, we get less distortion in the refractions, but less precision in the mesh geometry. From left to right: Rendering with smoothing, reference image, rendering without smoothing.

methods with a focus on two parts of the teapot scene. Smooth reconstruction refers to the procedure described in Section 2.D. The other procedure is simply to decimate the reconstructed mesh to 2.5% of the original vertices and apply Taubin smoothing [63]. This removes the high frequencies of the noise, but much noise is still present in the midranges, leading to wobbly refractions. Our method in Section 2.D reduces far more noise, but this is at the cost of greater changes to the overall shape. We note that a refractive object with a simple geometry is very hard to reconstruct automatically if fidelity and almost no noise are both required.

## 5. DISCUSSION

Since our pipeline enables us to compare renderings with photographs, we can identify problems in acquisition, reconstruction, and rendering that would otherwise have been hard to find. Camera calibration issues, for example, reveal themselves as error lines along edges (visible in Fig. 20). Color calibration issues reveal themselves as color shift. Such issues led us to more careful camera calibration procedures and the choice of root-polynomial color correction. Qualitative comparisons revealed artifacts in surface reconstruction, mesh intersections calling for deformation, misplacement of highlights, color shift due to chrome reflectance, and missing absorption in renderings (Figs. 16–19). Quantitative comparisons confirmed

**Fig. 20.** Material transitions: error lines along checker edges and along the boundary between tablecloth and backdrop.

**Fig. 21.** Effect of separating markers from glass (refracted light close to marker) and of not accounting for subsurface scattering (dark areas close to caustics).

improvement due to perspective unwrapping of light probe images and led to analysis by synthesis.

The comparison with reference photographs before and after deformation (Fig. 17) to some extent validates our soft object deformation technique. Further validation would be desirable, but it is difficult to come up with a different experiment. Some kind of soft, durable memory foam with a scannable surface would be required, as the soft object would otherwise change shape again once the hard object is removed. Our validation supports only that the cloth appearance (as observed through glass) is represented more faithfully after deformation.

We found analysis by synthesis useful for estimating parameters with an outset in physics-based initial guesses. The results in Fig. 12 show that we can estimate optical properties for a given material and use them in a different setting (right part of Fig. 1). The precision of the estimation varies with the impact of the property on the overall error, and the estimated parameters may compensate for unrelated errors. In this regard, specific scene configurations could be used to favor estimation of a particular parameter. The most important limitation of our method is that we describe materials as large patches of isotropic BRDFs. In our renderings, this assumption works well for the checkerboard backdrop but not for the cloth, where we have both subsurface scattering effects and probably anisotropy due to the weave structure of the cloth. Figure 21 reveals that the rendered image is too dark in areas surrounding caustics. As seen in the light refracted through the sphere in the vicinity of the marker, our processing of the glass object to separate glass from markers causes some imprecision in the geometry. We believe this mainly influences the shape of the caustic. The bleeding of the caustic to areas that are much darker in the rendered images looks like backscattering from the table beneath the cloth. We refer to this as a kind of subsurface scattering.

Another limitation is seen at the transition between non-connected elements. It is visible in the renderings at the boundary between the cloth and the backdrop (see Fig. 20). The problem derives from the fact that the cloth and the backdrop were too close to each other during dataset acquisition. This resulted in the Poisson mesh reconstruction interpreting them as a continuous object instead of two separate ones. The problems around markers (Fig. 21) are also due to transition of materials. The marker removal and whole closing in the glass surface reconstruction interrupts the original shape of the surface. Furthermore, the markers are glued onto the glass surface, and the glue is not considered in the reconstruction and renderings. The marker glue problem is magnified by the glass refraction.

## 6. CONCLUSION

We have proposed a pipeline for multimodal scene digitization. Our work addresses the entire process from acquisition of the original objects, through reassembly of the digital scene, to accurate modeling of camera and environment. While the pipeline required several non-trivial steps, the benefits are correspondingly great, since we can perform pixelwise comparisons between rendered images and photographs of the corresponding physical scene. This means that we have the means to quantitatively assess the accuracy of an acquired model based on comparison with empirical evidence. We believe this kind of quantitative assessment has not previously been possible for transparent objects. In applications such as cultural heritage preservation and industrial inspection, where the accuracy of a digitization is important, such comparison with empirical evidence is crucial.

To the best of our knowledge, our work is also the first work to quantify the photorealism of a heterogeneous scene requiring multimodal acquisition.

Our dataset is publicly available so that others can test new techniques for the different steps of the pipeline with quantitative feedback based on photorealistic rendering. The fact that one can use off-the-shelf rendering techniques for improving the different steps of a multimodal digitization pipeline is perhaps the most important benefit of our work. An application of the full pipeline is the virtual product placement in Fig. 1. Another important application is the estimation of radiometric properties through analysis by synthesis. The ability to accurately estimate optical properties through computation rather than measurement, which might require specialized equipment, is likely to greatly simplify the digitization of radiometrically complex objects. In this paper, we estimated absorption and refractive indices of transparent objects, but analysis by synthesis could be equally useful for other materials with non-trivial BRDFs. This is another key benefit of our work that we believe is well worth exploring in the future.

## APPENDIX A

Figure 22.

**Fig. 22.** Comparison of renderings and photographs as in Fig. 1 (left), but with more views.

**Funding.** Innovation Fund Denmark (IFD) (3067-00001B, 5163-00001B, 5163-00003B, 75-2014-1).

†These authors contributed equally for this work.

## REFERENCES

1. M. Weinmann and R. Klein, "Advances in geometry and reflectance acquisition (course notes)," in *SIGGRAPH Asia 2015 Courses* (ACM, 2015).
2. P. Debevec, "The light stages and their applications to photoreal digitial actors," in *SIGGRAPH Asia 2012 Technical Briefs* (2012).
3. L. Gomes, O. R. P. Bellon, and L. Silva, "3D reconstruction methods for digital preservation of cultural heritage: a survey," Pattern Recogn. Lett. **50**, 3–14 (2014).
4. L. Zhang, H. Dong, and A. E. Saddik, "From 3D sensing to printing: a survey," ACM Trans. Multimedia Comput. Commun. Appl. **12**, 27 (2016).
5. J. B. Nielsen, E. R. Eiriksson, R. L. Kristensen, J. Wilm, J. R. Frisvad, K. Conradsen, and H. Aanæs, "Quality assurance based on descriptive and parsimonious appearance models," in *Workshop on Material Appearance Modeling (MAM)* (The Eurographics Association, 2015), pp. 21–24.
6. B. T. Phong, "Illumination for computer generated pictures," Commun. ACM **18**, 311–317 (1975).
7. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," SIGGRAPH Comput. Graph. **18**, 213–222 (1984).
8. A. Takagi, H. Takaoka, T. Oshima, and Y. Ogata, "Accurate rendering technique based on colorimetric conception," SIGGRAPH Comput. Graph. **24**, 263–272 (1990).
9. G. W. Meyer, H. E. Rushmeier, M. F. Cohen, D. P. Greenberg, and K. E. Torrance, "An experimental evaluation of computer graphics imagery," ACM Trans. Graph. **5**, 30–50 (1986).
10. H. Rushmeier, G. Ward, C. Piatko, P. Sanders, and B. Rust, "Comparing real and synthetic images: some ideas about metrics," in *Conference: Proceedings of the Eurographics Workshop Rendering Techniques* (Springer, 1995), pp. 82–91.
11. K. F. Karner and M. Prantl, "A concept for evaluating the accuracy of computer generated images," in *Spring Conference on Computer Graphics (SCCG)* (1996).
12. S. N. Pattanaik, J. A. Ferwerda, K. E. Torrance, and D. P. Greenberg, "Validation of global illumination solutions through CCD camera measurements," in *Color Imaging Conference (CIC)* (1997), pp. 250–253.
13. N. L. Jones and C. F. Reinhart, "Parallel multiple-bounce irradiance caching," Comput. Graph. Forum **35**, 57–66 (2016).
14. N. L. Jones and C. F. Reinhart, "Experimental validation of ray tracing as a means of image-based visual discomfort prediction," Build. Environ. **113**, 131–150 (2017).
15. D. P. Greenberg, K. E. Torrance, P. Shirley, J. Arvo, J. A. Ferwerda, S. Pattanaik, E. Lafortune, B. Walter, S.-C. Foo, and B. Trumbore, "A framework for realistic image synthesis," in *SIGGRAPH 97* (ACM/Addison-Wesley, 1997), pp. 477–494.
16. F. Drago and K. Myszkowski, "Validation proposal for global illumination and rendering techniques," Comput. Graph. **25**, 511–518 (2001).
17. C. Ulbricht, A. Wilkie, and W. Purgathofer, "Verification of physically based rendering algorithms," Comput. Graph. Forum **25**, 237–255 (2006).
18. J. Meseth, G. Müller, R. Klein, F. Röder, and M. Arnold, "Verification of rendering quality from measured BTFs," in *Applied Perception in Graphics and Visualization (APGV)* (ACM, 2006), pp. 127–134.
19. A. I. Ruppertsberg and M. Bloj, "Rendering complex scenes for psychophysics using RADIANCE: how accurate can you get?" J. Opt. Soc. Am. A **23**, 759–768 (2006).
20. A. Dal Corso, J. R. Frisvad, T. K. Kjeldsen, and J. A. Bærentzen, "Interactive appearance prediction for cloudy beverages," in *Workshop on Material Appearance Modeling (MAM)* (The Eurographics Association, 2016), pp. 1–4.

21. B. Tunwattanapong, G. Fyffe, P. Graham, J. Busch, X. Yu, A. Ghosh, and P. Debevec, "Acquiring reflectance and shape from continuous spherical harmonic illumination," ACM Trans. Graph. **32**, 109 (2013).
22. T. Nöll, J. Köhler, G. Reis, and D. Stricker, "Fully automatic, omnidirectional acquisition of geometry and appearance in the context of cultural heritage preservation," J. Comput. Cult. Herit. **8**, 2 (2015).
23. H. Wu, Z. Wang, and K. Zhou, "Simultaneous localization and appearance estimation with a consumer RGB-D camera," IEEE Trans. Vis. Comput. Graph. **22**, 2012–2023 (2016).
24. I. Ihrke, K. N. Kutulakos, H. P. A. Lensch, M. Magnor, and W. Heidrich, "Transparent and specular object reconstruction," Comput. Graph. Forum **29**, 2400–2426 (2010).
25. A. Kolb, J. Zhu, and R. Yang, "Sensor fusion," in *Digital Representation of the Real World*, M. A. Magnor, O. Grau, O. Sorkine-Hornung, and C. Theobalt, eds. (CRC Press, 2015), Chap. 9, pp. 133–150.
26. V. Bhateja, H. Patel, A. Krishn, A. Sahu, and A. Lay-Ekuakille, "Multimodal medical image sensor fusion framework using cascade of wavelet and contourlet transform domains," IEEE Sens. J. **15**, 6783–6790 (2015).
27. A. Pamart, O. Guillon, J.-M. Vallet, and L. De Luca, "Toward a multimodal photogrammetric acquisition and processing methodology for monitoring conservation and restoration studies," in *Eurographics Workshop on Graphics and Cultural Heritage* (The Eurographics Association, 2016), pp. 207–210.
28. H. Aanæs and A. B. Dahl, "Accuracy in robot generated image data sets," in *Scandinavian Conference on Image Analysis (SCIA)*, Lecture Notes in Computer Science (Springer, 2015), Vol. **9127**, pp. 472–479.
29. H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, and A. B. Dahl, "Large-scale data for multiple-view stereopsis," Int. J. Comput. Vis. **120**, 153–168 (2016).
30. Z. Zhang, "A flexible new technique for camera calibration," IEEE Trans. Pattern Anal. Mach. Intell. **22**, 1330–1334 (2000).
31. P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *SIGGRAPH 97* (ACM/Addison-Wesley, 1997), pp. 369–378.
32. J. L. Posdamer and M. Altschuler, "Surface measurement by space-encoded projected beam systems," Comput. Graph. Image Process. **18**, 1–17 (1982).
33. J. Geng, "Structured-light 3D surface imaging: a tutorial," Adv. Opt. Photon. **3**, 128–160 (2011).
34. M. Kazhdan and H. Hoppe, "Screened Poisson surface reconstruction," ACM Trans. Graph. **32**, 29 (2013).
35. M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," IEEE Trans. Visualization Comput. Graph. **18**, 914–924 (2012).
36. J. F. Murray-Coleman and A. M. Smith, "The automated measurement of BRDFs and their application to luminaire modeling," J. Illum. Eng. Soc. **19**, 87–99 (1990).
37. J. B. Nielsen, H. W. Jensen, and R. Ramamoorthi, "On optimal, minimal BRDF sampling for reflectance acquisition," ACM Trans. Graph. **34**, 186 (2015).
38. W. Matusik, H. Pfister, M. Brand, and L. McMillan, "A data-driven reflectance model," ACM Trans. Graph. **22**, 759–769 (2003).
39. J. F. Barrett and N. Keat, "Artifacts in CT: recognition and avoidance," RadioGraphics **24**, 1679–1691 (2004).
40. T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of Hermite data," ACM Trans. Graph. **21**, 339–346 (2002).
41. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Commun. ACM **24**, 381–395 (1981).
42. L. Kobbelt, "$\sqrt{3}$-subdivision," in *SIGGRAPH 2000* (ACM/Addison-Wesley, 2000), pp. 103–112.
43. R. L. Cook, "The Reyes image rendering architecture," ACM SIGGRAPH Comput. Graph. **21**, 95–102 (1987).
44. T. Atherton and D. Kerbyson, "Size invariant circle detection," Image Vis. Comput. **17**, 795–803 (1999).
45. P. D. Sampson, "Fitting conic sections to "very scattered" data: an iterative refinement of the Bookstein algorithm," Comput. Graph. Image Process. **18**, 97–108 (1982).
46. J. C. Gower, "Generalized procrustes analysis," Psychometrika **40**, 33–51 (1975).
47. G. D. Finlayson, M. Mackiewicz, and A. Hurlbert, "Color correction using root-polynomial regression," IEEE Trans. Image Process. **24**, 1460–1470 (2015).
48. G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations," Color Res. Appl. **30**, 21–30 (2005).
49. J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. (Springer, 2006).
50. E. Reinhard, G. Ward, S. Pattanaik, P. Debevec, W. Heidrich, and K. Myszkowski, *High Dynamic Range Imaging: Acquisition, Display and Image-Based Lighting*, 2nd ed. (Morgan Kaufmann/Elsevier, 2010).
51. A. Stockman and L. T. Sharpe, "The spectral sensitivities of the middle- and long-wavelength-sensitive cones derived from measurements in observers of known genotype," Vis. Res. **40**, 1711–1737 (2000).
52. J. R. Frisvad, N. J. Christensen, and H. W. Jensen, "Computing the scattering properties of participating media using Lorenz-Mie theory," ACM Trans. Graph. **26**, 60 (2007).
53. C. Ulbricht and A. Wilkie, "A problem with the use of XYZ colour space for photorealistic rendering computations," in *Colour in Graphics, Imaging, and Vision (CGIV)* (2006), pp. 435–437.
54. J. Meng, F. Simon, J. Hanika, and C. Dachsbacher, "Physically meaningful rendering using tristimulus colours," Comput. Graph. Forum **34**, 31–40 (2015).
55. H. S. Fairman, M. H. Brill, and H. Hemmendinger, "How the CIE 1931 color-matching functions were derived from Wright-Guild data," Color Res. Appl. **22**, 11–23 (1997).
56. P. Debevec, "Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *SIGGRAPH 98* (ACM, 1998), pp. 189–198.
57. A. D. Rakić, A. B. Djurišić, J. M. Elazar, and M. L. Majewski, "Optical properties of metallic films for vertical-cavity optoelectronic devices," Appl. Opt. **37**, 5271–5283 (1998).
58. J. T. Kajiya, "The rendering equation," ACM SIGGRAPH Comput. Graph. **20**, 143–150 (1986).
59. M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. (Morgan Kaufmann/Elsevier, 2017).
60. S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "OptiX: a general purpose ray tracing engine," ACM Trans. Graph. **29**, 66 (2010).
61. M. Hejrati and D. Ramanan, "Analysis by synthesis: 3D object recognition by object reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 2449–2456.
62. D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," J. Global Optim. **13**, 455–492 (1998).
63. G. Taubin, "A signal processing approach to fair surface design," in *SIGGRAPH 95* (ACM, 1995), pp. 351–358.

Appendix III

# Interactive Directional Subsurface Scattering and Transport of Emergent Light

**The Visual Computer**
The final publication is available at Springer via http://dx.doi.org/10.1007/s00371-016-1207-2

# Interactive Directional Subsurface Scattering and Transport of Emergent Light

**Alessandro Dal Corso · Jeppe Revall Frisvad · Jesper Mosegaard · J. Andreas Bærentzen**

Received: date / Accepted: date

**Abstract** Existing techniques for interactive rendering of deformable translucent objects can accurately compute diffuse but not directional subsurface scattering effects. It is currently common practice to gain efficiency by storing maps of transmitted irradiance. This is however not efficient if we need to store elements of irradiance from specific directions. To include changes in subsurface scattering due to changes in the direction of the incident light, we instead sample incident radiance and store scattered radiosity. This enables us to accommodate not only the common distance-based analytical models for subsurface scattering but also directional models. In addition, our method enables easy extraction of virtual point lights for transporting emergent light to the rest of the scene. Our method requires neither preprocessing nor texture parameterization of the translucent objects. To build our maps of scattered radiosity, we progressively render the model from different directions using an importance sampling pattern based on the optical properties of the material. We obtain interactive frame rates, our subsurface scattering results are close to ground truth, and our technique is the first to include interactive transport of emergent light from deformable translucent objects.

**Keywords** subsurface scattering · global illumination · interactive rendering · translucent objects · turbid media

A. Dal Corso (✉) · J. R. Frisvad · J. A. Bærentzen
Technical University of Denmark, Kgs. Lyngby, Denmark
E-mail: alcor@dtu.dk

J. Mosegaard
The Alexandra Institute, Aarhus, Denmark

## 1 Introduction

Subsurface scattering of light is a physical phenomenon that occurs in translucent materials. Milk, honey, skin, marble, and candle wax are just a few examples of translucent materials. It is possible to produce the qualitative appearance of translucency using interactive volume rendering techniques [32], but such techniques are not quantitatively accurate. With the advent of analytical models for subsurface scattering [26], it became feasible to build more accurate techniques for interactive rendering of translucent objects. The first technique of this kind [33], and more recent ones that also work for deformable objects (see Section 2), consider diffuse subsurface scattering only. In practice, this means that subsurface scattering is computed by evaluating an integral over the object surface of an analytic dipole model [26] that only depends on the distance between the points of incidence and emergence. Single scattering and other dependencies of the subsurface scattering on the direction of the incident light are neglected. Recent work in offline rendering however shows that the directional effects are not negligible [50, 20, 10, 14].

We present an interactive technique that supports *directional* subsurface scattering without relying on precomputation or a grid for volumetric light propagation. To the best of our knowledge, our method is the first of its kind. Since the method does not rely on texture parameterization, it works for deformable and even procedurally generated geometry.

Due to reciprocity of light transport, we would ideally treat the directions of incident and emergent light equally. This is however too costly for an interactive technique. To achieve interactivity, we need caching of subsurface scattering computations. Existing techniques typically cache transmitted irradiance [25, 33]

diffuse subsurface scattering | transport of emergent light | directional subsurface scattering

**Fig. 1** Deforming translucent candle rendered interactively as with existing techniques (left block), with our transport of emergent light (middle block), and including directional subsurface scattering (right block). Our method is the first to support interactive rendering of the results in the right block (6 frames per second). For this scene, we use 28 scattered radiosity maps, 45 samples per direction, and 80 virtual point lights.

(total incoming light in a surface point) and use a precomputed filter to evaluate the subsurface scattering [33, 5, 29]. These techniques require that the subsurface scattering depends on distance only, whereas we need to use the direction of the incoming light. To cache another quantity, we note that subsurface scattering partly diffuses the light even if the incident light and the scattering are highly directional. Every ray of incoming light gives rise to a (non-diffuse) lobe of emergent light at all surface points. Adding up these lobes, the emergent light is in practice nearly diffuse. We therefore store scattered radiosity (outgoing light) instead of transmitted irradiance. Some of the directional subsurface scattering models also neglect dependency on the direction of emergence but still achieve improved accuracy [50, 20, 14]. Out of these, we can directly use the ones that do not rely on precomputation [20, 14].

In some existing techniques [33, 36, 4, 42], scattered radiosity is stored per vertex. To accommodate more detailed directional effects, we use more detailed maps of the scattered radiosity. We obtain these maps without requiring texture parameterization of the translucent object by rendering the object from multiple views using orthographic cameras. For each of these views, we compute a map of scattered radiosity. We can then efficiently render the translucent object from any view by look-ups into the scattered radiosity maps.

The scattered radiosity maps have two other important advantages. As long as the light source and the object are stationary, we can blend scattered radiosity maps and thereby progressively improve the rendering. Moreover, we can compute the transport of emergent light to the surrounding scene [40, 42]. To include these light paths while keeping the translucent object deformable, we generate a distribution of virtual point lights on the surface of the translucent object and set their intensity according to the scattered radiosity. These virtual point lights enable us to render the transported light using a many-light method [8]. Since we include transport of emergent light, our method is very useful for interactive rendering of scenes with the light source hidden behind a translucent object. Indirect illumination of a scene by light that has scattered through candle wax is one use case (Fig. 1). Another interesting example is light scattering through translucent lamp shades or light bulbs. To the best of our knowledge, we present the first interactive technique for transport of light emerging from deformable translucent objects.

## 2 Related Work

One way to obtain interactive subsurface scattering is by means of precomputation. Several early techniques rely on precomputed scattering factors that enable subsurface light transport between surface patches or from patch to vertex [33, 23, 4, 24]. These factors resemble form factors in radiosity algorithms and specify transport of transmitted irradiance to scattered radiosity. An extension of these radiosity-like techniques is to include transport of emergent light [42]. Other work is based on precomputed radiance transfer [43, 47, 49, 46], and some of this includes directional effects such as single scattering in the rendered result [43, 47, 49]. Another approach is to precompute a grid that can be used with a fast diffusion computation to render subsurface scattering in real-time [45, 48]. As opposed to our work,

all these precomputation-based methods cannot interactively render deformable translucent objects.

Some finite element methods are fast enough to enable interactive rendering of deformable translucent objects [36,34]. However, as these methods rely on diffuse incoming light (transmitted irradiance) and a multi-resolution mesh (triangular or tetrahedral), they are not easily adapted for directional subsurface scattering and would typically require some mesh preprocessing.

Volume rendering techniques can quite convincingly produce the qualitative appearance of translucency at high frame rates [32,3,2,13]. While such methods are inspired by the volume rendering equation [30], they only provide a rather rough approximation of its solution. In addition, the accuracy of the subsurface scattering is limited by the resolution of the volume or the grid. Some of the more advanced methods [3,13] also propagate light using low-order spherical harmonics that effectively diffuse the subsurface scattering contribution. Other techniques, which are based on separable filtering and a depth map, also achieve real-time subsurface scattering by aiming at the qualitative appearance and sacrificing quantitative accuracy [19,18].

Fast filtering techniques can be constructed so that they approximate diffuse subsurface scattering more accurately [12,5,21,29]. The filtering is done in texture space and thus requires texture parametrization of the object surface. To avoid texture space problems, similar filtering techniques are available for light space [9] and screen space [35,27,28,29]. The performance of all these filtering techniques, however, depends heavily on the assumption that the subsurface scattering is diffuse so that the convolution kernel is only a function of the distance between the points of incidence and emergence. Our work uses light space sampling [9], but removes the assumption that subsurface scattering is diffuse. If we were to remove this assumption from texture or screen space filtering techniques and adapt them for directional subsurface scattering, they would become texture space or screen space variations of the technique that we propose. The former variation would require texture parametrization of the object surface, the latter would be view-dependent.

Another interesting approach to interactive rendering of deformable translucent objects is based on splatting [41,6]. In this approach, surface points seen from the light source are splatted as screen-aligned quads. These splats contribute according to the subsurface scattering model where they overlap surface points in the geometry buffer of the camera. On first inspection, this seems an ideal approach for interactive rendering of directional subsurface scattering. However, the directional model requires larger splats as it varies not only with distance, and it is more expensive to evaluate as tabulation is impractical. We therefore found the splatting approach too expensive.

**Fig. 2** BSSRDF configuration on an object surface $A$. The diagram illustrates the notation we use: bold font as in $\boldsymbol{x}_o$ denotes a point, while arrow overline as in $\vec{\omega}_i$ denotes a normalized direction vector.

## 3 Method

We render translucent objects using a *bidirectional scattering-surface reflectance distribution function* (BSSRDF). In most BSSRDFs, a translucent material is defined by the following spectral optical properties: refractive index $\eta$, absorption coefficient $\sigma_a$, scattering coefficient $\sigma_s$, and asymmetry parameter $g$. As is common in graphics, we use trichromatic optical properties (rgb). In addition, the BSSRDF depends on the position $\boldsymbol{x}_i$ and the direction $\vec{\omega}_i$ of the incident light as well as the position $\boldsymbol{x}_o$ and the direction $\vec{\omega}_o$ of the emergent light. The configuration is illustrated in Fig. 2. When rendering a translucent object, we obtain the outgoing radiance $L_o$ by evaluating the following integral over all $\boldsymbol{x}_i$ in the surface area $A$ and over all $\vec{\omega}_i$ in the hemisphere around the surface normal $\vec{n}_i$ at $\boldsymbol{x}_i$ [26]:

$$L_o(\boldsymbol{x}_o,\vec{\omega}_o) = L_e(\boldsymbol{x}_o,\vec{\omega}_o) + \int_A\int_{2\pi} S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o)L_i(\boldsymbol{x}_i,\vec{\omega}_i)\cos\theta_i\,\mathrm{d}\omega_i\,\mathrm{d}A_i\,, \quad (1)$$

where $\cos\theta_i = \vec{\omega}_i\cdot\vec{n}_i$, $L_i$ is incident radiance, $L_e$ is emitted radiance, and $S$ is a BSSRDF. Disregarding surface reflection, as this can be incorporated using well-known techniques, the analytical BSSRDF can be written in the form:

$$S(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o,\vec{\omega}_o) = F_t(\vec{\omega}_o)(S_d(\boldsymbol{x}_i,\vec{\omega}_i;\boldsymbol{x}_o)+S^*)F_t(\vec{\omega}_i)\,, \quad (2)$$

where $F_t$ is Fresnel transmittance, $S_d$ is the diffusive part, which is typically modeled by a dipole, and $S^*$ (dependencies omitted) is the remaining light transport, that is, the part not included with $S_d$.

As in other interactive subsurface scattering techniques that are not based on precomputation, we now

assume that $S^*$ is insignificant. For most BSSRDF models [26,11,20], this means that single scattering is excluded entirely. However, if we use the directional dipole model [14], most single scattering is included with $S_d$. We therefore get a more accurate result with this model as the neglected $S^*$ contains a significantly smaller part of the scattered light.

In existing interactive techniques, it is common practice to move the BSSRDF outside the integration over directions of incidence $\vec{\omega}_i$ (in Equation 1) and define transmitted irradiance by [33,9,36,35,5,41,6,34,29]

$$E(\boldsymbol{x}_i) = \int_{2\pi} L_i(\boldsymbol{x}_i, \vec{\omega}_i) F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \,. \tag{3}$$

We would however like to support BSSRDFs that include directional effects [20,14]. Since such BSSRDFs depend on $\vec{\omega}_i$, we cannot perform this separation, but we can define scattered radiosity by

$$B(\boldsymbol{x}_o) = \pi \int_A \int_{2\pi} S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) L_i(\boldsymbol{x}_i, \vec{\omega}_i) \\ F_t(\vec{\omega}_i) \cos\theta_i \, \mathrm{d}\omega_i \, \mathrm{d}A_i \,. \tag{4}$$

This is an important quantity as the rendering equation (1) becomes

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = L_e(\boldsymbol{x}_o, \vec{\omega}_o) + \frac{1}{\pi} F_t(\vec{\omega}_o) B(\boldsymbol{x}_o) \,, \tag{5}$$

which enables view-independent rendering of translucent objects if we store scattered radiosity $B$. We note that $L_o$ is not fully view independent because of the Fresnel term $F_t$, but this is an inexpensive term that we can evaluate per pixel per frame at very little cost.

For simplicity, our initial assumption is of a scene consisting of a single object illuminated by a single directional light. In Section 3.3, we extend to point lights, and in Section 4, we show an example of using multiple lights. For surface points lit by a directional light with radiance $L_\ell$ and direction $\vec{\omega}_\ell$, we have

$$L_i(\boldsymbol{x}_i, \vec{\omega}_i) = L_\ell \, V(\boldsymbol{x}_i, -\vec{\omega}_\ell) \, \delta(\vec{\omega}_i + \vec{\omega}_\ell) \,, \tag{6}$$

where $V$ is visibility and $\delta$ is a Dirac delta function that makes the inner integral disappear, yielding

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_{A_{\mathrm{lit}}} S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, \mathrm{d}A_i \,, \tag{7}$$

where $\cos\theta_\ell = -\vec{\omega}_\ell \cdot \vec{n}_i$ and $A_{\mathrm{lit}}$ is the directly lit area of the surface (for unlit areas $L_i = V = 0$). Since we only need to integrate over the directly lit part of the surface area, we perform the integration in a geometry buffer (G-buffer) rendered from the point of view of the light source (a translucent shadow map [9]). Since we have a directional light, our G-buffer is an orthographic projection of the scene into the light's view plane, which has $\vec{\omega}_\ell$ as its normal.

In order to distribute samples in the G-buffer according to a distance $r$ and an angle $\alpha$, we assume a planar surface normal to the light direction and rewrite the integral in polar coordinates with origin $\boldsymbol{x}_o$:

$$B(\boldsymbol{x}_o) = \pi L_\ell \int_0^{2\pi} \int_0^{\infty} S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) \\ F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r \, \mathrm{d}r \, \mathrm{d}\alpha \,, \tag{8}$$

where $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and $\alpha$ is the angle between $\boldsymbol{x}_o - \boldsymbol{x}_i$ and the first basis vector of the light's view plane. This assumption is clearly often violated, but it is commonly used in derivation of BSSRDF models [26,11].

We evaluate the integral in Equation 8 by Monte Carlo integration. Our estimator for scattered radiosity is

$$B_N(\boldsymbol{x}_o) = \frac{\pi L_\ell}{N} \sum_{j=1}^{N} \frac{S_d(\boldsymbol{x}_i, -\vec{\omega}_\ell; \boldsymbol{x}_o) F_t(-\vec{\omega}_\ell) \cos\theta_\ell \, r_j}{p(r_j, \alpha_j)} \,, \tag{9}$$

where $p(r, \alpha)$ is the joint probability density function from which we draw the sample pairs $(r_j, \alpha_j)$. Starting from $\boldsymbol{x}_o$ transformed to the texture space of the light's camera, each sample pair corresponds to a texture space offset for looking up $\boldsymbol{x}_i$ and $\vec{n}_i$ in the light's G-buffer.

### 3.1 Sampling Distribution

BSSRDFs decay exponentially with the distance $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$. In particular, the asymptotic exponential falloff of the standard and directional dipoles [26,14] is $\exp(-\sigma_{\mathrm{tr}} d)$, where $d \to r$ for $r \to \infty$ and $\sigma_{\mathrm{tr}}$ is the *effective transport coefficient* defined by

$$\sigma_{\mathrm{tr}} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)} \,. \tag{10}$$

It is therefore highly beneficial to importance sample according to this exponential decay. We do importance sampling by choosing

$$p_{\exp}(r, \alpha) = p(r)p(\alpha) = \sigma_{\mathrm{tr}} e^{-\sigma_{\mathrm{tr}} r} \frac{1}{2\pi} \,, \tag{11}$$

which is easily sampled by

$$(r_j, \alpha_j) = \left( \frac{-\log \xi_1}{\sigma_{\mathrm{tr}}}, 2\pi\xi_2 \right) . \tag{12}$$

The symbols $\xi_1, \xi_2 \in [0, 1]$ denote canonical uniform random variables, which we obtain on the fly using a linear congruential pseudorandom number generator.

It is important to note that the effective transport coefficient $\sigma_{\mathrm{tr}}$ is different for different color bands. As a consequence, we use a separate set of position samples for each color band. In this way, we avoid color shifts, especially for materials with very different scattering coefficients in the different color bands (ketchup, for example).

**Fig. 3** Our three-step multipass technique for interactive rendering of directional subsurface scattering in deformable translucent objects. The scattered radiosity maps enable view-independence and transport of emergent light.

### 3.2 Rendering Technique

The diffusive part of the standard dipole BSSRDF depends only on $r = \|\boldsymbol{x}_o - \boldsymbol{x}_i\|$ and is therefore easily tabulated and used at runtime at nearly no expense. In directional subsurface scattering, on the other hand, the diffusive part of the BSSRDF depends on both $\boldsymbol{x}_o$, $\vec{n}_o$, $\boldsymbol{x}_i$, $\vec{n}_i$, and $\vec{\omega}_i$. This means that it is impractical to tabulate it and thus expensive to evaluate it. To limit the number of times that we need to evaluate the BSSRDF at runtime, we chose to exploit the opportunity to have view-independence by storing scattered radiosity in maps. In fact, as we noted in Equation 5, the scattered radiosity does not depend on the view direction $\vec{\omega}_o$. With view-independence, it is convenient to also make the update of the scattered radiosity maps progressive. By doing so, the rendered result improves over time if we are only moving the camera. Our technique is easily made progressive by adding more samples for each frame. This means that we have two render modes: (a) converged translucency with real-time fly-through and (b) fully flexible translucency rendered at interactive frame rates.

Our rendering technique is based on the rasterization pipeline of the graphics processing unit (GPU). In fully flexible mode, we use the three-step multipass algorithm illustrated in Fig. 3. In the first step, we create a G-buffer for each light source. In the second step, we compute scattered radiosity maps using these light G-buffers. In the third step, we sample the scattered radiosity maps and combine the look-ups. If nothing changed except the camera position, we also accumulate radiosity map results with the ones from the previous frames. When convergence is reached, we switch to converged mode and perform the third step only. In the following, we provide the details of the three steps.

In the first step, as in translucent shadow mapping [9], we render a G-buffer from the point of view of the light. For each pixel, we store positions and normals, as well as a material index (for global illumination purposes, Section 3.4). Each directional light has an orthographic camera and an associated G-buffer stored in a layered 2D texture. We compute all the light G-buffers in a single rendering pass, where each triangle is fed to each layer of a 2D layered texture in a geometry shader.

In the second step, we render the translucent object from $K$ directions using orthographic cameras. The number of directions is chosen so that the surface of the model is covered well. We place the cameras randomly on the bounding sphere of the object using a quasi-random Halton sequence [22]. We then configure the cameras to look at the center of the bounding sphere with a frustum that encapsulates the sphere. Also in this step, we use layered rendering in order to efficiently render scattered radiosity into the different maps in a single pass. For each fragment of the translucent object observed by an orthographic camera, we compute the scattered radiosity by generating $N$ samples per color band on-the-go (Equation 12), looking up into the light G-buffers with those samples to get $\boldsymbol{x}_i$ and $\vec{n}_i$, and using those to evaluate Equation 9. To avoid pattern repetition artifacts, we choose a seed for the random points using the pixel index in the scattered radiosity map as well as the current map and frame numbers.

To progressively update the scattered radiosity maps, we first perform a depth-only pass and then we render the model with writing into the depth buffer disabled. During the second step of the algorithm (except when the light condition is changing or the object is deforming), blending is enabled to allow accumulation in the scattered radiosity maps. We also generate mipmaps for the scattered radiosity maps so that we have the oppor-

**Algorithm 1:** Estimating the scattered radiosity in $\boldsymbol{x}_o$ using $K$ maps (step 3 of Fig. 3). Each map has a direction $\vec{d}_k$ and a world-to-texture conversion matrix $\boldsymbol{P}_k$. The variable $F$ counts the number of accumulated frames, which is needed to average the blending in step 2 of Fig. 3.

**Data:** $\boldsymbol{x}_o$, $\epsilon_{bias}$, $\epsilon_{comb}$, $F$, $K$
**Result:** $B$
$n = 0$
**color** $= (0, 0, 0)$
**for** $k \in [0, K)$ **do**
  $\cos\theta = \text{clamp}(\vec{n}_o \cdot \vec{d}_k, 0, 1)$
  $\bar{\boldsymbol{x}}_o = \boldsymbol{x}_o - \epsilon_{\text{comb}}(\vec{n}_o - \cos\theta\ \vec{d}_k)$
  $\bar{\boldsymbol{x}}_{o,\text{tex}} = \boldsymbol{P}_k\ \bar{\boldsymbol{x}}_o$
  $v = \texttt{multisampleVisibilityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}}, \epsilon_{\text{bias}})$
  **color** $=$ **color** $+ v \cdot \texttt{sampleRadiosityMap}_k(\bar{\boldsymbol{x}}_{o,\text{tex}}, \epsilon_{\text{bias}})$
  $n = n + v$
**end**
$B = \frac{\textbf{color}}{Fn}$

tunity to apply a cheap high-pass filter that smoothes high frequency noise.

In the third and final pass, we sample the scattered radiosity maps for each fragment of the translucent object observed by the actual camera. This process is described in the pseudo-code in Algorithm 1. We average the contributions from the various directions with the visibility of the point as a binary weight. In the third step of Fig. 3, the green and the red dots represent the visible and not visible contributions from the point $\boldsymbol{x}_o$, respectively. Storing depth with the scattered radiosity maps, we use shadow mapping to obtain a visibility function. To avoid artifacts, we choose a constant shadow bias $\epsilon_{bias}$ for the visibility function. Moreover, to avoid errors when sampling close to the borders of a scattered radiosity map, we multi-sample the shadow map and introduce an additional bias $\epsilon_{comb}$ that translates the sample position towards the negative normal direction $-\vec{n}_o$. After composition of the scattered radiosity $B$, we obtain outgoing radiance from Equation 5 and perform tone mapping to finalize the result.

Considering the procedure described in this section, we can get a better understanding of the parameter $N$. The total number of Monte Carlo samples used for computing the outgoing radiance ($L_o$) in a surface point observed by the camera is $3N$ times $K$ times the number of frames used for progressive updates. From the point of view of a surface point, $N$ can thus be thought of as the number of samples per frame per map direction per color band.

**Fig. 4** Effect of stereographic correction when a translucent object surrounds a point light. With planar sampling (a), we look up into the light's cube map G-buffer using $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. With stereographic correction (b), we use $\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell$ instead. The insets (a and b) show how the correction improves the final result (torus, potato material).

### 3.3 Point Lighting

A point light at some distance from the translucent object works much in the same way as a directional light. The light's camera simply uses perspective instead of orthographic projection and intensity falls off with the distance squared. One particularly important application of our work is however simulation of the light coming through candles, candleholders, and lamp shades (Fig. 1, for example). In these cases, the point light is surrounded by the translucent object and we then use omnidirectional shadow mapping [15] with a cube map G-buffer for the light.

With a cube map captured for a point light at $\boldsymbol{x}_\ell$, one would first get a sampled point $\boldsymbol{x}_j$ by using $(r_j, \alpha_j)$ to offset $\boldsymbol{x}_o$ in its tangent plane. A look-up into the cube map with $\boldsymbol{x}_j - \boldsymbol{x}_\ell$ would then provide the sampled $\boldsymbol{x}_i$ and $\vec{n}_i$. However, when observing a translucent object surrounding the light source, this planar sampling of the light's G-buffer is no longer a good approximation. To have a better approximation that enables sampling of the entire cube map for each $\boldsymbol{x}_o$ (instead of only a hemisphere), we use an inverse stereoscopic projection. With this stereoscopic correction, the direction used for look-up into the cube map becomes

$$\boldsymbol{x}_{\text{stereo}} - \boldsymbol{x}_\ell = (\boldsymbol{x}_\ell - \boldsymbol{x}_o) - 2\left[(\boldsymbol{x}_\ell - \boldsymbol{x}_o) \cdot \vec{\ell}\right]\vec{\ell}, \qquad (13)$$

**Fig. 5** Transport of emergent light from a translucent object (blue) to a diffuse object (red). We distribute VPLs (gray dots) on the outer surface of the translucent object, and use them to indirectly illuminate the remaining scene.

where

$$\vec{\ell} = \frac{(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)}{\|(\boldsymbol{x}_j - \boldsymbol{x}_\ell) - (\boldsymbol{x}_\ell - \boldsymbol{x}_o)\|}\,, \tag{14}$$

as illustrated in Fig. 4. The top right image (a) in Fig. 4 is an example of the sampling noise we get if we use $\boldsymbol{x}_j - \boldsymbol{x}_\ell$. The middle right image (b) shows how the stereoscopic correction betters this problem.

### 3.4 Transport of Emergent Light

We further extend our method to account for transport of emergent light using virtual point lights (VPLs) [31]. We distribute a set of $N_{\text{vpl}}$ points on the surface of the translucent object. Then, for each observed point $\boldsymbol{x}_o$, we add the contribution from all VPLs using

$$L_o(\boldsymbol{x}_o, \vec{\omega}_o) = \sum_{v=1}^{N_{\text{vpl}}} f_r(\boldsymbol{x}_o, -\vec{\omega}_v, \vec{\omega}_o) \\ G_b(\boldsymbol{x}_o, \boldsymbol{x}_v) V(\boldsymbol{x}_o, \boldsymbol{x}_v) I_v \tag{15}$$

with VPL intensity

$$I_v = \frac{1}{\pi} F_t(\omega_v) B(\boldsymbol{x}_v) A / N_{\text{vpl}}\,, \tag{16}$$

where $A$ is the surface area across which the VPLs were distributed, $G_b$ is the standard bounded geometry term [8], and $B$ is obtained from the scattered radiosity maps using Algorithm 1.

As in the previous section, we now take special steps to accommodate our key use case of a point light surrounded by a translucent material. Our approach is illustrated in Fig. 5. In this particular case, the scene illuminated by emergent light will most commonly be shadowed from surface points of the translucent object that are directly lit (as the source is surrounded). We therefore approximate the visibility term $V$ by distributing VPLs on backlit surfaces only. With this distribution of VPLs, we use the area of the bounding volume of the translucent object as an approximation of $A$. This is computed for each frame on the CPU.

Unfortunately, for a deformable object and a relatively small set of VPLs, the method is prone to flickering unless we ensure that the VPL positions are stable over time. Our solution is to render the outermost surface of the translucent object to a cube map whose center $\boldsymbol{c}$ coincides with the object's bounding box center. Each pixel in the cube map now contains the coordinates of a point on the surface of the translucent object. By sampling the cube map at a constant set of random directions, we obtain a stable set of surface positions that we use as VPL locations (Fig. 5).

## 4 Results

The implementation of our method interactively renders directional subsurface scattering in deformable objects and requires no preprocessing nor texture parameterization of the object surface. We use the diffusive part of the directional dipole [14] as $S_d$ or the photon beam diffusion model [20] when evaluating Equation 9. The directional dipole is significantly faster, so we use this one unless noted otherwise. We define the translucent objects in our scenes using measured optical properties from different sources [26,37,17].

To validate our results, we compare with Monte Carlo ray tracing implemented on the GPU using OptiX [38]. In this reference method, we render directional subsurface scattering using the progressive direct Monte Carlo integration technique described by Frisvad et al. [14]. As prescribed, we use a Russian roulette based on the asymptotic exponential falloff of the model to accept or reject samples. However, we do not equidistribute the samples using a dart throwing technique as a more brute force uniform sampling of the object surface is more well-suited for a GPU ray tracer. This implementation gave us a ground truth for comparison both in terms of quality and performance. However, when comparing performance, one should keep in mind that unlike our method the reference method is view dependent.

In all the following examples, performance is at interactive rates. If nothing changes except the camera, our method will converge over a number of frames and then run in real-time. The implementation switches back to interactive rates when something other than the camera changes. By 'interactive' we mean a rendering time below 166 milliseconds per frame (6 frames per second, fps), as specified by Akenine-Möller et al. [1]. All the tests were performed on an NVIDIA GeForce GTX 780 Ti graphics card (2880 cores). Unless otherwise indi-

**Fig. 6** Comparison of our method (rows 1 and 2) with the reference method (row 3) and diffuse subsurface scattering (row 4) for different materials. Row 1 is our results for a single frame at 6 fps, while row 2 is our view-independent result after convergence. All results use 31 maps.

**Fig. 7** Equal time comparison (left column) of our method with the reference method and qualitative comparison with diffuse subsurface scattering (upper right) and the converged reference solution (lower right). The scene is lit by a point light in a white grapefruit candle holder.

cated, our results use a $512 \times 512$ frame resolution for both radiosity and light maps.

Fig. 6 allows a visual comparison with ground truth (results obtained with the reference method). We chose one highly scattering material with isotropic phase function ($g = 0$), namely marble, and two forward scattering materials ($g > 0$), namely white grapefruit juice and strawberry shampoo. At convergence (second row), our method compares favorably to the directional dipole reference (third row). Our method improves the details

**Fig. 8** Zoom-ins and differences from Figures 6 and 7. Root-mean-squared error of the color bands $\sqrt{\Delta r^2 + \Delta g^2 + \Delta b^2}$ is used as error metric in the difference images.

of the subsurface scattering when compared with diffuse subsurface scattering, that is, the standard dipole [26] (fourth row), especially for white grapefruit juice and strawberry shampoo. We also show the results of our method after one frame rendered at interactive frame rates (first row). These results are similar to our converged solution except that there is a slight bit of sampling noise, which we reduce using mipmap filtering.

Fig. 7 compares the transport of emergent light obtained with our method to that obtained with the reference method. While the 200 VPLs used here do not provide a highly accurate result, they do provide something better than a constant ambient term. At 6 fps, our solution is similar to the reference and converges very quickly to a better result, while the OptiX solution has both high-frequency and low frequency noise, is view dependent, and converges very slowly.

Fig. 8 provides zoom-ins and difference images from Figs. 6 and 7. Our results in general seem to be missing a part of the light transport. As revealed by the difference images, the missing contribution is due to undersampling of the surface at grazing incidence and missing interreflections. This undersampling is the reason why

our method ray tracer reference

**Fig. 9** Stanford bunny with marble material at different scales (from top to bottom the scale is: 0.01, 0.1, and 1 meter). The left and middle columns show equal time results for our method and the ray tracer (1 frame at 6 fps). The right column shows the ray traced results after convergence. Here we use 16 maps and a $1024 \times 1024$ light map.

Mertens et al. [35] chose to sample in screen space instead of light space. However, sampling in screen space has other problems, as not all samples are lit. When considering transport of emergent light, the zoom-ins and difference images show missing shadows and inaccuracies due to the small number of VPLs. However, as graphics hardware improves, we will be able to use more VPLs and one of several fast VPL visibility techniques [8] to get better accuracy while retaining interactive frame rates.

In Fig. 9, we compare the quality reached by our solution with the quality reached by the ray traced solution in equal time. We perform this comparison for a marble bunny at three different scales. Generally, our method has a uniform behavior for different scales. For materials that are not optically thin (not at low scale), our method converges faster. The highly scattering materials (mid and high scale) are the more important cases to render well, as these are inside the range of materials for which the analytic subsurface scattering models are valid. At high scales, scattering effects become more localized, so our method is better at capturing the effect than the ray traced solution. At low scales, fewer G-buffer samples hit the object, which leads to a more noisy result with our solution.

**Fig. 10** Rendering with our method and a dynamically generated 3D surface ('blob') and transport of emergent light for three materials. The blob renders at 6 fps with 50 VPLs and 1500 samples per map in 6 maps. Materials from left to right: white grapefruit juice, soy milk, and glycerine soap.

In order to test the method using dynamically generated geometry, we created an implicit 3D surface [44] as the sum, $\Phi_t = \Sigma_i \phi_{i,t}(\vec{p})$, of 4 blobs,

$$\phi_{i,t}(\vec{p}) = \exp(-\sigma \|\vec{p} - \vec{p}_i(t)\|^2)\,,$$

where the position of each blob, $\vec{p}_i(t)$, is a periodic function. Since the periods are different, the period of the aggregate implicit $\Phi_t$ is potentially very large, and precomputation of the light transport inside the object would not be practical. Our method however applies, as it does not rely on precomputation, but we do need to rasterize the object. To do this, we compute a triangle mesh for an isosurface of $\Phi_t$ using dual contouring [16] implemented in a geometry shader. This is done in a pre-pass to each frame where the geometry shader evaluates $\Phi_t$ and its gradient directly based on the current time. The output triangle strips are streamed back to a vertex buffer object using transform feedback. Fig. 10 presents a rendered blob using different materials.

To justify our claimed need for scattered radiosity maps, we compare our method with an implementation without caching of subsurface scattering computations (as in translucent shadow mapping [9]). Note that this approach as opposed to ours is view dependent and pixel bound, and that unobserved VPLs would be more expensive to evaluate. Fig. 11 compares performance without considering view dependency and VPLs. Caching of scattered radiosity in maps is more efficient as soon as the translucent object occupies more than 5.8% of a $1024 \times 1024$ image.

The candle scene in Fig. 1 demonstrates the usefulness of our method. We scaled the optical properties of glycerine soap to approximate the scattering properties of candle wax. Our method creates a soft 'caustic' on the ground with varying intensity depending on the shape of the candle model. We thus enable a more realistic lighting of the scene than is obtainable with existing interactive techniques.

**Fig. 11** Chocolate milk blob occupying different percentages of the image (noted at the top). We compare our method (ours) with a view-dependent, caching-free implementation (no cache, meaning no scattered radiosity maps). We use 1000 samples per map in 10 maps when caching, per pixel when not caching. Equal frame rates (12 fps) occur when occupancy is 5.8% of the image.

**Fig. 12** Timing breakdowns for some of our renderings. Initialization times were negligible and were thus included with step 1 of Fig. 3. The evaluation of the BSSRDF and the VPLs (when present) dominate the rendering times.

To provide a performance breakdown of our technique, Fig. 12 lists render times dedicated to the different steps of our algorithm in our various results. BSSRDF evaluation (step 2 of Fig. 3) dominates all the timings, with the exception of Figs. 1 and 7, where the transport of emergent light dominates. Fig. 13 provides timings and coverage improvement of a bunny rendering with increasing $K$. The first seven directions cover most of the surface, while the remaining directions are necessary to cover small holes in the shading.

**Fig. 13** Converged renderings of a potato bunny ($N = 30$) and timings for increasing number of scattered radiosity maps $K$. We list $K$ followed by rendering time in milliseconds (ms) for each result. The first 7 maps cover most of the surface, while the following 4 cover small details (the small area just to the left of the bunny's hind leg, for example).

**Fig. 14** Converged results with scenes and parameters as in other figures, but this time rendered using the photon beam diffusion model [20]. For each rendering, we provide the factor that this model is slower than if we use the directional dipole.

To underline the versatility of our approach, Fig. 14 has a set of results rendered using the photon beam diffusion model [20]. The weak singularities in this model lead to fireflies (overly bright pixels) with our sampling approach. We avoid this problem by clamping the distance $d_r$ to a minimum of $0.25/(\sigma_a + \sigma_t)$ when it is used in a denominator. Factors that photon beam diffusion is slower than the directional dipole are included in the figure. These factors double if we use a graphics card with 512 cores (GTX 580) instead of 2880 cores.

Finally, Fig. 15 presents results with multiple directional lights. To approximate an environment light, we sample a number of representative directional light sources from the environment map using the method described by Pharr and Humphreys [39]. Contributions from all the directional lights are cached in the same scattered radiosity maps. In this example, we add specularly reflected light by looking up into the environment map using the direction of the reflected ray and multiplying by Fresnel reflectance.

## 5 Discussion

The resolution of a light's G-buffer (a light map) should be chosen carefully. If the range of the scattering ef-

**Fig. 15** Stanford Bunny illuminated by an environment map. The map was importance sampled and converted to eight different directional lights. Potato material, 16 maps.

fects (roughly $1/\sigma_{\mathrm{tr}}$) is smaller than the size of one pixel in the light map, the contributions from the directional dipole tend to cluster and form 'pearling' artifacts. A possible solution would be a variation of cascaded shadow maps [51] to provide a higher resolution light map when needed. Generally, a light map of $512 \times 512$ pixels is an acceptable size that can be brought to $1024 \times 1024$ in problematic cases.

User parameters of our method include the resolutions of the light map and the scattered radiosity maps, the two biases $\epsilon_{comb}$ and $\epsilon_{bias}$, the number of samples $N$, the number of scattered radiosity maps $K$, and the number of VPLs $N_{\mathrm{vpl}}$. We now provide some guidelines for setting parameters. The size of the light map was already discussed in the previous paragraph. For the scattered radiosity maps, a size of $512 \times 512$ is fine for most application, and $K = 16$ directions generally provide enough coverage for simple models (the dragon, with its complicated geometry, required $K = 31$ directions). Performance scales linearly with $K$ (Fig. 13), as we spend most of the time evaluating the BSSRDF (Fig. 12). The two biases $\epsilon_{comb}$ and $\epsilon_{bias}$ need to be tweaked manually. The numbers $N$ and $N_{\mathrm{vpl}}$ are usually set manually to get the desired performance once the other parameters have been settled.

For most of our results, we choose the directions $\vec{d}_k$ of the scattered radiosity maps automatically. This works well for objects that are roughly convex, but for more oddly shaped concave objects some part may be left uncovered. Tearing artifacts caused by insufficient coverage appear in the mouth of the dragon in Fig. 6 and in the supplementary video. Fig. 13 also illustrates the problem, and shows that increasing the number of directions or manually choosing them can often ease this problem.

The memory consumption of our technique is comparable to that of the texture space filtering techniques [12,5,21,29]. As such, the maps and buffers that we use easily fit in the memory of modern GPUs. We surprisingly use more memory than the volumetric techniques [32,3,2,13]. The reason is that they make do with very low resolution volumes ($32^3$ or $64^3$). It is however important to note that the added directionality and quality of details that we achieve cannot be achieved with such low resolution volumes. High resolution volumes would be needed with these techniques, which would lead to performance and memory issues.

Since we cache scattered radiosity, we cannot directly use a BSSRDF that fully depends on the direction of emergence $\vec{\omega}_o$ (the dual-beam model [10], for example). For such a BSSRDF, we would have to rely on the assumption that the emergent radiance integrates to a nearly diffuse distribution. We would then carry out a cosine-weighted integral over $\vec{\omega}_o$ when computing the scattered radiosity maps and otherwise use the same method. On the other hand, our concept of caching scattered radiosity instead of transmitted irradiance might be of interest in offline rendering techniques such as multiresolution radiosity caching [7]. This would enable use of directional subsurface scattering and inexpensive transport of emergent light in a movie production rendering solution.

## 6 Conclusion

We have presented a novel technique for interactive rendering of directional subsurface scattering. The method is view independent and applicable to deformable 3D models without requiring a texture parameterization of the object surface. While our method takes the direction of incident light into account, it also relies on the assumption that emergent light is not directional. This enables us to cache emergent light in so-called scattered radiosity maps. These maps enable us to control the output quality, to render progressively, and to illuminate the scene with light that has scattered through a translucent object.

**Acknowledgements** We would like to thank Christian Esbo Agergaard, Technical Director, Sunday Studios for the melting candle model. The Stanford Bunny and the Stanford Dragon models are courtesy of the Stanford University Computer Graphics Laboratory (http://graphics.stanford.edu/data/3Dscanrep/). The HDR environment map in Fig. 15 is courtesy of Tobias Grønbeck Andersen.

## References

1. Akenine-Möller, T., Haines, E., Hoffman, N.: Real-Time Rendering, 3rd edn. A K Peters (2008)
2. Bernabei, D., Hakke-Patil, A., Banterle, F., Benedetto, M.D., Ganovelli, F., Pattanaik, S., Scopigno, R.: A parallel architecture for interactively rendering scattering and

refraction effects. IEEE Computer Graphics and Applications **32**(2), 34–43 (2012)
3. Børlum, J., Christensen, B.B., Kjeldsen, T.K., Mikkelsen, P.T., Noe, K.Ø., Rimestad, J., Mosegaard, J.: SSLPV: Subsurface light propagation volumes. In: Proceedings of ACM SIGGRAPH Symposium on High Performance Graphics (HPG '11), pp. 7–14 (2011)
4. Carr, N.A., Hall, J.D., Hart, J.C.: GPU algorithms for radiosity and subsurface scattering. In: Proceedings of Graphics Hardware 2003, pp. 51–59 (2003)
5. Chang, C.W., Lin, W.C., Ho, T.C., Huang, T.S., Chuang, J.H.: Real-time translucent rendering using GPU-based texture space importance sampling. Computer Graphics Forum (Proceedings of Eurographics 2008) **27**(2), 517–526 (2008)
6. Chen, G., Peers, P., Zhang, J., Tong, X.: Real-time rendering of deformable heterogeneous translucent objects using multiresolution splatting. The Visual Computer **28**(6–8), 701–711 (2012)
7. Christensen, P.H., Harker, G., Shade, J., Schubert, B., Batali, D.: Multiresolution radiosity caching for efficient preview and final quality global illumination in movies. Tech. Rep. Pixar Technical Memo #12-06, Pixar (2012)
8. Dachsbacher, C., Křivánek, J., Hašan, M., Arbree, A., Walter, B., Novák, J.: Scalable realistic rendering with many-light methods. Computer Graphics Forum **33**(1), 88–104 (2014)
9. Dachsbacher, C., Stamminger, M.: Translucent shadow maps. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 197–201 (2003)
10. d'Eon, E.: A dual-beam 3D searchlight BSSRDF. In: ACM SIGGRAPH 2014 Talks, p. 65 (2014)
11. d'Eon, E., Irving, G.: A quantized-diffusion model for rendering translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011) **30**(4), 56:1–56:13 (2011)
12. d'Eon, E., Luebke, D., Enderton, E.: Efficient rendering of human skin. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2007), pp. 147–157 (2007)
13. Di Koa, M., Johan, H.: ESLPV: enhanced subsurface light propagation volumes. The Visual Computer **30**(6–8), 821–831 (2014)
14. Frisvad, J.R., Hachisuka, T., Kjeldsen, T.K.: Directional dipole model for subsurface scattering. ACM Transactions on Graphics **34**(1), 5:1–5:12 (2014)
15. Gerasimov, P.S.: Omnidirectional shadow mapping. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 12, pp. 193–203. Addison Wesley (2004)
16. Gibson, S.F.F.: Constrained elastic surface nets: Generating smooth surfaces from binary segmented data. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI'98, *Lecture Notes in Computer Science*, vol. 1496, pp. 888–898. Springer (1998)
17. Gkioulekas, I., Zhao, S., Bala, K., Zickler, T., Levin, A.: Inverse volume rendering with material dictionaries. ACM Transactions on Graphics **32**(6), 162:1–162:13 (2013)
18. Gosselin, D.R., Sander, P.V., Mitchell, J.L.: Real-time texture-space skin rendering. In: W. Engel (ed.) ShaderX$^3$: Advanced Rendering with DirectX and OpenGL, chap. 2.8, pp. 171–184. Charles River Media (2004)
19. Green, S.: Real-time approximations to subsurface scattering. In: R. Fernando (ed.) GPU Gems: Programming Techniques, Tips, and Tricks for Real-time Graphics, chap. 16, pp. 263–278. Addison Wesley (2004)
20. Habel, R., Christensen, P.H., Jarosz, W.: Photon beam diffusion: A hybrid Monte Carlo method for subsurface scattering. Computer Graphics Forum (Proceedings of EGSR 2013) **32**(4), 27–37 (2013)
21. Hable, J., Borshakov, G., Heil, J.: Fast skin shading. In: W. Engel (ed.) ShaderX$^7$: Advanced Rendering Techniques, chap. 2.4, pp. 161–173. Charles River Media (2009)
22. Halton, J.H.: Algorithm 247: Radical-inverse quasi-random point sequence. Communications of the ACM **7**(12), 701–702 (1964)
23. Hao, X., Baby, T., Varshney, A.: Interactive subsurface scattering for translucent meshes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics (i3D 2003), pp. 75–82 (2003)
24. Hao, X., Varshney, A.: Real-time rendering of translucent meshes. ACM Transactions on Graphics **23**(2), 120–142 (2004)
25. Jensen, H.W., Buhler, J.: A rapid hierarchical rendering technique for translucent materials. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002) **21**(3), 576–581 (2002)
26. Jensen, H.W., Marschner, S.R., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: Proceedings of ACM SIGGRAPH 2001, pp. 511–518 (2001)
27. Jimenez, J., Sundstedt, V., Gutierrez, D.: Screen-space perceptual rendering of human skin. ACM Transactions on Applied Perception **6**(4), 23:1–23:15 (2009)
28. Jimenez, J., Whelan, D., Sundstedt, V., Gutierrez, D.: Real-time realistic skin translucency. IEEE Computer Graphics and Applications **30**(4), 32–41 (2010)
29. Jimenez, J., Zsolnai, K., Jarabo, A., Freude, C., Auzinger, T., Wu, X.C., von der Pahlen, J., Wimmer, M., Gutierrez, D.: Separable subsurface scattering. Computer Graphics Forum (2015). To appear
30. Kajiya, J.T., Von Herzen, B.P.: Ray tracing volume densities. Computer Graphics (Proceedings of ACM SIGGRAPH 84) **18**(3), 165–174 (1984)
31. Keller, A.: Instant radiosity. In: Proceedings of ACM SIGGRAPH 97, pp. 49–56 (1997)
32. Kniss, J., Premože, S., Hansen, C., Ebert, D.: Interactive translucent volume rendering and procedural modeling. In: Proceedings of IEEE Visualization 2002, pp. 109–116 (2002)
33. Lensch, H.P.A., Goesele, M., Bekaert, P., Kautz, J., Magnor, M.A., Lang, J., Seidel, H.P.: Interactive rendering of translucent objects. In: Proceedings of Pacific Graphics (PG 2002), pp. 214–224 (2002)
34. Li, D., Sun, X., Ren, Z., Lin, S., Tong, Y., Guo, B., Zhou, K.: TransCut: Interactive rendering of translucent cutouts. IEEE Transactions on Visualization and Computer Graphics **19**(3), 484–494 (2013)
35. Mertens, T., Kautz, J., Bekaert, P., Reeth, F.V., Seidel, H.P.: Efficient rendering of local subsurface scattering. In: Proceedings of Pacific Graphics (PG 2003), pp. 51–58 (2003 )
36. Mertens, T., Kautz, J., Bekaert, P., Seidel, H.P., Reeth, F.V.: Interactive rendering of translucent deformable objects. In: Proceedings of Eurographics Symposium on Rendering (EGSR 2003), pp. 130–140 (2003)
37. Narasimhan, S.G., Gupta, M., Donner, C., Ramamoorthi, R., Nayar, S.K., Jensen, H.W.: Acquiring scattering properties of participating media by dilution. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2006) **25**(3), 1003–1012 (2006)

$$2C_1 \approx \begin{cases} 0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 \\ \qquad + 4.98554\eta^4 - 1.36881\eta^5, \eta < 1 \\ -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 \\ \qquad - 2.54396\eta^4 + 0.254913\eta^5, \eta \geq 1 \end{cases} \quad 3C_2 \approx \begin{cases} 0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 \\ \qquad + 0.236494\eta^4 + 0.145787\eta^5, \eta < 1 \\ -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} + 1213.67\eta \\ -568.556\eta^2 + 164.798\eta^3 - 27.0181\eta^4 + 1.91826\eta^5, \eta \geq 1 \end{cases}$$

**Fig. 16** Approximate fits for $2C_1$ and $3C_2$ by d'Eon and Irving [11].


38. Parker, S.G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., Stich, M.: OptiX: a general purpose ray tracing engine. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2010) **29**(4), 66:1–66:13 (2010)
39. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation, second edn. Morgan Kaufmann/Elsevier (2010)
40. Rushmeier, H.E., Torrance, K.E.: Extending the radiosity method to include specularly reflecting and translucent materials. ACM Transactions on Graphics **9**(1), 1–27 (1990)
41. Shah, M.A., Konttinen, J., Pattanaik, S.: Image-space subsurface scattering for interactive rendering of deformable translucent objects. IEEE Computer Graphics and Applications **29**(1), 66–78 (2009)
42. Sheng, Y., Shi, Y., Wang, L., Narasimhan, S.G.: A practical analytic model for the radiosity of translucent scenes. In: Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (i3D 2013), pp. 63–70 (2013)
43. Sloan, P.P., Hall, J., Hart, J., Snyder, J.: Clustered principal components for precomputed radiance transfer. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2003) **22**(3), 382–391 (2003)
44. Velho, L., Gomes, J., de Figueiredo, L.H.: Implicit objects in computer graphics. Springer (2002)
45. Wang, J., Zhao, S., Tong, X., Lin, S., Lin, Z., Dong, Y., Guo, B., Shum, H.Y.: Modeling and rendering of heterogeneous translucent materials using the diffusion equation. ACM Transactions on Graphics **27**(1), 9:1–9:18 (2008)
46. Wang, R., Cheslack-Postava, E., Wang, R., Luebke, D., Chen, Q., Hua, W., Peng, Q., Bao, H.: Real-time editing and relighting of homogeneous translucent materials. The Visual Computer **24**(7), 565–575 (2008)
47. Wang, R., Tran, J., Luebke, D.: All-frequency interactive relighting of translucent objects with single and multiple scattering. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005) **24**(3), 1202–1207 (2005)
48. Wang, Y., Wang, J., Holzschuch, N., Subr, K., Yong, J.H., Guo, B.: Real-time rendering of heterogeneous translucent objects with arbitrary shapes. Computer Graphics Forum (Proceedings of Eurographics 2010) **29**(2), 497–506 (2010)
49. Xu, K., Gao, Y., Li, Y., Ju, T., Hu, S.M.: Real-time homogenous translucent material editing. Computer Graphics Forum **26**(3), 545–552 (2007)
50. Yan, L.Q., Zhou, Y., Xu, K., Wang, R.: Accurate translucent material rendering under spherical Gaussian lights. Computer Graphics Forum **31**(7), 2267–2276 (2012)
51. Zhang, F., Sun, H., Nyman, O.: Parallel-split shadow maps on programmable GPUs. In: GPU Gems 3, chap. 10. Addison-Wesley (2007)


## A The Directional Dipole Model

[This appendix is not in the final publication.]

As other BSSRDF models, the directional dipole uses a number of inputs that are based on the optical properties of the translucent material ($\eta$, $\sigma_s$, $\sigma_a$, $g$):

$$\begin{aligned} &\sigma_t = \sigma_s + \sigma_a\,, && \sigma_s' = (1-g)\sigma_s\,, && \sigma_t' = \sigma_s' + \sigma_a\,, \\ &D = 1/(3\sigma_t')\,, && d_e = 2.131\,D\sqrt{\sigma_t'/\sigma_s'}\,, && \sigma_{\mathrm{tr}} = \sqrt{\sigma_a/D}\,, \\ &A = \tfrac{1-C_{\boldsymbol{E}}}{2C_\phi}\,, && C_\phi = \tfrac{1}{4}(1-2C_1)\,, && C_{\boldsymbol{E}} = \tfrac{1}{2}(1-3C_2)\,, \end{aligned}$$

where $C_1$ and $C_2$ are functions of $\eta$ listed in Fig. 16. The directional dipole formulas for $S_d$ are [14]:

$$S_d(\boldsymbol{x}_i, \vec{\omega}_i; \boldsymbol{x}_o) = S_d'(\boldsymbol{x}_o - \boldsymbol{x}_i, \vec{\omega}_{12}, d_r) - S_d'(\boldsymbol{x}_o - \boldsymbol{x}_v, \vec{\omega}_v, d_v)$$

and

$$\begin{aligned} &S_d'(\boldsymbol{x}, \vec{\omega}_{12}, r) = \\ &\tfrac{1}{4C_\phi(1/\eta)}\tfrac{1}{4\pi^2}\tfrac{e^{-\sigma_{\mathrm{tr}} r}}{r^3}\Bigg[C_\phi(\eta)\left(\tfrac{r^2}{D} + 3(1+\sigma_{\mathrm{tr}} r)\,\boldsymbol{x}\cdot\vec{\omega}_{12}\right) \\ &- C_{\boldsymbol{E}}(\eta)\Bigg(3D(1+\sigma_{\mathrm{tr}} r)\,\vec{\omega}_{12}\cdot\vec{n}_o \\ &- \left((1+\sigma_{\mathrm{tr}} r) + 3D\tfrac{3(1+\sigma_{\mathrm{tr}} r)+(\sigma_{\mathrm{tr}} r)^2}{r^2}\boldsymbol{x}\cdot\vec{\omega}_{12}\right)\boldsymbol{x}\cdot\vec{n}_o\Bigg)\Bigg], \end{aligned}$$

where the various $S_d'$ arguments are based on positions ($\boldsymbol{x}_i$, $\boldsymbol{x}_o$), direction of incidence ($\vec{\omega}_i$), and normals ($\vec{n}_i$, $\vec{n}_o$). For the real source, we have

$$\begin{aligned} \vec{\omega}_{12} &= \frac{1}{\eta}((\vec{\omega}_i\cdot\vec{n}_i)\vec{n}_i - \vec{\omega}_i) - \vec{n}_i\sqrt{1 - \frac{1}{\eta^2}(1-(\vec{\omega}_i\cdot\vec{n}_i)^2)} \\ d_r^2 &= \begin{cases} |\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + D\mu_0(D\mu_0 - 2d_e\cos\beta), & \text{for } \mu_0 > 0 \\ |\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + 1/(3\sigma_t)^2 & \text{, otherwise} \end{cases} \\ \mu_0 &= -\vec{n}_o\cdot\vec{\omega}_{12} \\ \cos\beta &= -\sqrt{\frac{|\boldsymbol{x}_o - \boldsymbol{x}_i|^2 - (\boldsymbol{x}\cdot\vec{\omega}_{12})^2}{|\boldsymbol{x}_o - \boldsymbol{x}_i|^2 + d_e^2}}\,, \end{aligned}$$

and for the virtual source,

$$\begin{aligned} \boldsymbol{x}_v &= \boldsymbol{x}_i + 2Ad_e\vec{n}_i^* \\ \vec{\omega}_v &= \vec{\omega}_{12} - 2(\vec{\omega}_{12}\cdot\vec{n}_i^*)\vec{n}_i^* \\ \vec{n}_i^* &= \begin{cases} \vec{n}_i & \text{, for } \boldsymbol{x}_o = \boldsymbol{x}_i \\ \dfrac{\boldsymbol{x}_o - \boldsymbol{x}_i}{|\boldsymbol{x}_o - \boldsymbol{x}_i|} \times \dfrac{\vec{n}_i \times (\boldsymbol{x}_o - \boldsymbol{x}_i)}{|\vec{n}_i \times (\boldsymbol{x}_o - \boldsymbol{x}_i)|} & \text{, otherwise}\,. \end{cases} \end{aligned}$$

Appendix IV

# Interactive Stable Ray Tracing

# Interactive Stable Ray Tracing

Alessandro Dal Corso
NVIDIA
Technical University of Denmark
alcor@dtu.dk

Marco Salvi
NVIDIA

Craig Kolb
NVIDIA

Jeppe Revall Frisvad
Technical University of Denmark

Aaron Lefohn
NVIDIA

David Luebke
NVIDIA

## ABSTRACT

Interactive ray tracing applications running on commodity hardware can suffer from objectionable temporal artifacts due to a low sample count. We introduce stable ray tracing, a technique that improves temporal stability without the over-blurring and ghosting artifacts typical of temporal post-processing filters. Our technique is based on sample reprojection and explicit hole filling, rather than relying on hole-filling heuristics that can compromise image quality. We make reprojection practical in an interactive ray tracing context through the use of a super-resolution bitmask to estimate screen space sample density. We show significantly improved temporal stability as compared with supersampling and an existing reprojection techniques. We also investigate the performance and image quality differences between our technique and temporal antialiasing, which typically incurs a significant amount of blur. Finally, we demonstrate the benefits of stable ray tracing by combining it with progressive path tracing of indirect illumination.

## CCS CONCEPTS

•**Computing methodologies** →**Ray tracing;**

## KEYWORDS

Reprojection, dynamic scene, caching, temporal stability, GPU

**ACM Reference format:**
Alessandro Dal Corso, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. 2017. Interactive Stable Ray Tracing. In *Proceedings of HPG '17, Los Angeles, CA, USA, July 28-30, 2017,* 10 pages.
DOI: 10.1145/3105762.3105769

## 1 INTRODUCTION

A rendered image will contain aliasing artifacts in regions where the underlying signal carries higher frequency content than the local sampling rate can capture. For example, light reflected from a highly specular surface can lead to aliasing if not sampled at sufficiently high rate. In addition, such aliasing artifacts will be perceived as particularly objectionable if high-frequency details are inconsistently sampled, causing sample values to change rapidly in time. To eliminate these artifacts, the underlying signal should ideally be bandlimited to remove frequencies beyond the local Nyquist limit. In general, however, robustly bandlimiting reflectance functions, visibility, and programmable shaders are open problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*HPG '17, Los Angeles, CA, USA*
© 2017 ACM. 978-1-4503-5101-0/17/07…$15.00
DOI: 10.1145/3105762.3105769

**Figure 1: In standard ray tracing, sampled screen space locations are kept fixed and shading locations vary. In stable ray tracing, shading locations are kept static while screen space locations vary.**

Stable shading is one strategy that can successfully mitigate aliasing artifacts in practice. In stable shading, shading calculations are performed in an object-local parametrization space, such as at the vertices of an underlying mesh, and the resulting values are interpolated across image pixels. These same object-local vertices are typically shaded again in subsequent frames, improving temporal stability in the presence of aliasing. Gouraud shading [1971] and the REYES rendering algorithm [1987], for example, use this approach to improve temporal image quality. However, these stable shading techniques do not work well with approaches such as ray tracing, wherein shading locations are determined independently of and without regard to any underlying local surface parametrization.

Stable ray tracing is a general technique that draws inspiration from previous stable shading approaches to improve the visual quality and/or reduce the computational cost of generating a sequence of images using ray tracing. Rather than using independent rays to sample the screen, shading locations from previous frames are re-used when possible, as shown in Figure 1. The fact that the points being shaded are temporally coherent results in fewer objectionable artifacts, even though the resulting images are still aliased. Furthermore, intermediate shading values can be cached along with the shading location, providing an additional performance benefit.

Our stable ray tracing is based on sample reprojection [Adelson and Hodges 1995; Badt 1988; Martin et al. 2002]. The main challenges in reprojection are verifying visibility of reprojected samples and avoiding large holes in the resulting screen space sampling pattern. We deal with the first issue by tracing visibility rays from the camera to the reprojected samples. For the second issue, we generate new samples on demand, where the demand is determined

using screen space sample density estimation. We perform this density estimation efficiently using a super-resolution bitmask that maps subpixel sample locations. This bitmask is also useful for removing samples to keep a uniform sample distribution. As an example application of stable ray tracing, we use amortized sampling to add progressively path traced indirect illumination to an image. We demonstrate how our stable ray tracing significantly improves temporal stability as compared with supersampling and as compared with an existing reprojection technique [Martin et al. 2002]. In addition, we use an image sharpness metric to verify that our technique avoids the blur of post-process filtering techniques.

## 2 RELATED WORK

The use of sample reprojection to exploit the temporal coherence of ray traced frames was first suggested by Badt [Badt 1988]. His technique is limited to viewpoint changes only, but he identifies the key issues of bad pixels and missed pixels. Bad pixels occur in regions where the colors of reprojected samples are no longer valid. Missed pixels are pixels that are not hit by reprojected samples. Badt suggests the interesting notion of a "recast mat", a one-bit-per-pixel mask pointing out the pixels for which we need new samples. We reverse this concept and use a super-resolution bitmask pointing out the subpixels that were hit by a reprojected sample.

Chapman et al. [1991] map out the spatio-temporal coherence of a predefined animation sequence by tracking sample trajectories across scene geometry. This is similar to sample reprojection and also works for moving objects. A reprojection based technique exploiting coherence between frames in a predefined animation sequence is also available for variance reduction in Monte Carlo ray tracing [Zhou and Chen 2015]. Gröller and Purgathofer [1991] present a spatial data structure for techniques like these that assume a predefined animation sequence. A more progressive approach is however required in interactive ray tracing, where the future scene dynamics are unknown. Murakami and Hirota [1992] present such an incremental approach, but only for a fixed viewpoint. They connect ray paths with objects using a hash index so that it is only necessary to recompute paths that interact with dynamic objects. We also connect samples to objects using an index.

Adelson and Hodges [1995] present a fully general reprojection technique for ray tracing with a screen space data structure containing one sample per pixel. We enhance this data structure by enabling a nonintegral number of samples per pixel. Adelson and Hodges [1995] also provide a careful description of the verification phase including the need for shadow and visibility rays to check for occlusion. We adopt their verification phase and make it practical for an interactive ray tracer running on graphics hardware.

The render cache concept [Walter et al. 2002, 1999; Zhu et al. 2005] achieves interactive frame rates through reprojection with different heuristics for handling bad and missed pixels. While the heuristics significantly improve performance, they also lead to objectionable visual artifacts.

Although reprojection started out as a way of exploiting temporal coherence to save computations, Martin et al. [2002] recognize it as an important technique for avoiding temporal aliasing. They find that reprojection achieves temporal stability similar to supersampling at a significantly lower computational cost. Their system only accounts for viewpoint changes and they apply temporal filtering using a box filter spanning three frames. Apart from this, their technique seems quite similar to that of Adelson and Hodges [1995]. Martin et al. [2002] also use one sample per pixel and pick the closest sample when multiple samples land in one pixel. This one-sample-per-pixel policy easily leads to scintillation artifacts due to insertion or removal of samples as objects rotate or move relative to the camera. Missed pixels and multiple samples in one pixel occur frequently when samples move across pixel boundaries (especially in perspective view) even if the local sample density is not changing much. We successfully mitigate this issue by estimating sample density in a 2-by-2 pixels area centered in every pixel. Our super-resolution bitmask strategy enables us to perform this density estimation efficiently.

In rasterization, the use of reprojection seems to be introduced in the context of warping one rendered image to the next [Chen and Williams 1993; Mark et al. 1997]. Rasterization-based techniques like the edge and point image [Bala et al. 2003; Velázquez-Armendáriz et al. 2006] achieve good results by adding edge information to the render cache information. However, this requires precomputation of an edge-based data structure [2003] or an additional edge rendering of the image [2006]. This becomes expensive in geometry-rich scenes where several edges may land in a pixel.

Inspired by the offline techniques [Adelson and Hodges 1995; Walter et al. 1999], reprojection finds an efficient implementation in a rasterization context with the reverse reprojection cache [Nehab et al. 2007] (also discovered by Scherzer et al. [2007] in a shadow mapping context and optimized by Sitthi-amorn et al. [2008a,b]). We keep forward reprojection, as this is better suited for ray tracing. As an add-on, these techniques [Nehab et al. 2007; Scherzer et al. 2007] introduce amortized sampling where pixel values are progressively updated over time. We use such amortized sampling for progressive sampling of indirect illumination.

Reprojection has also been used together with Monte Carlo ray tracing techniques like bidirectional path tracing and photon mapping [Havran et al. 2003; Tawara et al. 2004]. These techniques rely on stored sample points in any case, so no additional data structure is needed for the reprojection. In our case, we add a screen space data structure to support stable ray tracing. Our approach is thus well-suited for unidirectional Monte Carlo techniques.

In rasterization, Herzog et al. [2010] find that temporal finite differences are useful for amortized upsampling of images rendered with real-time global illumination techniques. They investigate screen-space ambient occlusion and indirect illumination from virtual point lights. In addition to better performance, they also find that their reprojection cache improves temporal stability.

On the side of temporal stability, recently introduced postprocessing filters like temporal supersampling [Karis 2014; Patney et al. 2016] efficiently hide temporal aliasing at the cost of introducing blur in the final image. Reprojection helps avoid excessive blurring and is effective in combination with sampling and filtering techniques from antialiasing [Jimenez et al. 2012] and from denoising [Iglesias-Guitian et al. 2016]. We set out to confirm that forward reprojection also has this ability to reduce temporal aliasing while preserving image sharpness. In addition, we exemplify the benefits of having stable samples in interactive ray tracing.

**Figure 2: Main building blocks of our algorithm. Data are in light green, compute phases are in dark green.**

## 3 OVERVIEW

In its most straight-forward implementation, stable ray tracing consists of four phases. *Reprojection* projects cached shading locations from the previous frame into screen space of the current frame, accounting for camera and object motion/deformation, to create a set of screen space sample locations. *Verification* constructs and traces primary visibility rays through the screen space sample locations to determine which of the reprojected shading positions are visible from the camera. Visible locations are then shaded, optionally caching intermediate results of the shading computation for later reuse. *Hole filling* generates screen space samples in regions where the density of visible reprojected points is low, and traces, shades, and caches hitpoint/shading information. Finally, *reconstruction* generates the final image for the current frame from the set of shaded samples.

The basic version of stable ray tracing improves temporal stability through the reuse of shading points across frames. However, there are a number of practical challenges to achieving interactive performance. In this section, we discuss these issues and associated tradeoffs, and briefly describe the choices we made in our system.

### 3.1 Sampling Rate and Uniformity

Sampling rate is the primary means of trading image quality for performance. Unlike conventional ray tracing, wherein screen sample locations are essentially independent of objects in the scene, in stable ray tracing screen space sampling density can be highly non-uniform due to the effects of camera and object movement on reprojected samples. Reprojection can lead to oversampling due to many points being reprojected to the same region of the screen, for example when an object moves away from the camera, or the camera zooms out. In such cases, maintaining performance requires that we ensure oversampling is kept to a minimum. Conversely, reprojection can also lead to undersampling due to disocclusions, or when sample density decreases due to a surface moving closer to the camera. In such cases, maintaining image quality requires that we ensure that enough samples are used. Highly non-uniform sampling can also lead to issues with resource contention (for example, multiple threads attempting to write to same cache location during reprojection) and load balancing. In addition, nonuniform sampling can produce artifacts when the sampling rate is very low compared to the reconstruction rate, as discussed in Section 7.

In order to ensure appropriate sampling rate and uniformity, our implementation adds an *analysis phase* prior to verification. The analysis phase efficiently estimates local sampling density and adds or removes samples to ensure the sampling rate falls within a specified range. As described in Section 4.2, the analysis phase makes use of a bitmask that encodes a quantized representation of the sampling pattern in each pixel, which allows us to estimate sampling density without having to read or recompute exact screen space locations for each sample.

### 3.2 Caching

Key to the efficiency and effectiveness of our implementation is the sample cache, which allows temporal re-use of shading locations and intermediate values. However, stable ray tracing's computational and memory overhead is proportional to the number of entries that are reprojected and potentially verified and shaded. As such, a cache eviction policy is needed that allows trading performance and memory use for temporal stability.

The simplest policy would be to evict points that are occluded or otherwise not used in the current frame. However, stability in the face of high-frequency visibility changes can be improved if occluded points remain in cache long enough to be re-used when they become visible again. As a result, there is a tradeoff between the space and reprojection cost of keeping occluded points in the cache and the temporal stability improvements to which such points may contribute in the future.

In addition to storing in the cache sufficient information to reconstruct world space position, we can also use the cache to avoid recomputation of expensive intermediate values required during shading (e.g., visibility or normals). Taken together, these values can cause each cache entry to be rather large. As such, minimizing overall size is important to performance, as is minimizing cache reads due to memory bandwidth constraints.

We use a two-phase cache eviction scheme that strives to strike a balance between overall performance and temporal stability. The first set of evictions occur in the reprojection phase (Section 4.1) and the second in the analysis phase (Section 4.2).

### 3.3 Ray Tracing

The basic stable ray tracing algorithm has two distinct ray tracing phases: verification and hole filling. The number of holes to be filled is typically small compared to the number of verification rays, and as a result the overhead associated with launching a separate hole-filling ray tracing pass can be non-trivial. As such, performance

**Figure 3: Reprojection phase for an M=4 set of subpixels from one frame, left, to the next, right, and evolution of the bitmasks. Both occluded and unoccluded samples are recorded in the occupancy bitmask. Occluded samples (red) also have a bit set in the occlusion bitmask. In the case of a collision between two samples (subpixel 7, right), the first unoccluded sample written to the subpixel is kept.**

could benefit if it were possible to combine the two ray tracing passes into one.

In our implementation, we fill verification-failure holes by using the occluding hitpoints discovered in the verification phase. This optimization improves performance over the naive implementation, at the cost of some sampling bias and an increase in sampling rate variance. However, the instability added is typically spatially incoherent and persists for a single frame, and as such is not usually objectionable.

## 4 METHOD

In this section, we discuss the details of our implementation, the design decisions we faced, and the choices we made. Our implementation is illustrated in Figure 2.

We store samples in two screen space data buffers, which serve as caches for the previous and current frame. At the beginning of each frame, samples are reprojected from the previous buffer to the current to account for object and camera motion. We analyze the outcome of the reprojection process and adaptively add or remove samples in the reprojection buffer in order to achieve a uniform sample distribution. The location samples are then verified, and finally shaded. The resulting color information is stored in a shading buffer, which is used by the reconstruction phase to resolve color.

### 4.1 Reprojection

Stable ray tracing requires that cached samples are updated to reflect scene dynamics such as camera motion and object motion and deformation. The data to be stored per sample in the reprojection buffers should thus be chosen according to the scene dynamics that one would like to support. We store a 3D position in object space coordinates and a transform ID to support affine transformations.

```
input : pixelDestination and subpixelDestination for a sample
        and associated data that isOccluded or not.
subpixel ← flatten (subpixelDestination);
bitOccupancy ← 1 ≪ subpixel;
bitOcclusion ← 1 ≪ (subpixel + M·M);
bitMask ← bitOccupancy ∨ (isOccluded? bitOcclusion: 0);
originalBitmask ← AtomicOr (pixelDestination, bitMask);
originalIsOccluded ← (bitOcclusion ∧ originalBitmask) ==
    bitOcclusion;
replace ← not isOccluded ∧ originalIsOccluded;
if not (isOccluded ∧ originalIsOccluded) then
    AtomicAnd (pixelDestination, ¬bitOcclusion)
end
originalExists ← (bitOccupancy ∧ originalBitmask) ==
    bitOccupancy;
if replace ∨ not originalExists then
    writeData( pixelDestination,data);
end
```

**Algorithm 1:** Pseudocode for sample reprojection storage.

More data would likely be required to support arbitrary object deformation. The ID we store is used to access an object-to-world transformation matrix for the current frame. This matrix is in turn used to transform the sample position to world space. We then project the world space position onto the screen using the current camera transformation, and we clip away samples that fall outside the screen area.

During reprojection, we take steps to ensure that not too many samples reproject to the same screen location in order to reduce resource contention, improve load balancing, and manage size of the cache. We also strive to preferentially keep samples that are visible over those that are occluded.

To do so, we divide each pixel in the reprojection buffer into $M \times M$ subpixels, as illustrated in Figure 3. We maintain a corresponding occupancy bitmask representing the occupancy state of each subpixel, which is cleared at the start of each frame. The occupancy bitmasks are also used during the analysis phase to determine approximate sample location and local sample density. We similarly maintain with each pixel an $M \times M$ bitmask that indicates if the sample in each subpixel is occluded; values in this occlusion bitmask are written during the verification phase. Storing these bitmasks separately from the cache values themselves allows us to reduce bandwidth required by the reprojection phase.

When a source sample reprojects into a given destination subpixel, we check the destination subpixel's corresponding occupancy bit in the bitmask. If the destination subpixel occupancy bit is zero, the sample is written to the destination location, the destination occupancy bit is set to one, and the destination subpixel occlusion bit is copied from the source bitmask. If the destination subpixel occupancy bit is one, we examine the destination subpixel occlusion bit. If the destination subpixel occlusion bit is one and the source occlusion bit is zero, the source sample is written to the destination, and the destination occlusion bit set to zero. Otherwise the source sample is not written to the destination buffer, effectively evicting

**Figure 4: Left: the spatial extent (light red) of our local density analysis. Right: Update pattern that arises from our analysis scheme. Example: hairball sequence with a right-rotating camera and $d_{\text{target}} = d_{\text{tolerance}} = 1$. Green pixels indicate areas where new samples are added, while red samples indicate where samples are evicted.**

it from the cache. A pseudocode outline of our eviction scheme is in Algorithm 1.

Data races due to competing threads working on the same sample can be avoided by atomically updating the per-sample data, potentially causing a large performance impact. We note instead that as we only perform atomic updates of the bitmasks a data race can only occur when a first occluded sample lands on a sample and second unoccluded one tries to overwrite it. In this rare case, we would simply store the occluded sample over the unoccluded, leading to reduced temporal stability. In practice we found these events to be rare and to have small impact on the final image quality.

Our sample rejection policy ensures that we cache at most $M \times M$ samples in any pixel, enforcing an upper bound on storage and subsequent processing costs, while maintaining a good screen-space distribution of samples, unlike, for example, simply keeping the first $M \times M$ samples that reproject into a given pixel would. The mechanism also ensures that unoccluded samples are preferentially cached over occluded samples.

## 4.2 Sample Analysis

In regions that are oversampled, analysis chooses which samples to remove, and adds new samples in undersampled regions to meet the desired sampling rate.

To help ensure a good spatial distribution of samples, we divide each pixel in a number of strata (in our implementation, 4). For each stratum, we count the number of samples. To remove samples, we choose from the substratum with the most number samples, selecting randomly in the case of a tie. Similarly, we progressively add samples to the substratum with the fewest samples. This process allows us to stratify the samples across the pixel. Within a substratum, new samples are placed in the center, with a small random offset in order to avoid correlation in the screen space location of the samples.

To minimize the overall performance impact of analysis, we use the occupancy and occlusion bitmasks to determine whether samples should be added or removed. To determine how many to add or remove, we analyze the local sample density $d = N/A$, where $N$ is the number of unoccluded samples in an area of $A = 2 \times 2$ pixels around the current pixel. The user can then specify two parameters, $d_{\text{target}}$ and $d_{\text{tolerance}}$. The algorithm will not add or remove samples if the density is within $[d_{\text{target}} - d_{\text{tolerance}}, d_{\text{target}} + d_{\text{tolerance}}]$. Otherwise, we add or remove enough unoccluded samples $\Delta N$ to bring the density within limits:

$$\Delta N = \begin{cases} \operatorname{sgn}(d_{\text{target}} - d) \lceil |d_{\text{target}} - d| \rceil & \text{if } |d_{\text{target}} - d| \geq d_{\text{tolerance}} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where the sign of $\Delta N$ tells us whether we need to add or remove samples. Figure 4 illustrates a typical pattern of sample addition and removal for a dynamic scene.

It is necessary to modify the cache when we add a new sample, since in the next phase we need to distinguish between new and cached samples. To remove a sample, we simply set the corresponding occupancy bit to zero. For a new sample, we write $(\text{NaN}, \mathbf{p}^x, \mathbf{p}^y)$ instead of its object space position. The NaN marks the sample as new. Since we have to store the new sample in memory, we also store the chosen screen space coordinates for the sample $(\mathbf{p}^x, \mathbf{p}^y)$.

## 4.3 Verification and Shading

The verification phase processes the location samples to generate shading samples for the reconstruction phase. Our algorithm works on top of any ray tracing framework that provides programmable camera and closest hit stages. We define a standard ray as a tuple $\mathbf{r} = (\mathbf{o}, \vec{d}, t_{\text{min}}, t_{\text{max}})$, where the quantities represent origin, direction, and minimum and maximum intersection distances, respectively.

In this step, we distinguish between cached samples and newly generated samples with screen space coordinates $(\text{NaN}, \mathbf{p}^x, \mathbf{p}^y)$ in the cache. We trace these new samples with a closest hit ray, using the stored screen space position to generate a corresponding world space direction $\vec{d}$ according to our camera model. Given the camera position $\mathbf{c}$, our ray becomes $\mathbf{r} = (\mathbf{c}, \vec{d}, \epsilon, +\infty)$. Once the ray tracing operation terminates, we store the hitpoint object space position and transform ID in the reprojection cache, and the corresponding shade in the shading cache.

For existing samples with cached position $\mathbf{x}_{\text{object}}$, we first compute its corresponding world space position $\mathbf{x}_{\text{world}}$. Then, we cast a closest hit ray $\mathbf{r}_{\text{cached}} = (\mathbf{c}, (\mathbf{x}_{\text{world}} - \mathbf{c})/\|\mathbf{x}_{\text{world}} - \mathbf{c}\|, \epsilon, \|\mathbf{x}_{\text{world}} - \mathbf{c}\| + \epsilon)$. When we hit the closest surface, we verify that the sample is still visible in the current frame. If the sample is still visible, the intersected $t$ should match the cached $t = \|\mathbf{x}_{\text{world}} - \mathbf{c}\|$.

Occluded samples can cause numerical instability in the shading distribution, in particular around geometric edges. In our implementation, we normally mark such samples as occluded and keep them in the cache. However, if an occluded sample is the last one remaining in a pixel, we replace its hitpoint with the one from the occluding surface. This allows us to maintain a minimum sample density without requiring a new ray to be traced, as discussed in Section 3.3.

Once a sample is verified, or if it is new, we shade it according to our rendering algorithm, and store the results in the shading buffer, alongside its subpixel position.

| Supersampling, 4 spp | Supersampling, 4 spp + temporal antialiasing | Stable ray tracing, 2 spp | Stable ray tracing, 2 spp + temporal integration | Supersampling, 32 spp |
|---|---|---|---|---|
| sharpness: 0.8142 | sharpness: 0.6610 | sharpness: 0.8056 | sharpness: 0.7783 | sharpness: 0.8054 |

**Figure 5: Comparison of frames rendered for the hairball video. For each technique, we report the number of samples per pixel (spp) and the CPBD-based image sharpness. Stable ray tracing strikes a compromise between sharpness and temporal stability at the price of added spatial aliasing.**

| Technique | Reprojection | Analysis | Verification / Shading | Reconstruction | Total |
|---|---|---|---|---|---|
| Stable ray tracing, $d_{\text{target}} = 1$ | 1.05 ms | 0.28 ms | 18.91 ms | 0.71 ms | **20.94 ms** |
| Stable ray tracing, $d_{\text{target}} = 2$ | 1.23 ms | 0.38 ms | 28.88 ms | 0.82 ms | **31.31 ms** |
| Stable ray tracing, $d_{\text{target}} = 4$ | 1.73 ms | 0.62 ms | 47.48 ms | 0.90 ms | **50.73 ms** |
| Supersampling, 1 spp | - | - | 13.35 ms | 0.21 ms | **13.56 ms** |
| Supersampling, 2 spp | - | - | 20.94 ms | 0.38 ms | **21.32 ms** |
| Supersampling, 3 spp | - | - | 28.36 ms | 0.54 ms | **28.90 ms** |
| Supersampling, 4 spp | - | - | 35.86 ms | 0.71 ms | **36.57 ms** |
| Supersampling, 5 spp | - | - | 43.40 ms | 0.88 ms | **44.28 ms** |
| Supersampling, 6 spp | - | - | 50.91 ms | 1.04 ms | **51.95 ms** |

**Table 1: Average time spent per frame in the hairball video for each phase of the different techniques. All results use GPU timers. The additional price for stable ray tracing is a slowdown of the overall rendering time between 1.4x and 1.5x. Temporal integration is performed on the resulting image, at an additional cost of 0.67 ms.**

### 4.4 Reconstruction

Each color sample stored in the previous step carries an RGB color and subpixel position. We then filter our resulting color using a $3 \times 3$ truncated spatial Gaussian filter. Our algorithm does not guarantee uniform sampling rate, since it trades off a uniform rate for temporal stability. A nonuniform sampling density can lead to challenges in reconstruction, such as pixels with no samples. At low sampling densities, the use of this simple reconstruction filter can lead to blurring and apparent thickening of edges. We discuss the artifacts resulting from trading spatial uniformity for temporal coherence in Section 7.

After reconstruction, an additional post processing step may be performed. In Section 6, we discuss how our method fares with a temporal reconstruction scheme on top, namely temporal integration. When performing this additional step, we calculate and store motion vectors in the shading cache, picking the one with maximum length during reconstruction.

## 5 IMPLEMENTATION DETAILS

Our reprojection and analysis phases are implemented as OpenGL compute shaders. The reprojection shader transfers data between two identically deep screen sized buffers. The verification and shading step is implemented on the GPU in the camera program using the NVIDIA OptiX ray tracing engine [2010]. The programmable ray tracing pipeline of OptiX allows us to insert our cache management. The reconstruction and post processing were implemented as full screen passes in OpenGL shaders.

We compress our samples as 16-bytes elements of which 12 are reserved for 3 floating point elements defining position in object space. Due to OpenGL-OptiX interoperability limitations, we were not able to write the occlusion bit in the bitmask in the verification and shading phase directly. So we use one of the remaining 32 bits to store occlusion for the sample. Note that this does not change performance, since we have to fetch the sample anyways in the reprojection phase. The remaining 31 bits are reserved for a transform ID to allow affine transformations. The existence and occlusion statuses of the samples are stored in the bitmasks, for which we use $M = 4$. We use the two halves of a 32 bits unsigned integer to store both 16 bits bitmasks. The shading samples are stored as 8 bytes elements: 3 bytes for the tone-mapped color, 4 bytes for a motion vector (16 bits per component) and 1 byte for the subpixel position and flags (3+3 bits for position in a 8x8 grid, plus 1 bit for an existence flag).

**Figure 6: Three examples of sample distributions generated by our algorithm in the areas marked by the colored squares. Blue circles represent visible samples, red circles represent occluded samples.**

Stable ray tracing, 1 spp sharpness: 0.8182 | [Martin et al. 2002] sharpness: 0.6957

**Figure 7: Quality comparison with Martin et al. [Martin et al. 2002] for frame 526 of the martin_comparison.mp4 video. Their technique produces a blurrier result and is also more temporally unstable.**

## 6 RESULTS

Given the dynamic nature of our algorithm, we provide some of our results in a video (hairball.mp4) of a static hairball [McGuire 2011] captured with a moving camera. The hairball has a standard glossy material applied, and is illuminated by a single point light to which a shadow ray is traced per shading evaluation. The frames of the video were captured individually and then assembled to create a video of 60 frames per second. All our results were generated using an NVIDIA GeForce GTX 1080 graphics card. We report rendering times for a 1080x1080 image frame.

The hairball video compares stable ray tracing with supersampling of similar performance. In addition, to measure the impact of a recent temporal noise reduction scheme, we apply temporal integration with color clamping in the variant proposed by Patney et al. [2016]. For stable ray tracing, samples are not jittered and we choose an integration factor of $\alpha = 0.25$. For supersampling, we use $\alpha = 0.1$ and do full temporal antialiasing by including sample jittering in the temporal integration. The larger $\alpha$ used for stable ray tracing incurs a smaller amount of blur, which we can get away with because our input values are more stable. If we use $\alpha > 0.1$ with supersampling, the temporal antialiasing cannot hide the underlying temporal instability. A single frame of the hairball video is provided in Figure 5. Here, we compare image sharpness using a CPBD-based sharpness metric [Narvekar and Karam 2011]. The sharpness score measures the percentage of edges at which blur (probably) cannot be detected. The video shows a reference rendering, rendered as 32 samples per pixels.

Comparing supersampling and stable ray tracing, we first observe that while stable ray tracing does not completely remove temporal artifacts (in particular around the strands of the hairball), the final result perceptually improves in temporal stability. This is especially true at the beginning of the video, where the camera is only rotating. Sharpness of stable ray tracing and supersampling is similar to that of the reference, with supersampling being slightly greater than reference. Once we apply temporal integration to both results, the situation reverses. Supersampling with temporal integration is more temporally stable (although some underlying noise is still present), but it is also significantly more blurry. Temporal integration applied to stable ray tracing reduces some of the higher frequency noise, but it also better preserves sharpness while retaining temporal stability.

Since our algorithm trades temporal stability for an irregular spatial sampling pattern, we want to validate the aliasing artifacts that are generated by the algorithm. An example of the kind of distribution of samples we achieve with our algorithm is shown in Figure 6, for three different areas of a single frame of the hairball video. We compare the quality for different target densities of our algorithm in Figure 8, for three different scenes (hairball, plane with text and ogre). The images were taken after 25 frames of an animated video, to allow stable ray tracing to set into a nonstandard sampling pattern. We provide closeups to better show the artifacts generated at a pixel level. For the lowest sample count (1 spp averages), we can see that stable ray tracing introduces artifacts. In the hairball frames, we can see that this manifests as thickened edges. In the plane with text frame, the artifacts manifest as broken edges and letters. In the ogre scene, they manifest as weirdly shaped specular highlights. For averages of 2 spp, the differences reduce and it almost disappears with averages of 4 spp.

We compare the performance of stable ray tracing against supersampling in Table 1. All results were obtained using OpenGL GPU timers, averaging the milliseconds spent in each phase over the whole sequence in the hairball video. From the totals in the table, we can see that stable ray tracing generally performs 1.4 to 1.5 times slower than the equivalent supersampling. This is similar to the performance cost of a factor of around 1.35 reported by Martin et al. [2002]. The overhead of reprojection and analysis phases is between 1 and 3 milliseconds. We note that the reconstruction phase for stable ray tracing has a higher impact than the one in supersampling, given that we need to adapt it to the irregular number of samples we have per pixel.

We made a comparison with Martin et al. [2002], tweaking the algorithm to fit modern GPU pipelines. For each sample, we generate

**Figure 8: Quality comparisons between standard supersampling (SS) and stable ray tracing (SRT), for different number of samples. We use the same Gaussian reconstruction filter for all images. For low sample counts, stable ray tracing gives a result that is more temporally stable, at the price of introducing spatial aliasing artifacts.**

a single vertex, rendered as a 1x1 pixel splat in the final destination pixel. This allows us to use the depth buffer to find the closest sample in the case of multiple samples landing in one pixel. If a sample does not exist, we simply generate a vertex outside of the view frustum. Then, a ray tracing step generates a sample in the middle of the pixel if it does not find one, and traces the ray. Relatively, our implementation is a bit faster than the original method, being only 1.2 times slower than the equivalent supersampling. The results are in a video (martin_comparison.mp4) and in Figure 7, where we provide a comparison with our method for similar sample counts. On the left-hand side of the video, we compare the two techniques for a panning view of a bump mapped plane. In this case, the quality of the two techniques is similar, except for the blurring due to the temporal filter employed by Martin et al. [2002]. If we consider the hairball (right-hand side of the video), our method is significantly more temporally stable. In addition, since we do not use an averaging temporal filter, our method produces sharper images (see Figure 7).

## 6.1 Application: Progressive Path Tracing

Our screen space sample data structure serves a double purpose: nearby samples in the data structure are close in world space, and

| Supersampling, 2 spp | Supersampling, 2 spp + temporal antialiasing | Stable ray tracing, 1 spp | Stable ray tracing, 1 spp + temporal integration | Supersampling, 32 spp |
|---|---|---|---|---|
| sharpness: 0.7924 | sharpness: 0.5348 | sharpness: 0.7085 | sharpness: 0.6060 | sharpness: 0.6771 |

**Figure 9: Including indirect illumination, the different techniques are here applied to a frame in the Sponza video.**

the majority of samples are consistent in world space across frames. These properties make stable ray tracing suitable for accumulating view-independent but time-dependent information, such as diffuse indirect illumination.

As a proof of concept, we apply our technique on top of standard unidirectional path tracing to cache diffuse indirect illumination in a dynamic scene. For performance reasons, our path tracing has a fixed maximum trace depth. For each frame, we choose a random direction, trace a new path in that direction, and accumulate the final result. Directions are sampled using a cosine-weighted hemispherical distribution. For a completely static scene, we could give equal importance to all frames. Since we want to be able to react to dynamic content in the scene, we use a simple exponential moving average [Nehab et al. 2007; Scherzer et al. 2007] with integration factor 0.1. Our focus is here to illustrate the virtues of stable ray tracing in accumulation. More complicated sampling schemes are possible, such as accumulating indirect illumination to allow convergence when camera and scene are static, or from literature [Herzog et al. 2010; Yang et al. 2009].

Like the hairball video, we provide a similar video comparison for our global illumination method. In this video (sponza.mp4), we compare our progressive path tracing to a similar performance supersampling with 2 samples per pixel. As in the previous section, we provide comparisons with and without the temporal integration schemes. In this comparison, we observe that stable ray tracing improves temporal stability for a scene with a dynamic moving light. Some noise is still present, mostly due to fireflies generated by the new shading directions chosen for each sample. Since we use a screen space data structure, results must be re-generated upon disocclusion. This is why the flagpoles in the video leave trails of higher variance content. Figure 9 compares a cutout of a still frame of the Sponza video. One should note that the blur incurred by the temporal post-processing filters is good at hiding the stochastic noise of the path tracing. However, as is clear from visual comparison and the CPBD-based image sharpness measurements, the blurring of temporal antialiasing on top of supersampling is too much. We also note how the reference rendering in this case also has a lower sharpness score than results without temporal post-processing filters. This is mainly due to the noise in these two images being considered as sharpness.

initial distribution thickened edge

**Figure 10: Edge thickening. Blue samples belong to one of the hairball strands, yellow samples belong to the background, and red samples are occluded. In the right image, the camera has moved upwards, so that the apparent motion in screen space of the edge is downwards. The low local density in the area above the strand causes the thickening.**

## 7 DISCUSSION

Stable ray tracing improves temporal stability while retaining sharpness (hairball video and Figure 5). Our algorithm offers an intermediate solution between supersampling, which is sharp but temporally unstable, and temporal antialiasing, which is too blurry. The reason for this excessive blurriness is the high temporal instability in the input from supersampling. Since we do not have this temporal instability, we can apply a more relaxed temporal filtering (larger $\alpha$) and thus strike a compromise between stability and sharpness. On the other hand, we cannot use jittering and therefore pay the price of spatial aliasing artifacts. These artifacts are particularly evident at lower sampling rates, resulting in broken or thickened edges and changed highlight patterns (Figure 8).

Spatial aliasing artifacts arise from the fact that we do not estimate the screen space coverage of each sample, but rather give them the same weight in the reconstruction phase. As we illustrate in Figure 10, this causes edge thickening. The distribution of samples changes a bit, but not enough to change the density. New samples are therefore added. The small gap introduced by the change in distribution is filled as possible by the reconstruction algorithm, causing the edge to thicken. A lower $d_{\text{tolerance}}$ could mitigate this

problem by fixing the distribution more quickly wherever necessary. However, lowering this parameter would cause samples to get recycled more often, leading as well to temporal instability. This screen space coverage problem is partly to blame for the residual temporal instability of stable ray tracing, since each sample would have a different estimated coverage every frame.

As previously noted, the overhead of our technique is similar to that of Martin et al. [2002]. In our video comparison, we see how we reduce temporal artifacts, by allowing an irregular number of samples per pixel in our technique. This allow us to remove the originally proposed scene-based temporal filter, increasing the sharpness of the final image in the process. Although the overhead added by the reprojection and analysis phases are relatively low, there is an additional verification overhead when comparing on an iso-sample-rate basis. This penalty is due to load balancing issues resulting from the nonuniform screen-space sampling patterns, and subsequent varying amount of per-pixel work, generated by reprojection. We expect that the ray tracing overhead can be reduced by performing a load balancing step prior to tracing rays.

Our Sponza video exemplifies the potential of stable ray tracing as a technique for caching indirect light. In this example, due to the nature of our accumulation scheme, the fireflies generated by the path tracing procedure cause an additional level of temporal instability. However, our algorithm still retains its qualities, retaining a higher temporal stability (at least when temporal filtering is not used to hide it) and better image sharpness (Figure 9).

## 8 CONCLUSION

We presented a new practical technique for stable shading in interactive ray tracing. Our technique is based on sample reprojection and introduces low cost sample analysis for generating and evicting samples in the reprojection cache. The stable ray tracing that we propose is useful for striking a balance between temporal stability and image sharpness in interactive ray tracing applications. This comes at the cost of spatial aliasing and around a factor 1.5 hit to the performance. If the rendering budget allows a target sample density of just 4 samples per pixel, our technique can eliminate most spatial aliasing artifacts and provide a visually pleasing (sharp, antialiased) and fairly temporally stable result. Since we have stable shading in a ray tracing context, we can use our shading cache to add global illumination effects such as progressively path traced indirect illumination. In general, our algorithm eases the use of progressive techniques when a scene is dynamic.

## REFERENCES


Stephen J. Adelson and Larry F. Hodges. 1995. Generating exact ray-traced animation frames by reprojection. *IEEE Computer Graphics and Applications* 15, 3 (1995), 43–52.

Sig Badt, Jr. 1988. Two algorithms for taking advantage of temporal coherence in ray tracing. *The Visual Computer* 4, 3 (1988), 123–132.

Kavita Bala, Bruce Walter, and Donald P. Greenberg. 2003. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)* 22, 3 (July 2003), 631–640.

John Chapman, Thomas W. Calvert, and John Dill. 1991. Spatio-temporal coherence in ray tracing. In *Proceedings of Graphics Interface (GI '91)*. 101–108.

Shenchang Eric Chen and Lance Williams. 1993. View interpolation for image synthesis. In *Proceedings of SIGGRAPH 93*. ACM, 279–288.

Robert L. Cook, Loren Carpenter, and Edwin Catmull. 1987. The Reyes image rendering architecture. *Computer Graphics (Proceedings of SIGGRAPH 87)* 21, 4 (July 1987), 95–102.

Henri Gouraud. 1971. Continuous shading of curved surfaces. *IEEE Trans. Comput.* 20, 6 (June 1971), 623–629.

Eduard Gröller and Werner Purgathofer. 1991. Using temporal and spatial coherence for accelerating the calculation of animation sequences. In *Proceedings of Eurographics (EG '91)*, Vol. 91. 103–113.

Vlastimil Havran, Cyrille Damez, Karol Myszkowski, and Hans-Peter Seidel. 2003. An efficient spatio-temporal architecture for animation rendering. In *Rendering Techniques 2003 (Proceedings of EGSR 2003)*, Per H. Christensen and Daniel Cohen-Or (Eds.). Eurographics Association, 106–117.

Robert Herzog, Elmar Eisemann, Karol Myszkowski, and Hans-Peter Seidel. 2010. Spatio-temporal upsampling on the GPU. In *Proceedings of Interactive 3D Graphics and Games (I3D '10)*. ACM, 91–98.

Jose A. Iglesias-Guitian, Bochang Moon, Charalampos Koniaris, Eric Smolikowski, and Kenny Mitchell. 2016. Pixel history linear models for real-time temporal filtering. *Computer Graphics Forum (Proceedings of Pacific Graphics 2016)* 35, 7 (October 2016), 363–372.

Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. 2012. SMAA: enhanced subpixel morphological antialiasing. *Computer Graphics Forum (Proceedings of Eurographics 2012)* 31, 2pt1 (May 2012), 355–364.

Brian Karis. 2014. High-quality temporal supersampling. In *Advances in Real-Time Rendering in Games, Part I*. Number 10 in ACM SIGGRAPH 2014 Courses. http://advances.realtimerendering.com/s2014/

William R. Mark, Leonard McMillan, and Gary Bishop. 1997. Post-rendering 3D warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics (I3D '97)*. ACM, 7–16.

William Martin, Peter Shirley, Steven Parker, William Thompson, and Erik Reinhard. 2002. Temporally coherent interactive ray tracing. *Journal of Graphics Tools* 7, 2 (2002), 41–48.

Morgan McGuire. 2011. Computer Graphics Archive. (August 2011). http://graphics.cs.williams.edu/data

Koichi Murakami and Katsuhiko Hirota. 1992. Incremental ray tracing. In *Photorealism in Computer Graphics (Proceedings of EGWR 1990)*, K. Bouatouch and C. Bouville (Eds.). Springer, 17–32.

N. D. Narvekar and L. J. Karam. 2011. A no-reference image blur metric based on the cumulative probability of blur detection (CPBD). *IEEE Transactions on Image Processing* 20, 9 (September 2011), 2678–2683.

Diego Nehab, Pedro V. Sander, Jason Lawrence, Natalya Tatarchuk, and John R. Isidoro. 2007. Accelerating real-time shading with reverse reprojection caching. In *Proceedings of Graphics Hardware (GH 2007)*. 25–36.

Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. 2010. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)* 29, 4 (July 2010), 66:1–66:13.

Anjul Patney, Marco Salvi, Joohwan Kim, Anton Kaplanyan, Chris Wyman, Nir Benty, David Luebke, and Aaron Lefohn. 2016. Towards foveated rendering for gaze-tracked virtual reality. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* 35, 6 (November 2016), 179:1–179:12.

Daniel Scherzer, Stefan Jeschke, and Michael Wimmer. 2007. Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings of EGSR 2007)*. Eurographics Association, 45–50.

Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, and Diego Nehab. 2008a. An improved shading cache for modern GPUs. In *Proceedings of Graphics Hardware (GH 2008)*. Eurographics Association, 95–101.

Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, Diego Nehab, and Jiahe Xi. 2008b. Automated reprojection-based pixel shader optimization. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)* 27, 5 (December 2008), 127:1–127:11.

Takehiro Tawara, Karol Myszkowski, Kirill Dmitriev, Vlastimil Havran, Cyrille Damez, and Hans-Peter Seidel. 2004. Exploiting temporal coherence in global illumination. In *Proceedings of Spring Conference on Computer Graphics (SCCG 2004)*. ACM, 23–33.

Edgar Velázquez-Armendáriz, Eugene Lee, Kavita Bala, and Bruce Walter. 2006. Implementing the render cache and the edge-and-point image on graphics hardware. In *Proceedings of Graphics Interface 2006 (GI '06)*. Canadian Information Processing Society, 211–217.

Bruce Walter, George Drettakis, and Donald P. Greenberg. 2002. Enhancing and optimizing the render cache. In *Proceedings of the Eurographics Workshop on Rendering (EGWR 2002)*. ACM Press, 37–42.

Bruce Walter, George Drettakis, and Steven Parker. 1999. Interactive rendering using the render cache. In *Rendering techniques '99 (Proceedings of EGWR 1999)*. Springer, 19–30.

Lei Yang, Diego Nehab, Pedro V. Sander, Pitchaya Sitthi-amorn, Jason Lawrence, and Hugues Hoppe. 2009. Amortized supersampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)* 28, 5 (December 2009), 135:1–135:12.

Peng Zhou and Yanyun Chen. 2015. Variance reduction using interframe coherence for animated scenes. *Computational Visual Media* 1, 4 (December 2015), 343–349.

Tenghui Zhu, Rui Wang, and David Luebke. 2005. A GPU accelerated render cache. In *Proceedings of Pacific Graphics 2005 (short paper)*.

Appendix V

# Virtual Reality inspection and painting with measured BRDFs

# Virtual Reality inspection and painting with measured BRDFs

Alessandro Dal Corso
Technical University of Denmark

Jonathan Dyssel Stets
Technical University of Denmark

Andrea Luongo
Technical University of Denmark

Jannik Boll Nielsen
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

Henrik Aanæs
Technical University of Denmark

**Figure 1: Pictures illustrating our VR demo application, with an in-game screenshot (left) and a picture of the setup (right). Paintbucket, table and lightbuld models ©TurboSquid.com, environment maps ©HDRMaps.com and ©Joost Vanhoutte.**

## CCS CONCEPTS

• **Computing methodologies → Virtual reality**;

## KEYWORDS

Virtual Reality, material appearance

**ACM Reference Format:**
Alessandro Dal Corso, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. 2017. Virtual Reality inspection and painting with measured BRDFs. In *Proceedings of SA '17 VR Showcase, Bangkok, Thailand, November 27-30, 2017,* 2 pages.
https://doi.org/10.1145/3139468.3139472

## 1 INTRODUCTION

This is a virtual reality (VR) painting application that enables the user to paint on 3D models with real measured materials, much like in the physical world. A scanned physical object can be imported into the VR application, and the user can paint on the surface of the object with a virtual hand-controlled paint brush. The user is presented with several paint buckets, each containing a material known from the physical world. These materials are measured bidirectional reflectance distribution functions (BRDFs) of real physical materials. The materials and objects present in VR are thus represented as they would be in the physical world, and the user can control both the environment lighting and a single light source. The application enables analog artists to apply their skills directly on a digital model and it enables engineers to directly inspect BRDFs in a fast and intuitive way. Figure 1 is a screenshot from the application showing models that have been painted with BRDFs.

## 2 MOTIVATION

The realization of our VR demo was mainly driven by two objectives. The first was to have an intuitive and practical way of inspecting and visualizing measured BRDFs. Our application enables the user to apply BRDFs to 3D objects in a simple way that mimics the action of painting in real life. The user can also interact with the environment and the objects through an intuitive interface. Some of the actions the user can perform consist of modifying the light source intensity and position, the environment map, and the position and orientation of the objects. All these features allow us to quickly visualize and inspect a chosen BRDF under different lighting conditions without having to interact with a complicated interface.

Our second objective was to have an application useful in industrial and artistic design [Wald et al. 2006]. We provide a new way for artists, both digital and analog, to transfer their skills to the 3D domain by painting materials directly on 3D models that they might have created or scanned from real objects. By providing a real-time rendering environment, we enable artists to immediately see the final results of their creations under different viewing perspectives and lighting conditions. With this VR application, we aim at moving

*SA '17 VR Showcase, November 27-30, 2017, Bangkok, Thailand*
© 2017 Copyright held by the owner/author(s).
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of SA '17 VR Showcase, November 27-30, 2017*, https://doi.org/10.1145/3139468.3139472.

**Figure 2: Interaction (left) and paintbrush (right) controller. Note the icons allowing to change the model and the brush size, respectively. See Figure 1 for copyright notice.**

a first step towards the creation of a tool that will allow artists to apply materials and colors in a way that is similar to what they would do in the real world. Thus, we support the more traditional artistic workflow rather than inventing new artistic workflows for editing BRDFs [Colbert et al. 2006].

A user entering the virtual world will see a simple scene: a table, calibrated to match the position and height of a table present in the real world, a set of paint buckets containing different materials based on measured BRDFs, several 3D models, and a light-bulb. The user interacts with the scene through two handheld trackable controllers, one used as a painting tool and the other one as a grabbing and interaction tool. He/she can grab, move, rotate, and scale the 3D models with one hand and paint and select a different material with the other. The user can also move the position of the light-bulb and its intensity to get an immediate feedback on the appearance of their work. Furthermore, we use haptic feedback to enhance the interaction between user and objects in the scene. Our application thus creates a bridge between the digital and the physical domain, with an interface known from the physical world that the user is already familiar with. All these features help users immerse themselves and unfold their creativity in an environment similar to what they would see in a real artist's studio, and we have aimed to make this environment as interactive as possible.

## 3 DETAILS

Our demo is an HTC Vive setup (https://www.vive.com/), using the two provided controllers as interaction devices. One of the controllers is used for interaction, while the other acts as a paintbrush. The interaction is activated using the trigger button on the controller, and is used for grabbing and moving objects, including the light bulb above the table and the paint buckets. We use the small spheres on the left side of the table to change environment map. Finally, we have an undo button on the right to undo/redo the last action performed. Dipping the paintbrush into a bucket changes the BRDF that it will paint with. Based on the controller touchpads (see Figure 2), we also added interactions for changing object size (while grabbed), paintbrush size (on the paintbrush controller), and light intensity (while the light is grabbed).

We use measured BRDFs both from the MERL database [Matusik et al. 2003] and from our own laboratory. The scene has two light sources: an environment map and a movable point light in the form of a light bulb. The environment map contributes with background, reflections, and an ambient term. The ambient term is computed through standard spherical harmonics multiplied by the bihemispherical reflectance $\rho$ of each measured BRDF, calculated in a preprocessing step using Monte Carlo integration. We multiply the environment map reflected color by $\rho$ and by the factor $\min(1, \frac{f_{\max}}{C\rho_{\text{avg}}} - 1)$, where $f_{\max}$ is the peak value in the measured material, $\rho_{\text{avg}}$ is the average of the three channels of $\rho$, and $C$ is a user-defined constant. In case of a material with a strong reflection peak (such as a metallic paint), $\rho_{\text{avg}} \ll f_{\max}$ and the factor will be equal to one. In a case of a more diffuse material, $\rho_{\text{avg}} \approx f_{\max}$ and the reflection term will not be included. To paint the material, we intersect a sphere with the vertices of the model. For materials without a UV map, we write a material label on a per-vertex data structure. If a UV map is present, we first generate a secondary texture that maps vertex coordinates to UVs and then write the labels into a texture using the generated mapping.

## 4 USER FEEDBACK AND CONCLUSION

We invited an analog artist, a design engineer, and two 3D artists to test our application. They all found the interface intuitive to work with. Most noticeably, the analog artist used the full system without any previous VR experience after a one minute verbal instruction. This supports the objective of our application to enable transfer of artistic skills from the analog to the digital domain. Users noted the bulkiness of the controller compared to real-life painting tools such as a brush or a pencil. We accept this limitation of the system, hoping for smaller, lighter, or more customizable solutions in the future, like the stylus presented by Jackson and Keefe [2016].

Our VR painting application enables the user to paint on 3D models with measured BRDFs. Our users praised how our application is intuitive to use, and how it creates a bridge between the analog and digital skills. Furthermore, it enables a fast and intuitive way to inspect BRDFs under various lighting conditions.

## Acknowledgements

Paint bucket, table, and light bulb models are courtesy of TurboSquid.com. Environment maps: studio, garden, and sunset forest from hdrmaps.com, lobby and night city square from Joost Vanhoutte. We use chrome, blue paint, gold paint and red fabric BRDFs from the MERL database. Ogre model courtesy of Keenan Crane. VIVE and related assets are property of HTC, Inc. and Valve Corporation. We would like to thank users Felicia Frisvad, Jon Murray Vinther, Sam Surplice, Christian Ahm for their valuable feedback.

## REFERENCES

M. Colbert, S. Pattanaik, and J. Krivanek. 2006. BRDF-Shop: creating physically correct bidirectional reflectance distribution functions. *IEEE Computer Graphics and Applications* 26, 1 (Jan 2006), 30–36. https://doi.org/10.1109/MCG.2006.13

B. Jackson and D. F. Keefe. 2016. Lift-off: Using reference imagery and freehand sketching to create 3D models in VR. *IEEE Transactions on Visualization and Computer Graphics* 22, 4 (2016), 1442–1451. https://doi.org/10.1109/TVCG.2016.2518099

W. Matusik, H. Pfister, M. Brand, and L. McMillan. 2003. A Data-Driven Reflectance Model. *ACM Transactions on Graphics* 22, 3 (July 2003), 759–769.

I. Wald, A. Dietrich, C. Benthin, A. Efremov, T. Dahmen, J. Gunther, V. Havran, H. p. Seidel, and P. Slusallek. 2006. Applying Ray Tracing for Virtual Reality and Industrial Design. In *2006 IEEE Symposium on Interactive Ray Tracing*. 177–185. https://doi.org/10.1109/RT.2006.280229

Appendix VI

# VirtualTable: a projection augmented reality game

# VirtualTable: a projection augmented reality game

A. Dal Corso M. Olsen K. H. Steenstrup J. Wilm S. Jensen R. R. Paulsen E. Eiríksson
J. Nielsen J. R. Frisvad G. Einarsson H. M. Kjer

Department of Applied Mathematics and Computer Science, Technical University of Denmark

**Figure 1:** *(left) Our setup using projector (red frustum) and a Kinect camera (green frustum). (middle,right) Pictures of the gameplay.*

VirtualTable is a projection augmented reality installation where users are engaged in an interactive tower defense game.The installation runs continuously and is designed to attract people to a table, which the game is projected onto. Any number of players can join the game for an optional period of time. The goal is to prevent the virtual stylized soot balls, spawning on one side of the table, from reaching the cheese. To stop them, the players can place any kind of object on the table, that then will become part of the game. Depending on the object, it will become either a *wall*, an obstacle for the soot balls, or a *tower*, that eliminates them within a physical range. The number of enemies is dependent on the number of objects in the field, forcing the players to use strategy and collaboration and not the sheer number of objects to win the game.

Our installation is an example of a combination of tangible user interfaces [Shaer and Hornecker 2010] and projection augmented reality [Mine et al. 2012]. Leitner et al. [2008] presented IncreTable, a tabletop game that includes multiple inputs from different devices, including physical objects. Molla and Lepetit [2010] present a similar concept of augmented board game, but in their case the output is shown on a screen and not re-projected on the game. Compared to Leitner et al. [2008], our interaction design can be learned by exploration and thus requires no instructions. This is important in child-computer interaction and in combination with the use of tangibles it empowers the shift from "learning by being told" to "learning by doing" [Hourcade 2008]. We thus believe that our VirtualTable is an excellent concept for development of immersive and engaging learning games for children.

## Our approach

VirtualTable uses a computer unit attached to both a Kinect camera and a projector. We first process the input from the Kinect depth camera, then we pass it to the actual game to display the output.

The objects are recognized using the depth camera of the Kinect. We automatically calibrate our software once before the game is actually started, to estimate both a ground depth and the Kinect-projector homography. After the calibration, objects of any significant depth (at least 0.5 cm) can be recognized. The output of the depth camera is used to create a bounding box around the objects. We exclude objects that are connected to the border, to avoid recognizing the players' hands. In Figure 1 (left) we see that the Kinect camera covers an area bigger than the game area, to not accidentally exclude objects lying on the border.

We transmit the identified bounding boxes to the actual game using a custom made protocol. In the virtual game, wall objects are invisible and affect only the behavior of the soot balls. We project a red glow around the towers to distinguish them and give a visual feedback on their range (see Figure 1 (right)). When we update the set of recognized boxes, we compare it with the existing set. Matching boxes have their position updated, interpolating it with their old position to avoid flickering. The remaining boxes are either added or removed to the game accordingly. Objects are distinguished only by shape: elongated objects are walls, square-like objects towers.

The behavior of the soot balls is simulated using Unity Engine's built-in navigation system on navigation meshes. A tower, with a given frequency, shoots bullets to the soot balls within its range, removing them from the game.

The game explores the concept of augmented reality games, combining the tangible sensation of the pieces from board games and the immediate visual feedback from modern computer games.

## References

HOURCADE, J. P. 2008. Interaction design and children. *Found. Trends Hum.-Comput. Interact. 1*, 4 (Apr.), 277–392.

LEITNER, J., HALLER, M., YUN, K., WOO, W., SUGIMOTO, M., AND INAMI, M. 2008. IncreTable, a mixed reality tabletop game experience. In *Proceedings of Advances in Computer Entertainment Technology (ACE)*, 9–16.

MINE, M., VAN BAAR, J., GRUNDHOFER, A., ROSE, D., AND YANG, B. 2012. Projection-based augmented reality in Disney theme parks. *Computer 45*, 7 (July), 32–40.

MOLLA, E., AND LEPETIT, V. 2010. Augmented reality for board games. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*.

SHAER, O., AND HORNECKER, E. 2010. Tangible user interfaces: Past, present, and future directions. *Found. Trends Hum.-Comput. Interact. 3*, 1;2 (Jan.), 1–137.

©ACM 2015. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in
*SA'15 Posters*, November 02-06, 2015, Kobe, Japan
ACM 978-1-4503-3926-1/15/11.
http://dx.doi.org/10.1145/2820926.2820950

Appendix VII

# Our 3D Vision Data-Sets in the Making

# Our 3D Vision Data-Sets in the Making

H. Aanæs[1] K. Conradsen[1] A. Dal Corso[1] A. B. Dahl[1] A. Del Bue[2] M. Doest[1]
J. R. Frisvad[1] S. H. N. Jensen[1] J. B. Nielsen[1] J. D. Stets[1]
G. Vogiatzis[3]
[1] Technical University of Denmark [2] Istituto Italiano di Tecnologia [3] Aston University, UK

## 1. Introduction

Over the previous years, we have at the Section for Image Analysis and Computer Graphics at the Technical University of Denmark been working on generating high quality data sets for computer vision via our lab setup using a 6-axis industrial robot. This has provided a new data set aimed at feature matching [1, 4], and two data sets aimed at multiple view stereo [14, 16]. The resulting data sets are publicly available via `http://roboimagedata.compute.dtu.dk/`.

The evaluation of computer vision algorithms on these data sets has provided useful insights on realistic scenarios by setting a rigorous framework for evaluation. The results of these efforts have been well received by the community and the hardware and software platform associated with the robot is now well developed. We are currently in the process of making three new data sets aimed at 3D vision, with a special focus on the more challenging aspects, such as radiometry and the modelling of non-rigid objects. The construction of these data sets all leverage on our robotic setup's ability to produce ground truth camera and surface geometry, as briefly outlined in Section 2, and there is a great deal of commonality in the making of the data sets.

This abstract describes our current ongoing work on this data set construction for 3D vision. The data sets include:

1. A direct extension of our large multiple view stereo (MVS) data set [14], where we are now including transparent and semi transparent objects into the scenes, Section 3. A challenge in doing this is getting the ground truth geometry of the transparent objects.

2. A data set addressing the radiometric challenges in 3D vision as presented in Section 4 where we aim at extending our MVS data set by explicitly measure the bidirectional reflectance distribution function (BRDF) of the surfaces. This will have the additional feature to finally give a data set for evaluating photometric stereo with a ground truth.

3. An extension of our data set on feature matching to evaluate these algorithm with non-rigid objects, (Section 5) where we use actuators to make stop motion 3D data sets. This data set will also evaluate Non-rigid Structure from Motion (NRSfM) with realistic objects.

Figure 1. Photos of the 6-axis industrial robot mounted with two cameras and a projector. Cameras allow for MVS, and in conjunction with the projector SL provides ground truth point clouds.

## 2. Brief System Overview

Our experimental setup, cf. [1], is built around a 6-axis ABB IRB 1600 industrial robot, providing a flexible, precise, and highly repeatable camera pose. The robot is mounted with two Point Grey Grasshopper3 $3376 \times 2704$ 8-bit RGB cameras and a projector (for previously published datasets the cameras were $1600 \times 1200$ 8-bit Point Grey Scorpion cameras). From each position ground truth surface point clouds are obtained using structured light (SL), and stereo images with a 32 cm baseline are captured with the camera pair. Five individually controlled 6500K LED tube lights allow for soft natural illumination of scenes from varying directions. Figure 1 shows the robot.

Previous evaluations of our system [14] have shown that the ground truth samples obtained through SL have good accuracy with a surface standard deviation of 0.14 mm. We expect similar or better performance in this data set. Positioning repeatability of the robot is very high, with a standard deviation of 0.0031 mm over two months.

Additional instruments used for generating the data include a CT (Computed Tomography) scanner for ground truth geometry of transparent objects (described in Section 3) and an illumination arch for controlled directional light-

Figure 2. Preliminary images from our data set. In the first row, three glass objects (sphere, bowl, teapot) with markers placed on. On the second row, three calibration and rendering tools part of the pipeline: a black and white checkerboard (coordinate estimation), an X-Rite ColorChecker® (color balance compensation) and a chrome sphere (environment light evaluation).

ing (described in Section 4).

## 3. Transparent Objects

Our goal is to extend our original MVS dataset to account for transparent objects where the focus is on reconstruction of geometry and appearance. Usually, the radiometric behavior of the objects used in 3D reconstructions is assumed diffuse and opaque. This leads to a number of simplifications that we cannot apply to transparent objects. In the case of transparent objects, refraction and reflection cause distortion effects that complicate reconstruction.

Previous methods acquire data sets useful for image-based rendering of a transparent object [18, 11]. However, these methods do not produce an actual triangle mesh and require special rendering techniques for reconstruction of the appearance of the transparent object. A survey on methods that do provide a triangle mesh is available [13]. In this survey, they note that CT scanning of refractive objects like glass is costly but straight forward. Thus, we use CT scanning to obtain ground truth geometry. Another way is to acquire shape and pose of a transparent object from motion [3]. In any case, there seems to be no data set, like the one we propose, which is useful for multiple view reconstruction of transparent objects.

### 3.1. Data

Our data set contains a set of multiple view HDR images of three glass objects with different radiometric properties (top row of Figure 2). We use a solid sphere, a bowl with lid (composed of two parts) and a teapot with multiple thin glass layers (composed of three parts). The walls of the bowl and the teapot have different thickness. A diffuse backdrop is provided for the objects. We have made this as a gradient checkerboard, so that one half of the squares varies in color from left to right, and the other half varies in color from top to bottom. In this way, we can see how light reflects, refracts and scatters through the objects. The refractive index of the glass objects will be estimated directly from the scanned images, or, if this is unsuccessful, by the use of a refractometer. We marked the objects with small black plastic spheres, in order to easily determine their position relative to the scene. In our data set, we also provide high-resolution triangle meshes generated from CT scans. We use these scans as ground truth for either geometrical reconstruction algorithms or physically based rendering algorithms for appearance modelling.

Our current data set creation procedure is as follows. First, we choose a sequence of camera positions and orientations for our industrial robot. The robot enables us to reproduce a given set of positions and orientations with a very high precision. Then, we capture a first set of images placing a black and white checkerboard in the scene. This is done to obtain the camera positions relative to the scanned objects and calculate camera parameters for the setup. Secondly, we scan a commercial color checker, which allows us to compensate for color channel alterations in the final images. Finally, we scan a chrome sphere to get an HDR environment map of the surroundings. We use the resulting map as a light source in our rendering algorithms [5], so we can simulate the resulting scene with high precision. After these three calibration steps, we can finally scan the glass objects using the same pre-defined path used for the calibration images.

Once compiled, we are planning to use this data set to verify that the radiometric models [9] properly describe the radiometric properties of the scene. To do this we plan to feed the ground truth of our data into a custom-built renderer based on the NVIDIA OptiX library [20], and see how well it reproduces the images. If successful, we have a validated computational model, which in principle we 'just' have to invert to do 3D reconstruction of transparent objects. Following this we plan at applying state of the art 3D reconstruction algorithms and quantify how far the state of the art has come toward solving this central 3D vision reconstruction problem.

## 4. BRDF measurements and Photometric Stereo

The radiometric behaviour of an object plays a crucial role in MVS. Often this behaviour has been ignored or at most assumed Lambertian. This allows for acceptable reconstructions of geometry, but often poor recovery of the reflectance. For more accurate MVS and reflectance capture, the BRDF of an object should be taken into account and this is a problem that receives a growing amount of

attention [24, 15]. Within the field of photometric stereo, the reflectance of an object is the key element in recovering surface normals and thereby indirectly the object's geometry. Also here, assumptions about reflectance are made, these include e.g. Lambertian behaviour [27] or isotropic BRDFs [12].

For both of the above areas, a multi-view data set having ground-truth reflectance behaviour would be of great value, and does, to our knowledge, not currently exist. We are therefore now working on a MVS data set where not only the ground-truth geometry is given, but also a densely sampled BRDF ground-truth for all materials in the scene. In the following, we will elaborate on the details of how this data set will be acquired and what it will include.

### 4.1. Concept

Figure 3. Schematic of BRDF capturing setup. Setup includes a 6-axis industrial robot holding a CCD (stereo) camera for view, and an arc in conjunction with a turntable for illumination.

Capturing the reflectance of a material generally requires four degrees of freedom: polar and azimuthal angle of illumination-direction, and polar and azimuthal angle of view-direction, $\rho(\omega_i, \phi_i, \omega_v, \phi_v)$. Utilizing our lab-facility's 6-axis industrial robot, mounted with a stereo-camera setup, all view directions $(\omega_v, \phi_v)$ can effectively be captured. For illumination directions $(\omega_i, \phi_i)$, we utilize an illumination arc and a rotation-table. The arc holds a range of halogen light-bulbs and is capable of covering the polar angle $\phi_i$ in $7.5°$ intervals. The rotation-table turns the target sample with a resolution of $< 1°$, thus densely covering $\theta_i$. Figure 3 shows a schematic of the BRDF capturing setup, and Figure 4 is a photo of an actual acquisition scene.

Using the above described setup, we intend to densely sample the BRDFs of a collection objects whose surfaces consist of one or a few, isotropic, BRDFs. The BRDFs of each material will be stored in the 3-dimensional Rusinkiewicz frame for isotropic BRDFs [21], as also done in the MERL database[17], although with a coarser resolution of $7.5°$ in each dimension. In conjunction with the densely sampled BRDFs, stereo images of scenes containing the sampled objects will be acquired for a wide range of directions. Objects will be of relatively low geometric complexity, and scenes will consist of one or more of the objects.

Figure 4. Capturing the BRDF of an object with known geometry. All illumination directions and view-directions are covered for each type material present on the object.

## 5. Non-Rigid Structure from Motion

Evaluating Non-rigid feature matching and NRSfM algorithms[1] in a quantitative manner has in the literature proven to be problematic. Deformations are inherently a dynamic process and subject to the physical properties of the objects in consideration. Thus, evaluating deformation modelling algorithms require a reasonable number of different objects and set of motions. Also, given the dynamic deformation objects might change their topology (e.g. stretching and tearing) and easily self-occluded some parts of the shape. For this reason, many approaches have provided several models that fit specific types of deformation, but that cannot comprise all of them. For this reason understanding the real performance of methods on realistic deformations is necessary to push forward advancements in this field.

The central problem of producing reference ground truth has been approached from many different angles. Several works compare their methods using synthetically generated images, as the true 3D geometry is readily available[26, 22, 19, 10]. Another popular approach is using MOCAP data, mainly human motion, for generating both test video sequence with 3D reference points [7, 26, 2, 10, 25]. Both falls short, as the former often lacks the complexity found in real-life scenes and the latter provides only a sparse set of reference points that are likely not to be possible to detect from images because of occlusions. As stated in [22, 8], there is a lack of and a need for a real-life NRSfM sequence with a dense 3D reference.

[1] A review on NRSfM methods, updated to 2010, can be found here: [23]

Figure 5. Robot arm carrying cameras for capturing stop motion frame and structured light data. A Gray code pattern is currently being projected onto the object.

We seek to remedy this situation by providing a video recording of real objects with dense 3D ground truth for each frame. It will be accomplished using a stop motion like animation techniques and structured light 3D scanning, combined in our unique recording setup.

### 5.1. Concept

We wish to simulate motion in a manner similar to stop motion animated films. Here a rigid object is moved into a certain pose, an image is taken, the object is slightly changed with a deformation, another image is taken etc. The result is a sequence that, when played at an interactive frame rate, provides the illusion of motion. We will apply the same principle here, in generating a benchmarking data set for NRSfM with ground truth.

Now one may ask, why not just record the motion using ordinary video format? After all, stop motion techniques does not properly reproduce motion blur artifacts that are present in standard recorded video sequences. Our approach has several significant advantages that greatly outweigh the loss of motion blur. Most importantly, we can obtain a 3D ground truth for each frame. After adjusting the object into its current frame position and acquiring an image for the stop motion sequence, we will perform a 3D scan using structured light. Utilizing gray code patterns we obtain a dense ground truth so obtaining both the image frame and a 3D reference for benchmarking and validation.

Another advantage is that we can obtain data from multiple views by acquiring images at different angles thus providing data for evaluating multi-view NRSfM (e.g. [6]). Furthermore, this procedure provides a great degree of control over both camera movement and object pose. As each frame is recorded independently, time in between becomes a non-issue.

Figure 6. Actuators for manipulating the geometry of the mask. The image of the mask has been superimposed on an image of the actuators, illustrating their functionality.

### 5.2. Implementation

Such data could be acquired by pure manual effort, however that would be extremely time consuming and error-prone. As such, a robotics solution is currently being developed with a the data acquisition procedure that is predictably and reproducibly implemented. In detail, a robotic arm move the camera and the projector needed for data acquisition and structured light scan. From this the view position can be determined with high precision and reproducibility. Figure 5 illustrates this setup.

Additionally, object deformation will also be automated and Figure 6 shows an example with an object where a mask resembling a human face is put on top of two actuators. Manipulating the actuators deforms the mask geometry, simulating facial movement. Similar results can be obtained with cloth, paper and other deformable materials.

## 6. Concluding Remarks

We have here presented our ongoing work on making high quality data sets for evaluating and developing methods for 3D vision. A motivation for doing this is that we see a need for this, especially with respect to making data sets that are large enough, so that it is possible to reasonably determine if differences in performance are a statistical fluke, or are in fact statistically significant.

By presenting our ongoing work in this forum, we hope to get valuable and constructive feedback on how these data sets in the making could be adapted to serve the needs of the computer vision communities as best possible.

## References


[1] H. Aanæs, A. Dahl, and K. Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012.

[2] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(7):1442–1456, 2011.

[3] M. Ben-Ezra and S. Nayar. What does motion reveal about transparency? In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 1025–1032, 2003.

[4] A. Dahl, H. Aanæs, and K. Pedersen. Finding the best feature detector-descriptor combination. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 318–325, 2011.

[5] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH 98*, pages 189–198, 1998.

[6] A. Del Bue and L. Agapito. Stereo non-rigid factorization. *International Journal of Computer Vision*, 66(2):193–207, February 2006.

[7] J. Fayad, L. Agapito, and A. Del Bue. Piecewise quadratic reconstruction of non-rigid surfaces from monocular sequences. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 297–310. Springer, 2010.

[8] K. Fragkiadaki, M. Salas, P. Arbelaez, and J. Malik. Grouping-based low-rank trajectory completion and 3D reconstruction. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 55–63. Curran Associates, Inc., 2014.

[9] A. S. Glassner. Surface physics for ray tracing. In A. S. Glassner, editor, *An Introduction to Ray Tracing*, chapter 4, pages 121–160. Academic Press Ltd., London, UK, 1989.

[10] P. F. U. Gotardo and A. M. Martinez. Kernel non-rigid structure from motion. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pages 802–809. IEEE, 2011.

[11] T. Hawkins, P. Einarsson, and P. E. Debevec. A dual light stage. *Rendering Techniques 2005 (Proceedings of EGSR 2005)*, pages 91–98, 2005.

[12] M. Holroyd, J. Lawrence, G. Humphreys, and T. Zickler. A photometric approach for estimating normals and tangents. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, 27(5):133, 2008.

[13] I. Ihrke, K. N. Kutulakos, H. Lensch, M. Magnor, and W. Heidrich. Transparent and specular object reconstruction. *Computer Graphics Forum*, 29(8):2400–2426, 2010.

[14] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 406–413, 2014.

[15] H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo beyond Lambert. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I:171–178. IEEE, 2003.

[16] S. Kim, H. Aanæs, A. Dahl, K. Conradsen, R. Jensen, and S. Kim. Multiple view stereo by reflectance modeling. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, 2012.

[17] W. Matusik, H. Pfister, M. Brand, and L. McMillan. A data-driven reflectance model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):759–769, 2003.

[18] W. Matusik, H. Pfister, R. Ziegler, A. Ngan, and L. McMillan. Acquisition and rendering of transparent and refractive objects. pages 267–278, 2002.

[19] S. I. Olsen and A. Bartoli. Implicit non-rigid structure-from-motion with priors. *Journal of Mathematical Imaging and Vision*, 31(2-3):233–244, 2008.

[20] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich. OptiX: a general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 29(4):66:1–66:13, July 2010.

[21] S. Rusinkiewicz. A new change of variables for efficient BRDF representation. In *Rendering Techniques (Proceedings of EGWR 1998)*, June 1998.

[22] C. Russell, J. Fayad, and L. Agapito. Dense non-rigid structure from motion. In *Proceedings of 3DIMPVT 2012*, pages 509–516. IEEE, 2012.

[23] M. Salzmann and P. Fua. Deformable surface 3d reconstruction from monocular images. *Synthesis Lectures on Computer Vision*, 2(1):1–113, 2010.

[24] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528. IEEE, 2006.

[25] L. Tao and B. J. Matuszewski. Non-rigid structure from motion with diffusion maps prior. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1530–1537. IEEE, 2013.

[26] J. Taylor, A. D. Jepson, and K. N. Kutulakos. Non-rigid structure from locally-rigid motion. *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 2761–2768, 2010.

[27] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):191139, 1980.

Appendix VIII

# Derivation of standard and directional dipole quantities

# Derivation of standard and directional dipole quantities

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

Thomas Kim Kjeldsen
The Alexandra Institute

September 2017

## 1 Integrating the radiative transfer equation

We start from the radiative transfer equation:

$$(\nabla \cdot \vec{\omega}) L(\mathbf{x}, \vec{\omega}) = -\sigma_t L(\mathbf{x}, \vec{\omega}) + \sigma_s \int_{4\pi} p(\vec{\omega}, \vec{\omega}') L(x, \vec{\omega}') \, d\omega' + Q(\mathbf{x}, \vec{\omega})$$

Then we integrate over $\vec{\omega}$:

$$\int_{4\pi} (\nabla \cdot \vec{\omega}) L(\mathbf{x}, \vec{\omega}) \, d\omega = \int_{4\pi} -\sigma_t L(\mathbf{x}, \vec{\omega}) \, d\omega + \int_{4\pi} \sigma_s \int_{4\pi} p(\vec{\omega}, \vec{\omega}') L(x, \vec{\omega}') \, d\omega' \, d\omega + \int_{4\pi} Q(\mathbf{x}, \vec{\omega}) \, d\omega$$

Rearranging we obtain:

$$\nabla \cdot \left( \int_{4\pi} \vec{\omega} L(\mathbf{x}, \vec{\omega}) \, d\omega \right) = -\sigma_t \int_{4\pi} L(\mathbf{x}, \vec{\omega}) \, d\omega + \sigma_s \int_{4\pi} \left( \int_{4\pi} p(\vec{\omega}, \vec{\omega}') \, d\omega \right) L(x, \vec{\omega}') \, d\omega' + q(\mathbf{x})$$

Where we used the regularity of the functions involved to switch gradient and integral operation on the left side, and to switch the integrals on the right hand side. The integral of the phase function is 1, since it is normalized, so by further simplifying and applying the definitions we obtain:

$$\nabla \cdot \vec{E}(\mathbf{x}) = -\sigma_t \phi(\mathbf{x}) + \sigma_s \int_{4\pi} L(x, \vec{\omega}') \, d\omega' + q(\mathbf{x})$$

$$\nabla \cdot \vec{E}(\mathbf{x}) = -\sigma_t \phi(\mathbf{x}) + \sigma_s \phi(\mathbf{x}) + q(\mathbf{x})$$

$$\nabla \cdot \vec{E}(\mathbf{x}) = -\sigma_a \phi(\mathbf{x}) + q(\mathbf{x}) \tag{1}$$

Where in the last passage we used $\sigma_t = \sigma_a + \sigma_s$. The last one is equation (1) in Jensen.

## 2 The diffusion approximation

To get the diffusion approximation, we approximate radiance using spherical harmonics.

$$L(\mathbf{x}, \vec{\omega}) \approx \sum_{n=0}^{1} \sum_{m=-n}^{n} L_{n,m}(x, \vec{\omega}) Y_{n,m}(\vec{\omega})$$

Where $Y_{n,m}(\vec{\omega})$ is the $n-m$ spherical harmonics base function[1], and $L_{n,m}(x, \vec{\omega})$ is the projection of $L$ over this function:

$$L_{n,m}(x, \vec{\omega}) = \int_{4\pi} L(\mathbf{x}, \vec{\omega}) Y_{n,m}(\vec{\omega}) \; d\omega$$

For $n = 0$ the integral is trivial. The first term of the sum becomes:

$$L_{0,0}(x, \vec{\omega}) Y_{0,0}(\vec{\omega}) = \sqrt{\frac{1}{4\pi}} \int_{4\pi} \sqrt{\frac{1}{4\pi}} L(\mathbf{x}, \vec{\omega}) \; d\omega = \frac{1}{4\pi} \phi(\mathbf{x})$$

As for the other coordinates, we use the cartesian form. For the first harmonics we have:

$$L_{-1,1}(x, \vec{\omega}) Y_{-1,1}(\vec{\omega}) = \sqrt{\frac{3}{4\pi}} \omega_x \int_{4\pi} \sqrt{\frac{3}{4\pi}} \omega_x L(\mathbf{x}, \vec{\omega}) \; d\omega = \frac{3}{4\pi} \omega_x E_x(x)$$

Where the $x$ subscript indicates the first component. Similarly we obtain:

$$L_{0,1}(x, \vec{\omega}) Y_{0,1}(\vec{\omega}) = \frac{3}{4\pi} \omega_z E_z(x)$$

$$L_{1,1}(x, \vec{\omega}) Y_{1,1}(\vec{\omega}) = \frac{3}{4\pi} \omega_y E_y(x)$$

By applying the approximation, we finally obtain:

$$L(x, \vec{\omega}) \approx \frac{1}{4\pi} \phi(\mathbf{x}) + \frac{3}{4\pi} \omega_x E_x(x) + \frac{3}{4\pi} \omega_y E_y(x) + \frac{3}{4\pi} \omega_z E_z(x) = \frac{\phi(\mathbf{x})}{4\pi} + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(\mathbf{x})$$

## 3 The diffusion equation

We substitute the diffusion approximation into the radiative transfer equation:

$$\begin{aligned}(\nabla \cdot \vec{\omega}) \left( \frac{\phi(\mathbf{x})}{4\pi} + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(\mathbf{x}) \right) &= -\sigma_t \left( \frac{\phi(\mathbf{x})}{4\pi} + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(\mathbf{x}) \right) \\ &\quad + \sigma_s \int_{4\pi} p(\vec{\omega}, \vec{\omega}') \left( \frac{\phi(\mathbf{x})}{4\pi} + \frac{3}{4\pi} \vec{\omega}' \cdot \vec{E}(\mathbf{x}) \right) \; d\omega' + Q(\mathbf{x}, \vec{\omega})\end{aligned}$$

We will need the following three identities to go on:

$$\int_{4\pi} \vec{\omega} \; d\omega = 0$$

$$\int_{4\pi} \vec{\omega} (\vec{\omega} \cdot \vec{A}) \; d\omega = \frac{4\pi}{3} \vec{A} \tag{2}$$

[1] `https://www.cs.dartmouth.edu/~wjarosz/publications/dissertation/appendixB.pdf`

$$\int_{4\pi} \vec{\omega}[\vec{\omega} \cdot \nabla(\vec{\omega} \cdot \vec{A})] \; d\omega = 0$$

We now further simplify the equation above:

$$\frac{1}{4\pi}\vec{\omega}\cdot\nabla\phi(\mathbf{x})+\frac{3}{4\pi}\vec{\omega}\cdot\nabla(\vec{\omega}\cdot\vec{E}(\mathbf{x})) = -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}-\sigma_t\frac{3}{4\pi}\vec{\omega}\cdot\vec{E}(\mathbf{x})+\sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega},\vec{\omega}') \; d\omega' + \frac{3}{4\pi}\sigma_s\int_{4\pi} p(\vec{\omega},\vec{\omega}')\vec{\omega}' \cdot \vec{E}(\mathbf{x}) \; d\omega' + Q(\mathbf{x},\vec{\omega})$$

Now we multiply each term by $\vec{\omega}$ and integrate on the sphere. We take all the terms separately, then put them together at the end.

$$\int_{4\pi} \frac{1}{4\pi}\vec{\omega} \cdot \nabla\phi(\mathbf{x})\vec{\omega} \; d\omega = \frac{1}{4\pi}\frac{4\pi}{3}\nabla\phi(\mathbf{x}) = \frac{\nabla\phi(\mathbf{x})}{3}$$

$$\int_{4\pi} \frac{3}{4\pi}\vec{\omega} \cdot \nabla(\vec{\omega} \cdot \vec{E}(\mathbf{x}))\vec{\omega} \; d\omega = \frac{3}{4\pi}\int_{4\pi} \vec{\omega}[\vec{\omega} \cdot \nabla(\vec{\omega} \cdot \vec{E}(\mathbf{x}))] \; d\omega = 0$$

$$\int_{4\pi} -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}\vec{\omega} \; d\omega = -\sigma_t\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} \vec{\omega} \; d\omega = 0$$

$$\int_{4\pi} -\sigma_t\frac{3}{4\pi}\vec{\omega} \cdot \vec{E}(\mathbf{x})\vec{\omega} \; d\omega = -\sigma_t\frac{3}{4\pi}\int_{4\pi} \vec{\omega}(\vec{\omega} \cdot \vec{E}(\mathbf{x})) \; d\omega = -\sigma_t\frac{3}{4\pi}\frac{4\pi}{3}\vec{E}(\mathbf{x}) = -\sigma_t\vec{E}(\mathbf{x})$$

$$\int_{4\pi} \sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} p(\vec{\omega},\vec{\omega}') \; d\omega'\vec{\omega} \; d\omega = \sigma_s\frac{\phi(\mathbf{x})}{4\pi}\int_{4\pi} \vec{\omega} \; d\omega = 0$$

$$\int_{4\pi} \frac{3}{4\pi}\sigma_s\int_{4\pi} p(\vec{\omega},\vec{\omega}')\vec{\omega}' \cdot \vec{E}(\mathbf{x}) \; d\omega'\vec{\omega} \; d\omega = \int_{4\pi} \frac{3}{4\pi}\sigma_s g(\vec{\omega} \cdot \vec{E}(\mathbf{x}))\vec{\omega} \; d\omega = g\sigma_s\vec{E}(\mathbf{x})$$

$$\int_{4\pi} Q(\mathbf{x},\vec{\omega})\vec{\omega} \; d\omega = \vec{\mathbf{Q}}(\mathbf{x},\vec{\omega})$$

Putting everything together:

$$\frac{\nabla\phi(\mathbf{x})}{3} + 0 = 0 - \sigma_t\vec{E}(\mathbf{x}) + 0 + g\sigma_s\vec{E}(\mathbf{x}) + \vec{\mathbf{Q}}(\mathbf{x},\vec{\omega})$$

$$\nabla\phi(\mathbf{x}) = -3\sigma_t'\vec{E}(\mathbf{x}) + 3\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) \tag{3}$$

Where we used $\sigma_t' = \sigma_s' + \sigma_a = \sigma_s(1-g) + \sigma_a = \sigma_t - g\sigma_s$. We combine 3 and 1 we obtain the final diffusion equation. First we rearrange 3:

$$\vec{E}(\mathbf{x}) = 3D\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) - D\nabla\phi(\mathbf{x})$$

Where $D = -\frac{1}{3\sigma_t'}$. Inserting it into 1:

$$\nabla \cdot (3D\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) - D\nabla\phi(\mathbf{x})) = -\sigma_a\phi(\mathbf{x}) + q(\mathbf{x})$$

$$3D\nabla \cdot \vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) - D\nabla^2\phi(\mathbf{x}) = -\sigma_a\phi(\mathbf{x}) + q(\mathbf{x})$$

$$D\nabla^2\phi(\mathbf{x}) = \sigma_a\phi(\mathbf{x}) - q(\mathbf{x}) + 3D\nabla \cdot \vec{\mathbf{Q}}(\mathbf{x},\vec{\omega})$$

Which is the final diffusion equation in Jensen.

# 4 Boundary condition

In the case of a scattering medium in a finite region of space, we impose the classic boundary condition that the net inward flux on each surface point $x_s$ with normal $\vec{n}_s$ is zero:

$$\int_{2\pi_-} L(x_s, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s)\ d\omega = 0$$

We use the diffusion approximation:

$$\int_{2\pi_-} \left( \frac{\phi(\mathbf{x})}{4\pi} + \frac{3}{4\pi} \vec{\omega} \cdot \vec{E}(\mathbf{x}) \right) (\vec{\omega} \cdot \vec{n}_s)\ d\omega = 0$$

$$\phi(\mathbf{x}) \int_{2\pi_-} (\vec{\omega} \cdot \vec{n}_s)\ d\omega + 3 \int_{2\pi_-} (\vec{\omega} \cdot \vec{E}(\mathbf{x}))(\vec{n}_s \cdot \vec{\omega})\ d\omega = 0$$

Given the standard spherical coordinates convention, $n_s = (0, 0, 1)$ and $\vec{\omega} = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$. We obtain then:

$$\int_{2\pi_-} (\vec{\omega} \cdot \vec{n}_s)\ d\omega = \int_0^{2\pi} \int_{\frac{\pi}{2}}^{\pi} \cos\theta \sin\theta d\theta d\phi = -\pi$$

And:

$$\int_{2\pi_-} (\vec{\omega} \cdot \vec{E}(\mathbf{x}))(\vec{n}_s \cdot \vec{\omega})\ d\omega = -\int_0^{2\pi} \int_0^{\frac{\pi}{2}} (\cos\phi \sin\theta E_x + \sin\phi \sin\theta E_y + \cos\theta E_z) \cos\theta \sin\theta d\theta d\phi$$

$$= -\frac{2\pi}{3} E_z = -\frac{2\pi}{3} \vec{n}_s \cdot \vec{E}(\mathbf{x})$$

Using the last two results and simplifying, we get:

$$\phi(\mathbf{x})(-\pi) + 3 \left( -\frac{2\pi}{3} \vec{n}_s \cdot \vec{E}(\mathbf{x}) \right) = 0$$

$$\phi(\mathbf{x}) + 2\vec{n}_s \cdot \vec{E}(\mathbf{x}) = 0$$

From Equation 3, assuming no point source, we have the $\vec{\mathbf{Q}}(\mathbf{x}, \vec{\omega})$ term go to zero, so $\vec{E}(\mathbf{x}) = -D\nabla\phi(\mathbf{x})$. Inserting this result and simplifying, we get the final boundary condition:

$$\phi(\mathbf{x}) - 2D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x}) = 0$$

# 5 Different media assumption

To include surfaces of different boundaries, we need to change the above equations. The boundary condition then becomes:

$$I_- = \int_{2\pi_-} L(x_s, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s)\ d\omega = \int_{2\pi_+} R(\eta, \vec{\omega}) L(x_s, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s)\ d\omega = I_+$$

Keeping the above conventions, we define $R$ as a step function:

$$R(\eta, \vec{\omega}) = \begin{cases} 1 & \text{for} \quad \frac{\pi}{2} \leq \theta \leq \pi - \theta_c \\ R_0 & \text{for} \quad \pi - \theta_c \leq \theta \leq \pi \end{cases}$$

Where $\theta_c$ is the critical angle, and $R_0 = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$, where $n_1$ and $n_2$ are indices of refraction of vacuum and medium, respectively. We use $\pi - \theta_c$ as the critical angle is conventionally defined from the normal pointing *inside* the surface.

The left side of equation 5 is:

$$I_- = \pi\phi(\mathbf{x}) - 2\pi D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x})$$

The other side is more tricky, since it requires splitting the integration along the different angles. We proceed as before, introducing the diffusion approximation:

$$I_+ = \phi(\mathbf{x}) \int_{2\pi_+} R(\eta, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s)\ d\omega + 3 \int_{2\pi_+} R(\eta, \vec{\omega})(\vec{\omega} \cdot \vec{E}(\mathbf{x}))(\vec{n}_s \cdot \vec{\omega})\ d\omega$$

As before, we calculate the integral separately. The first part:

$$\int_{2\pi_+} R(\eta, \vec{\omega})(\vec{\omega} \cdot \vec{n}_s)\ d\omega = \int_0^{2\pi} \int_{\frac{\pi}{2}}^{\pi - \theta_c} \cos\theta \sin\theta d\theta d\phi + \int_0^{2\pi} \int_{\pi - \theta_c}^{\pi} R_0 \cos\theta \sin\theta d\theta d\phi =$$
$$= ((R_0 - 1)\cos^2\theta_c - R_0)\pi$$

The second part (only on the z coordinate, since the other coordinates always go to zero):

$$\int_{2\pi_+} R(\eta, \vec{\omega})(\vec{\omega} \cdot \vec{E}(\mathbf{x}))(\vec{n}_s \cdot \vec{\omega})\ d\omega =$$
$$E_z \int_0^{2\pi} \int_{\frac{\pi}{2}}^{\pi - \theta_c} \cos^2\theta \sin\theta d\theta d\phi + E_z \int_0^{2\pi} \int_{\pi - \theta_c}^{\pi} R_0 \cos^2\theta \sin\theta d\theta d\phi =$$
$$\frac{2\pi}{3} E_z (R_0(1 - \cos^3\theta_c) + \cos^3\theta_c)$$

Performing all simplifications, we finally get:

$$I_+ = ((R_0 - 1)\cos^2\theta_c - R_0)\pi\phi(x) + 2\pi D(R_0(1 - \cos^3\theta_c) + \cos^3\theta_c)(\vec{n}_s \cdot \nabla)\phi(\mathbf{x})$$

We can now impose $I_+ = I_-$:

$$\pi\phi(\mathbf{x}) + 2\pi D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x}) = ((R_0 - 1)\cos^2\theta_c - R_0)\pi\phi(x) + 2\pi D(R_0(1 - \cos^3\theta_c) + \cos^3\theta_c)(\vec{n}_s \cdot \nabla)\phi(\mathbf{x})$$

That we can simplify as:

$$\phi(\mathbf{x}) + 2\frac{(1 - (R_0(1 - \cos^3\theta_c) + \cos^3\theta_c))}{(1 - ((R_0 - 1)\cos^2\theta_c - R_0))} D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2\frac{\left(\frac{2}{1 - R_0} - 1 + \cos^3\theta_c\right)}{1 - \cos^2\theta_c} D(\vec{n}_s \cdot \nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2AD(\vec{n}_s \cdot \nabla)\phi(\mathbf{x}) = 0$$

So, to handle the different media, we need to add a corrective factor $A$ in our boundary condition.

### 5.1 Approximating the corrective factor

We can rewrite the $I_+$ term in equation 5 as:

$$\int_{2\pi_+} R(\eta,\vec{\omega})L(x_s,\vec{\omega})(\vec{\omega}\cdot\vec{n}_s)\,d\omega \approx \int_{2\pi_+} R(\eta,\vec{\omega})(\vec{\omega}\cdot\vec{n}_s)\,d\omega \int_{2\pi_+} L(x_s,\vec{\omega})(\vec{\omega}\cdot\vec{n}_s)\,d\omega$$

$$= F_{dr}(\eta)\ [\pi\phi(\mathbf{x}) + 2\pi D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x})]$$

The $F_{dr}(\eta)$ integral function can be approximated as:

$$F_{dr}(\eta) = -\frac{1.440}{\eta^2} + \frac{0.710}{\eta} + 0.668 + 0.0636\eta$$

So we can express the boundary condition as:

$$\pi\phi(\mathbf{x}) - 2\pi D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = F_{dr}(\eta)\ [\pi\phi(\mathbf{x}) + 2\pi D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x})]$$

$$\phi(\mathbf{x}) - 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = F_{dr}[\phi(\mathbf{x}) + 2D(\vec{n}_s\cdot\nabla)\phi(\mathbf{x})]$$

$$\phi(\mathbf{x})(1-F_{dr}) - 2D(1+F_{dr})(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2D\frac{1+F_{dr}}{1-F_{dr}}(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

$$\phi(\mathbf{x}) - 2AD(\vec{n}_s\cdot\nabla)\phi(\mathbf{x}) = 0$$

With $A = \frac{1+F_{dr}}{1-F_{dr}}$.

## 6 Solutions for an infinite medium

From the telegraph equation:

$$(D\nabla^2 - \sigma_a)\phi(\mathbf{x}) = -q(\mathbf{x}) + 3D\nabla\cdot\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega})$$

$$(\nabla^2 - \sigma_{tr}^2)\phi(\mathbf{x}) = -\frac{q(\mathbf{x})}{D} + 3\nabla\cdot\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega})$$

The equation above is a particular case of screened Poisson equation, that has a generic solution:

$$\phi(\mathbf{x}) = \frac{1}{4\pi}\iiint_{\mathbb{R}^3} \frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left(\frac{q(\mathbf{r}')}{D} - 3\nabla\cdot\vec{\mathbf{Q}}(\mathbf{r}',\vec{\omega})\right)d^3\mathbf{r}'$$

### 6.1 Point source solution

If we use a simple point source model, we have:

$$q(\mathbf{x}) = \Phi_i\sigma_s'\delta(x)\delta(y)\Theta(z)e^{-\sigma_t' z}$$

$$\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) = 0$$

Plugging in the above sources:

$$\phi(\mathbf{x}) = \frac{1}{4\pi}\iiint_{\mathbb{R}^3} \frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left(\frac{\Phi_i\sigma_s'\delta(x)\delta(y)\Theta(z)e^{-\sigma_t' z}}{D}\right)d^3\mathbf{r}'$$

Applying the deltas:

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D}\int_0^{+\infty} \frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}\left(\sigma_s' e^{-\sigma_t' z}\right) dz$$

Given the assumption of point source, we can assume $\|\mathbf{x}-z\mathbf{e}_z\| \approx \|\mathbf{x}\| = r$. So we get:

$$\phi(\mathbf{x}) = \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\sigma_s'\int_0^{+\infty} e^{-\sigma_t' z}dz = \frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\frac{\sigma_s'}{\sigma_t'} = \alpha'\frac{\Phi_i}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}$$

Where $\alpha' = \frac{\sigma_s'}{\sigma_t'}$ is the reduced albedo. So we get the first solution in an infinite medium (the one used in Jensen):

$$\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}$$

Where $\Phi = \alpha'\Phi_i$.

## 6.2 Ray source solution

In case of ray sources, we have the following equations for the source terms:

$$q(\mathbf{x}) = \Phi\tilde{\sigma}_s\delta(x)\delta(y)\Theta(z)e^{-\tilde{\sigma}_t z}$$

$$\vec{\mathbf{Q}}(\mathbf{x},\vec{\omega}) = \tilde{g}q(\mathbf{x})\mathbf{e}_z$$

Plugging it in in 6, we get two integrals. $\phi(\mathbf{x}) = \phi_1(\mathbf{x}) + \phi_2(\mathbf{x})$, where:

$$\begin{aligned}\phi_1(\mathbf{x}) &= \frac{1}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left(\frac{\Phi\tilde{\sigma}_s\delta(x)\delta(y)\Theta(z)e^{-\tilde{\sigma}_t z}}{D}\right)d^3\mathbf{r}' \\ &= \frac{\Phi\tilde{\sigma}_s}{4\pi D}\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}e^{-\tilde{\sigma}_t z}dz \\ &= \frac{3\tilde{\sigma}_s\Phi}{4\pi}(\tilde{\sigma}_s(1-\tilde{g})+\sigma_a)\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}e^{-\tilde{\sigma}_t z}dz.\end{aligned}$$

And:

$$\phi_2(\mathbf{x}) = -\frac{3}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\nabla\cdot\left(\Phi\tilde{\sigma}_s\tilde{g}\delta(x)\delta(y)\Theta(z)e^{-\tilde{\sigma}_t z}\right)\mathbf{e}_z d^3\mathbf{r}'.$$

We now apply the divergence operator:

$$\phi_2(\mathbf{x}) = -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\left[\delta(x)\delta(y)\frac{\partial}{\partial z}(\Theta(z)e^{-\tilde{\sigma}_t z})\right]d^3\mathbf{r}'.$$

Note that we have only the $z$ term given that we multiply by $\mathbf{e}_z$. Using that $\frac{\partial\Theta(z)}{\partial z} = \delta(z)$:

$$\phi_2(\mathbf{x}) = -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\delta(x)\delta(y)\left[\delta(z)e^{-\tilde{\sigma}_t z} - \tilde{\sigma}_t\Theta(z)e^{-\tilde{\sigma}_t z}\right]d^3\mathbf{r}'.$$

Given the three deltas in the first term, we can explicitly solve the integral for $\mathbf{r}' = (x, y, z) = \mathbf{0}$.

$$\phi_2(\mathbf{x}) = -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\Phi\tilde{\sigma}_s\tilde{g}\tilde{\sigma}_t}{4\pi}\iiint_{\mathbb{R}^3}\frac{e^{-\sigma_{tr}\|\mathbf{x}-\mathbf{r}'\|}}{\|\mathbf{x}-\mathbf{r}'\|}\delta(x)\delta(y)\Theta(z)e^{-\tilde{\sigma}_t z}d^3\mathbf{r}'.$$

$$= -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\Phi\tilde{\sigma}_s\tilde{g}\tilde{\sigma}_t}{4\pi}\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}e^{-\tilde{\sigma}_t z}dz.$$

Putting it together:

$$\phi(\mathbf{x}) = -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\tilde{\sigma}_s\Phi}{4\pi}(\tilde{\sigma}_s(1-\tilde{g})+\sigma_a+\tilde{\sigma}_s\tilde{g}+\sigma_a\tilde{g})\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}e^{-\tilde{\sigma}_t z}dz$$

$$\phi(\mathbf{x}) = -\frac{3\Phi\tilde{\sigma}_s\tilde{g}}{4\pi}\frac{e^{-\sigma_{tr}r}}{r} + \frac{3\tilde{\sigma}_s\Phi}{4\pi}(\tilde{\sigma}_s+\sigma_a(1+\tilde{g}))\int_0^{+\infty}\frac{e^{-\sigma_{tr}\|\mathbf{x}-z\mathbf{e}_z\|}}{\|\mathbf{x}-z\mathbf{e}_z\|}e^{-\tilde{\sigma}_t z}dz$$

We can interpret the second term in Eq. (6.2) as the fluence from an exponentially decaying line source along the $z$ axis. Due to this exponentially decaying factor, the integrand will only have a significant weight for small $z$. Hence, in the asymptotic limit, $r \gg 1/\tilde{\sigma}_s$, we can approximate the distances in the integrand

$$\begin{aligned}|\mathbf{x}-z\mathbf{e}_z| &= \sqrt{r^2+z^2-2zr\cos\theta} = r\left(1-2\frac{z}{r}\cos\theta+\frac{z^2}{r^2}\right)^{\frac{1}{2}} \\ &\approx r\left(1-\frac{z}{r}\cos\theta\right) = r - z\cos\theta,\end{aligned}$$

and

$$\begin{aligned}\frac{1}{|\mathbf{x}-z\mathbf{e}_z|} &= \frac{1}{\sqrt{r^2+z^2-2zr\cos\theta}} = \frac{1}{r}\left(1-2\frac{z}{r}\cos\theta+\frac{z^2}{r^2}\right)^{-\frac{1}{2}} \\ &\approx \frac{1}{r}\left(1+\frac{z}{r}\cos\theta\right),\end{aligned}$$

$$\begin{aligned}\int_0^\infty \frac{e^{-(\tilde{\sigma}_s+\sigma_a)z}e^{-\sigma_{tr}|\mathbf{x}-z\mathbf{e}_z|}}{|\mathbf{x}-z\mathbf{e}_z|}dz &\approx \frac{e^{-\sigma_{tr}r}}{r}\int_0^\infty e^{-(\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta)z}\left(1+\frac{z}{r}\cos\theta\right)dz \\ &= \frac{e^{-\sigma_{tr}r}}{r}\left(\frac{1}{\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta}+\frac{\cos\theta}{r}\frac{1}{(\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta)^2}\right).\end{aligned}$$

In an highly scattering medium $\sigma_a \ll \tilde{\sigma}_s$. For $g \neq 1$ this will imply and $\sigma_a \ll \sigma_{tr} \ll \tilde{\sigma}_s$, and, hence

$$\frac{1}{\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta} = \frac{1}{\tilde{\sigma}_s}\left(1+\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)^{-1} \approx \frac{1}{\tilde{\sigma}_s}\left(1-\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right) \tag{4}$$

and

$$\frac{1}{(\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta)^2} = \frac{1}{\tilde{\sigma}_s^2}\left(1+\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)^{-2} \approx \frac{1}{\tilde{\sigma}_s^2}\left(1-2\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right). \tag{5}$$

The integral can now be approximated by

$$\begin{aligned}
\int_0^\infty \frac{e^{-(\tilde{\sigma}_s+\sigma_a)z}e^{-\sigma_{tr}|\mathbf{x}-z\mathbf{e}_z|}}{|\mathbf{x}-z\mathbf{e}_z|}dz &\approx \frac{e^{-\sigma_{tr}r}}{r}\left(\frac{1}{\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta}+\frac{\cos\theta}{r}\frac{1}{(\tilde{\sigma}_s+\sigma_a-\sigma_{tr}\cos\theta)^2}\right)\\
&\approx \frac{e^{-\sigma_{tr}r}}{\tilde{\sigma}_s r}\left(1-\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s r}\left(1-2\frac{\sigma_a-\sigma_{tr}\cos\theta}{\tilde{\sigma}_s}\right)\right)\\
&= \frac{e^{-\sigma_{tr}r}}{\tilde{\sigma}_s r}\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s}\left(\sigma_{tr}+\frac{1}{r}-2\frac{\sigma_a}{\tilde{\sigma}_s r}\right)+2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}\right).
\end{aligned}$$

Inserting this expression in Eq. (6.2)

$$\begin{aligned}
\frac{\phi(\mathbf{x})}{\Phi} &\approx -\frac{3\tilde{g}\tilde{\sigma}_s e^{-\sigma_{tr}r}}{4\pi r}\\
&\quad+\frac{3\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)e^{-\sigma_{tr}r}}{4\pi r}\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}+\frac{\cos\theta}{\tilde{\sigma}_s}\left(\sigma_{tr}+\frac{1}{r}-2\frac{\sigma_a}{\tilde{\sigma}_s r}\right)+2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}\right)\\
&= \frac{3e^{-\sigma_{tr}r}}{4\pi r}\left(\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)\left(1-\frac{\sigma_a}{\tilde{\sigma}_s}\right)-\tilde{g}\tilde{\sigma}_s\right)\\
&\quad+\frac{3e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(1+(\tilde{g}+1)\frac{\sigma_a}{\tilde{\sigma}_s}\right)\left(\sigma_{tr}+\frac{1}{r}-\frac{2\sigma_a}{\tilde{\sigma}_s r}\right)\\
&\quad+\frac{3\left(\tilde{\sigma}_s+(\tilde{g}+1)\sigma_a\right)e^{-\sigma_{tr}r}}{4\pi r}2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s^2 r}.
\end{aligned}$$

Now we neglect terms $\sigma_a/\tilde{\sigma}_s$ compared to unity

$$\frac{\phi(\mathbf{x})}{\Phi}\approx\frac{3e^{-\sigma_{tr}r}}{4\pi r}\tilde{\sigma}_s\left(1-\tilde{g}\right)+\frac{3e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(\sigma_{tr}+\frac{1}{r}\right)+\frac{3e^{-\sigma_{tr}r}}{4\pi r}2\frac{\sigma_{tr}\cos^2\theta}{\tilde{\sigma}_s r}.$$

Introducing the reduced scattering coefficient $\tilde{\sigma}_t=\tilde{\sigma}_s(1-\tilde{g})=\mu(1-g)$ and neglecting terms $\sigma_{tr}/\tilde{\sigma}_s\cdot 1/(\tilde{\sigma}_s r)$ compared to unity

$$\phi(\mathbf{x})\approx\frac{3\Phi e^{-\sigma_{tr}r}}{4\pi r}\tilde{\sigma}_t+\frac{3\Phi e^{-\sigma_{tr}r}}{4\pi r}\cos\theta\left(\sigma_{tr}+\frac{1}{r}\right)=\frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r}\left(1+3D\frac{1+\sigma_{tr}r}{r}\cos\theta\right).$$

For highly scattering media, the reduced scattering coefficient, $\tilde{\sigma}_s$ can be approximated by the reduced extinction coefficient, because absorption will be negligible compared to scattering.

# 7 Fresnel integrals

We define the first two order Fresnel integrals, to aid in our calculations:

$$C_\phi(\eta)=\frac{1}{4\pi}\int_{2\pi}T_{21}(\eta,\theta_o)\cos\theta_o\mathrm{d}\vec{\omega}_o$$

$$C_{\mathbf{E}}(\eta)=\frac{3}{4\pi}\int_{2\pi}T_{21}(\eta,\theta_o)\cos^2\theta_o\mathrm{d}\vec{\omega}_o$$

Where $\cos\theta_o=\vec{n}_o\cdot\vec{\omega}_o$, and the integral is on the whole hemisphere where $\vec{n}_o\cdot\vec{\omega}_o>0$.

# 8 BSSRDF theory

The BSSRDF is defined as the ration of an element of emergent radiance $L_r$ and an element of incident flux $\Phi_i$:

$$S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = \frac{\mathrm{d}L_r(\mathbf{x}_o, \vec{\omega}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

There are various approximations available for the BSSRDF. For the directional dipole, we use the delta-Eddington approximation:

$$S = T_{12}(S_{\delta E} + S_d)T_{21}$$

Let us consider only the diffusive part, $S_d$. The emergent radiance due to diffusion events is given by

$$L_{r,d}(\mathbf{x}_o, \vec{\omega}_o) = \eta^2 T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21})$$

Where $\vec{\omega}_{21}$ is the refracted vector corresponding to $\vec{\omega}_o$:

$$\vec{\omega}_{21} = \frac{\vec{\omega}_o}{\eta} - \left( \frac{\vec{n}_o \cdot \vec{\omega}_o}{\eta} - \sqrt{1 - \frac{1 - (\vec{n}_o \cdot \vec{\omega}_o)^2}{\eta^2}} \right) \vec{n}_o$$

Combining the above equations we obtain:

$$S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = T_{12} S_d T_{21} = \frac{\mathrm{d}L_{r,d}(\mathbf{x}_o, \vec{\omega}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} = \eta^2 \frac{\mathrm{d}T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21})}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

We now integrate both sides over the cosine weighted hemisphere where $\vec{n}_o \cdot \vec{\omega}_o > 0$:

$$\int_{2\pi} T_{12} S_d T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o = \int_{2\pi} \eta^2 \frac{\mathrm{d}T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21})}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} \cos\theta_o \mathrm{d}\vec{\omega}_o$$

We now assume no directional dependency, $S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o)$:

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) \int_{2\pi} T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o = \eta^2 \frac{\mathrm{d} \int_{2\pi} T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21}) \cos\theta_o \mathrm{d}\vec{\omega}_o}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) 4\pi \frac{C_\phi(\eta)}{\eta^2} = \frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

$$T_{12} S_d(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o) 4\pi C_\phi(1/\eta) = \frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)}$$

Let us calculate the diffuse exitant radiance first. Plug in the diffusion approximation into the exitant radiance formula:

$$M_d(\mathbf{x}_o) = \int_{2\pi} T_{21} L_d(\mathbf{x}_o, \vec{\omega}_{21}) \cos\theta_o \mathrm{d}\vec{\omega}_o = \int_{2\pi} T_{21} \left( \frac{\phi(\mathbf{x}_o)}{4\pi} - \frac{3}{4\pi} D \; \vec{\omega}_{21} \cdot \nabla\phi(\mathbf{x}_o) \right) \cos\theta_o \mathrm{d}\vec{\omega}_o$$

$$= \underbrace{\frac{\phi(\mathbf{x}_o)}{4\pi} \int_{2\pi} T_{21} \cos\theta_o \mathrm{d}\vec{\omega}_o}_{I_\phi} - \underbrace{\frac{3}{4\pi} D \int_{2\pi} \vec{\omega}_{21} \cdot \nabla\phi(\mathbf{x}_o) \cos\theta_o \mathrm{d}\vec{\omega}_o}_{I_{\mathbf{E}}}$$

And, in a straightforward way,

$$I_\phi = C_\phi(\eta)\phi(\mathbf{x}_o)$$

The second part is more complicated. Without loss of generality, we rotate the reference coordinate system so that $\vec{\omega}_o = (\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)$ and $\vec{n}_o = (0,0,1)$. Given this reference system, the refracted vector becomes:

$$\vec{\omega}_{21} = \frac{(\cos\phi\sin\theta, \sin\phi\sin\theta, \cos\theta)}{\eta} - \left(\frac{\cos\theta}{\eta} - \sqrt{1 - \frac{\sin^2\theta}{\eta^2}}\right)(0,0,1)$$

$$= \left(\frac{\cos\phi\sin\theta}{\eta}, \frac{\sin\phi\sin\theta}{\eta}, \sqrt{1 - \frac{\sin^2\theta}{\eta^2}}\right)$$

When plugged into the dot product in equation $I_{\mathbf{E}}$, we get a sum of three components. The first two terms of this sum go to zero, since we can first integrate over $\phi$:

$$\int_0^{2\pi} \cos\phi \mathrm{d}\phi = \int_0^{2\pi} \sin\phi \mathrm{d}\phi = 0.$$

So we are left only with the cumbersome $z$ term:

$$I_{\mathbf{E}} = \frac{3}{4\pi} D(\nabla\phi_z(\mathbf{x}_o)) \int_0^{2\pi}\int_0^{\frac{\pi}{2}} T_{21}(\eta,\theta_o)\sqrt{1 - \frac{\sin^2\theta_o}{\eta^2}} \cos\theta_o \sin\theta_o \mathrm{d}\theta_o \mathrm{d}\phi$$

Where we note that $\nabla\phi_z(\mathbf{x}_o) = \nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o$. Note that without loss of generality we can assume $\eta > 1$, so that the argument of the square root is never negative. We now perform a substitution using Snell's law, $\sin\theta_o = \eta\sin\theta_i$. We obtain the following identities:

$$\mathrm{d}\theta_o = \frac{\eta\cos\theta_i}{\cos\theta_o}\mathrm{d}\theta_i, \quad T_{21}(\eta,\theta_o) = T_{21}(1/\eta,\theta_i), \quad \sqrt{1 - \frac{\sin^2\theta_o}{\eta^2}} = \cos\theta_i.$$

If we introduce a critical angle $\theta_c = \arcsin(1/\eta)$. We finally get:

$$I_{\mathbf{E}} = \frac{3}{4\pi} D\nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o \int_0^{2\pi}\int_0^{\theta_c} T_{21}(1/\eta,\theta_i)\cos\theta_i\cos\theta_o\eta\sin\theta_i\frac{\eta\cos\theta_i}{\cos\theta_o}\mathrm{d}\theta_i\mathrm{d}\phi$$

$$= D\nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o\frac{3}{4\pi}\eta^2\int_0^{2\pi}\int_0^{\theta_c} T_{21}(1/\eta,\theta_i)\cos^2\theta_i\sin\theta_i\mathrm{d}\theta_i\mathrm{d}\phi$$

$$= D\nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o\eta^2 C_{\mathbf{E}}(1/\eta) = C_{\mathbf{E}}(\eta)D\nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o.$$

We can finally get our final expression for $M_d(\mathbf{x}_o)$

$$M_d(\mathbf{x}_o) = C_\phi(\eta)\phi(\mathbf{x}_o) - C_{\mathbf{E}}(\eta)D\nabla\phi(\mathbf{x}_o)\cdot\vec{n}_o$$

Inserting into the expression, we derive the BSSRDF in an infinite medium:

$$S_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o) = \frac{1}{4\pi T_{12}C_\phi(1/\eta)}\frac{\mathrm{d}M_d(\mathbf{x}_o)}{\mathrm{d}\Phi_i(\mathbf{x}_i,\vec{\omega}_i)}$$

## 8.1 Jensen's dipole

Using the expression from Jensen:

$$\phi(\mathbf{x}) = \alpha' \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r}$$

We get the gradient as:

$$\nabla\phi(\mathbf{x}) = -\alpha' \frac{\Phi_i}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r^3}(1 + \sigma_{tr} r)\mathbf{x}$$

So we can calculate the BSSRDF in the infinite medium:

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{1}{4\pi C_\phi(1/\eta)} \frac{\alpha'}{4\pi^2} \frac{e^{-\sigma_{tr} r}}{r^3} \left[ C_\phi(\eta) \frac{r^2}{D} + C_{\mathbf{E}}(\eta)(1 + \sigma_{tr} r)\mathbf{x} \cdot \vec{n}_o \right]$$

Where we used $\Phi_i = T_{12}\Phi_i(\mathbf{x}_i, \vec{\omega}_i)$. This is the better dipole from Eugene d'Eon. To get Jensen's dipole, we further approximate the above expression so that $C_\phi(1/\eta) = 1/4$, $C_\phi(\eta) = 0$, $C_{\mathbf{E}}(\eta) = 1$:

$$S_d(\mathbf{x}_i, \mathbf{x}_o) = \frac{\alpha'}{4\pi^2} \frac{e^{-\sigma_{tr} r}}{r^3}(1 + \sigma_{tr} r)\mathbf{x} \cdot \vec{n}_o$$

This is Jensen's final dipole in an infinite medium.

## 8.2 Directional dipole

For the directional dipole, we have the following form:

$$\phi(\mathbf{x}) = \frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r} \left( 1 + 3D \frac{1 + \sigma_{tr} r}{r} \cos\theta \right).$$

Is is more convenient to do the gradient in spherical coordinates:

$$\nabla\phi(\mathbf{x}) = \frac{\partial}{\partial r}\phi(\mathbf{x})\vec{e}_r + \frac{1}{r}\frac{\partial}{\partial \theta}\phi(\mathbf{x})\vec{e}_\theta + \frac{1}{r\sin\theta}\frac{\partial}{\partial \phi}\phi(\mathbf{x})\vec{e}_\phi$$

Where:

$$\frac{\partial}{\partial r}\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r^2} \left( 3D \frac{2(1 + \sigma_{tr} r) + (\sigma_{tr} r)^2}{r} \cos\theta + (1 + \sigma_{tr} r) \right)$$

And:

$$\frac{\partial}{\partial \theta}\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r^2} 3D(1 + \sigma_{tr} r)\sin\theta$$

Finally, $\frac{\partial}{\partial \phi}\phi(\mathbf{x}) = 0$. Given our choice of basis for derivation, we can use the following identities:

$$\vec{e}_r = \frac{\mathbf{x}}{r}$$

$$-\vec{e}_\theta \sin\theta = \vec{\omega}_{12} - \vec{e}_r \cos\theta$$

Inserting the identities, we get the final expression for the gradient:

$$\nabla\phi(\mathbf{x}) = -\frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r^2} \left( 3D \frac{2(1 + \sigma_{tr} r) + (\sigma_{tr} r)^2}{r} \cos\theta + (1 + \sigma_{tr} r) \right) \vec{e}_r + \frac{1}{r} \frac{\Phi}{4\pi D} \frac{e^{-\sigma_{tr} r}}{r^2} 3D(1 + \sigma_{tr} r)(-\vec{e}_\theta \sin\theta)$$

$$\nabla\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^3}\Bigg[\left(-3D\frac{2(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta - (1+\sigma_{tr}r)\right)r\vec{e}_r$$
$$-3D(1+\sigma_{tr}r)\cos\theta\vec{e}_r + 3D(1+\sigma_{tr}r)\vec{\omega}_{12}]$$

$$\nabla\phi(\mathbf{x}) = \frac{\Phi}{4\pi D}\frac{e^{-\sigma_{tr}r}}{r^3}\Bigg[-\left(3D\frac{3(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r}\cos\theta + (1+\sigma_{tr}r)\right)\mathbf{x}$$
$$+3D(1+\sigma_{tr}r)\vec{\omega}_{12}]$$

We can do now the same as above, obtaining the BSSRDF:

$$S'_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o) = \frac{1}{4\pi C_\phi(1/\eta)}\frac{1}{4\pi^2}\frac{e^{-\sigma_{tr}r}}{r^3}\Bigg[C_\phi(\eta)(\frac{r^2}{D} + 3(1+\sigma_{tr}r)\mathbf{x}\cdot\vec{\omega}_{12})$$
$$-C_{\mathbf{E}}(\eta)\left[3D(1+\sigma_{tr}r)\vec{\omega}_{12}\cdot\vec{n}_o - \left((1+\sigma_{tr}r)+3D\frac{3(1+\sigma_{tr}r)+(\sigma_{tr}r)^2}{r^2}\mathbf{x}\cdot\vec{\omega}_{12}\right)\mathbf{x}\cdot\vec{n}_o\right]\Bigg]$$

Alternative form, using $t = 1+\sigma_{tr}r$:

$$S'_d(\mathbf{x}_i,\vec{\omega}_i,\mathbf{x}_o) = \frac{1}{4\pi C_\phi(1/\eta)}\frac{\sigma_{tr}}{4\pi^2}\frac{e^{1-t}}{(t-1)^3}\Bigg[\frac{C_\phi}{D}(t-1)^2 + 3C_\phi\sigma_{tr}t(t-1)(\vec{x}\cdot\vec{\omega}_{12})$$
$$-3C_{\mathbf{E}}D\sigma_{tr}^2 t(\vec{n}_o\cdot\vec{\omega}_{12}) + C_{\mathbf{E}}\sigma_{tr}^2(t+1)(\vec{x}\cdot\vec{n}_o) + C_{\mathbf{E}}D\sigma_{tr}^3\frac{t^2+t+1}{t-1}(\vec{x}\cdot\vec{\omega}_{12})(\vec{x}\cdot\vec{n}_o)\Bigg]$$

Appendix IX

# Point cloud method for rendering BSSRDFs

# Point cloud method for rendering BSSRDFs

Jeppe Revall Frisvad
Technical University of Denmark

Alessandro Dal Corso
Technical University of Denmark

March 2018

This is a short note on rendering a triangular mesh onto which we apply a BSSRDF model. The method works on arbitrary static triangular meshes and is unbiased.

We first define some quantities relative to our mesh. We consider a triangle mesh a set $M = \{T_i, i \in [0, N_\Delta - 1]\}$ composed of $N_\Delta$ triangles. Each triangle $T_i$ is composed of three vertices:

$$T_i = \{\mathbf{v}_0^i, \mathbf{v}_1^i, \mathbf{v}_2^i\}$$

From these quantities it straightforward to define two derived quantities, the normal $\vec{n}_i$ and the area $A_i$ of the triangle.

$$\vec{n}_i = \frac{(\mathbf{v}_1^i - \mathbf{v}_0^i) \times (\mathbf{v}_2^i - \mathbf{v}_0^i)}{\|(\mathbf{v}_1^i - \mathbf{v}_0^i) \times (\mathbf{v}_2^i - \mathbf{v}_0^i)\|}$$

$$A_i = \frac{1}{2}\|(\mathbf{v}_1^i - \mathbf{v}_0^i) \times (\mathbf{v}_2^i - \mathbf{v}_0^i)\|$$

Once we have defined our triangles, we can start describing out our solution.

Theoretically, we solve the standard reflectance equation for BSSRDFs:

$$L_r(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_\Omega S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_i, \vec{\omega}_i)(\vec{n}_i \cdot \vec{\omega}_i) dA_i d\vec{\omega}_i$$

We now solve the integral with Monte Carlo integration with importance sampling. We take $K$ samples from the area integral and one sample from the directional integral:

$$\begin{aligned} \hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) &\approx \sum_{k=1}^{K} \frac{S(\mathbf{x}_{i,k}, \vec{\omega}_{i,k}, \mathbf{x}_o, \vec{\omega}_o) L_i(\mathbf{x}_{i,k}, \vec{\omega}_{i,k})(\vec{n}_{i,k} \cdot \vec{\omega}_{i,k})}{\mathrm{pdf}(\vec{\omega}_{i,k})\mathrm{pdf}(\mathbf{x}_{i,k})} = \\ &= \sum_{k=1}^{K} S(\mathbf{x}_{i,k}, \vec{\omega}_{i,k}, \mathbf{x}_o, \vec{\omega}_o) \Phi_{i,k}(\mathbf{x}_{i,k}, \vec{\omega}_{i,k}) \end{aligned} \tag{1}$$

The algorithm is composed of two phases. In the first phase, we generate the $K$ samples on the surface of the model, storing for each position $\mathbf{x}_{i,k}$, direction $\vec{\omega}_{i,k}$ and incoming flux $\Phi_{i,k}$. In the second phase, we render the final image evaluating the contribution at each pixel.

In phase one, we produce $K$ samples. The $k$-th sample is composed of a position $\mathbf{x}_{i,k}$, a direction $\vec{\omega}_{i,k}$ and a flux $\Phi_{i,k}$. First, we choose a triangle $T_j$. We choose the triangle uniformly, so given a random number $\xi \in [0,1)$, $j = \lfloor \xi N_\Delta \rfloor$. To get $\mathbf{x}_k$, we sample a point uniformly on the triangle $T_j$:

$$\mathbf{x}_{i,k} = (1-\sqrt{\xi_0})\mathbf{v}_0^j + (1-\xi_1)\sqrt{\xi_0}\mathbf{v}_1^j + \xi_1\sqrt{\xi_0}\mathbf{v}_2^j$$

Where $\xi_0, \xi_1 \in [0,1)$.

Now, the sampling direction $\vec{\omega}_{i,k}$ and the flux $\Phi_{i,k}$ need to be evaluated. We leave the choice of the sampling distribution for $\vec{\omega}_{i,k}$ to implementation, since it is not important for the method. From sampling the light source we obtain $\vec{\omega}_{i,k}$ and an irradiance $E_{i,k}$:

$$E_{i,k}(\mathbf{x}_{i,k}, \vec{\omega}_{i,k}) = \frac{L_i(\mathbf{x}_{i,k}, \vec{\omega}_{i,k})(\vec{n}_{i,k} \cdot \vec{\omega}_{i,k})}{\mathrm{pdf}(\vec{\omega}_{i,k})}$$

We need the pdf of choosing the point $\mathrm{pdf}(\mathbf{x}_{i,k})$. Since we sampled the triangle and then the point, we combine the two independent probabilities:

$$\mathrm{pdf}(\mathbf{x}_{i,k}) = \frac{1}{N_\Delta A_j}$$

Putting it all together:

$$\Phi_{i,k}(\mathbf{x}_{i,k}, \vec{\omega}_{i,k}) = \frac{E_{i,k}(\mathbf{x}_{i,k}, \vec{\omega}_k)}{\mathrm{pdf}(\mathbf{x}_{i,k})} = N_\Delta A_j E_{i,k}(\mathbf{x}_{i,k}, \vec{\omega}_{i,k})$$

Once we have stored this quantity in the sample, we are ready to proceed to the second and final phase. For each pixel, we ray trace a camera ray, obtaining a point $\mathbf{x}_o$ and a direction towards the camera $\vec{\omega}_o$. We consider the scattering medium with coefficients $\sigma_a, \sigma_s, g$ and relative index of refraction $\eta$ as a pure Fresnel interface with added scattering.

In our path tracing implementation, we perform russian roulette on the Fresnel coefficient $R$ to choose reflected or refracted direction. In the case of reflection, we continue tracing the reflected ray. In the case of refraction, we calculate the final radiance as:

$$\hat{L}_o(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + L_t(\mathbf{x}_o, \vec{\omega}_o) + \hat{L}_r(\mathbf{x}_o, \vec{\omega}_o)$$

Where $L_e$ is the emitted radiance from the medium, $L_t$ is the radiance due to direct transmission and $\hat{L}_r$ comes from 1.

To efficiently render the model and save expensive BSSRDF evaluations, we do a last optimization. We evaluate the BSSRDF stochastically with probability $\exp(-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_k\|)$, where $\sigma_{tr} = \sqrt{3\sigma_a(\sigma_a + (1-g)\sigma_s)}$ is the effective transport coefficient. So, only samples that are very close to the exit point $\mathbf{x}_o$ will be actually evaluated. In formulas:

$$\hat{L}_r(\mathbf{x}_o, \vec{\omega}_o) \approx \sum_{k=1}^{K} S(\mathbf{x}_k, \vec{\omega}_k, \mathbf{x}_o, \vec{\omega}_o)\Phi_{i,k}(\mathbf{x}_k, \vec{\omega}_k) V(\xi, e^{-\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_k\|})\, e^{\sigma_{tr}\|\mathbf{x}_o - \mathbf{x}_k\|}$$

Where:

$$V(\xi, d) = \begin{cases} 1 & \text{if } \xi < d \\ 0 & \text{otherwise} \end{cases}$$

In case of a spectral RGB rendering, we pick $\sigma_{tr}$ as the mean of the three rgb components.

Appendix X

# On BSSRDF estimation

# On BSSRDF estimation

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

January 2018

Figure 1: Configuration for random walk procedure.

The configuration of the bidirectional scattering-surface reflectance distribution function (BSSRDF) is illustrated in Figure 1. Our aim is to estimate the BSSRDF for a specific spatial area bin with area $A_o$ and for a specific angular bin with solid angle $\Omega_o$. In a measurement environment [3], we have the following incoming flux:

$$\Phi_i(A_i, \Omega_i) = \int_{A_i} \int_{\Omega_i} L_i(\mathbf{x}'_i, \vec{\omega}'_i)(\vec{\omega}'_i \cdot \vec{n}_i)\, d\omega'_i\, dA'_i\,,$$

where $L_i$ is radiance incident somewhere on the surface area $A_i$ from the solid angle $\Omega_i$. As illustrated in Figure 1, $\vec{n}_i$ is the surface normal at a point of incidence $\mathbf{x}'_i$, while $\vec{\omega}'_i$ is a direction of incidence. The corresponding responsivity-weighted emergent flux is:

$$\Phi_o(A_o, \Omega_o) = \int_{A_o} \int_{\Omega_o} \int_{A_i} \int_{\Omega_i} L_i(\mathbf{x}'_i, \vec{\omega}'_i) S(\mathbf{x}'_i, \vec{\omega}'_i, \mathbf{x}'_o, \vec{\omega}'_o) R(\mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_i \cdot \vec{n}_i)\, d\omega'_i\, dA_i\, (\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,,$$

where $R$ is the relative responsivity of the instrument used for a measurement and $S$ is the BSSRDF. Note that we choose our notation so that $\Phi_o(A_o, \Omega_o)$ rep-

resents the emergent flux arriving in the bin of interest. In an idealized computational setting, we ignore the responsivity of the sensor and set $R(\mathbf{x}'_o, \vec{\omega}'_o) = 1$. If we choose a unit response function and unit incident radiance $L_i(\mathbf{x}_i, \vec{\omega}_i) = \delta(\mathbf{x}_i - \mathbf{x}'_i)\delta(\vec{\omega}_i - \vec{\omega}'_i)$, the ratio of emergent to incident flux is

$$G(A_o, \Omega_o) = \frac{\Phi_o(A_o, \Omega_o)}{\Phi_i(A_i, \Omega_i)} = \int_{A_o} \int_{\Omega_o} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,.$$

Once we have obtained $G$, we can rearrange the above equation to obtain the BSSRDF:

$$S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = \frac{dL_o(\mathbf{x}_o, \vec{\omega}_o)}{d\Phi_i(\mathbf{x}_i, \vec{\omega}_i)} \approx \frac{\Phi_o(A_o, \Omega_o)}{\Phi_i(A_i, \Omega_i) A_o \Omega_o (\vec{\Omega}_o \cdot \vec{n}_o)} = \frac{G(A_o, \Omega_o)}{A_o \Omega_o (\vec{\Omega}_o \cdot \vec{n}_o)}\,, \tag{1}$$

where $\vec{\Omega}_o$ is the direction around which the bin solid angle is centered. This means that we can represent the light scattering between two points, the BSSRDF, as the ratio of emergent to incoming flux divided by the cosine-weighted area and the solid angle.

The BSSRDF of a medium can be defined as an operator that includes all radiance resulting from the scattering events in the medium [1]. This radiance is described locally by the radiative transfer equation [1]. Thus, we can evaluate the scattering process given by the BSSRDF using the radiative transfer equation. So, the above quantity $G$ becomes all flux carried by a flux packet arriving in a bin after going through a random walk. This is the procedure used by Wang et al. [4].

The random walk procedure works as follows:

1. Create a new photon with flux $\Phi_t = 1/N$, where $N$ is the number of photons. Start assigning $\mathbf{x}_p = \mathbf{x}_i$ and $\vec{\omega}_p = \vec{\omega}_{12}$, the refracted incoming direction.

2. Sample a new distance to the next scattering event $s$. We assume a homogenous medium, so we use an exponential distribution with decay $\sigma_t$, that can be easily importance sampled:

$$s = \frac{-\log(1-\xi)}{\sigma_t}\,, \quad \text{with } \xi \in [0, 1).$$

3. Check if the next scattering event $\mathbf{x}'_p = \mathbf{x}_p + s\vec{\omega}_p$ is within the medium.

4. If the next scattering event is within the medium:

   (a) Terminate the path with probability $1 - \alpha$. If killed, start from step 1.

   (b) If not terminated, sample a new direction $\vec{\omega}_p$ according to the phase function.

   (c) Continue from step 2.

5. If the next scattering event is outside of the medium:

6. Calculate the reflection Fresnel coefficient $R_{21}$ and update $\mathbf{x}_p$ to be the intersection point on the surface of the medium.

(a) With probability $R_{21}$ reflect the path around $-\vec{n}_o$, and continue from point 2.

(b) With probability $1 - R_{21}$ refract the path outside, storing $\Phi_t$ in the corresponding bin. Start from step 1.

Note that all the sampling is chosen to cancel out the terms that would change the stored flux. So, the flux within the medium should change for each scattering event as:

$$\Phi_t' = \Phi_t \sigma_s p(\vec{\omega}_p \cdot \vec{\omega}_p) e^{-\sigma_t s}$$

Although, dividing by the pdfs, $p(\vec{\omega}_p \cdot \vec{\omega}_p')$ cancels out if we sample $\vec{\omega}_p$ according to the correct phase function, $e^{-\sigma_t s}$ cancels with the pdf of the distance sampling, leaving $\sigma_s/\sigma_t = \alpha$. This last factor is divided out by the path continuation probability, leading to $\Phi_t' = \Phi_t$ as expected. If the flux goes outside the medium, the Fresnel coefficients cancel with the probabilities of reflection or transmission.

# 1 Variance reduction through connections

Figure 2: Configuration for the connection event described in the text.

The procedure from the previous section provides an unbiased estimate of the BSSRDF. However, this process requires a huge number of photons to converge to an acceptable solution. To reduce convergence times, we connect each scattering event (except the first, to measure multiple scattering only), to a point $\mathbf{x}_o$ on the surface. We then use the three point form of the local scattering equation [2] to calculate the resulting measurement:

$$L(\mathbf{x}_o, \vec{\omega}_o) = L_e(\mathbf{x}_o, \vec{\omega}_o) + L(\mathbf{x}_p \to \mathbf{x}_o) f(\mathbf{x}_p \to \mathbf{x}_o \to \mathbf{x}_z) G(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o)\,.$$

We assume a non emissive body ($L_e(\mathbf{x}_o, \vec{\omega}_o) = 0$). In our case, we have a point in the volume generating a scattering event ($\mathbf{x}_p$), a point in a bin on the

surface ($\mathbf{x}_o$), and a point specifying a direction into the bin ($\mathbf{x}_z$), see Figure 2. So, we have the incoming radiance:

$$L(\mathbf{x}_p \to \mathbf{x}_o) = \alpha \Phi_t(\mathbf{x}_p) p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g)\,,$$

the three point scattering function corresponding to the bidirectional transmittance distribution function (BTDF) of a perfectly smooth surface:

$$\begin{aligned} f(\mathbf{x}_p \to \mathbf{x}_o \to \mathbf{x}_z) &= T_{21}(\eta, \vec{\omega}_o) \frac{\delta(\vec{\omega}_o - \vec{\omega}_t)}{|\vec{\omega}_{21} \cdot \vec{n}_o|}\,, \\ \vec{\omega}_t &= \eta^{-1}(|\vec{\omega}_{21} \cdot \vec{n}_o| \vec{n}_o - \vec{\omega}_{21}) - \vec{n}_o \sqrt{1 - \eta^{-2}(1 - (\vec{\omega}_{21} \cdot \vec{n}_o)^2)}\,, \end{aligned}$$

the geometry term:

$$G(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = \frac{|\vec{\omega}_{21} \cdot \vec{n}_o|}{\|\mathbf{x}_o - \mathbf{x}_p\|^2}\,,$$

and the generic visibility term:

$$V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = V'(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) \exp\left( - \int_0^{\|\mathbf{x}_o - \mathbf{x}\|} \sigma_t \left(\mathbf{x}_p + t\vec{\omega}_{21}\right) dt \right)\,.$$

Here, $T_{21} = 1 - R_{21}$ is the Fresnel transmittance at the point of emergence, and one should note that $T_{21}(\eta^{-1}, -\vec{\omega}_{21}) = T_{21}(\eta, \vec{\omega}_o)$.

In our configuration of a semi infinite plane, the binary visibility function $V'(\mathbf{x}_p \leftrightarrow \mathbf{x}_o)$ is always one (i.e., it is always possible to connect to the emergent point). Since we have a homogenous medium, we can rewrite $V$ as the beam transmittance:

$$V(\mathbf{x}_p \leftrightarrow \mathbf{x}_o) = \exp(-\sigma_t \|\mathbf{x}_o - \mathbf{x}_p\|)\,.$$

And the final radiance of the $p$th scattering event of the $k$th photon becomes

$$L_{k,p}(\mathbf{x}_o, \vec{\omega}_o) = \alpha \Phi_t(\mathbf{x}_p) p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) T_{21}(\eta, \vec{\omega}_o) \frac{\delta(\vec{\omega}_o - \vec{\omega}_t)}{\|\mathbf{x}_o - \mathbf{x}_p\|^2} \exp(-\sigma_t \|\mathbf{x}_o - \mathbf{x}_p\|)$$

To get the collective contribution of a photon path, we need to sum all the elements of the random walk:

$$L_k(\mathbf{x}_o, \vec{\omega}_o) = \sum_p L_{k,p}(\mathbf{x}_o, \vec{\omega}_o)\,.$$

And to get the final emergent flux, we need to solve the measurement equation for each bin. This is integration of incident flux (given by the three point form of the local scattering equation) across the cosine weighted area and the solid angle of the bin:

$$\Phi_o(A_o, \Omega_o) = \int_{A_o} \int_{\Omega_o} L(\mathbf{x}'_o, \vec{\omega}'_o)(\vec{\omega}'_o \cdot \vec{n}_o)\, d\omega'_o\, dA'_o\,.$$

Now, since the three point scattering function is a delta function with respect to direction, the integration over bin solid angle cancels with this delta. As for the area integral, we solve it using Monte Carlo integration. The samples for the integral then become the photons, giving

$$\Phi_o(A_o, \Omega_o) = \frac{1}{N} \sum_{k=1}^{N} \sum_p \alpha \Phi_t p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) T_{21}(\eta, \vec{\omega}_o) \frac{\vec{\omega}_o \cdot \vec{n}_o}{\|\mathbf{x}_o - \mathbf{x}_p\|^2} e^{-\sigma_t \|\mathbf{x}_o - \mathbf{x}_p\|} \frac{\|\mathbf{x}_o - \mathbf{x}_i\|}{\mathrm{pdf}(\mathbf{x}_o)}\,,$$

where we sampled $\mathbf{x}_o$ according to a uniform pdf in polar coordinates. If we think our spatial bin $A_o$ as a circular sector delimited by $r_{\min}$ and $r_{\max}$ in radius, and $\theta_{s,\min}$ and $\theta_{s,\max}$. We have $\text{pdf}(\mathbf{x}_o) = 1/(\Delta r \Delta\theta_s)$, where $\Delta r = r_{\max} - r_{\min}$ and $\Delta\theta_s = \theta_{s,\max} - \theta_{s,\min}$. Since we use a circular sector, we obtain:

$$A_o = \iint_A dA = \int_{r_{\min}}^{r_{\max}} \int_{\theta_{s,\min}}^{\theta_{s,\max}} r dr d\theta_s = \frac{r_{\max}^2 - r_{\min}^2}{2} \Delta\theta_s = \frac{r_{\min} + r_{\max}}{2} \Delta r \Delta\theta_s$$

Also, since we are using a uniform sampling of the hemisphere, we obtain a constant solid angle for all directional bins:

$$\Omega_o = \frac{2\pi}{N_{bins}}$$

Plugging everything in the original BSSRDF equation (1) and simplifying, we obtain the final BSSRDF:

$$\begin{aligned} S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o) = & \frac{1}{N} \sum_{k=1}^{N} \sum_{p} \alpha \Phi_t p(\vec{\omega}_p \cdot \vec{\omega}_{21}, g) \frac{T_{21}(\eta, \vec{\omega}_o)}{\vec{\Omega}_o \cdot \vec{n}_o} \frac{\vec{\omega}_o \cdot \vec{n}_o}{\|\mathbf{x}_o - \mathbf{x}_p\|^2} e^{-\sigma_t \|\mathbf{x}_o - \mathbf{x}_p\|} \\ & \|\mathbf{x}_o - \mathbf{x}_i\| \frac{2}{r_{\min} + r_{\max}} \frac{N_{bins}}{2\pi} \end{aligned}$$

Note that we used $\Phi(\mathbf{x}_i, \vec{\omega}_i) = 1$.

# References


[1] Radiative transfer on discrete spaces. by r. w. preisendorfer. oxford, london, edinburgh, new york, paris, frankfurt. (pergamon press), (1st edn.) 1965. pp. xi, 462: Many figures, tables. £7. *Quarterly Journal of the Royal Meteorological Society*, 93(395):148–148, 1967.

[2] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In Alexander Keller, Stefan Heinrich, and Harald Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–605, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[3] William H. Venable, Jr. and Jack J. Hsia. Optical Radiation Measurements: Describing Spectrophotometric Measurements. 1974.

[4] Lihong Wang, Steven L. Jacques, and Liqiong Zheng. Mcml—monte carlo modeling of light transport in multi-layered tissues. *Computer Methods and Programs in Biomedicine*, 47(2):131 – 146, 1995.

Appendix XI

# Camera space BSSRDF importance sampling

# Camera space BSSRDF importance sampling

Alessandro Dal Corso
Technical University of Denmark

Jeppe Revall Frisvad
Technical University of Denmark

September 2017

This note explain the importance sampling technique in camera space from Mertens et al. [1].

## 1 Planar surfaces

We start from the classical BSSRDF form of the rendering equation:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_A \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i dA$$

Where $\Omega(\mathbf{x}_i)$ is an hemisphere placed in $\mathbf{x}_i$ with normal $\vec{n}_i$. Our first step is to convert the integration into polar coordinates. We assume for now that the integration happens on a plane. In polar coordinates, we have $dA = rdrd\theta$. The integral becomes:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_0^{2\pi} \int_0^{+\infty} \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i r dr d\theta$$

Where $r = \|\mathbf{x}_o - \mathbf{x}_i\|$. Also, $\mathbf{x}_i = \mathbf{x}_o + r\cos\theta\ \vec{t}_o + r\sin\theta\ \vec{b}_o$, where $\vec{t}_o$ and $\vec{b}_o$ form an orthonormal basis with $\vec{n}_o$.

We can now perform Monte Carlo integration, that gives the estimator:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{1}{NM} \sum_{p=1}^{M} \sum_{q=1}^{N} \frac{L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_{i,q} \cdot \vec{n}_{i,p}) r}{\mathrm{pdf}(r, \theta)\mathrm{pdf}(\omega_{i,q})}$$

Now, we can importance sample our disc coordinates $(r, \theta)$ using the joint pdf:

$$\mathrm{pdf}(r, \theta) = \frac{1}{2\pi} \sigma_{tr} e^{-\sigma_{tr} r}$$

And we can sample the incoming light direction using a cosine weighted hemisphere:

$$\mathrm{pdf}(\omega_{i,q}) = \frac{\vec{\omega}_{i,q} \cdot \vec{n}_{i,p}}{\pi}$$

So the integral can be finally approximated as:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{2\pi^2}{NM\sigma_{tr}} \sum_{p=1}^{M} \sum_{q=1}^{N} L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) r e^{\sigma_{tr} r}$$

Note that the $re^{\sigma_{tr} r}$ cancels out some terms inside the BSSRDF.

## 2 Non planar surfaces

We now assume that the surface is no longer planar. So we want to make a change of variables so that our integral becomes tractable, i.e. we are able to perform it in the tangent plane. Let us call $dA_{tan}$ an element of area in the tangent plane. The change of variables simply becomes:

$$L_o(\mathbf{x}_o, \vec{\omega}_o) = \int_{A_{tan}} \int_{\Omega(\mathbf{x}_i)} L_i(\mathbf{x}_i, \vec{\omega}_i) S(\mathbf{x}_i, \vec{\omega}_i, \mathbf{x}_o, \vec{\omega}_o)(\vec{\omega}_i \cdot \vec{n}_i) d\omega_i \left| \frac{dA}{dA_{tan}} \right| dA_{tan}$$

Where $dA$ is the corresponding area element on the real surface, continuing a camera ray. We call the camera ray $\vec{d} = \frac{\mathbf{x}_i - \mathbf{e}}{\|\mathbf{x}_i - \mathbf{e}\|}$, where $\mathbf{e}$ is the camera position.

We only need to estimate the Jacobian of the change of variables $\left| \frac{dA}{dA_{tan}} \right|$. We observe that, since we are using a pinhole camera, the solid angles subtended by the projected area elements must be equal. We calculate them and put them equal, that gives:

$$\frac{dA(\vec{n}_i \cdot -\vec{d})}{d^2} = \frac{dA_{tan}(\vec{n}_o \cdot -\vec{d})}{d_{tan}^2}$$

Where $d = \|\mathbf{x}_i - \mathbf{e}\|$ and $d_{tan} = \|\mathbf{x}_{i,tan} - \mathbf{e}\|$ are the distances with the real point and the point in tangent space, respectively. The Jacobian then becomes:

$$\left| \frac{dA}{dA_{tan}} \right| = \frac{(\vec{n}_o \cdot -\vec{d})}{(\vec{n}_i \cdot -\vec{d})} \frac{d^2}{d_{tan}^2}$$

Note that for a plane, $\vec{n}_i = \vec{n}_o$ and $d = d_{tan}$, giving a unitary Jacobian as expected.

After the change of variables, we can now proceed as in the previous section, and obtaining the final estimator:

$$L_o^{N,M}(\mathbf{x}_o, \vec{\omega}_o) = \frac{2\pi^2}{NM\sigma_{tr}} \sum_{p=1}^{M} \sum_{q=1}^{N} L_i(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}) S(\mathbf{x}_{i,p}, \vec{\omega}_{i,q}, \mathbf{x}_o, \vec{\omega}_o) \frac{(\vec{n}_o \cdot -\vec{d})}{(\vec{n}_i \cdot -\vec{d})} \frac{d^2}{d_{tan}^2} r e^{\sigma_{tr} r}$$

## References


[1] Tom Mertens, Jan Kautz, Philippe Bekaert, Hans-Peter Seidel, and Frank Van Reeth. Interactive rendering of translucent deformable objects. In *Proceedings of Eurographics Symposium on Rendering (EGSR 2003)*, pages 130–140, June 2003.

# Bibliography


[ACDC+15] Henrik Aanæs, Knut Conradsen, Alessandro Dal Corso, Anders Bjorholm Dahl, A Del Bue, Mads Emil Brix Doest, Jeppe Revall Frisvad, Sebastian Hoppe Nesgaard Jensen, Jannik Boll Nielsen, Jonathan Dyssel Stets, et al. Our 3d vision datasets in the making. In *CVPR Workshop: The Future of Datasets in Vision 2015*, 2015.

[AK90] James Arvo and David Kirk. Particle transport and image synthesis. *Computer Graphics (Proceedings of SIGGRAPH '90*, 24(4):63–66, August 1990.

[Cha50] Subrahmanyan Chandrasekhar. *Radiative Transfer*. Oxford, Clarendon Press, 1950. Unabridged and slightly revised version published by Dover in 1960.

[Cha58] Subrahmanyan Chandrasekhar. On the diffuse reflection of a pencil of radiation by a plane-parallel atmosphere. *Proceedings of the National Academy of Sciences of the United States of America*, 44(9):933–940, September 1958.

[DCFKB16] Alessandro Dal Corso, Jeppe Revall Frisvad, Thomas Kim Kjeldsen, and Jakob Andreas Bærentzen. Interactive Appearance Prediction for Cloudy Beverages. In Reinhard Klein and Holly Rushmeier, editors, *Workshop on Material Appearance Modeling*. The Eurographics Association, 2016.

[DCFMB17] Alessandro Dal Corso, Jeppe Revall Frisvad, Jesper Mosegaard, and J. Andreas Bærentzen. Interactive directional subsurface scat-

tering and transport of emergent light. *The Visual Computer*, 33(3):371–383, Mar 2017.

[DCOS+15] A. Dal Corso, M. Olsen, K. H. Steenstrup, J. Wilm, S. Jensen, R. R. Paulsen, E. Eiríksson, J. Nielsen, J. R. Frisvad, G. Einarsson, and H. M. Kjer. Virtualtable: A projection augmented reality game. In *SIGGRAPH Asia 2015 Posters*, SA '15, pages 40:1–40:1, New York, NY, USA, 2015. ACM.

[DCSK+17] Alessandro Dal Corso, Marco Salvi, Craig Kolb, Jeppe Revall Frisvad, Aaron Lefohn, and David Luebke. Interactive stable ray tracing. In *Proceedings of High Performance Graphics*, HPG '17, pages 1:1–1:10, New York, NY, USA, 2017. ACM.

[DCSL+17] Alessandro Dal Corso, Jonathan Dyssel Stets, Andrea Luongo, Jannik Boll Nielsen, Jeppe Revall Frisvad, and Henrik Aanæs. Virtual reality inspection and painting with measured brdfs. In *SIGGRAPH Asia 2017 VR Showcase*, SA '17, pages 15:1–15:2, New York, NY, USA, 2017. ACM.

[d'E12] Eugene d'Eon. A better dipole. Publicly available technical report. http://www.eugenedeon.com/?project=a-better-dipole, 2012.

[dI11] Eugene d'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4):56:1–56:13, July 2011.

[DLR+09] Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. An empirical bssrdf model. *ACM Trans. Graph.*, 28(3):30:1–30:10, July 2009.

[EVNT78] G. Eason, A. R. Veitch, R. M. Nisbet, and F. W. Turnbull. The theory of the back-scattering of light by blood. *Journal of Physics D: Applied Physics*, 1978(10):1463–1479, July 1978.

[FHK14] Jeppe Revall Frisvad, Toshiya Hachisuka, and Thomas Kim Kjeldsen. Directional dipole model for subsurface scattering. *ACM Transactions on Graphics*, 34(1):5:1–5:12, November 2014.

[FPW92] Thomas J. Farrell, Michael S. Patterson, and Brian Wilson. A diffusion theory model of spatially resolved, steady-state diffuse reflectance for the noninvasive determination of tissue optical properties *in vivo*. *Medical Physics*, 19(4):879–888, July/August 1992.

[FWP03] Alexander L. Fetter, John Dirk Walecka, and Physics. *Theoretical Mechanics of Particles and Continua (Dover Books on Physics)*. Dover Publications, 2003.

[HG40] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Annales d'Astrophysique*, 3:117–137, 1940. Also in *The Astrophysical Journal 93*, 1941.

[JB02] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3):576–581, July 2002.

[JMLH01] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, pages 511–518. ACM, August 2001.

[KA13] Tero Karras and Timo Aila. Fast parallel construction of high-quality bounding volume hierarchies. In Kayvon Fatahalian and Christian Theobalt, editors, *Proceedings of High Performance Graphics (HPG '13)*, pages 89–99. ACM, 2013.

[Kaj86] James T. Kajiya. The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):143–150, August 1986.

[KW08] Malvin H. Kalos and Paula A. Whitlock. *Monte Carlo Methods*. Wiley-VCH, 2008.

[MPBM03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2003)*, 22(3):759–769, 2003.

[NRH+77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometrical considerations and nomenclature for reflectance. Technical report, National Bureau of Standards (US), October 1977.

[PBD+10] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M Stich. OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 29(4):66:1–66:13, 2010.

[Pre65] R W. Preisendorfer. *Radiative Transfer on Discrete Spaces*. Pergamon Press, 1965.

[Pre76] R.W. Preisendorfer. *Hydrologic optics*. U.S. Dept. of Commerce, National Oceanic and Atmospheric Administration, Environmental Research Laboratories, Pacific Marine Environmental Laboratory, 1976.

[Rus88] Holly Edith Rushmeier. *Realistic image synthesis for scenes with radiatively participating media.* PhD thesis, Cornell University, Ithaca, NY, USA, 1988.

[Rus98] Szymon Rusinkiewicz. A new change of variables for efficient BRDF representation. In *Eurographics Rendering Workshop*, pages 11–22, 1998.

[SDN+17] Jonathan Dyssel Stets, Alessandro Dal Corso, Jannik Boll Nielsen, Rasmus Ahrenkiel Lyngby, Sebastian Hoppe Nesgaard Jensen, Jakob Wilm, Mads Brix Doest, Carsten Gundlach, Eythor Runar Eiriksson, Knut Conradsen, Anders Bjorholm Dahl, Jakob Andreas Bærentzen, Jeppe Revall Frisvad, and Henrik Aanæs. Scene reassembly after multimodal digitization and pipeline evaluation using photorealistic rendering. *Appl. Opt.*, 56(27):7679–7690, Sep 2017.

[Sta95] Jos Stam. Multiple scattering as a diffusion process. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 41–50. Springer, June 1995. Proceedings of the Sixth Eurographics Workshop on Rendering.

[Sto09] George Gabriel Stokes. *On the perfect Blackness of the Central Spot in Newton's Rings, and on the Verification of Fresnel's Formula for the intensities of Reflected and Reflacted Rays*, volume 2 of *Cambridge Library Collection - Mathematics*, page 89–103. Cambridge University Press, 2009.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[WREE67] Chris Wylie, Gordon Romney, David Evans, and Alan Erdahl. Half-tone perspective drawings by computer. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, AFIPS '67 (Fall), pages 49–58, New York, NY, USA, 1967. ACM.