FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Extending Separable Subsurface Scattering to Arbitrary Materials

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Christian Freude
Matrikelnummer 0728278

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Mitwirkung: Projektass.(FWF) Károly Zsolnai, BSc MSc

Wien, 02.12.2014

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Extending Separable Subsurface Scattering to Arbitrary Materials

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Christian Freude
Registration Number 0728278

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Associate Prof. Dipl.-Ing. Dipl.-Ing. Dr.techn. Michael Wimmer
Assistance: Projektass.(FWF) Károly Zsolnai, BSc MSc

Vienna, 02.12.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Christian Freude
Fleischmarkt 18, 1010 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

First and foremost, I want to thank my supervisors Károly Zsolnai and Michael Wimmer for their outstanding support and patience, and for giving me the opportunity to contribute to their research as part of my thesis. I also want to thank Thomas Auzinger for his insightful advice and support on numerous occasions. Thanks to Jorge Jimenez, Adrian Jarabo and Diego Gutierrez for their collaboration.

Furthermore, I want to thank my family and especially my parents for supporting me and my studies, as well as God for his help and guidance. In addition, I also want to thank all my fellow students who accompanied and helped me throughout my studies.

Additionally, thanks to the creators (listed in Table A.2) of several 3D models used for this thesis, and Joao Henriques for sharing his useful 'textborder' function for MATLAB with the community.

Abstract

This thesis proposes extensions for the Separable Subsurface Scattering algorithm to support arbitrary materials. Four separable (rank-1) kernel models for the approximation of physically based diffuse reflectance profiles are presented. Each model offers different approximation quality and controllability. The first two models are based on singular value decomposition and a custom analytic pre-integration scheme. They enable fast deterministic kernel computation and provide fixed-quality solutions. Two additional parametrized models are based on automatic and manual optimization and provide more control over the approximation quality but are more time-consuming to generate. Higher rank approximations can be computed using the approach based on singular value decomposition.

All four kernel models are used to compute approximations for physically measured diffuse reflectance profiles of different materials and tested using several special-case irradiance signals and complex proof-of-concept scenes. The results are compared to the state of the art in real-time rendering of subsurface scattering, showing comparable approximation quality at lower computational cost. The proposed extensions enable rendering of physically based subsurface scattering for arbitrary materials and dynamic scenes in real time.

Kurzfassung

In dieser Diplomarbeit werden Erweiterungen des Separable Subsurface Scattering Algorithmus vorgestellt, welche es ermöglichen, diesen für beliebige Materialien zu verwenden. Darunter befinden sich vier separable (Rang-1) Filter-Modelle, welche die Approximation von physikalisch-basierten diffusen Reflexionsprofilen ermöglichen. Jedes Modell bietet hierbei einen unterschiedlichen Grad an Approximationsqualitat und Anpassungsmöglichkeit. Die ersten beiden Modelle ermöglichen eine schnelle und deterministische Berechnung der Filter und bieten eine fixe Approximationsqualität. Zwei weitere parametrisierte Modelle basieren auf automatischer und manueller Optimierung und ermöglichen mehr Kontrolle über die Approximationsqualität, wobei diese mehr Zeit zur Filtergenerierung benötigen. Approximationen von höherem Rang können mittels Singulärwertzerlegung berechnet werden.

Alle vier Filtermodelle werden verwendet, um gemessene diffuse Reflexionsprofile von verschiedenen Materialien zu approximieren und getestet, mittels der Berechnung von verschieden einfachen und komplexen Szenen. Die Ergebnisse werden mit dem heutigen Stand der Technik zur Berechnung von Subsurface Scattering in Echtzeit verglichen, und zeigen vergleichbare Approximationsqualität bei verringertem Berechungsaufwand. Die vorgestellten Erweiterungen ermöglichen das Rendern von physikalisch-basiertem Subsurface Scattering für beliebige Materialien und dynamische Szenen in Echtzeit.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Contributions . . . 3
- 1.4 Structure of the Work . . . 3

2 Basic Concepts **5**
- 2.1 Subsurface Scattering . . . 5
- 2.2 Volume Rendering Theory . . . 6
- 2.3 SSS Rendering . . . 9

3 Related Work **13**
- 3.1 Offline Rendering . . . 13
- 3.2 Real-time Rendering . . . 14
- 3.3 Separable Subsurface Scattering . . . 16

4 Separable Subsurface Scattering **19**
- 4.1 Main Idea . . . 19
- 4.2 Algorithm Overview . . . 21
- 4.3 SSS for Human Skin . . . 22
- 4.4 Implementation Details . . . 24
- 4.5 Additional Notes . . . 26
- 4.6 First Extension Attempts . . . 27

5 Separable Kernel Models **31**
- 5.1 Singular Value Decomposition . . . 32
- 5.2 Pre-Integration . . . 42
- 5.3 Guided Optimization . . . 44
- 5.4 Manual Approximation . . . 57

6 Results **61**
- 6.1 Initial Preparations . . . 61
- 6.2 Approximation Kernels . . . 63
- 6.3 Test Renderings . . . 65

6.4 Discussion 85

7 Conclusion **95**

A Appendix **97**

Bibliography **141**

CHAPTER 1

Introduction

1.1 Motivation

One interesting challenge in computer graphics is the computation of photo-realistic images and animations. Throughout the last decade, research and development of sophisticated algorithms and faster hardware has led to the point where many computer-generated images and animations are indistinguishable from real-life footage. Photo-realistic rendering is not only needed for the generation of special effects or even feature-length films for the movie industry, but has become more and more important for industrial visualisation. Furthermore, due to the incredibly fast development and huge processing power of modern GPUs, even video games often feature 3D graphics, that are close to photorealism.

However, whereas the film industry can afford to use offline rendering, with rendering times up to multiple minutes or hours per frame, games have to deliver real-time performance, meaning that one frame has to be computed in a few milliseconds. This constraint leads to the fact that in modern computer games, many rendering effects which are affordable in offline rendering are simplified or even ignored in order to achieve real-time performance. One example of such an effect is *subsurface scattering* (SSS), which plays an important role in the realistic appearance of many materials. It is therefore an interesting and challenging research endeavour to make SSS rendering feasible for use in games and other real-time applications.

Subsurface Scattering

Many physical materials are translucent to some extent. This has the effect that light is not reflected entirely at the surface, but some fraction is transmitted into or through the material. Inside, this light may get scattered multiple times before some fraction may leaf the material at some point or is completely absorbed. Highly translucent materials absorb only a small amount of light, while for materials with very low translucency, light is usually scattered multiple times inside the material and therefore its absorption becomes very likely. The latter are also referred to as optically thick, in which case the volumetric propagation of light inside the material is

referred to as subsurface scattering. Well-known examples are e.g. wax, marble, milk or human skin.

For the simulation and rendering of SSS it is necessary to take the material's volume into account and how light propagates throughout its medium. In a more general context, this is often referred to as rendering of participating media or volume rendering. Even in the context of offline rendering, computation of light propagation in a medium is a hard problem (outlined in more detail in Chapter 2), and rendering of a single image can take from several seconds up to multiple hours. Real-time applications, however, often have only a very limited time budget of a few milliseconds, and it is therefore necessary to develop sophisticated techniques in order to make real-time SSS rendering possible.

1.2 Problem Statement

Despite the importance of SSS for the realistic appearance of many surfaces and materials, it is often ignored in the context of real-time rendering to increase rendering performance. Although SSS is challenging, even for offline rendering, a variety of different methods for real-time SSS rendering have been developed over the past years (outlined in Section 3). A lot of effort was put into the approximation of SSS in human skin, as rendering of faces is quite a common and important task in the context of games.

In particular, one implementation of a technique by Jimenez and Gutierrez [23, 25], called *Separable Subsurface Scattering* (SSSS), demonstrated that SSS in human skin can be computed in approximately one millisecond (outlined in more detail in Chapter 4). This method approximates SSS via a post-processing step, in which the surface irradiance is blurred in screen space, in order to mimic light transmittance below the surface due to SSS. In general, such an approximation requires an expensive 2D convolution with a filter kernel specific to the rendered material. The main idea of the SSSS algorithm is to approximate the 2D filter kernel via a corresponding separable filter kernel, which makes it possible to perform the expensive 2D blurring operation via two fast 1D convolution passes. In case of the original SSSS method, the used separable filter kernel is derived via a parametrized model which is specifically designed and tuned to approximate SSS in human skin. Although the parameters provide some form of control over the SSS effect, the model is quite limited and therefore not suited for close approximation of materials other than human skin. The SSSS algorithm enables fast rendering via use of the separable convolution approach, whereas the limited filter kernel model hinders its application for the approximation of SSS in arbitrary materials.

Therefore, it is the aim of this thesis to propose a more general separable filter kernel model in order to extend this technique to support SSS rendering for arbitrary materials. For this purpose, it is necessary to find a separable filter kernel model which is able to approximate corresponding 2D filters, representing a wide variety of different materials. In general, arbitrary 2D filters are not separable, and therefore a close separable approximation may not be possible. However, the particular properties and form of filters used for SSS rendering via irradiance blurring enable a close approximation via low-rank or separable (rank-1) filter kernels.

1.3 Contributions

The main contribution of this work are several filter kernel models which can be used to extend the previously mentioned SSSS algorithm to support arbitrary materials. These proposed extensions of the SSSS algorithm were researched and developed in the course of a research project in close collaboration with Károly Zsolnai, Thomas Auzinger and Michael Wimmer from the Vienna University of Technology, Adrian Jarabo and Diego Gutierrez from the University of Zaragoza and Jorge Jimenez, Xian-Chun Wu and Javier von der Pahlen from Activision-Blizzard.

In order to support different materials, it is necessary to find an appropriate model which supports the close approximation of 2D filter kernels for arbitrary materials via a separable (rank-1) filter kernel. Over the course of multiple tests of various approaches, four different separable kernel approximation models were researched and developed, which act as replacements for the limited skin model of the original SSSS algorithm. These models enable the approximation of SSS for arbitrary materials, as they are more general and less limited than the model used in the original SSSS method.

The first of the proposed models provides a separable approximation of a 2D filter kernel via compression based on singular value decomposition. Furthermore, this model also supports higher-rank solutions, which enable higher approximation quality at the cost of increased rendering time. The second separable approximation approach uses a custom analytic pre-integration scheme, and is exact for a special class of irradiance signals. These two models support fast kernel generation while providing fixed-quality solutions. More control over the approximation quality, at the cost of increased kernel generation speed, is provided by two additional models that are based on optimization and manual approximation. The optimization-based approach computes the separable kernel by minimizing a parametrized function, which can be used to control the kernel shape and subsequently the approximation quality. The last kernel model was developed by Jorge Jimenez and is described in this thesis for the sake of completeness. It is based on manual approximation by the user, and provides a few intuitive parameters, which can be used to fully control the separable filter kernel. All four models offer varying levels of quality, computation speed and controllability, which makes it possible to select the approach that is most suited for a particular application.

In order to evaluate the models and the corresponding filter kernels, a series of rendering tests were performed. The different filters were applied to several artificial test signals for easy comparison. Furthermore, more complex proof-of-concept scenes were assembled and rendered to prove the practical application of the proposed models in the context of real-time SSS rendering of arbitrary materials in dynamic scenes.

1.4 Structure of the Work

The following chapter will outline some of the basic concepts and theories of subsurface scattering and volume rendering as well as common concepts used for SSS rendering. Chapter 3 gives and overview over existing SSS rendering methods and approaches, while Chapter 4 includes a more detailed explanation of the original SSSS algorithm. The proposed extensions to arbitrary

materials are presented in Chapter 5, and their evaluation for different materials using artificial tests and practical proof-of-concept renderings is shown in Chapter 6. This chapter also includes the discussion of the results and outlines several limitations. The last chapter represents a high-level overview of the proposed extensions and draws a final conclusion based on the presented results.

CHAPTER 2

Basic Concepts

This chapter provides an overview over basic concepts and theories regarding subsurface scattering and volume rendering in general. First, the basic concept of SSS is explained, followed by the introduction to volumetric rendering theory. Finally, a few aspects of practical SSS rendering and corresponding concepts are discussed.

2.1 Subsurface Scattering

In the real world many materials are more or less translucent. When light hits the surface of such a translucent material some part of the light gets reflected directly on the surface and some other part enters the medium below the surface. Inside this medium the light may get scattered multiple times before getting absorbed or leaving the material again at a certain point on the surface. This is illustrated in Figure 2.1. The more light gets absorbed by a certain material the less translucent it is. For materials with very low translucency, light is usually scattered multiple times inside its medium with the effect that its absorption becomes more and more likely after every scattering event. Such materials with high absorption and low translucency are often called optically thick. This volumetric propagation of light inside optically thick materials, including scattering and absorption, is often referred to as subsurface scattering. In Figure 2.2 examples of real-world subsurface scattering in human skin, marble and milk are shown, where the translucency and SSS is especially visible in the thinner edge regions. This are of course only a few examples and there are many more materials which exhibit similar subsurface scattering. For simulation and rendering of such materials it is necessary to model how light propagates throughout the material's volume and how light interacts with the medium. The corresponding theory and basic concepts are explained in the following section.

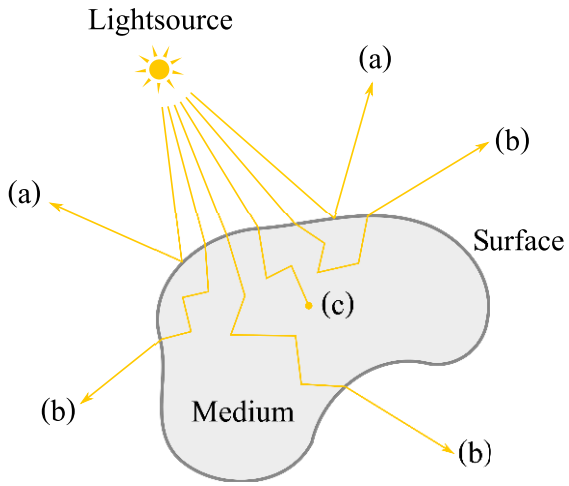


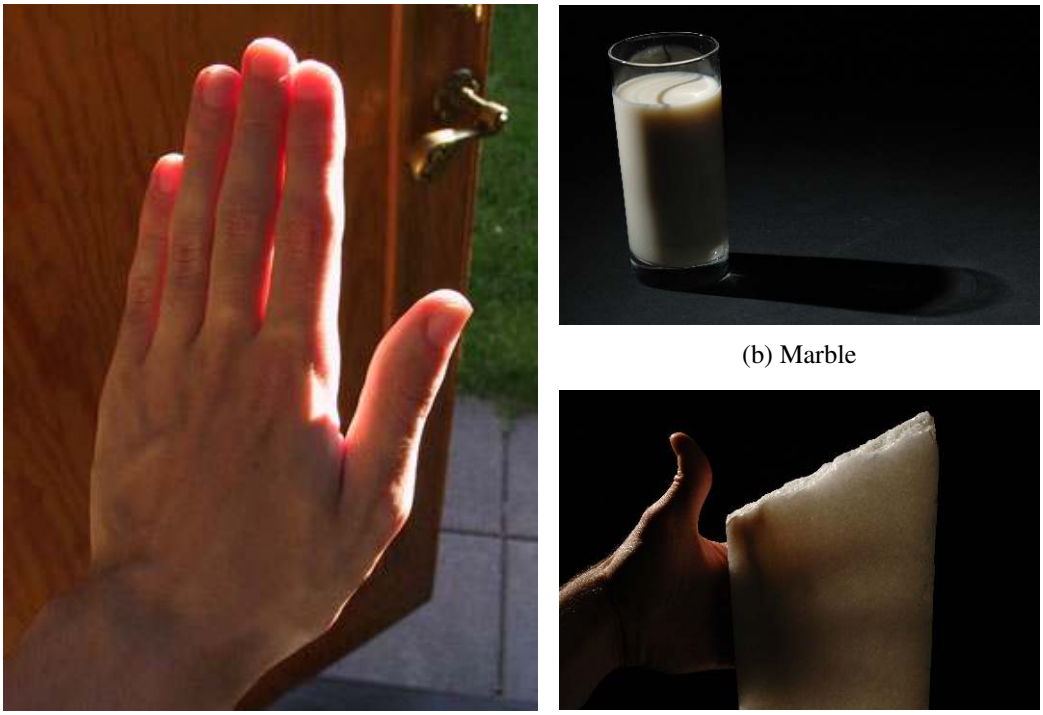
Figure 2.1: This is an illustration of subsurface scattering showing different examples of possible light paths annotated as follows: (a) denotes light that is directly reflected at the surface, (b) corresponds to light that was scattered inside the medium but never absorbed, and (c) shows a light path that never leaves the medium again and is completely absorbed.

2.2 Volume Rendering Theory

For the simulation of light propagation inside a material it is important to know how light interacts with the medium. In order to model these interactions the medium is often described statistically and assumed to be comprised of little particles which interact with light. These particles may emit, absorb or scatter light as it travels through the medium. For simulation and rendering it is necessary to know how the light or radiance changes throughout the medium due to such particle–light interactions. Given this model, let's consider a single ray of light with a certain direction which travels trough a point inside the medium. The change of radiance in this point along the ray direction is determined by the combined effect of the following interaction events:

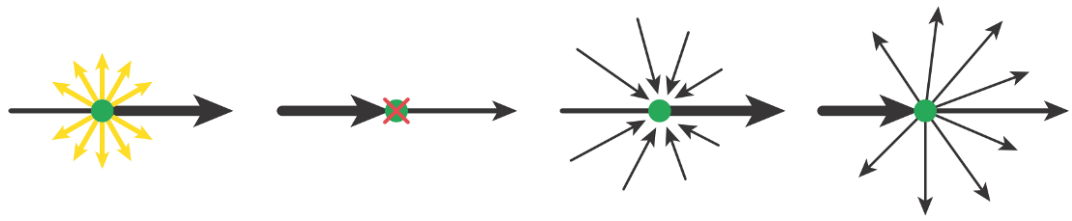
Emission:	Light is emitted.
Absorption:	Light is absorbed.
In-Scattering:	Light is scattered into the direction of the ray.
Out-Scattering:	Light is scattered into a different direction.

An illustration of these four interaction events can be seen in Figure 2.3. In this context, a medium is usually defined probabilistically using the so called absorption coefficient σ_a, scattering coefficient σ_s and phase function p, which are described in more detail in the following paragraph.



(a) Skin (b) Marble (c) Milk

Figure 2.2: Real-world examples of subsurface scattering. (Image sources: [11, 15])



(a) Emission (b) Absorption (c) In-Scattering (d) Out-Scattering

Figure 2.3: Illustrations of light–medium interactions. (Images from Jarosz [18])

Radiative transfer equation: The four particle–light interaction events and medium properties, described earlier, are incorporated by the so called *Radiative Transfer Equation* (RTE) shown in Equation 2.1. Although, its theoretical foundation was introduced by Chandrasekhar [3], the explanations, equations and notations of this section are based on work by Jarosz [18].

The RTE describes the change in radiance L along a given direction $\vec{\omega}$ in a given point $\mathbf{x}$.

$$\begin{aligned}(\vec{\omega} \cdot \nabla)L(\mathbf{x} \rightarrow \vec{\omega}) \; = \; - \; & \underbrace{\underbrace{\sigma_a(\mathbf{x}) \; L(\mathbf{x} \rightarrow \vec{\omega})}_{\text{absorption}} \; - \; \underbrace{\sigma_s(\mathbf{x}) \; L(\mathbf{x} \rightarrow \vec{\omega})}_{\text{out-scattering}}}_{\text{extinction}} \\ & + \; \underbrace{\sigma_a(\mathbf{x}) \; L_e(\mathbf{x} \rightarrow \vec{\omega})}_{\text{emission}} \; + \; \underbrace{\sigma_s(\mathbf{x}) \; L_i(\mathbf{x} \rightarrow \vec{\omega})}_{\text{in-scattering}}\end{aligned} \tag{2.1}$$

$L(\mathbf{x} \rightarrow \vec{\omega})$ denotes the radiance propagating in direction $\vec{\omega}$ in point $\mathbf{x}$. The absorption coefficient σ_a and the scattering coefficient σ_s are characteristic for a given medium and represent the probability that light is scattered or absorbed in point $\mathbf{x}$ of the medium's volume. Absorption and out-scattering can be furthermore combined to form the so called extinction: $\sigma_t L(\mathbf{x} \rightarrow \vec{\omega})$, where $\sigma_t = \sigma_a + \sigma_s$.

The term L_e represents the emitted light, while L_i describes the light scattered into the direction $\vec{\omega}$ at point $\mathbf{x}$. As L_i represents in-scattered light from all directions, it is necessary to integrate the radiance over the whole sphere around point $\mathbf{x}$. This leads to the expanded term of L_i shown in Equation 2.2.

$$L_i(\mathbf{x} \rightarrow \vec{\omega}) = \int_{\Omega_{4\pi}} p(\mathbf{x}, \vec{\omega}' \rightarrow \vec{\omega}) \; L(\mathbf{x} \leftarrow \vec{\omega}') \; d\vec{\omega}' \tag{2.2}$$

The term $L(\mathbf{x} \leftarrow \vec{\omega}')$ represents the light arriving in point $\mathbf{x}$ from direction $\vec{\omega}'$. It is scaled by the so called phase function $p(\mathbf{x}, \vec{\omega}' \rightarrow \vec{\omega})$ which denotes how much light is scattered into the different directions. The coefficients σ_a and σ_s or the phase function p can be constant or may vary with respect to $\mathbf{x}$, for which the medium is referred to as homogeneous or heterogeneous, respectively.

Phase functions can be of arbitrary form as long as they satisfy the two conditions of being normalized and reciprocal as defined in Equation 2.3.

$$\begin{aligned} &\text{Reciprocity:} && p(\mathbf{x}, \vec{\omega}' \rightarrow \vec{\omega}) = p(\mathbf{x}, \vec{\omega}' \leftarrow \vec{\omega}) \\ &\text{Normalization:} && \int_{\Omega_{4\pi}} p(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega}) \; d\vec{\omega}' = 1, \; \forall \vec{\omega} \end{aligned} \tag{2.3}$$

In case of the constant phase function $p_I(\mathbf{x}, \vec{\omega}' \leftrightarrow \vec{\omega}) = \frac{1}{4\pi}$ the medium scatters light uniformly in all direction and is therefore called isotropic. Phase functions which are not constant with respect to the direction model anisotropic scattering. Depending on the predominant scattering direction the medium may be referred to as back-scattering or forward-scattering. One very common example of an anisotropic phase function is the so called Henyey-Greenstein phase function [17] shown in Equation 2.4.

$$p_{HG}(\mathbf{x}, \theta) = \frac{1 - g^2}{4\pi \; (1 + g^2 - 2g \; \cos\theta)^{1.5}} \tag{2.4}$$

Here θ denotes the scattering angle and g is a parameter in the interval $[-1, 1]$ which can be used to define the anisotropy and describes the average cosine of the scattering directions. The

phase function is isotropic for $g = 0$, forward-scattering for $0 < g \leq 1$ and back-scattering for $-1 \leq g < 0$. The advantages of this function are its simplicity and the fact that it can be used to approximate other more complicated scattering functions. The anisotropy parameter g is also used for the definition of the so called reduced scattering coefficient defined as ${\sigma_s}' = (1 - g)\sigma_s$ and the reduced excision coefficient defined as ${\sigma_t}' = \sigma_a + {\sigma_s}'$, according to Pharr and Humphreys [34].

Volume rendering equation: Unfortunately, the RTE is a differential equation and therefore not suitable to be evaluated directly. However, it is possible to derive, based on the RTE, the so called *Volume Rendering Equation* (VRE), shown in Equation 2.5, which can be evaluated by Monte Carlo integration.

$$\begin{aligned} L(\mathbf{x} \leftarrow \vec{\omega}) = & \underbrace{T_r(\mathbf{x} \leftrightarrow \mathbf{x}_s)\ L(\mathbf{x}_s \leftarrow -\vec{\omega})}_{\text{reduced surface radiance}} + \\ & \underbrace{\int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\ \sigma_a(\mathbf{x})\ L_e(\mathbf{x} \rightarrow -\vec{\omega})\ dt}_{\text{accumulated emmited radiance}} + \\ & \underbrace{\int_0^s T_r(\mathbf{x} \leftrightarrow \mathbf{x}_t)\ \sigma_s(\mathbf{x}_t)\ L_i(\mathbf{x}_t \rightarrow -\vec{\omega})\ dt}_{\text{accumulated in-scattered radiance}} \end{aligned} \tag{2.5}$$

The VRE expresses the radiance L arriving at point $\mathbf{x}$ from direction $\vec{\omega}$. This equation basically represents the sum of the light from the nearest surface arriving at point $\mathbf{x}$ in direction $\vec{\omega}$ and the emitted or in-scattered light along the ray from the surface to point $\mathbf{x}$, attenuated by absorption and out-scattering. Here, $\mathbf{x}_s$ is the nearest surface point in direction $\vec{\omega}$ and s is its distance to $\mathbf{x}$. Points between $\mathbf{x}$ and $\mathbf{x}_s$ are denoted as $\mathbf{x}_t$. The T_r terms represent the so called transmittance between two points, with Equation 2.6 as its expanded from.

$$\begin{gathered} T_r(\mathbf{x}' \leftrightarrow \mathbf{x}) = e^{-\tau(\mathbf{x}' \leftrightarrow \mathbf{x})} \\ \text{where} \\ \tau(\mathbf{x}' \leftrightarrow \mathbf{x}) = \int_0^d \sigma_t(\mathbf{x} + t\vec{\omega})\ dt \end{gathered} \tag{2.6}$$

The transmittance T_r represents the fraction of light left after travelling distance d between point $\mathbf{x}$ and $\mathbf{x}'$. This fraction depends on the extinction coefficient σ_t, as the radiance may get reduced by absorption and out-scattering.

2.3 SSS Rendering

In principle any algorithm which supports volumetric rendering may be used to render SSS. For example it is possible to evaluate the VRE using a Monte-Carlo based renderer. However, in case of subsurface scattering it is mandatory to compute multiple scattering, which is significantly harder than single scattering. Therefore, rendering of SSS can become very time consuming due to the necessary computation of multiple scattering in optically thick media.

Due to the high number of scattering events inside an optically thick material, and the accompanying attenuation, most of the rays which enter the surface simply terminate without ever reaching a light source. If we would consider to use e.g. a simple volumetric path tracer for rendering of SSS, it will become apparent that a vast amount of computed paths are simply wasted and do not contribute to the final solution, which leads to slow image convergence. Although more sophisticated and efficient algorithm exist, this simple example illustrates that simulation and rendering of SSS can be quite difficult and time consuming.

The fact that performing a full volumetric SSS simulation is hard and time consuming lead to the development of alternative approaches for which the majority is based upon concepts described in the following paragraphs.

Diffusion: A characteristic feature of multiple scattering in optically thick materials is that every scattering event basically blurs the light distribution in the medium. As a consequence the light distribution becomes uniform after a sufficient amount of scattering events, even for highly anisotropic materials. This makes it possible to compute an approximation of multiple scattering based on diffusion theory, which replaces the exact but time consuming simulation. This diffusion theory is used and applied in various ways by different papers mentioned in the next chapter. One approach is to compute the multiple scattering component by performing light diffusion on a discretisation of the material's volume. Another approach is to approximate diffusion via dipoles or multipoles used in BSSRDF shading models.

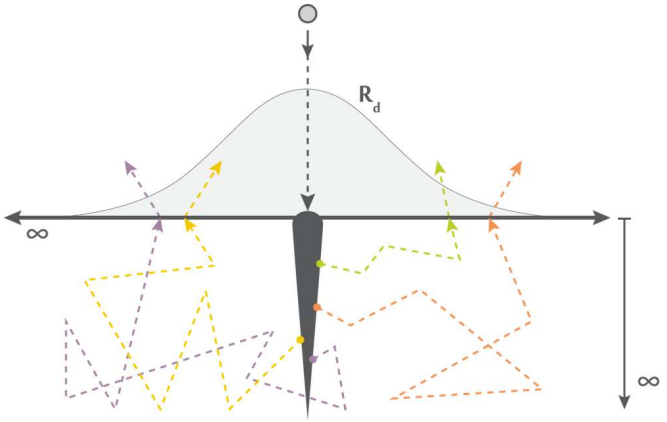
BSSRDF: In order to render SSS without the need for full volumetric simulation of the participating media it is possible to use an advanced surface shading model incorporating the so called *Bidirectional Scattering Surface Reflectance Distribution Function* (BSSRDF). It is basically a function which tells you how much light hitting a surface at one location from a certain direction is reflected from the surface at another location in a certain direction. This is formally described in Equation 2.7.

$$dL_o(x_o, \vec{\omega}_o) = S(x_i, \vec{\omega}_i; x_o, \vec{\omega}_o)\ d\Phi_i(x_i, \vec{\omega}_i) \tag{2.7}$$

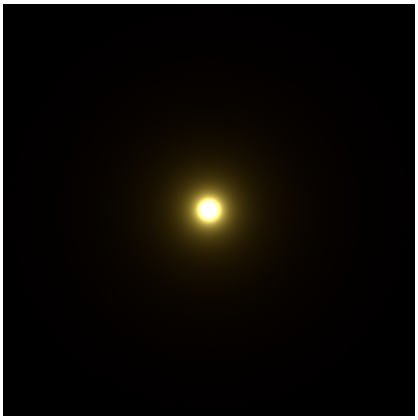
This equation states that the outgoing radiance L_o at point x_o in direction $\vec{\omega}_o$ is determined by the incident flux Φ_i at point x_i in direction $\vec{\omega}_i$ scaled by the BSSRDF S parametrized with the corresponding locations and directions. This model can be seen as a more general version of the commonly used *Bidirectional Reflectance Distribution Function* (BRDF) which assumes that light that hits a surface is directly reflected at the same location ($x_o = x_i$). Similar to the various existing BRDF models, there are also different BSSRDF models proposed in the literature, of which some notable examples are mentioned in the next chapter.

Irradiance Filtering: Given that offline rendering algorithms need a significant amount of time to compute SSS, it is not surprising that real-time rendering of SSS can be even more challenging. Due to the time constraint of only a few milliseconds per frame, it is currently not possible to compute a full SSS simulation in real time, but some kind of fast approximation has to be used.

One characteristic feature of SSS is that it blurs surface detail and illumination. It is therefore a very common real-time approach to approximate subsurface light diffusion by filtering the



(a) 1D illustration of a diffuse reflectance profile



(b) 2D diffuse reflectance profile

Figure 2.4: These two figures demonstrate the basic concepts behind a diffuse reflectance profile. Figure (a) shows an illustration of how a diffuse reflectance profile is generated. A light beam hits the surface of an infinitely wide and thick half-space, enters the material and is attenuated and scattered below the surface. Eventually, light that was not absorbed leaves the material's surface again at different locations, forming the diffuse reflectance profile R_d. Please note that this illustration was inspired by previous work of Habel et al. [14]. An example for an actual 2D diffuse reflectance profile is shown in (b), which was generated via particle tracing. (Images courtesy of Károly Zsolnai)

irradiance on the surface. Various different filter kernels ranging from simple Gaussians to more complex functions are proposed and used throughout the literature. Some notable examples are mentioned in the next chapter.

Diffuse Reflectance Profile: Using only a simple Gaussian blur to approximate SSS in real time is expectably a very crude approximation. In order to perform a physically based SSS approximation it is necessary to use a so called *Diffuse Reflectance Profile* (DRP) for the convolution of the irradiance on the surface. A DRP represents the amount of diffuse reflected light from the surface of a material illuminated by a normally incident infinitesimal light beam. This is further illustrated in Figure 2.4. Alternatively, it might also be interpreted as the diffuse impulse response of the surface or material. The DRP is in general a 2D function $R_d(x, y)$ with the origin at the incident location of the light beam. For isotropic and homogeneous materials this function is radially symmetric and can therefore also be expressed as a 1D function $R_d(r)$ parametrized by the distance to the origin (radius), where $R_d(r) = R_d(\|(x, y)\|)$. DRPs for different materials can be generated using e.g. brute-force Monte-Carlo simulation and are commonly computed for an infinitely extending half-space. DRPs may be applied for irradiance filtering according to Equation 2.8.

$$M_e(x, y) = \int_{\mathbb{R}^2} E(x', y') R_d(x - x', y - y') dx' dy' \tag{2.8}$$

Here, $M_e(x, y)$ denotes the radiant exitance for arbitrary surface points (x, y), which is the result of the convolution of the surface irradiance $E(x, y)$ with the diffuse reflectance profile $R_d(x, y)$. Using 2D convolution notation, Equation 2.8 may be expressed as $M_e(x, y) = (E * R_d)(x, y)$. By using DRPs as filters for the surface irradiance convolution it is possible to perform fast approximation and rendering of physically based SSS.

The concepts described previously in this chapter are common foundations for the methods discussed in the next chapter which gives an overview of the state of the art in SSS rendering and outlines important real-time algorithms.

CHAPTER 3

Related Work

For the simulation and rendering of SSS various approaches have been developed. This chapter gives an overview over some of the more significant methods. At first an overview over algorithms suitable for offline rendering is given, followed by a more detailed discussion of real-time techniques including algorithms on which the extended method is build upon.

3.1 Offline Rendering

Chandrasekhar [3] laid the foundation for rendering of participating media and consequential SSS. In principle, it is possible to use brute-force path tracing introduced by Kajiya [26] or variations of more sophisticated algorithms like photon mapping proposed by Jensen [20], bidirectional path tracing developed by Lafortune and Willems [28] or metropolis light transport introduced by Veach and Guibas [37].

However, for optically thick materials many of those rendering algorithms are quite inefficient and may result in increased rendering times and slow convergence. Therefore, finding more efficient ways to render participating media is and was an active and challenging research area. The following are a few significant methods developed to increase efficiency for offline rendering of participating media and especially SSS.

Stam [36] introduced the diffusion approximation for multiple scattering in participating media and presented practical techniques for its evaluation. Jensen et al. [21] developed a BSSRDF model including exact single scattering and an approximation for multiple scattering based on dipole diffusion. The model was verified using a specially developed technique for the measurement of optical material properties. Furthermore, the model's usage in the context of ray-tracing was discussed. Improving on the previous method Jensen and Buhler [19] presented a two pass approach for rendering of translucent materials which also builds upon the dipole diffusion approximation. The first pass caches irradiance samples on the surface which are then used in the second pass for an efficient hierarchical evaluation of the diffusion approximation. Due to this two pass approach it is easily integrateable into scanline or ray-tracing based renderers and supports indirect illumination.

Dipoles used for the diffusion approximation are only able to model homogeneous semi-infinite slaps and therefore are not suitable if different material layers need to be modelled. In order to eliminate this issue, Donner and Jenson [8] introduced a technique to render translucent materials composed of multiple thin layers by developing a multipole diffusion approximation. Each layer is modelled by a multipole which are combined by convolution in frequency space to form the reflectance and transmittance profiles of a layered material. Another paper by Donner and Jenson [9] presented a method which combined photon tracing with diffusion to render more advanced global illumination effects like volumetric caustics and shadows in translucent materials. The basic idea is to store photons at their first scattering location inside the translucent material, which are later used as sources for the diffusion. They further adaptively blend between a dipole, multipole and quadpole to improve the quality of the approximation for complex geometry.

D'Eon and Irving [7] developed a new BSSRDF based on an improved diffusion approach called quantized diffusion which models the reflectance and transmittance profiles of individual layers as sum of Gaussians. It enables rendering of translucent materials with very thin layers for which previous dipole or multi-pole models deliver inaccurate results. The application of the new BSSRDF for rendering was demonstrated using the two-pass approach of Jensen and Buhler [19]. Kulla and Fajardo [27] introduced a new importance sampling strategy for rendering of participating media. It supports illumination of homogeneous or heterogeneous media by point and area light sources and integrates well into ray-tracing based renderers, but introduces small bias. The sampling strategy places samples along a ray according to the incoming radiance and special probability density functions, constructed during ray-marching, aid rendering efficiency and sample placement. Habel et al. [14] combined the diffusion approximation with a Monte Carlo integration scheme based on photon beams. Compared to quantized diffusion, it is more accurate, robust and faster and even handles arbitrary incident light directions. Furthermore, it was integrated into various rendering frameworks to show its practicality and usability.

Although the mentioned algorithms all are able to deliver high quality results while being increasingly efficient, they still need, depending on the actual scene and algorithm, at least a few seconds to compute the final image. This is obviously still to slow for real-time rendering of SSS an therefore slightly different approaches taking more radical approximations have to be used for real-time performance.

3.2 Real-time Rendering

During the research and development phase for the film 'Matrix Reloaded' Borshukov and Lewis [2] where in need of a fast SSS rendering approach for CG models of the actors faces. As current methods at that time did not meet their computation-speed constraints they developed a simple diffusion approximation in image space. This idea laid the foundation for various real-time SSS rendering algorithms which adopted this approach.

An examples of an early adoption of this idea in the context of real-time rendering is the method proposed by Green [13] which performs the diffusion by simple irradiance convolution with a single Gaussian filter in texture space. A similar example is the approach introduced by Gosselin et al. [12] which uses Poisson disk filtering to approximate SSS in skin. Another

sophisticated algorithm for real-time SSS rendering in skin was presented by d'Eon et al. [5, 6] which is based on texture-space diffusion and adapted shadow maps. They derived a very efficient sum of Gaussian approximation of the dipole and multipole model of Donner and Jenson [8] for multi-layered materials. In order to compute the light reflected of a surface due to SSS, the irradiance is stored in a texture and convolved with the reflectance profile represented as a sum of Gaussians. Light transmitted through thin geometry is approximated by clever use of the already convolved irradiance textures and the sum of Gaussians parametrized by the thickness derived from shadow maps. As Gaussians are separable, the required convolutions can be computed fast enough to deliver real-time frame rates and this method even works for animated light and geometry. Hable et al. [16] built upon the method by d'Eon et al. [5,6] and adapted it to support efficient rendering in console game production environments. They used a single convolution pass with jittered sampling to increase rendering performance and introduced additional optimizations to avoid convolution of back-facing surfaces.

In contrast to approximation of diffusion in a 2D space like e.g. texture space, other methods solve the diffusion approximation on some form of discretization of the object's volume. Wang et al. [38] developed a technique for measurement and real-time rendering of heterogeneous translucent materials based on the diffusion approximation. They obtain the heterogeneous material parameters by solving the inverse diffusion problem via optimization. For rendering, the diffusion approximation is solved on the GPU based on a volumetric polygrid. A similar but improved real-time approach for heterogeneous translucent materials was presented by Wang et al. [40]. It is also based on the diffusion approximation, but uses a QuadGraph for discretisation, which can be built automatically and supports topology-preserving deformations. Li et al. [29] introduced a method for interactive manipulation and rendering of heterogeneous translucent materials. It uses a hierarchical tetrahedral representation of the object and a specially developed solver for the discrete diffusion equation, both implemented on the GPU. Since these computations are performed every frame it is possible to perform cutting and fracturing operations on the geometry as well as painting of the heterogeneous material properties at interactive frame rates.

The main drawback of the previously mentioned real-time methods is that they scale poorly with the number of translucent objects in the scene, as the algorithm has to consider every object separately. To eliminate this strong dependence on scene complexity Jimenez et al. [22, 24] improved upon the approach by d'Eon et al. [5, 6] by performing the diffusion in screen space instead of texture space. This alleviates multiple drawbacks of the texture-space approach and improves rendering performance. They also conducted a psychophysical experiment which showed that there is no significant perceptual difference of the screen-space approach compared to texture-space diffusion. The beneficial properties of screen-space rendering are also utilized by various other methods. Mertens et al. [30] developed an importance sampling strategy for the BSSRDF of Jensen et al. [21] and applied it in screen space. Building on the same BSSRDF model Shah et al. [35] used a three-pass splatting approach, efficiently computed on the GPU, which supports multiple and single scattering. In the first two passes visible surface points from the light and the camera are computed. This information is used in the third pass to splat precomputed and camera-aligned reflectance profiles centered at the surface locations visible from the light, to compute SSS at the surface locations visible from the camera. Mikkelsen [31]

approximated SSS by blurring the irradiance using a cross-bilateral filter in image space. Here the filter, a single empirically chosen Gaussian, is applied in two separable 1D passes, where its size is adapted per pixel, based on the underlying geometry.

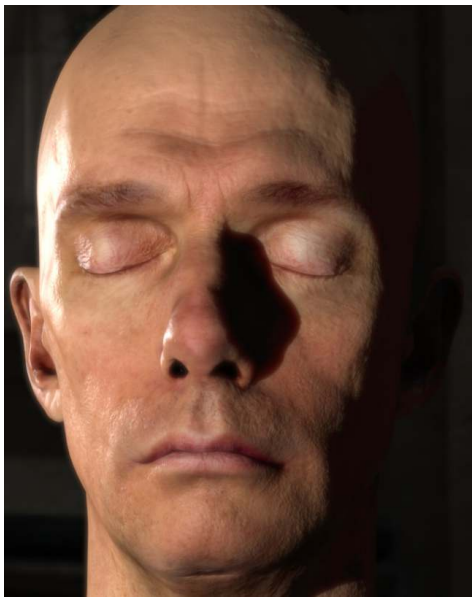
The majority of the previously mentioned real-time algorithms use a runtime integration scheme based on convolution. A quite different and notable algorithm based on pre-integration was presented by Penner and Borshukov [33]. It enables real-time skin rendering via the decomposition and pre-integration of the SSS effect into three different components representing low and high frequency surface variations as well as shadow edges. SSS variations caused by low frequency changes in curvature and illumination are pre-integrated into a lookup table. The high frequency variations due to fine surface details are pre-integrated by normal map filtering. And illumination variation caused by shadows is handled via penumbra pre-integration into an additional lookup table. These precomputed lookup tables and filtered normal maps are then used together in the final surface shading computations to approximate the SSS effect in skin.

Three notable examples of rendered images, generated by a subset of the discussed algorithms, are shown in Figure 3.1. These images show different materials rendered with SSS and demonstrate that today's rendering algorithms are able to produce high quality renderings which can be considered as being photorealistic.

3.3 Separable Subsurface Scattering

A recent implementation of a real-time SSS algorithm which builds upon the screen-space convolution approach and shares some similarities with previous methods was introduced by Jimenez and Gutierrez [23, 25]. This method is, as numerous others, specifically tuned for human skin. It is based upon the SSS screen-space method by Jimenez et al. [22] and uses the approach of Jimenez and Gutierrez [24] for translucency. The main difference is that it uses a parametrized and separable kernel which is specially designed to approximate the 2D diffuse reflectance profile for human skin. As it uses only two separable blur passes in screen space it is able to compute SSS in approximately one millisecond or less, while being almost independent of the scene complexity. These properties make this method very attractive for use in real-time rendering for games, where multiple rendering effects and additional game related simulations have to be computed using only a few milliseconds. Due to the fact that this method is especially designed for human skin, it is an interesting challenge to extend it for SSS rendering of arbitrary materials. The following chapter includes a more detailed explanation of this algorithm and outlines the context in which the proposed extensions are presented.

(a) Offline rendering



(b) Real-time rendering

(c) Offline rendering

Figure 3.1: This figure show rendered images from the following publications: (a) shows an image rendered offline via the method introduced by Donner and Jenson [9], image (b) was generated by the real-time algorithm of d'Eon et al. [6], and (c) is an image computed using the offline algorithm by d'Eon and Irving [7].

CHAPTER 4

Separable Subsurface Scattering

This chapter provides a more detailed explanation of the implementation of real-time SSS rendering for human skin by Jimenez and Gutierrez [23, 25], often referred to as Separable Subsurface Scattering (SSSS). First, the main idea of the method is explained in comparison to the state-of-the-art approach. Then, a general overview over the algorithm and the rendering pipeline is given, followed by a more detailed explanation of the SSS model for human skin and noteable aspects of the algorithm and its implementation. Finally, initial attempts to find a straightforward extension are discussed.

4.1 Main Idea

As outlined in the previous chapters, many real-time algorithms approximate SSS by convolution of the surface irradiance via filter kernels of varying complexity. The application of such a filter kernel would usually require a costly 2D convolution which is not feasible for most real-time rendering scenarios.

There exists, however, a special class of filters for which the costly 2D convolution can be replaced by two consecutive 1D convolutions. Such filters are called separable, because they can be decomposed into two components according to Equation 4.1.

$$A(x, y) = a(x)\, b(y) \tag{4.1}$$

This decomposition makes it possible to replace the 2D convolution with filter A by two consecutive 1D convolutions (horizontal and vertical) with a and b. For radially symmetric filters, such as diffuse reflectance profiles of homogeneous and isotropic materials, $a = b$.

This separability scheme is, among others, the reason why d'Eon and Luebke [5] proposed to approximate physically based filter kernels derived from diffuse reflectance profiles using a mixture of 2D Gaussians. A two-dimensional Gaussian kernel has the convenient property of being the only separable kernel which is also radially symmetric. Due to the separability property, it is possible to apply such a 2D Gaussian mixture via multiple 1D convolution passes

(two per Gaussian). This principle is applicable to any 2D filter kernel, e.g., diffuse reflectance profile, which can be decomposed into multiple separable components as illustrated in Equation 4.2,

$$\begin{aligned}
&\int_{\mathbb{R}^2} E(x', y')\ R_d(x - x', y - y')\ dx'dy' \approx \\
&\int_{\mathbb{R}^2} E(x', y')\ A(x - x', y - y')\ dx'dy' = \\
&\int_{\mathbb{R}^2} E(x', y')\ \sum_{i=1}^{N} a_i(x - x')\ a_i(y - y')\ dx'dy' = \\
&\sum_{i=1}^{N} \int_{\mathbb{R}^2} E(x', y')\ a_i(x - x')\ a_i(y - y')\ dx'dy' = \\
&\sum_{i=1}^{N} \int_{\mathbb{R}} \int_{\mathbb{R}} E(x', y')\ a_i(x - x')\ dx'\ a_i(y - y')\ dy'
\end{aligned} \tag{4.2}$$

and Equation 4.3 using a shorter notation.

$$\begin{gathered}
(E * R_d)(x, y) \approx (E * A)(x, y) = \sum_{i=1}^{N} ((E * a_i) * a_i)(x, y) \\
\text{with} \\
A(x, y) = \sum_{i=1}^{N} a_i(x) a_i(y)
\end{gathered} \tag{4.3}$$

Here, the convolution of the irradiance E with the diffuse reflectance profile R_d is approximated by a convolution of E with approximation A which is comprised of multiple components a_i. It is, therefore, possible to apply the different components a_i via the combination of multiple consecutive 1D convolutions. In case of d'Eon and Luebke [5], the approximation A is represented by a Gaussian mixture approximation A_g of the diffuse reflectance profile R_d, as shown in Equation 4.4.

$$R_d(x, y) \approx A_g(x, y) = \sum_{i=1}^{N} w_i G(x, y; \tau_i) \tag{4.4}$$

In this equation, A_g represents a 2D Gaussian mixture with weights w_i, and G denotes the zero-mean 2D Gaussians with variances τ_i. This approximation A_g is applicable according to Equation 4.3, as each individual 2D Gaussian G is separable.

Although this approach already offers a significant performance improvement compared to naive 2D convolution, computing multiple 1D convolution passes may still not be feasible for certain real-time rendering scenarios. Given the 2D Gaussian mixture approach, the fastest solution would be to use only one Gaussian, which would result in two 1D convolutions based on a single separable Gaussian. Unfortunately, diffuse reflectance profiles are in general not well approximated by only a single 2D Gaussian, which is the reason why d'Eon and Luebke [5] used up to six.

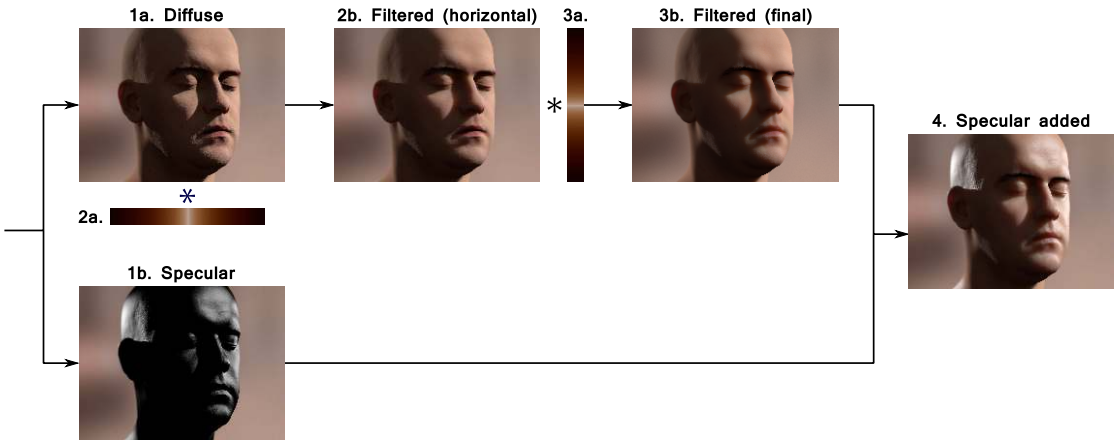


Figure 4.1: Visualisation of the SSSS rendering pipeline. (1a) and (1b) show the diffuse and specular input computed in the first pass. (2a) represents the first horizontal convolution pass, with (2b) as its output. The subsequent vertical convolution (3a) results in the filtered diffuse output (3b). The final image (4) is generated by adding the specular component to the filtered diffuse output. Please note that the images for (2a) and (3a) are only illustrative mock-ups and do not represent the actual filter kernel.

In order to enable reasonable approximation of SSS via only two 1D convolutions, according to Equation 4.5,

$$(E * R_d)(x, y) \approx (E * A)(x, y) = (E * a) * a)(x, y) \tag{4.5}$$

the main idea of SSSS is to use a custom separable, but not necessarily radially symmetric, kernel A_s which approximates the diffuse reflectance profile R_d as close as possible, according to Equation 4.6.

$$R_d(x, y) \approx A_s(x, y) = a_s(x)\, a_s(y) \tag{4.6}$$

The particular form of a_s is explained in more detail in Section 4.3.

By not using a single Gaussian as the separable kernel, but a different more specialized function a_s, this SSSS filter kernel is not radially symmetric. This radial asymmetry, also discussed in Section 5.1, is a property of all separable kernels which are not Gaussians, and manifests itself as a pronounced cross pattern of relatively higher values aligned with the kernels main axis. This cross pattern can lead to undesirable artefacts, as shown in Section 6.4.

The separable kernel model used by the SSSS algorithm is further explained in more detail in Section 4.3, while the following section outlines the general steps necessary to apply the separable convolution approach for SSS rendering.

4.2 Algorithm Overview

The SSSS algorithm is able to approximate subsurface scattering for human skin and fully dynamic scenes in real time. This is achieved by computing an approximation of SSS as a post-processing step, which makes this approach almost independent of the scene's complexity. It

primarily consists of a convolution using a separable filter kernel, as outlined previously, which can be applied in two 1D convolution passes. It can therefore be easily integrated into existing rendering pipelines by implementing the following changes and additional passes.

1. Pass — Diffuse Shading: The scene is rendered from the viewpoint of the camera into a render-target texture using diffuse shading only, including additional effects like texture-, normal- and shadow mapping. The specular component may be stored in a separate render-target texture to be reapplied after the next two convolution passes. This diffuse pass also includes the calculation of the translucency component based on shadow mapping.

2. + 3. Pass — Separable Convolution: The diffuse render-target texture from the previous pass is convolved using the separable filter kernel in two 1D convolution passes (one horizontal and vertical). The size of the kernel is adaptively scaled based on various factors, explained in more detail in Section 4.4. The weight of the individual kernel samples may be scaled to account for geometric discontinuities. For pseudocode of the convolution pass see Algorithm 4.1.

4. Pass — Reapply Specular: This final pass is used to reapply the specular shading, computed and stored in the first pass, by blending it on top of the convolved diffuse shading.

An intuitive visualisation of the SSSS rendering pipeline can be found in Figure 4.1. At first this pipeline may seem almost like a normal post-processing effect that just blurs the illuminated surfaces in screen space using a simple filter kernel. However, as SSS is a more complex effect than a simple blur, special care has to be taken to account for the underlying surface geometry, specular reflection and appropriate filter kernel size. These important details of the individual passes are explained in Section 4.4, preceded by a more detailed explanation of the special separable filter kernel model in the following section.

4.3 SSS for Human Skin

The SSSS algorithm uses a carefully designed filter kernel to approximate SSS in human skin. This special filter is separable, and, therefore, can be applied using two 1D convolutions, which makes fast SSS rendering possible. The basic idea of SSSS is to find a separable filter which approximates the diffuse reflectance profile of human skin as close as possible according to Equation 4.7.

$$R_d(x, y) \approx A_s(x, y) = a_s(x)\, a_s(y) \tag{4.7}$$

The ground-truth 2D diffuse reflectance profile is represented by R_d and approximated by A_s, which represents the separable filter determined by a_s. This term is defined as a special parametrized function, shown in Equation 4.8, which can be used to tweak the separable kernel A_s. The basic approach is to vary the parameter of a_s such that A_s matches R_d as close as

possible.

$$a_s(r; w, t, f) = p\left(\frac{w\ r}{0.001 + f}\right) t + \delta(1 - t)$$

where

$$p(r) = \sum_{i=0}^{n} w_i\ G(r, v_i) \tag{4.8}$$

with

$$G(r; v) = \frac{1}{2\pi v} e^{-\frac{r^2}{2v}}$$

The function p is based on the Gaussian mixture approximation of the radially symmetric 1D diffuse reflectance profile of the 3-layer model for skin introduced by d'Eon and Luebke [5]. It uses only the red channel of this 1D profile, which is transformed using the three parameters w, t and f. The parameter w stands for width, as it can be used to control the global size or falloff width of the function. The parameter t can be interpreted as the function's strength or magnitude. It influences the interpolation between the 1D Gaussian mixture profile $p(r)$ and the delta function, and can therefore be used to continuously switch between the two. The third parameter f defines the falloff width separately for each channel. Finally, r simply denotes the radius at which the function is evaluated. The internal profile $p(r)$ is a 1D Gaussian mixture with zero mean, where w_i and v_i represent the individual weights and variances, respectively. In this particular case, the weights and variances are fixed to the (red channel) values of the 1D Gaussian mixture approximation of the 3-layer skin diffuse reflectance profile from d'Eon and Luebke [5]. Please note that the a_s function represents only a single RGB channel, and that t and f are defined for each channel individually, while only the w parameter is the same across all channels.

The goal for SSSS is to find a separable approximation A_s of the 2D ground-truth diffuse reflectance profile R_d for human skin, according to Equation 4.9, by variation or optimization of the model parameters.

$$R_d(x, y) \approx A_s(x, y) = a_s(x; w, t, f)\ a_s(y; w, t, f) \tag{4.9}$$

For the original implementation the best parameter set (w, t and f) for a_s, such that the resulting separable kernel A_s approximates R_d as close as possible, was found by optimization according to Equation 4.10.

$$\underset{w,t,f}{\operatorname{argmin}} \left\{ \int_{\Re^2} (R_d(x, y) - a_s(x; w, t, f)\ a_s(y; w, t, f))^2\ dx\ dy \right\} \tag{4.10}$$

Additionally, the implementation provides an UI interface for interactive manipulation of the parameters, which makes it possible to tweak the kernel by hand until a desired approximation or effect has been achieved. The latter makes this model suited for artistic control, which in principle enables the approximation of any kernel.

However, it is worth noting that the transformed internal profile p used for this implementation is the red channel of a 3-layer skin profile. Furthermore, the w, t and f parameter set, which

transforms this fixed internal profile, is specially optimized to result in a kernel modelling SSS in human skin. It is therefore not safe to assume that a suitable parameter set can be found for any arbitrary material, let alone that this may be achieved in a straightforward manner.

This separable kernel A_s defined by the optimized parameter set is used to replace a costly 2D convolution or multiple 1D convolutions by only two 1D convolution passes with a_s, as outlined in Section 4.1 by Equation 4.5 and 4.6. Depending on the approximation quality, the differences between the results of the full 2D convolution and the separable (1D) two-pass convolution may be insignificant. However, in the application of such a separable kernel for convolution in image space, special care has to be taken, as outlined by the following section.

4.4 Implementation Details

This section discusses several implementation details of the SSSS algorithm, and explains different aspects of the rendering pipeline in more detail.

Diffuse–Specular Separation: The first pass of the SSS rendering pipeline, outlined in Section 4.2, includes the computation of diffuse and specular shading components for the rendered surfaces. As the specular component represents light which is directly reflected at the surface of the material, it would be incorrect to include it in the SSS approximation and the convolution with the SSS filter kernel. Therefore, the specular component is stored in a separate render-target and reapplied after convolution in an additional pass. In case the use of an additional render-target is prohibited by constraints of the particular application or used hardware, it is also possible to simply combine the specular and diffuse component before convolution. However, this approach may lead to washed-out specular highlights in the final render.

Translucency: The approximation of SSS via convolution in screen space primarily accounts for local subsurface scattering where light hits and leaves the surface on approximately the same side of the visible surface. The contribution of light which may enter the material at back-facing surfaces or travels over a longer distance through the material is generally not taken into account. This missing SSS effect is approximated by an additional translucency component, which is computed in the first pass using the approach by Jimenez and Gutierrez [24]. It is simply added to the diffuse render-target texture and therefore also included in the subsequent convolution.

Kernel Size: For a simple screen-space blur filter, it is usually sufficient to use a constant kernel size. This, however, is not possible in case of SSS, as the kernel usually represents some physically based diffuse reflectance profile. It is therefore necessary to scale the kernel to an appropriate size corresponding to the area of the visible surface in screen space.

The kernel size used for convolution is determined per pixel by the product of three factors, namely *Kernel Scale*, *SSS Width* and *SSS Strength*. A naive convolution using a fixed kernel size for all pixels would result in a constant overall blur, and would completely neglect surface area distortions due to perspective projection. For each pixel the represented surface may have varying distance to the camera, and consequently the surface area represented by each individual

screen-space pixel varies according to the perspective projection. It is therefore necessary to scale the kernel per pixel to fit the actual area represented by the corresponding surface region. In each pixel the projected surface area is estimated and combined with the pixel depth to compute the corresponding *Kernel Scale* factor k_{sc} via Equation 4.11.

$$k_{sc} = \frac{1}{\tan(\frac{1}{2}\, f_y)\, p_d} \tag{4.11}$$

The factor f_y denotes the field of view angle along the Y axis and p_d is the pixel depth. This equation basically represents the distance to the projection window divided by the pixel depth. As a consequence, the kernel size gets smaller as the distance of the geometry (depth of the pixel) increases.

The *SSS Width* factor represents an additional user-definable parameter. It modulates the global scale of the SSS effect, making it possible to adjust the kernel size to objects of different sizes. And the so-called *SSS Strength* factor enables the artist to define per-pixel surface regions with varying strength of the SSS effect. This scaler can be encoded into the alpha channel of the diffuse texture, for example.

Geometry-aware Filtering: Due to the fact that the algorithm uses a screen-space convolution to approximate local SSS, it is necessary to take the surface geometry into account. Otherwise the kernel would blur the diffuse illumination across regions which may be close in screen space but are far apart in object space, e.g., silhouette edges. In order to prevent this, the contribution of off-center samples is scaled according to their estimated object-space distance to the kernel center. This is done by computing the final sample color s_c via interpolation between the original input pixel color i_c at the kernel's center position and the off-center sample color o_c, based on their depth difference and additional factors, according to Equation 4.12.

$$\begin{gathered} s_c = o_c\,(1-t) + i_c\, t \\ \text{where} \\ t = \frac{C\ \mathit{SSS}_w\ |i_d - o_d|}{\tan(\frac{1}{2}\, f_y)} \end{gathered} \tag{4.12}$$

The factors i_d and o_d represent the depth of the input source pixel and the off-center sample, respectively. C is an implementation-specific constant and SSS_w represents the *SSS Width*. This approach ensures that the contribution of off-center samples with significant depth differences is continuously discarded and replaced by the original input source color.

Kernel Computation and Convolution: The SSSS algorithm uses a separable kernel model including a parameter set that is optimized for SSS in human skin, as described in Section 4.3. In order to apply this separable kernel A_s for SSS rendering via two 1D convolution passes, the actual implementation precomputes the filter kernel by sampling the kernel function a_s using a user-specified sample count. Although the separable convolution approach already cuts down the necessary sample count from $N * N$ (naive 2D) to $N + N$, it is still necessary to choose a moderately small filter size N (sample count) in order to ensure real-time performance. Results

Algorithm 4.1: Pixel shader pseudocode of the separable 1D convolution render pass

```
Sample diffuse color and depth;
Compute kernel size;
for all (1D) kernel samples do
    Compute sample offset modulated by the kernel size;
    Sample diffuse color at computed offset;
    if geometry-aware filtering requested then
        Compute interpolation factor;
        Replace current diffuse sample color by interpolation between the original center
        input pixel and the current sample;
    end
    Accumulate diffuse sample color scaled by the kernel sample weight;
end
return filtered result;
```

from the original SSSS implementation, along with the results shown in Chapter 6 indicate that a sample count as low as $N = 17$ is sufficient in most cases. This low sample count is, furthermore, enabled by importance sampling of a_s. As the shape of a_s is basically a high peak with a long lower falloff, importance sampling of a_s is applied by computing more samples near the center ($r = 0$). This is done by computation of equally spaced samples using the chosen sample count, and redistribution of the sample positions via a quadratic mapping function.

In order to be able to change the size of the kernel in screen space without recomputation, the w parameter is not used during precomputation, but later applied during shader evaluation. For the precompuation step $w = 1$ is assumed, and the kernel's sample weights and offsets are precomputed via importance sampling of a_s. Furthermore, after the evaluation of the internal profile p, and just before the application of the t parameter, the weights of the 1D kernel samples are normalized.

The precomputed filter kernel is passed to the pixel shader as a simple array of 4D vectors that represent the individual samples of the kernel. The XYZ components of each vector contain the RGB weights, and the W component encodes the sample position (radius offset), as they are not uniform due to importance sampling. The actual w parameter value is used in the pixel shader to offset the sampling positions during convolution. A single pixel shader is used for both 1D convolution passes by simply providing a normalized offset vector (2D) parameter that is scaled by the kernel offsets to compute the corresponding (vertical or horizontal) sampling positions.

4.5 Additional Notes

The implementation of this algorithm by Jimenez and Gutierrez [23, 25], showcasing real-time SSS for human skin applied to a scanned model of a human head, was uploaded to GitHub [25]. A screenshot of the original implementation can be seen in Figure 4.2. The extensions

Figure 4.2: Screenshot of the original SSSS implementation.

to arbitrary materials, proposed in the next chapter, are derived and developed based on this codebase. The used technologies include C++, DirectX 10 and HLSL and the utilization of functions and GUI elements commonly used for DirectX feature demonstrations. The rendered model is a high-resolution scan of a human head including diffuse and normal textures, as well as an additional texture encoding ambient occlusion and specular parameters. Additionally, the implementation includes features like shadow mapping, high-dynamic range (HDR) rendering, depth of field and enhanced subpixel morphological antialiasing (SMAA). The final renderings are furthermore enhanced using effects like HDR bloom and film-grain noise. Detailed on-screen profiling output is provided, reporting a per-frame computation time for the SSS rendering passes of about one millisecond.

As this method and its implementation are carefully designed for SSS rendering of human skin, its possible application for other materials presents an interesting research challenge. The next chapter, therefore, includes a detailed description of proposed extensions to this technique, which make it capable of rendering SSS for arbitrary materials, while the following section outlines first extension attempts.

4.6 First Extension Attempts

As mentioned previously in Section 4.3, the SSSS algorithm introduced by Jimenez and Gutierrez [23, 25] uses a kernel model highly adapted to SSS in human skin. It is therefore quite unapparent if the extension to arbitrary materials will be possible in a straightforward manner.

For the purpose of testing different approaches and kernels, an optimization framework, based on MATLAB and its included optimization functions, was developed. For the majority

of tests, the optimization goal was to minimize the root mean square (RMS) error between the approximating kernel and the ground-truth kernel obtained via MCML simulations.

The first approach was to see if it is possible to approximate the diffuse reflectance profile of the simulated materials by simply optimizing the w, t and f parameters of the original model, according to Equation 4.13.

$$\begin{gathered}\underset{w,t,f}{\operatorname{argmin}} \left\{ \int_{\Re^2} \left(R_d(x,y) - A_s(x,y;w,t,f)\right)^2 \, dx \, dy \right\} \\ \text{with} \\ A_s(x,y;w,t,f) = a_s(x;w,t,f) \; a_s(y;w,t,f)\end{gathered} \tag{4.13}$$

Unfortunately, it became apparent that this does not produce satisfactory approximations. One possible explanation is that this model has only a few degrees of freedom and the possible shapes of the profile are very limited. The model is only capable of scaling the fixed internal skin profile p (see Equation 4.8) along r, using the w or f parameter, and to some extent along its magnitude, using the parameter t. It is therefore difficult to approximate profiles which have a shape different than the used internal skin profile.

The next step was to perform the optimisation and approximation using different models for a_s with more degrees of freedom, different parameters and based on alternative functions. This is generally described by Equation 4.14, where u denotes the parameters of the chosen model.

$$\begin{gathered}\underset{u}{\operatorname{argmin}} \left\{ \int_{\Re^2} \left(R_d(x,y) - A_s(x,y;u)\right)^2 \, dx \, dy \right\} \\ \text{with} \\ A_s(x,y;u) = a_s(x;u) \; a_s(y;u)\end{gathered} \tag{4.14}$$

The tests ranged from using different sets of the Gaussian mixture parameters of the internal skin profile as optimization variables, to the direct use of Gaussian mixtures or exponentials of varying complexity. The end result of these initial tests was to define a_s based on simple zero-mean 1D Gaussian mixtures (see Equation 4.16) with a moderately low amount of components (up to 6), which offered a good tradeoff between approximation quality and optimization time.

This is similar to the approach of d'Eon and Luebke [5] with the difference that they computed a higher rank approximation A_g based on 2D Gaussian mixtures, according to Equation 4.15,

$$R_d(x,y) \approx A_g(x,y) = \sum_{i=1}^{N} w_i G(x,y;\tau_i) \tag{4.15}$$

while the initial test results, mentioned above, were based on a separable (rank-1) approximation A_s using 1D Gaussian mixtures, according to Equation 4.16.

$$\begin{gathered}R_d(x,y) \approx A_s(x,y) = a_s(x) \, a_s(y) \\ \text{with} \\ a_s(r) = \sum_{i=1}^{N} w_i G(r;\tau_i)\end{gathered} \tag{4.16}$$

First tests using kernels based on the 1D Gaussian mixture approximation looked promising, but as research progressed, this approach was replaced by several different kernel models which are described in the following chapter.

CHAPTER 5

Separable Kernel Models

This chapter describes the proposed extensions to the SSSS algorithm. Four kernel models for arbitrary materials are discussed, which offer different properties regarding approximation quality and kernel construction speed.

The general goal of the proposed kernel models is to provide a separable approximation A of an arbitrary diffuse reflectance profile R_d, according to Equation 5.1.

$$R_d(x, y) \approx A(x, y) = a(x)\, a(y) \tag{5.1}$$

The only exception is the model described in Section 5.1, since it also supports higher-rank approximations. It is, therefore, necessary to identify suitable forms for a, which enable a close separable approximation of R_d. The resulting kernel A can then be used to replace R_d, according to Equation 5.2,

$$M_e(x, y) = (E * R_d)(x, y) \approx (E * A)(x, y) = ((E * a) * a)(x, y) \tag{5.2}$$

in the computation of the radiant exitance M_e via convolution of the surface irradiance E. The separability property of A makes it possible to perform the usually required 2D convolution with A by using two consecutive 1D convolutions with a. This enables the fast approximation of SSS in real time.

In the following subsections, different models for extending SSSS to arbitrary materials are proposed and described in more detail. In order to access and explain the properties, approximation quality and motivation for the proposed models, example convolutions based on representative images and kernels for the Skin1 material are included. Please note that the convolution results shown in this chapter were computed in MATLAB and used the full-resolution kernels for illustrational purposes, while actual real-time rendering results are included in the next chapter.

The first two of the approximation models described in the following sections are deterministic and can be computed efficiently, while the last two allow more user control, but take more time to generate, as outlined in more detail in the corresponding sections. Table 5.1 shows a

Model	Visual quality	1D convolutions	Separable	Closed-form solution
d'Eon 1 Gaussian	low	2	✓	✗
d'Eon 2+ Gaussians	high	2 per Gaussian	✗	✗
SVD rank-1	low	2	✓	✓
SVD rank-2+	excellent	2 per rank	✗	✓
Kernel pre-integration	high	2	✓	✓
Guided optimization	controllable	2	✓	✗
Manual approximation	controllable	2	✓	✓

Table 5.1: Overview and comparison of the proposed kernel approximation models and their properties. The attribute 'Closed-form solution' also indicates if the kernel computation requires optimization (✗) or not (✓).

quick overview of the different kernel models and their characteristics, and also includes the properties of the state-of-the-art method by d'Eon et al. [5, 6], for easy comparison.

5.1 Singular Value Decomposition

The initial tests mentioned previously resulted in a separable kernel model which was based on a 1D Gaussian Mixture (see Equation 4.16) and minimization of the commonly used RMS error metric in kernel space (see Equation 4.14). However, one general drawback of using optimization is that, depending on the function and its complexity, the computation of one single kernel can be very time consuming and sometimes even impractical. Therefore, it would be more desirable to have a deterministic scheme which allows the computation of a separable approximation and automatically (inherently) minimizes the kernel-space RMS error.

Fortunately, mathematics offers such a scheme, which is called singular value decomposition (SVD), which can be used to decompose a matrix M into the product of three specific matrices, often denoted as U, Σ and V, according to Equation 5.3. M represents an arbitrary matrix of size $m \times n$. Matrices U and V are both orthogonal and of size $m \times m$ and $n \times n$, respectively, and Σ is a diagonal matrix of size $m \times n$ with non-negative entries.

$$M = U\Sigma V^T \tag{5.3}$$

This decomposition can also be applied to the discrete form of a diffuse reflectance profile $R_d \in \mathbb{R}^{m \times m}$, which is illustrated in Equation 5.4

$$\begin{gathered} R_d = U\Sigma V^T \\ \text{with} \\ U = \left(u^{(1)}|u^{(2)}|\cdots|u^{(m)}\right) \\ V = \left(v^{(1)}|v^{(2)}|\cdots|v^{(m)}\right) \\ \Sigma = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_m) \end{gathered} \tag{5.4}$$

One very useful application of SVD is that it can be used to generate a low-rank approximation of R_d. Such an approximation A_N can be computed by only using a subset (first N values) of the singular values of Σ in the reconstruction, according to Equation 5.5.

$$\begin{gathered} A_N = U\Sigma_N V^T \\ \text{where} \\ \Sigma_N = \text{diag}(\sigma_1, \cdots, \sigma_N, 0, \cdots, 0) \end{gathered} \tag{5.5}$$

Using this scheme, it is possible to compute an approximation of R_d with a rank as low as one. This is particularly useful since a filter kernel of rank one is separable. Furthermore, following from Eckart and Young [10], the best solution A to the minimization problem shown in Equation 5.6,

$$\begin{gathered} \min_A \|R_d - A\|_F \\ \text{subject to} \\ \text{rank}(A) = N \end{gathered} \tag{5.6}$$

with respect to the Frobenius norm (see Equation 5.7),

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij}^2} = \sqrt{\sum_{i=1}^{m} \sigma_i^2} \tag{5.7}$$

is given by the SVD approximation A_N (see Equation 5.5.

In the previously mentioned initial tests, discussed in Section 4.6, separable solutions which minimized the RMS error were optimized based on 1D Gaussian mixtures (see Equation 4.16 and 4.14). As the Frobenius norm corresponds to the RMS error, it was possible to replace these previous 1D Gaussian mixture approximations by a simple and deterministic rank-1 approximation A_1 based on SVD, shown in Equation 5.8.

$$\begin{gathered} R_d \approx A_1 = u^{(1)}\sigma_1 {v^{(1)}}^T \\ \text{or equivalently} \\ A_s = a_s\, {a_s}^T \\ \text{where} \\ a_s = u^{(1)}\sqrt{\sigma_1} \end{gathered} \tag{5.8}$$

Please note that U and V are identical, since the approximated 2D kernel is radially symmetric. It is therefore sufficient to simply use $u^{(1)}$ and the square root of σ_1 for a_s.

The quality of such a rank-1 approximation is closely related to the magnitude of the singular values represented by the diagonal entries of matrix Σ. The number of non-zero singular values corresponds to the rank of the matrix. Consequently, the lower the rank, the more likely that a low-rank approximation with an acceptable quality can be computed. Furthermore, the magnitude of the singular values and the total energy distribution among them also influences the quality of low-rank approximations. In case the majority of the energy is only contained in

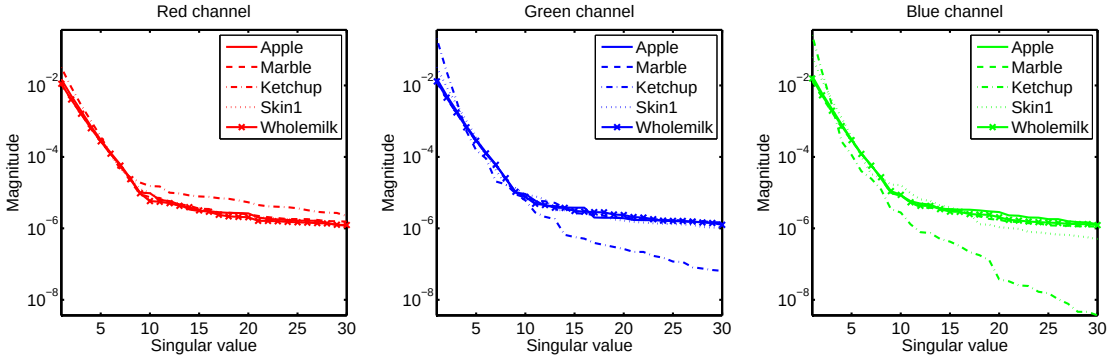


Figure 5.1: This figure shows the first 30 singular values of the individual channels for the ground-truth kernels of the five main materials used for final rendering. The plots indicate a rapid decay of the singular-value magnitude for all materials, which makes low-rank approximations feasible.

a few of the first singular values, approximations with very low rank are feasible. This is the case for most of the tested material profiles and makes low-rank approximations possible. An overview for the singular values of the ground-truth kernels of the five main materials used for final rendering can be seen in Figure 5.1.

Rank-one approximations have the advantage that they can be applied for SSS rendering by using only two 1D convolution passes, as described by Equation 5.9.

$$M_e(x,y) = (E * R_d)(x,y) \approx (E * A_s)(x,y) = ((E * a_s) * a_s)(x,y) \tag{5.9}$$

However, depending on the approximated diffuse reflectance profile, it is possible that such a rank-1 approximation, due to its low rank, does not provide the desired quality needed for a particular rendering scenario. In such a case, it is possible to simply compute higher rank approximations and apply them using multiple passes, as shown in Equation 5.10, similar to the application of the 2D Gaussian mixture approach by d'Eon and Luebke [5].

$$\begin{gathered}(E * R_d)(x,y) \approx (E * A_N)(x,y) = \sum_{i=1}^{N}((E * a_i) * a_i)(x,y) \\ \text{where} \\ a_i = u^{(i)}\sqrt{\sigma_i}\end{gathered} \tag{5.10}$$

A higher rank filter kernel is simply applied by using a different SVD component a_i for each separable convolution pass, based on the corresponding U column and Σ entry. The drawback of this higher rank approach is that it requires two 1D-convolutions per rank and, therefore, more time for rendering, which may be not feasible for certain real-time applications.

Energy Conservation:

One disadvantage of the SVD-based approach is that it does not take energy conservation into account. In this context energy conservation means that the 1-norm (see Equation 5.11) of the

approximation kernel A is not equal to the 1-norm of the approximated ground-truth diffuse reflectance profile R_d. For the discrete representation of R_d the 1-norm is computed by a simple summation of all absolute values according to Equation 5.11.

$$\|R_d\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{m} |a_{ij}| \tag{5.11}$$

In case an approximation A has a different 1-norm than R_d, $\|R_d\|_1 \neq \|A\|_1$, and an equal 1-norm may be artificially enforced by proper scaling of the approximation kernel according to Equation 5.12.

$$\tilde{A} = A \frac{\|R_d\|_1}{\|A\|_1} \tag{5.12}$$

However, this may also introduce distortions, and, in consequence, does not conserve the property of the SVD-based approximation of being minimal in terms of the RMS error metric. In general, SVD-based rank-1 approximations have different 1-norm in comparison to the approximated profile R_d, In case of our implementation the ground-truth profiles are all normalized, meaning $\|R_d\|_1 = 1$. Since the SSSS algorithm also normalizes the kernels prior to convolution, a kernel that is not energy conserving in the above sense, may get distorted in an undesirable way. In case of the SVD-based approximation approach, this issue gets more and more insignificant as higher-rank approximations are used, as they are closer to the ground truth and therefore also have similar 1-norm.

For SVD-based rank-1 approximations, however, the difference in the 1-norm is more significant, as outlined by the following example. Figure 5.2 shows the RMS errors of three different SVD-based approximation kernels for the example material Skin1. The errors represent the difference of the approximation kernel A_N to the ground-truth diffuse reflectance profile R_d, which may also be referred to as the difference in kernel space. A separable (rank-1) and energy conserving approximation kernel, which is introduced and discussed further below in Section 5.2, is included for comparison.

These plots show that the normalization increases the kernel-space error of all SVD-based approximations, while the error difference decreases with increasing rank, as higher rank approximations exhibit less error, and, therefore, a more similar 1-norm. The normalization destroys the property of the SVD-based approximation of being minimal in therms of kernel-space RMS error. This example illustrates that energy conservation is important, since the SSSS algorithm normalizes all kernels before the convolution, and therefore SVD-based kernels may get distorted. This can also be seen in the corresponding plots shown in Figure 5.3–5.4, which illustrate the distortions by comparing the normalized and not normalized SVD-based kernels. The normalization of the kernel is necessary to ensure that the energy of the input signal (irradiance) is not changed by the convolution operation, in order to satisfy the energy conservation of light-transport.

The low quality of the SVD-based rank-1 approximation is further illustrated by comparisons of the images included in Figure 5.6. Below each image a corresponding difference image shows the absolute difference between the ground-truth convolution result and its approximation. This is basically the difference between the radiant exitance $M_e(x, y) = (E * R_d)(x, y)$

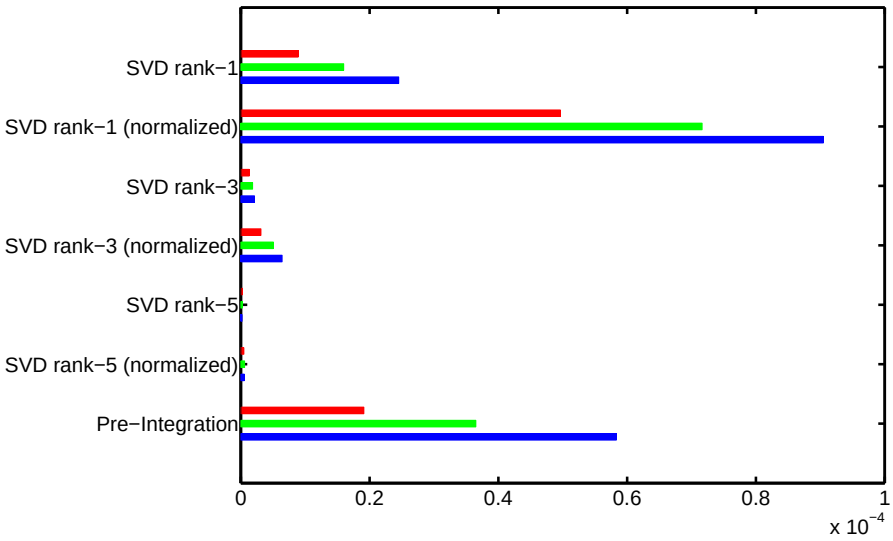


Figure 5.2: This figure shows a comparison of the RMS errors for three SVD-based approximation kernels of the Skin1 material. It shows that normalization of the SVD-based kernels increases the RMS errors, and that the difference vanishes with increasing rank. Subsequently, the property of SVD-based approximations of being minimal in terms of the RMS error is destroyed. An energy-conserving kernel model based on pre-integration, which is discussed in Section 5.2, was included for comparison. Please note that the different colors correspond to the RGB channels of each kernel. The corresponding kernel plots can be seen in Figures 5.3–5.4 and A.30.

(convolution with the ground truth) and $\widetilde{M}_e(x, y) = (E * A)(x, y)$ (convolution with the approximation), which can also be referred to as the difference in image-space. The convolutions and the corresponding difference images indicate that the rank-1 SVD-based approximation is of low quality, as it fails to model the falloff region of the ground-truth kernel. Alternative kernel models which are separable, energy-conserving, and are able to approximate such falloff regions more closely are proposed later on in this chapter.

Figure 5.3, in particular, shows that for the SVD-based rank-1 kernel the lack of energy conservation in combination with normalization leads to distortions. Apart from these distortions the plots also indicate that the SVD-based rank-1 kernel fails to approximate the ground-truth, especially at the diagonal. The comparison of the convolution results (Figure 5.6) with respect to the additional energy-conserving rank-1 example kernel (pre-integration), indicate that, especially in case of rank-1 approximations, a minimal RMS error in kernel space does not guarantee that the final rendering result will be of sufficient quality.

As the final rendering quality is the main criteria for the assessment of the approximation models, it is, therefore, important to analyse the kernel approximations in image space. Furthermore, special care has to be taken in order to assure energy conservation, which is the reason why the remaining kernel models, described in this chapter, ensure that the 1-norm of the ap-

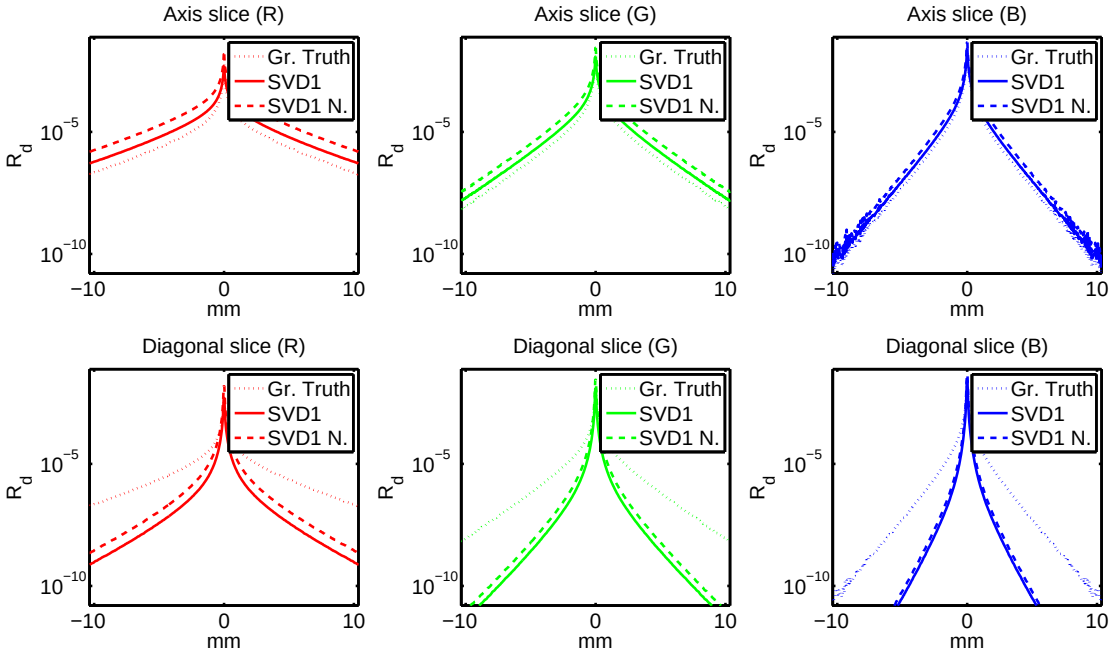


Figure 5.3: This figure shows plots of the SVD1 kernel approximation for the Skin1 material before (SVD1) and after normalization (SVD1 N.). As this rank-1 SVD-based approximation kernel does not preserve energy by construction, the forced normalization distorts the kernel, leading to higher RMS errors as shown in Figure 5.2.

proximation kernel is equal to that of the approximated ground-truth profile.

An additional limitation of SVD-based approximation kernels is that they are in general not radially symmetric, as shown in Figure 5.7. This radial asymmetry vanishes with increasing rank, and is most noticeable for separable (rank-1) SVD-based approximations. Furthermore, any separable (rank-1) kernel that is different from a 2D Gaussian kernel exhibits radial asymmetry to some extent, as the latter is the only kernel which is both separable and radially symmetric. This can lead to cross-pattern artefacts that were also previously outlined in Section 4.1, and are demonstrated in Section 6.4. The comparable approach by d'Eon and Luebke [5] does not suffer from this problem, as it uses a 2D Gaussian mixture where each Gaussian is radially symmetric.

The discussed properties of the SVD-based approximation scheme are illustrated by several examples and results included in Chapter 6, which indicate that rank-1 SVD-based approximations appear to be of insufficient quality, while higher-rank SVD approximations quickly converge to the ground truth with increasing rank. Furthermore, in comparison to the state-of-the-art Gaussian mixture approach of d'Eon and Luebke [5], the SVD-based approximation scheme has a favorable convergence rate with increasing rank and with respect to equal count of required convolution passes.

However, the application of low-rank approximations results in several convolutions per

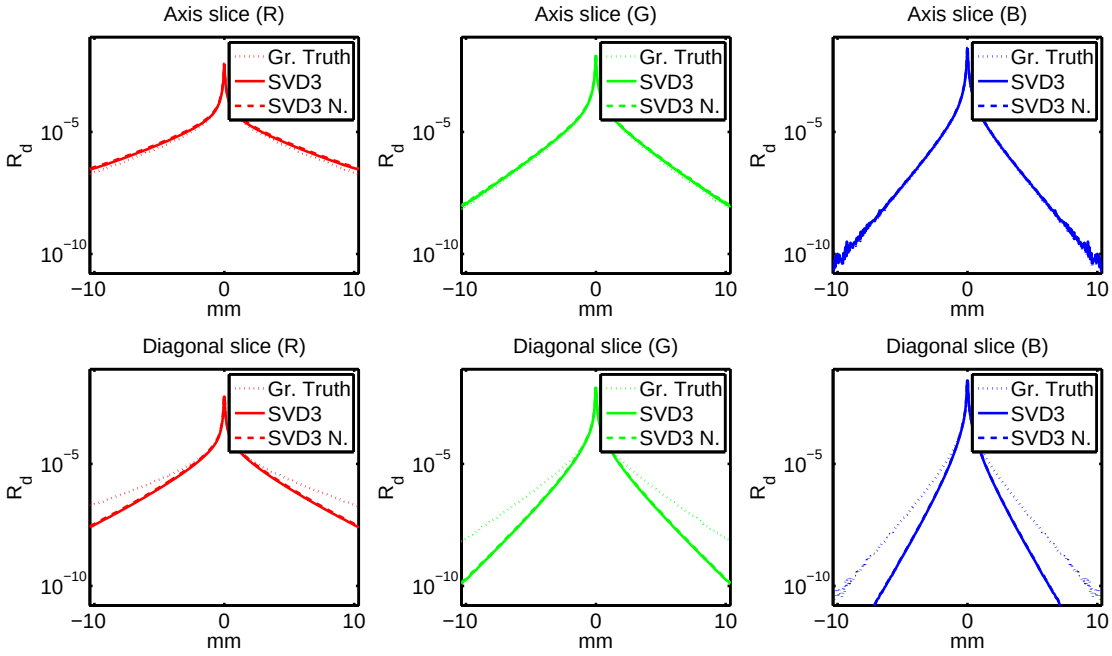


Figure 5.4: This figure shows plots of the SVD3 kernel approximation for the Skin1 material before (SVD3) and after normalization (SVD3 N.). This rank-3 SVD-based approximation kernel show less distortion by the normalization, as indicated by the kernel-space RMS errors shown in Figure 5.2.

frame, which may still be too time consuming for certain rendering scenarios. It is therefore desirable to find, in the spirit of the SSSS algorithm, a rank-1 kernel model of sufficient quality. Consequently, an energy-conserving rank-1 (only) kernel model is presented in the following section.

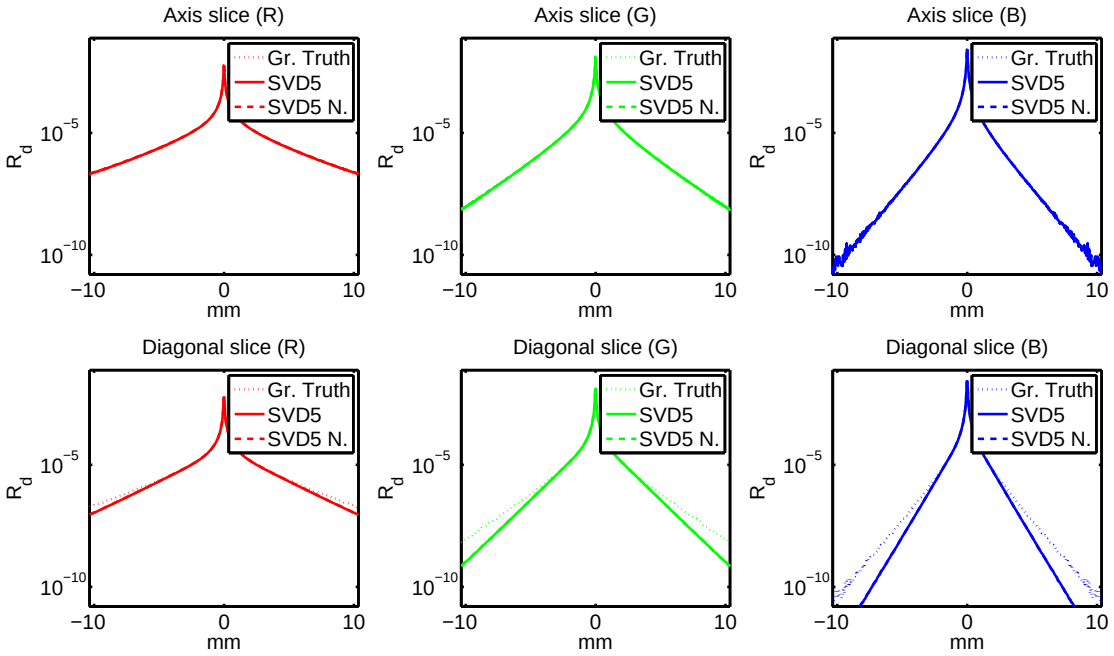
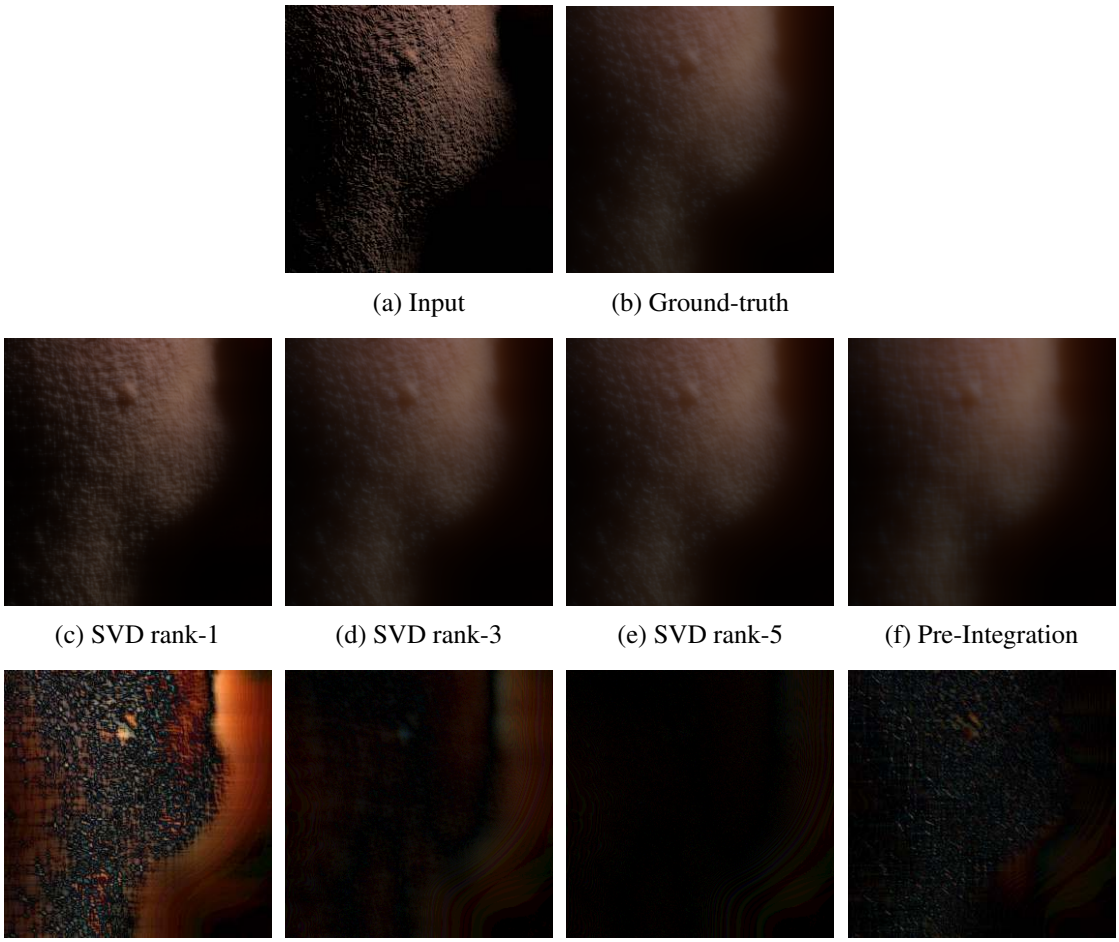


Figure 5.5: This figure shows plots of the SVD5 kernel approximation for the Skin1 material before (SVD5) and after normalization (SVD5 N.). Since this rank-5 SVD-based approximation kernel has a low RMS error, as shown in Figure 5.2, the distortions due to normalization are minimal.

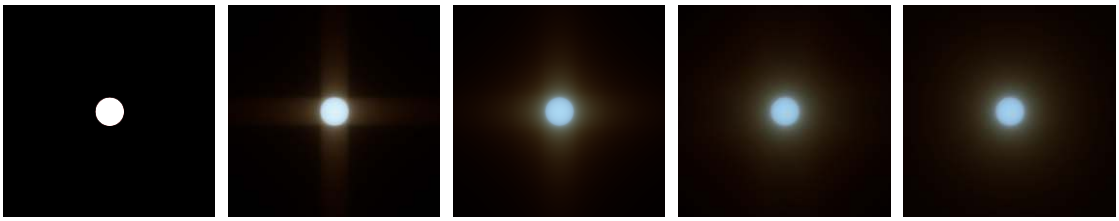


(a) Input (b) Ground-truth

(c) SVD rank-1 (d) SVD rank-3 (e) SVD rank-5 (f) Pre-Integration

(g) SVD rank-1 (h) SVD rank-3 (i) SVD rank-5 (j) Pre-Integration

Figure 5.6: This figure shows convolutions and difference images for the kernel approximations of the Skin1 material used as examples for the SVD-based approximation issues discussed in Section 5.1. Images (a) and (b) represent the input and ground-truth convolution results, receptively. The second row shows the convolution results for the example kernels, while the corresponding difference images are presented in the last row. Please note that the difference images represent the absolute differences which are normalized across all images. Therefore, most difference images are considerably dark in comparison to the rank-1 SVD-based difference image, which has the higher error.



(a) Input (b) SVD rank-1 (c) SVD rank-3 (d) SVD rank-5 (e) Ground-truth

Figure 5.7: This figure illustrates the radial asymmetry of SVD-based approximations of the Skin1 material as an example. The radial asymmetry is most pronounced for the rank-1 approximation, and vanishes with increasing rank.

5.2 Pre-Integration

The rank-1 kernel approximation model based on SVD decomposition presented in the previous section enables the computation of approximation kernels that are optimal in terms of the kernel-space RMS error. However, as outlined previously, the simple kernel-space RMS metric is not sufficient to assess the final approximation quality, since the rank-1 SVD-based approach does not deliver satisfactory approximation quality in image space, as indicated by Figure 5.6, and is not energy-conserving. Therefore, a rank-1 kernel approximation model based on analytic pre-integration, that is optimal in terms of the image-space RMS error in case of certain irradiance signals, and energy-conserving per definition, is proposed in this section.

The main idea of this approach is to derive an approximation kernel A_p in such a way that it produces the exact same convolution result as the original profile R_d for a special class of input irradiance signals. Let's consider an idealised vertical or horizontal shadow boundary or edge as an example for such a special irradiance signal. Since the signal does not change along the direction of the edge, the convolution result signal is also constant along the same direction. This makes it possible to achieve the same convolution result by using a 1D kernel that is derived by simple pre-integration of the 2D kernel along the corresponding direction, as will be shown later. Thus, the resulting kernel is separable. Although such a kernel would only be exact for a very specific class of idealized irradiance signals, general irradiance signals could be loosely considered as being comprised of multiple different edges. Therefore, the application of such a kernel for arbitrary signals seems feasible, and is supported by the results included in the next chapter.

The goal is to derive an approximation kernel A_p that is exact in the sense that it produces the same radiant exitance M_e as R_d for a special class of irradiance signals E, according to Equation 5.13.

$$M_e(x, y) = (E * R_d)(x, y) = (E * A_p)(x, y) \tag{5.13}$$

As special class of signals we consider so-called additively separable functions, i.e.,

$$E(x, y) = E_1(x) + E_2(y). \tag{5.14}$$

Such signals have the property that $\frac{\partial E}{\partial x \partial y} = \frac{\partial E}{\partial y \partial x} = 0$. An example would be an irradiance signal which is symmetric about one principal axis, e.g., a vertical or horizontal step, edge or shadow boundary. The convolution of such a signal with the kernel A_p would match exactly the ground truth result based on convolution with R_d.

Such an approximation kernel A_p can be computed via pre-integration of R_d, as explained

later, and the corresponding kernel model can be derived according to Equation 5.15.

$$
\begin{aligned}
M_e(x,y) &= \int\int E(x',y')\; R_d(x-x',y-y')\; dx'\; dy' = \\
&= \int\int (E_1(x') + E_2(y'))\; R_d(x-x',y-y')\; dx'\; dy' = \\
&= \int E_1(x') \underbrace{\int R_d(x-x',y-y')\; dy'}_{a_p(x-x')}\; dx' \\
&+ \int E_2(y') \underbrace{\int R_d(x-x',y-y')\; dx'}_{a_p(y-y')}\; dy' = \\
&= \int E_1(x')\; a_p(x-x') \underbrace{\frac{1}{\|a_p\|_1} \int a_p(y-y')\; dy'}_{=1}\; dx' \\
&+ \int E_2(y')\; a_p(y-y') \overbrace{\frac{1}{\|a_p\|_1} \int a_p(x-x')\; dx'}\; dy' = \\
&= \int\int E(x',y')\; \frac{1}{\|a_p\|_1}\; a_p(x-x')\; a_p(y-y')\; dx'\; dy'
\end{aligned}
\tag{5.15}
$$

Let's assume we want to compute the radiant exitance M_e by convolving an additively separable irradiance signal E with the ground-truth diffuse reflectance profile R_d. Due to the fact that the irradiance signal E is separable, it is possible to split the integral into two terms. The inner integrals can then be simplified and replaced by the terms $a_p(x-x')$ and $a_p(y-y')$, which can be computed by pre-integrating R_d along the corresponding independent dimension x and y, respectively. So a_p basically represents the integration of R_d along one of its dimensions (x or y), and the result is the same for both dimensions, since R_d is radially symmetric. This, subsequently, enables to express $R_d(x-x',y-y')$ as $\frac{1}{\|a_p\|_1}\; a_p(x-x')\; a_p(y-y')$, where $\|a_p\|_1$ represents the 1-norm of a_p. Therefore, $M_e(x,y)$ can be finally represented according to Equation 5.16.

$$
\begin{aligned}
M_e(x,y) &= \int\int E(x',y')\; R_d(x-x',y-y')\; dx'\; dy' = \\
&= \int\int E(x',y')\; \frac{1}{\|a_p\|_1}\; a_p(x-x')\; a_p(y-y')\; dx'\; dy'
\end{aligned}
\tag{5.16}
$$

Since a_p is the integration of R_d along one axis, it has the same 1-norm as R_d ($\|a_p\|_1 = \|R_d\|_1$). Due to the (radial) symmetry of R_d it is possible to write the proposed analytic pre-integrated

kernel approximation A_p according to Equation 5.17.

$$\begin{gathered} A_p(x,y) = \frac{1}{\|R_d\|_1}\, a_p(x)\, a_p(y) \\ \text{with} \\ a_p(x) = a_p(y) = \int_{\mathbb{R}} R_d(x,y)\, dx = \int_{\mathbb{R}} R_d(x,y)\, dy \end{gathered} \tag{5.17}$$

The approximation kernel A_p is separable, and the scaling factor $\frac{1}{\|R_d\|_1}$ ensures that it is energy-conserving with respect to the 1-norm ($\|A_p\|_1 = \|R_d\|_1$). This pre-integration scheme may also be applied to more general filters, however, in such a case the approximation kernel would include two different a components.

The proposed pre-integrated approximation kernel can be applied for convolution according to Equation 5.18.

$$\begin{gathered} M_e(x,y) = (E * R_d)(x,y) \approx (E * A_p)(x,y) = ((E * a) * a)(x,y) \\ \text{with} \\ R_d(x,y) \approx A_p(x,y) = a(x)\, a(y) \\ \text{where} \\ a(r) = \sqrt{\frac{1}{\|R_d\|_1}}\, a_p(r) \end{gathered} \tag{5.18}$$

Here, E represents an arbitrary irradiance signal, and, therefore, the ground-truth radiant exitance M_e is only approximated.

This pre-integration kernel model offers an energy-conserving separable (rank-1) approximation that provides, per definition, an optimal solution in case of additively separable input signals with respect to the image-space RMS error, but not for general input signals. However, it is possible to use this kernel even for arbitrary irradiance signals, as shown by the final results included in Chapter 6.

An additional drawback of this approach is that its quality is not scalable like the SVD-based approach, by using multiple convolution passes. It only provides a single separable solution without any user control over the approximation quality, which is not optimal in the general case. A more general energy-conserving rank-1 kernel model which provides a certain amount of control over the approximation quality is presented in the following section.

5.3 Guided Optimization

The two kernel models described in the previous sections both provide separable (rank-1) approximations, while only the pre-integrated kernel model is energy-conserving and only optimal for a special class of irradiance signals, but not in general. The SVD-based approach is only approximately energy-conserving at higher ranks, while providing some control over the approximation quality at the cost of additional convolution passes. In this section a separable (rank-1)

and energy-conserving kernel model based on optimization is proposed, which is more general than the pre-integration approach, and, additionally, allows more control over the approximation quality.

It is, as mentioned previously, important to obtain a good approximation quality in image space, since simple minimization of the RMS error in kernel space is not sufficient, as illustrated by the rank-1 SVD-based approximation, which is the optimal solution in terms of the kernel-space RMS error. Therefore, the optimal approach would be to optimize directly in image space. Unfortunately, this approach poses problems in terms of increased computation time and dependence on the actual input irradiance signal, which are discussed later in this section. The alternative approach used for the proposed model is, therefore, to guide the approximation in kernel space, based on the minimization of a weighted RMS term in combination with an additional constraint which ensures that the approximation kernel has the same 1-norm as the approximated profile.

The motivation for this guided optimization approach stems from the fact that the rank-1 SVD-based approximations are really bad at capturing the falloff of diffuse reflectance profiles, do not have the same 1-norm as the approximated profile, and allow no control over the approximation quality, except by using higher-rank approximations. The bad falloff approximation is illustrated by the example plots given in Figure 5.8, which show the red channels of multiple rank-1 kernels of the Skin1 material, for which corresponding example images are included in Figure 5.10 and 5.12. Not only do these plots and images illustrate that a rank-1 SVD-based approximation poorly captures the falloff region, which is a characteristic feature of SSS, but they also indicate that rank-1 approximations of higher quality are possible. One example is the pre-integration model, described in the previous section, which unfortunately only provides a fixed-quality solution. Due to the fact that the rank-1 SVD-based approach is already optimal in terms of the kernel-space RMS error, but exhibits low falloff approximation quality, the idea was to optimize a weighted RMS term in order to enable better falloff approximation. The RMS term is weighted by a parametrized function that allows shifting of the approximation accuracy between different regions of the kernel, which enables control over the approximation quality. The optimization also includes an additional constraint to ensure that the approximation kernel has the same 1-norm as the approximated profile, which is not the case for rank-1 SVD-based approximations.

The main goal of the optimization is to find a separable kernel $A_s(x, y) = a(x)a(y)$ that approximates the diffuse reflectance profile $R_d(x, y)$ as close as possible and additionally provides an equal 1-norm. The corresponding optimization approach is represented by Equation 5.19.

$$\begin{gathered} \underset{a}{\operatorname{argmin}} \int_{\mathbb{R}^2} \Gamma(x, y; k)\ (R_d(x, y) - a(x)a(y))^2\ dx\ dy \\ \text{subject to} \\ \|R_d\|_1 = \|a\|_1^2 \end{gathered} \tag{5.19}$$

The objective function includes a weighted RMS term combined with an additional constraint that ensures an equal 1-norm for the approximation A_s and the profile R_d. The term a represent a generic function and is the parameter subject to optimization. Furthermore, Γ (see Equation

5.20) is a weighting function defined over $\mathbb{R}^2$, which can be used to assign weights to the different regions of the kernel via the k parameter. With increasing value of k, the outer regions of the kernel (falloff) are approximated more closely, at the expense of a lower approximation quality at the center, which allows to guide and control the optimization, the approximation quality and the kernel shape. The particular form of the weighting function and the parameters are explained later in this section, after a more detailed motivation for this guided optimization approach using a custom weighting function.

The motivation for additional control over the optimization via the weighting function (shown in Equation 5.20) can be illustrated and explained via Figure 5.9, which shows two sets of images-space RMS errors for approximation kernels optimized using different parameters for the weighting function, and additional rank-1 kernels for comparison. The image-space errors were computed based on two example images showing patches of skin, where one is uniformly lit (uniform), while the other includes more pronounced shading (shaded).

Let's consider the RMS errors for the guided optimization kernels corresponding to a weighting function with $k = 0$ (Guided optimization, k=0) and $k = 2$ (Guided optimization, k=2). The former represents $\Gamma(x, y) = 1$ (equal weights) and, therefore, the optimization corresponds, apart from the additional constraint, to the minimization of the classical RMS error, while the latter ($k = 2$) applies higher weights to regions further away from the center, in which case the optimization does not minimize the classical RMS error. The comparison of the corresponding errors reveals that the image-space errors for $k = 0$ are higher than those for $k = 2$ for the shaded skin patch, and almost (at least for the red and green channel) vice versa for the uniformly lit patch of skin.

This may be explained via the input images, as they represent two quite different irradiance signals, as illustrated in Figure 5.10 and 5.12. Choosing $k = 0$ results in equal weights for all kernel regions and, in combination with the 1-norm constraint, gives a kernel with higher approximation quality at the center regions than in the falloff. By choosing $k = 2$, the falloff is approximated more closely than the center regions, due to the shifted weights distribution. This basically means that with higher k, the outer regions (falloff) of the kernel are approximated more closely, at the expense of lower approximated quality at the center. Therefore, $k = 0$ results in a lower image-space RMS error for the uniformly lit skin patch in comparison to $k = 2$, since there is no pronounced shadow region where a low falloff quality could be visible, and the high-frequency details are preserved, which is not the case for $k = 2$. And for the shaded skin patch, $k = 2$ gives a lower image-space RMS error in comparison to $k = 0$, since falloff regions are the predominant image feature, which $k = 0$ fails to approximate at high quality.

Not only does this illustrate the dependence of the image-space error on the used input image, but also that even with an additional norm-1 constraint, the minimization of the classical uniformly-weighted RMS error ($k = 0$) is, in general, not always sufficient in order to find an appropriate separable approximation kernel with a low image-space RMS error. Since the image-space error does not solely depend on the kernel, but also on the used input image for the convolution, and this error type measures the quality of the final result, the optimal goal would be to optimize the kernel in image space using multiple representative input images. This approach however can be considered as impractical, since multiple input images and the

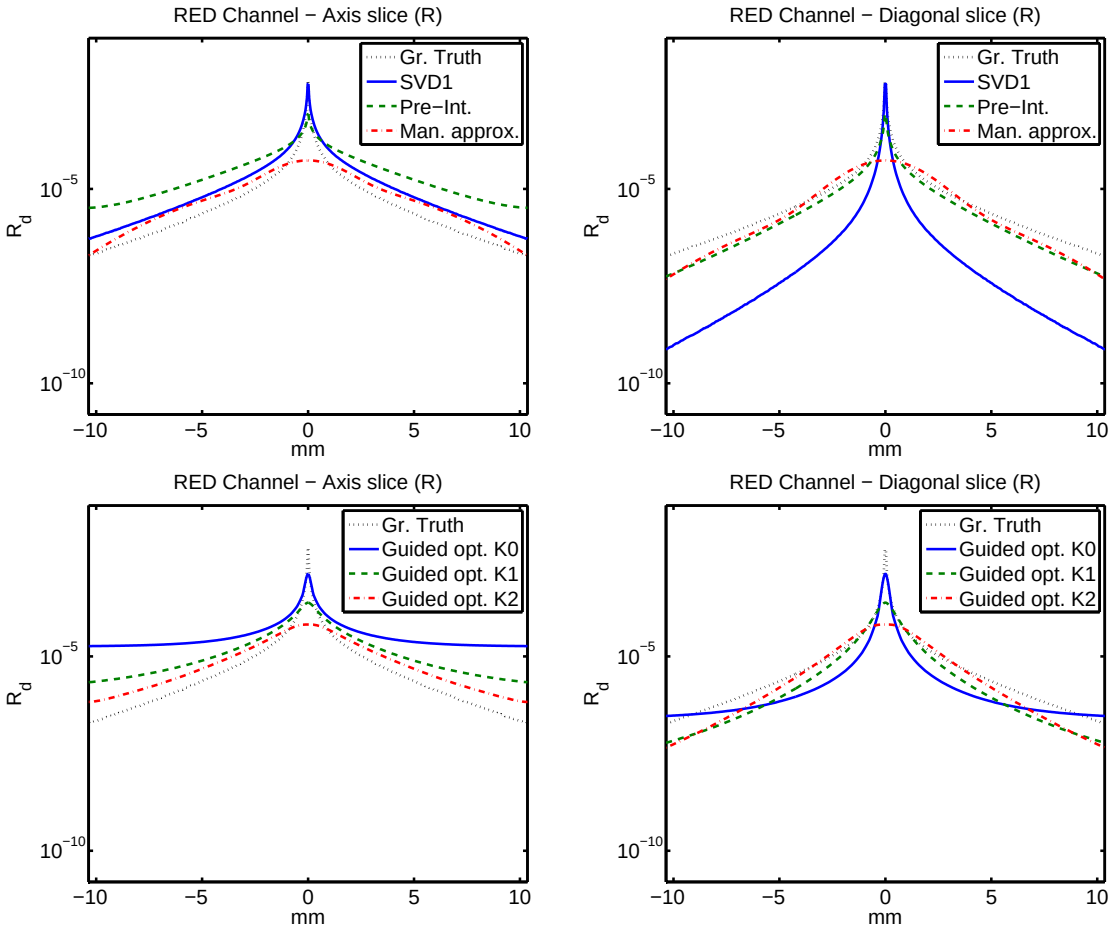


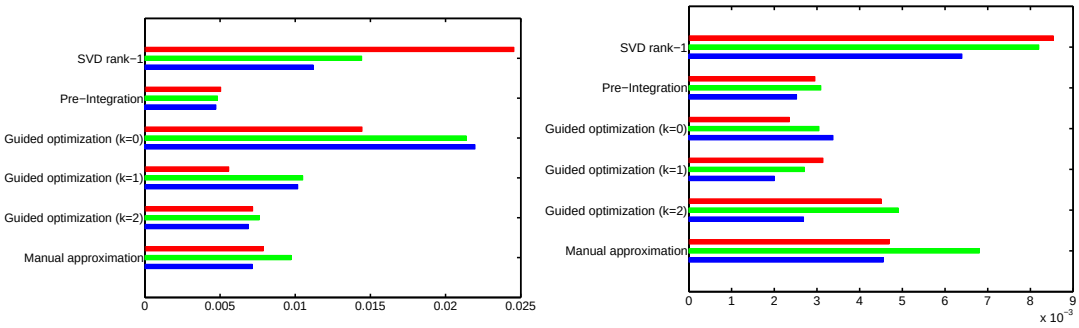
Figure 5.8: This figure shows plots of different kernels for the Skin1 material corresponding to the images shown in Figure 5.10 and 5.12. Please note that only the red channel is shown for easy comparison, and that the line colors do not correspond to the RGB channels.

required convolutions in combination with the high dimensionality of the optimization problem would render the automatic optimisation unusable due to a significant increase in computation time.

Therefore, an alternative kernel-space approach, proposed via the optimization scheme shown in Equation 5.19, was chosen, namely to guide the optimization in kernel space via a weighting function. This weighting function $\Gamma(x, y; k)$ can be used to emphasize different kernel regions based on the parameter k, according to Equation 5.20.

$$\Gamma(x, y; k) = \left(x^2 + y^2\right)^{k/2} \left(1 - e^{-bx^2}\right) \left(1 - e^{-by^2}\right) \tag{5.20}$$

By variation of parameter k it is possible to either use equal weight distribution ($k = 0$) or assign a higher weighting to the off-center regions ($k \geq 1$), based on the distance (radius) to the kernel center, as outlined previously.



(a) Shaded — Image-space RMS error

(b) Uniform — Image-space RMS error

Figure 5.9: This figure shows a comparison of image-space RMS errors for all separable (rank-1) approximation kernel models, using the Skin1 material as an example. Please note that the different colors correspond to the RGB channels of each kernel. The RMS errors were computed based on two different example input images, i.e. a skin patch showing pronounced shading (shaded) and a uniformly lit patch of skin (uniform). The corresponding input images and convolution results are shown in Figure 5.10 and 5.12, and the difference images are included in Figure 5.11 and 5.13. Please note that the manual approximation model is introduced later on in Section 5.4

The additional term $(1-e^{-bx^2})\ (1-e^{-by^2})$ evolved from attempts to imitate the analytic pre-integration-based kernel using the more general optimization approach. For this purpose, a guide function $\Gamma_p(x, y)$ was derived that yields the analytic pre-integration kernel as the optimization result. This guide function $\Gamma_p(x, y)$ has a quite complicated structure as shown in Figure 5.14a. In order to mimic this complex function, Equation 5.20 provides a simple parametrized alternative. By using $\Gamma(x, y; 1.55)$ with $b = 50$, it is possible to generate a guide function similar to $\Gamma_p(x, y)$, as shown in Figure 5.14b. Here, the parameter b is used to apply lower weights to the main axis regions similar to the $\Gamma_p(x, y)$ function.

Figure 5.15 shows that choosing $k = 1.55$ and $b = 50$ produces an approximation that is similar to the pre-integration kernel. This hints at the potential of the guided approximation approach, which is quite general, as arbitrary weighting functions may be used. Please note that for this particular case, the parameter value of 50 is defined with respect to a kernel-space interval of $x, y \in [-1, 1]$, and is, for the actual optimization, adapted according to the kernel's sampling intervals.

Figure 5.15, furthermore, illustrates that by application of lower weights to the main axis regions via $b > 0$, the k parameter no longer directly controls the falloff and center approximation, as the image corresponding to $k = 0$ (Figure 5.15c) no longer preserves high-frequency images features, but looks similar to $k = 2$ (Figure 5.15d). Therefore, all examples, apart from Figure 5.15, do not use this additional term and use $b = 0$, in order to maintain the controllability of the falloff and center approximation via k. Please note that in the actual implementation the case $b = 0$ is treated as a special case in which the exponential terms are not used in order to prevent them from scaling down the weight function to zero.

The proposed guide function $\Gamma(x, y; k)$ can be used to compute separable (rank-1) approximations which are energy-conserving and have image-space errors comparable to the other separable kernel models, as shown in Figure 5.9. Furthermore, by variation of the parameter k it is possible to control the approximation quality and compute different kernel approximations, which either provide higher approximation quality in the center or the falloff region, depending on the value of k.

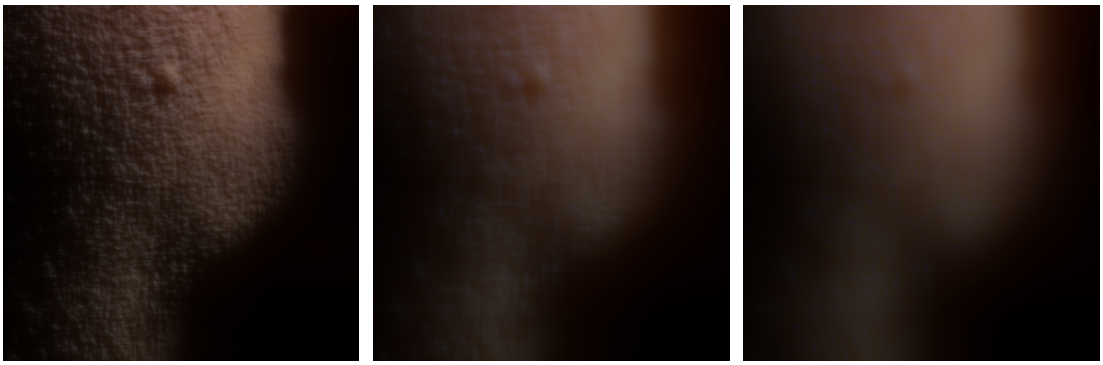
This is illustrated by Figure 5.10 and 5.12, which show the convolutions of two example images. i.e., a patch of skin with pronounced shading and a uniformly lit skin patch. Plots of the corresponding kernels (red channel only) are provided in Figure 5.8 in order to illustrate the different kernel shapes. The convolution and difference images indicate that in case of the shaded skin patch (Figure 5.10), a closer falloff approximation ($k = 1$ or $k = 2$) provides better image-space results than using $k = 0$. However, in case of the more uniformly lit skin patch, a closer approximation of the center region ($k = 0$ or $k = 1$), in order to preserve more surface detail, offers a better image-space approximation than choosing $k = 2$. For both cases, $k = 1$ seems like a reasonable trade-off between the center ($k = 0$) and falloff ($k = 2$) approximation. This is also indicated by the corresponding difference images, shown in Figure 5.11 and 5.13, and image-space errors included in Figure 5.9. These example results also show that the approximation kernel corresponding to $k = 1$ has a lower image-space error than the rank-1 SVD-based approximation in both cases. Furthermore, it is worth noting that the final image-space error is significantly influenced by the actual input image, which can be seen by comparison of the image-space RMS errors corresponding to the shaded and uniform skin patch (Figure 5.9).

Since this model represents a rank-1 kernel approximation, A_s can be applied like any separable kernel, as described previously by Equation 5.1 and 5.2. The drawback of this model is that optimization has to be used, which in the context of high-resolution kernels can be quite difficult and time consuming to compute, as outlined in Section 6.4, since the dimensionality of the optimization increases with the kernel size. In order to tackle this problem, a customized optimization framework was developed. It uses an iterative approach were the initial optimization is performed using a down-sampled version of the ground-truth kernel. All successive optimization steps are then initialised using the solution from the previous step until the full resolution is reached. For energy conservation, a soft constraint was used in order to support the efficient computation of usable solutions. This means that the norm-1 constraint was included in the objective function and was not supplied as a separate constraint to the optimization framework. It is, furthermore, worth noticing that it was necessary to weaken the influence of the soft 1-norm constraint with respect to the weighted RMS term in order to obtain reasonable approximations. This may be only necessary due to the particular implementation of the optimization framework, however, it illustrates that finding parameters which work in general is certainly no trivial task, and can make the usage of this optimization approach difficult.

The proposed guided approximation approach builds on optimization in kernel space, as automatic image-space optimization can be considered impractical, due to its high computational demands. This assessment is supported by the fact that the proposed kernel-space optimization can already take up to ~19 minutes in the worst case. This increased computation time and the

relative moderate flexibility of the guided optimization model might still be too restrictive to be used by artists in a production environment. Therefore, a more artist-friendly model which can be interpreted as a manual optimization in image space is described in the following section.

(a) Input (b) Ground-truth

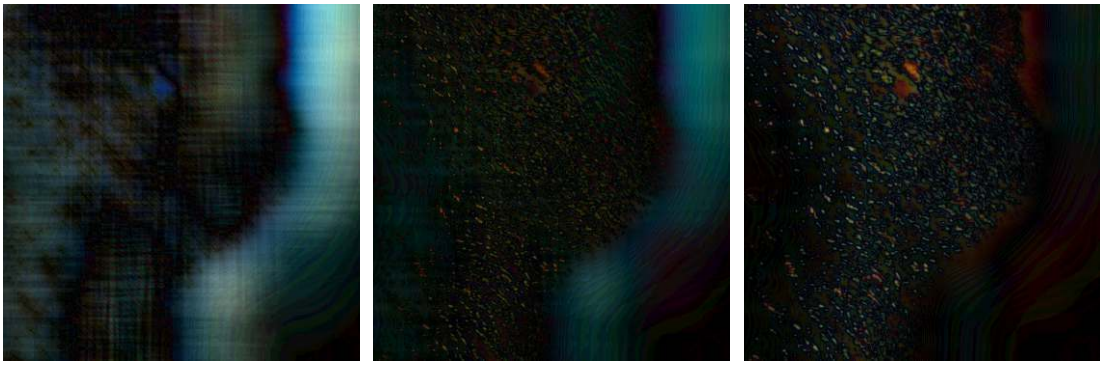


(c) SVD rank-1 (d) Pre-Integration (e) Manual approximation

(f) Guided optimization (k=0) (g) Guided optimization (k=1) (h) Guided optimization (k=2)

Figure 5.10: This figure shows images of the shaded skin patch convolved with different kernel approximations of the Skin1 material. The corresponding image-space RMS errors are shown in Figure 5.9. Images (a) and (b) represent the input and ground-truth convolution results, receptively. The remaining images show the convolution results for the example kernels, while the corresponding difference images are included in Figure 5.11.

(a) SVD rank-1 (b) Pre-Integration (c) Manual approximation



(d) Guided optimization (k=0) (e) Guided optimization (k=1) (f) Guided optimization (k=2)

Figure 5.11: This figure shows difference images corresponding to the convolutions of the shaded skin patch shown in Figure 5.10. Please note that these images represent the absolute differences which are normalized across all images.

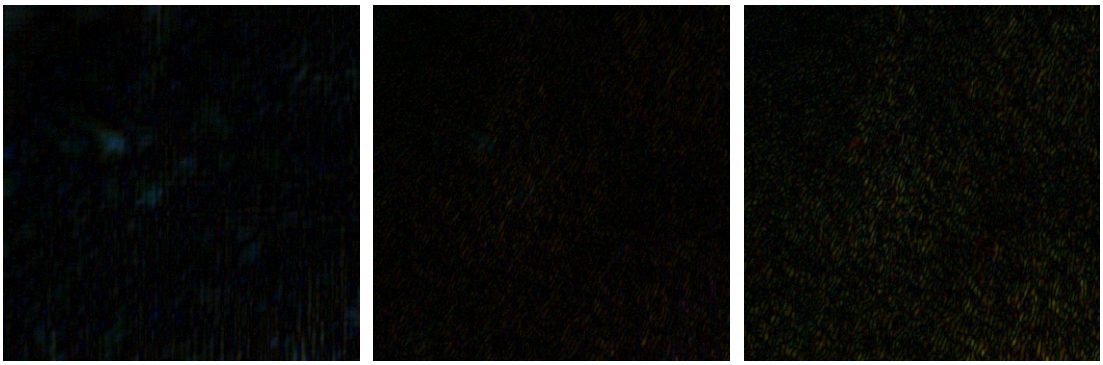
(a) Input (b) Ground-truth

(c) SVD rank-1 (d) Pre-Integration (e) Manual approximation

(f) Guided optimization (k=0) (g) Guided optimization (k=1) (h) Guided optimization (k=2)

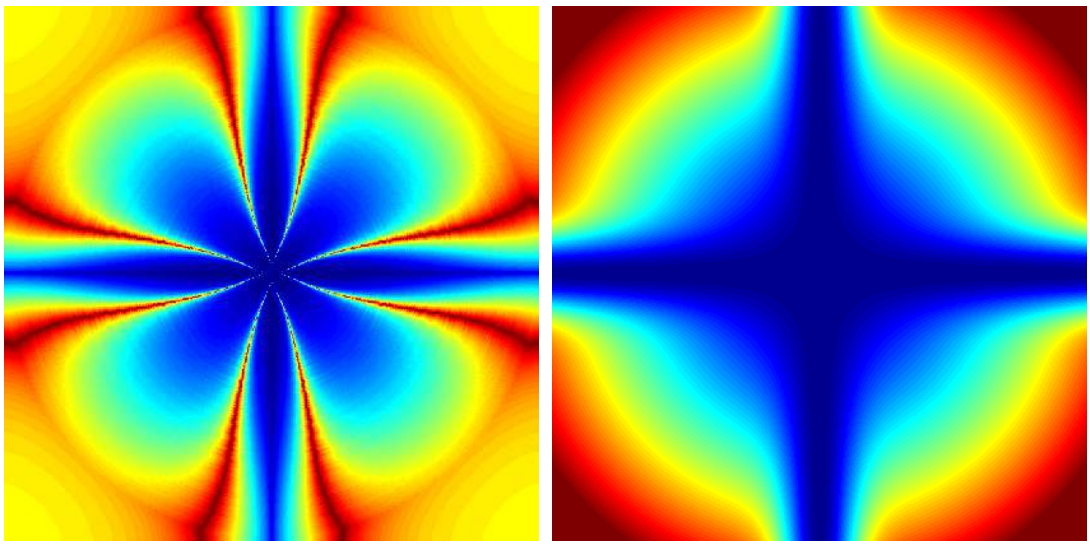
Figure 5.12: This figure shows images of the uniform skin patch convolved with different kernel approximations of the Skin1 material. The corresponding image-space RMS errors are shown in Figure 5.9. Images (a) and (b) represent the input and ground-truth convolution results, receptively. The remaining images show the convolution results for the example kernels, while the corresponding difference images are included in Figure 5.13.

(a) SVD rank-1 (b) Pre-Integration (c) Manual approximation



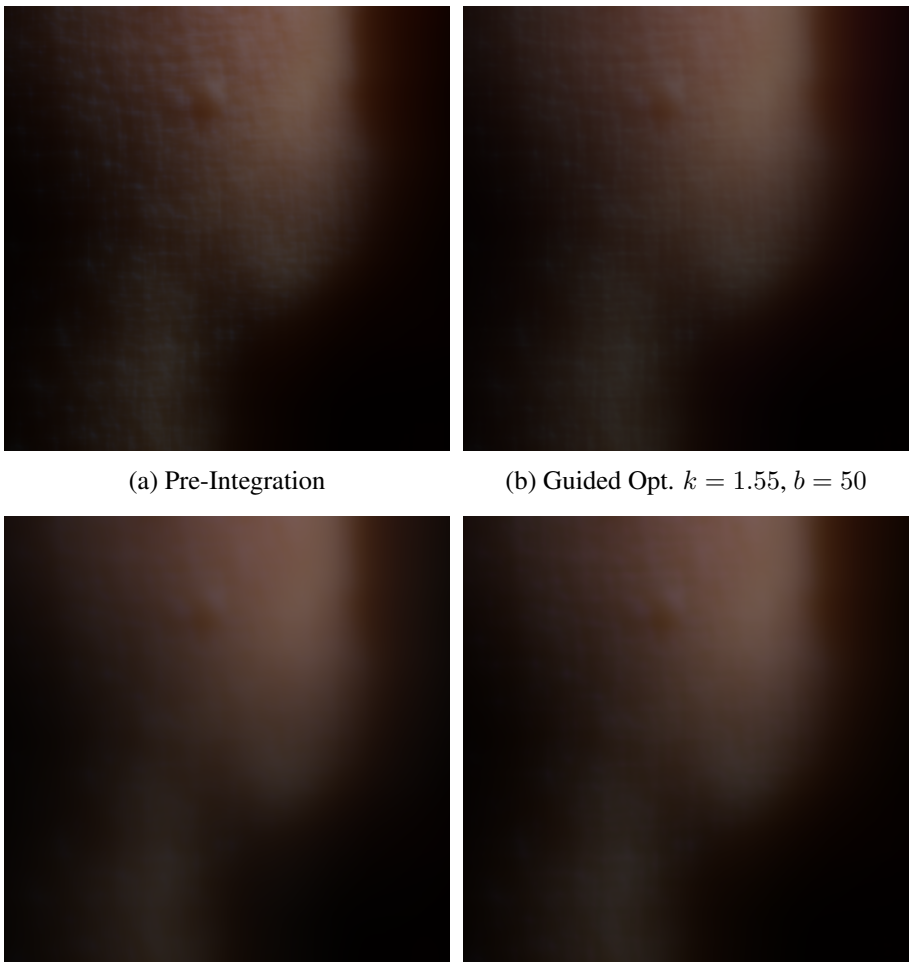
(d) Guided optimization (k=0) (e) Guided optimization (k=1) (f) Guided optimization (k=2)

Figure 5.13: This figure shows difference images corresponding to the convolutions of the shaded skin patch shown in Figure 5.12. Please note that these images represent the absolute differences which are normalized across all images.



(a) $\Gamma_p(x, y)$ (b) $\Gamma(x, y; 1.55)$

Figure 5.14: Comparison of the pre-integration-based guide function $\Gamma_p(x, y)$ and the simpler approximation using the parametrized approach $\Gamma(x, y; 1.55)$.



(a) Pre-Integration

(b) Guided Opt. $k = 1.55$, $b = 50$

(c) Guided Opt. $k = 0$, $b = 50$

(d) Guided Opt. $k = 2$, $b = 50$

Figure 5.15: This figure shows a comparison between convolutions of the shaded skin patch, using the pre-integration kernel (a) and guided approximation kernels corresponding to different parameters, where (b) shows the kernel corresponding to the parameters $k = 1.55$ and $b = 50$, which shows a similar results compared to the pre-integration kernel. Images (c) and (d) are two additional results, for which parameter b is the same as in image (b), and only the k parameter was varied. This two images illustrated that, in comparison to Figure 5.10, the influence of the k parameter is weakened, and the direct control over closer falloff or center approximation is not maintained in case the additional b parameter is used.

5.4 Manual Approximation

The kernel model described in the previous section used a guided optimization approach in kernel space in order to find kernel approximations which provide reasonable image-space results. An additional motivation for this approach was that automatic image-space optimization would pose highly impractical computational demands.

However, in this section, a simple kernel model is described that allows a manual image-space optimization. This model was developed by Jorge Jimenez and is described for the sake of completeness. The main goal of this model is to provide the user (or artist) with a small number of intuitive parameters that are manually controlled in order to find a separable approximation kernel which approximates the ground truth as close as possible, or in any other way intended by the artist.

The artist-friendly kernel approximation model A_m, described in Equation 5.21, has only three parameters, where a_m is based on a mixture of two 1D Gaussians.

$$\begin{gathered} A_m(x,y) = a_m(x)\; a_m(y) \\ \text{with} \\ a_m(x) = w\; G(x,\tau_n) + (1-w)\; G(x,\tau_f) \end{gathered} \tag{5.21}$$

The rank-1 approximation is represented by the separable kernel A_m, where the a_m term is basically a 1D Gaussian mixture including two components which are linearly interpolated via the w parameter. By adjustment of the variances of the two Gaussians and their interpolation w, it is possible to approximate the near and far scattering component of the diffuse reflectance profile via τ_n and τ_f, respectively.

This is further illustrated in Figure 5.16, which shows a simple 1D illustration explaining the Gaussian mixture a_m which is represented by the green function. By choosing appropriate variances τ_n and τ_f for the two Gaussians, represented by the orange and blue functions, respectively, it is possible to approximate the near and far scattering regions of the grey ground-truth profile function. Please note that this figure does not show an optimal approximation and also represents a simplified 1D illustration. The manual approximation kernel with its form $A_m(x,y) = a_m(x)\, a_m(y)$ is slightly more complex, but, nevertheless, the conceptual relation of the parameters τ_n and τ_f to near and far scattering, respectively, is still appropriate.

The ability to adjust the interpolation between the far and near scattering Gaussian via the w parameter is further illustrated in Figure 5.17. By adjustment of the w parameter it is possible to seamlessly transition from predominant near scattering which preserves sharper features, to emphasised far scattering which offers a more pronounced falloff.

In order to find appropriate parameters, the user (or artist) can simply tweak the parameters by hand and visually verify the resulting approximation. This can either be done at runtime, given that the rendering framework provides the necessary parameter interface, or by use of an external offline framework, in which case the found parameter or kernel is simply imported at runtime. An example for such an offline framework was developed in MATLAB and can be seen in Figure 5.18.

The manual approximation kernel A_m can be applied for convolution like every other separable kernel, accordion to Equation 5.1 and 5.2. Although this model does not explicitly include

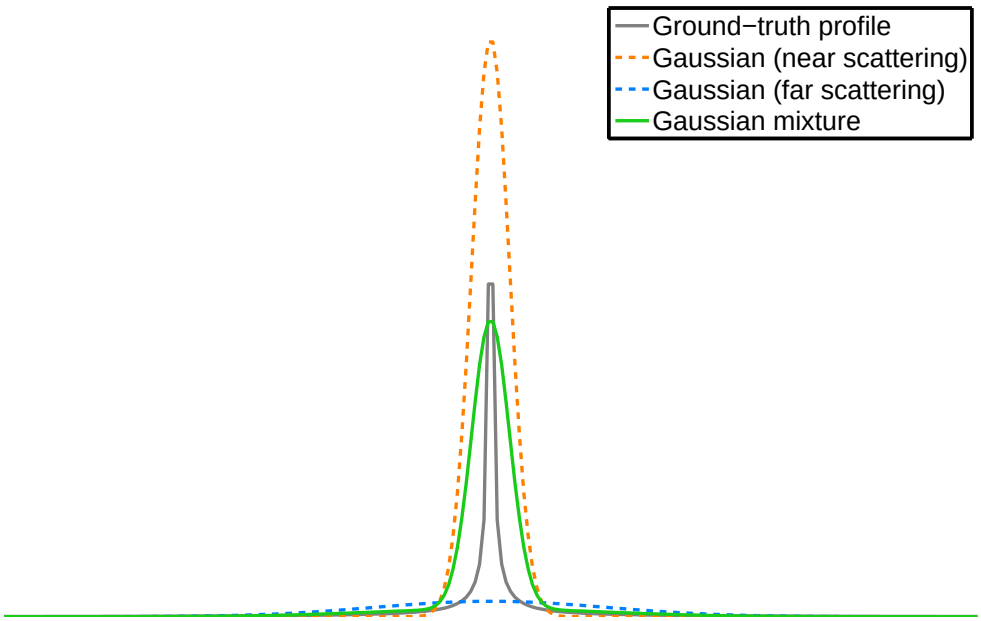


Figure 5.16: This figure illustrates the manual approximation approach based on a_m via a simple 1D example. In this plot the grey function represents an arbitrary example for a ground-truth profile that should be approximated. The green function denotes the approximating Gaussian mixture, composed out of the two weighted Gaussian functions for near (orange) and far (blue) scattering. Please note that this figure does not show an optimal approximation and is for illustration purposes only.

a constraint or term which ensures energy conservation, the resulting kernel can be considered as such, since the parameters are manually optimized in image space, which ensures that the resulting kernel has the desired shape after enforced energy conservation via normalization.

This artist-friendly model was originally developed in the context of real-time rendering for games by Jorge Jimenez and was reimplemented and tested during research for the SSSS extension presented in this thesis. The model is, due to its low parameter count, fairly simple and provides an intuitive interpretation of the two Gaussians as the near and far scattering components, which makes adjustments by hand quite intuitive. Furthermore, this model does not constrain artists to physical correctness, but provides a certain degree of artistic freedom in the choice for the parameter values. Depending on the skill of the user or artist, the search for the particular parameters that produce the desired approximation can take up to several minutes, but can be supported by instant feedback via evaluation and application of the kernel model at runtime, as mentioned earlier.

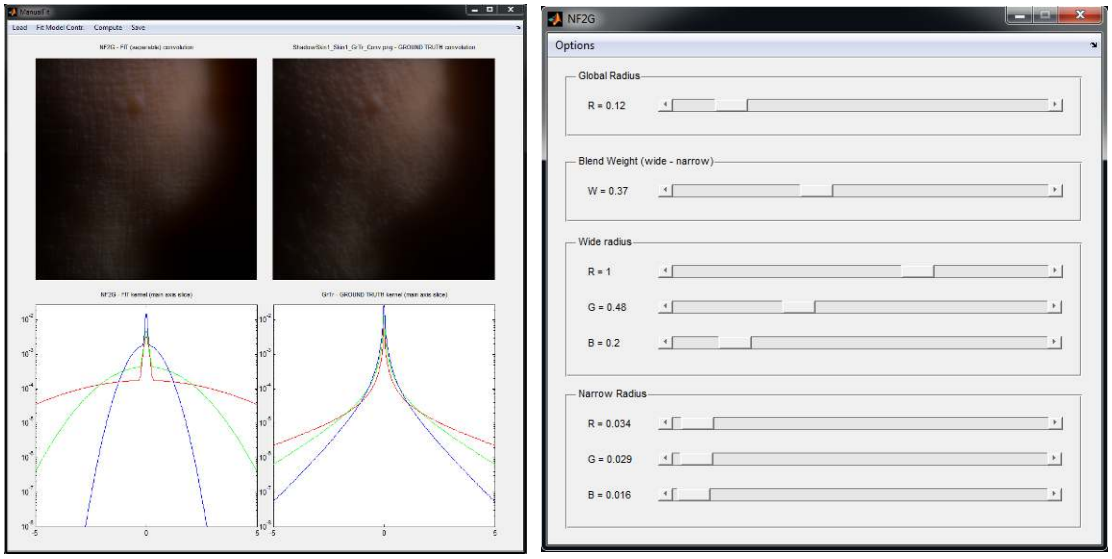
One drawback of this model is that the approximation quality is limited, as it only uses two Gaussians. However, as shown by the final results included in the next chapter, the approximation quality may be sufficient for most rendering scenarios.

(a) Input (b) Ground-truth

(c) Predominant near scattering (d) Balanced near–far scattering (e) Predominant far scattering

Figure 5.17: This figure illustrates the linear interpolation of the two Gaussians of the manual approximation model via the variation of the w parameter. For reference, image (a) shows a patch of skin and (b) displays the filtered result using the ground-truth Skin1 kernel. The application of the manual approximation kernel with varying w parameter is shown in the second row. Here, different w settings are illustrated, showing emphasised near scattering (c), balanced near–far scattering (d), and emphasised far scattering (e).

The models for the approximation of diffuse reflectance profiles, described in the previous sections, all have different advantages and drawbacks. Therefore, it depends highly on the specific application which kernel might be suited best for a specific rendering scenario. In order to compare and verify the models for different materials and various scenes, a series of rendering test were performed. Detailed test results and images are provided in the following chapter.



(a) Main window (b) Parameter controls

Figure 5.18: This figure shows two screenshots of the manual approximation model in MATLAB. The main window of the UI is shown in (a), which displays axis-aligned kernel plots and convolution results for the ground-truth kernel and the manual approximation kernel, for easy comparison and adjustment. The actual model parameter controls can be changed via a separate window, shown in (b).

CHAPTER 6

Results

This chapter includes visualisations, renderings and comparisons of the different approximation kernels, based on the models described in the previous chapter. First, initial preparations which were necessary for the development and testing of the extensions are discussed. Then, plots of the simulated ground-truth 1D diffuse reflectance profiles, computed via MCML simulation, are presented. And finally, the application of the generated approximation kernels for various test signals and scenes is shown using several rendered images, and the results are, furthermore, compared and discussed.

Given the vast amount of material-kernel combinations, this chapter only includes the results for the five materials, i.e., Apple, Marble, Ketchup, Skin1 and Wholemilk, which were used for the final scene rendering. Furthermore, the manual approximation kernel was only constructed for these main materials. Results for the remaining materials are included in the appendix.

6.1 Initial Preparations

In order to extend the SSSS algorithm to support arbitrary materials, a few initial preparations were necessary. First, physically based material measurements had to be obtained and used to derive ground-truth diffuse reflectance profiles for different materials. This was achieved by using the measured material parameters from Jensen et al. [21] as an input for brute-force simulation using MCML [39]. It supports simulation of multi-layered cylindrically symmetric tissue models, and outputs the diffuse reflectance profile as a 1D function, $R_d(r)$.

MCML Setup: All material parameters, taken from Jensen et al. [21], were converted to centimeter or cm^{-1}, respectively, and used to build the MCML input files. For each material, the RGB channels were simulated separately, and aside from the material properties (see Table 6.2), each channel used the same parameters specified in Table 6.1. The mean free path (MFP) was computed as $\frac{1}{\sigma_t'} = \frac{1}{\sigma_a + \sigma_s'}$, with the reduced scattering coefficient being trivial, i.e., $\sigma_s' = \sigma_s$, since only isotropic scattering ($g = 0$) was considered.

No. Photons:	10^7
Grid spacing:	$d_r = \frac{\min(MFP_{rgb})}{20}$
No. of grid elements:	$n_r = \lceil \frac{32 \ \max(MFP_{rgb})}{d_r} \rceil$
Thickness:	10^8 cm (quasi-infinite).

Table 6.1: MCML parameters used for all materials and each RGB channel. In this table MFP_{rgb} denotes the mean free path of the three RGB channels.

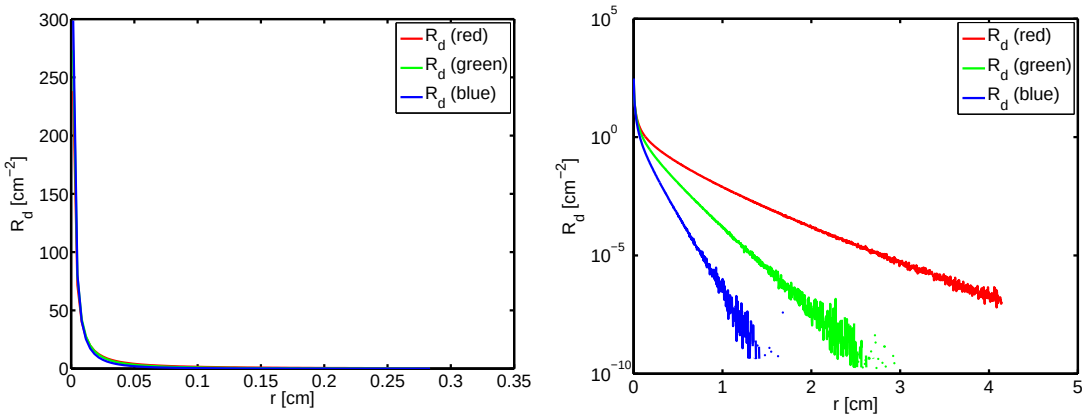
An example of such a simulated raw ground-truth diffuse reflectance profile can be seen in Figure 6.1, while plots for all materials are included in the appendix (Figure A.1–A.11). The left plots show a close view of the region near zero on a linear scale, while the right plots show the complete simulation interval on a logarithmic scale. Some plots may exhibit noise in the falloff regions, which can be explained as follows.

Diffuse reflectance profiles simulated using Monte-Carlo simulation tend to become more and more noisy with increasing radius. As light travels further through a material, its absorption becomes increasingly likely, which can lead to a low number of samples in regions far from the center, and subsequently to noise. The noise level usually depends on different parameters, such as simulation interval and photon count as well as the material properties. For the included results, the noise is insignificant, because it includes only very small values.

Based on this simulated 1D ground-truth diffuse reflectance profiles $R_d(r)$, the 2D diffuse reflectance profiles $R_d(x, y)$, from which the kernel approximations are derived, were computed by 'simple rotation' of $R_d(r)$ about the Y axis. A corresponding example plot can be seen in Figure 6.2 where the ground-truth profile is indicated by the dotted line, for easy comparison.

Name	σ_a (cm^{-1}) [RGB]	σ_s (cm^{-1}) [RGB]	g	η	Thickness (cm)
Apple	[0.03, 0.034, 0.46]	[22.9, 23.9, 19.7]	0	1.3	10^8
Chicken1	[0.15, 0.77, 1.9]	[1.5, 2.1, 3.8]	0	1.3	10^8
Chicken2	[0.18, 0.88, 2]	[1.9, 2.5, 3.2]	0	1.3	10^8
Cream	[0.002, 0.028, 0.163]	[73.8, 54.7, 31.5]	0	1.3	10^8
Ketchup	[0.61, 9.7, 14.5]	[1.8, 0.7, 0.3]	0	1.3	10^8
Marble	[0.021, 0.041, 0.071]	[21.9, 26.2, 30]	0	1.5	10^8
Potato	[0.024, 0.09, 1.2]	[6.8, 7, 5.5]	0	1.3	10^8
Skimmilk	[0.014, 0.025, 0.142]	[7, 12.2, 19]	0	1.3	10^8
Skin1	[0.32, 1.7, 4.8]	[7.4, 8.8, 10.1]	0	1.3	10^8
Skin2	[0.13, 0.7, 1.45]	[10.9, 15.9, 17.9]	0	1.3	10^8
Wholemilk	[0.011, 0.024, 0.14]	[25.5, 32.1, 37.7]	0	1.3	10^8

Table 6.2: Material parameters used for the MCML simulation. In this table, g denotes the anisotropy and η represents the index of refraction.



(a) Linear

(b) Logarithmic

Figure 6.1: An example plots of the simulated raw 1D diffuse reflectance profile for the material Skin1. The left plot (a) shows a subregion near zero on a linear scale, while the right plot (b) show the complete simulation interval on a logarithmic scale. The corresponding derived 2D diffuse reflectance profile can be seen in Figure 6.2.

Aside from ground-truth diffuse reflectance profiles, it was also necessary to prepare additional scenes for rendering tests. For this purpose, various object meshes were obtained from online sources like Blend Swap [32]. Blender [1] and MeshLab [4] were used to further prepare and combine the object meshes for later use in the SSSS rendering framework, which was extended to support multiple scenes and texture sets. The scenes include the original model of a human head, a marble dragon statue, whole milk, fruits on a plate, a plant and ketchup on a plate. For a list of authors and sources of the used models please see Table A.2 in the appendix. Final images showing the scenes rendered with SSS using the corresponding materials are included in Section 6.3.

6.2 Approximation Kernels

Based on the 2D ground-truth kernels, derived from the simulated radially symmetric 1D diffuse reflectance profiles, various approximations were computed using the models introduced in the previous chapter. An example plot of an approximation kernel is shown in Figure 6.2, while additional plots of the kernels and materials used for the examples included in this and the previous chapter can be found in the appendix (Figure A.12–A.64). Please note that all ground-truth kernels were normalized prior to approximation. This means that the discrete values of each kernel sum up to 1 (per channel). For comparison with the SVD-based approach, additional kernels using the Gaussian mixture model from d'Eon et al. [5, 6] were computed. The model parameters found for the manual approximation kernels are included in the appendix (Table A.1).

For additional visualisation, the different approximation kernels were used to convolve a

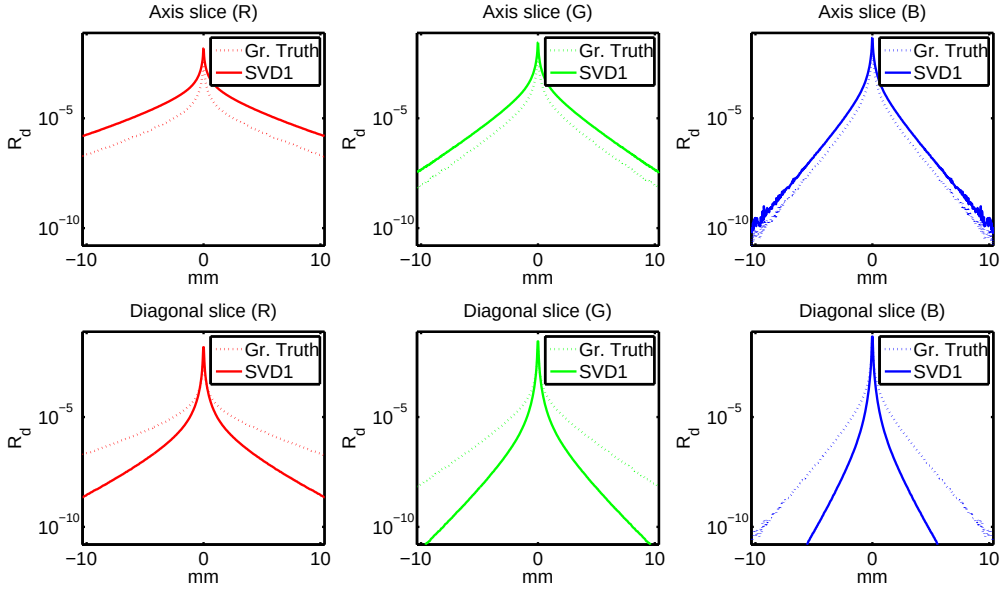


Figure 6.2: Example plot of the SVD rank-1 kernel approximation for material Skin1. The dotted function (Gr. Truth) represents the corresponding ground-truth 2D diffuse reflectance profile. The first row show slices trough the 2D kernel along the main axis for all three channels (RGB), while the second row shows the corresponding diagonal slices. Please note that this example plot and all additional kernel approximation plots included in the appendix show normalized kernels.

simple vertical white–black edge. A corresponding overview for easy comparison of the main materials can be seen in Figures 6.3–6.5, while the images for the additional materials are included in the appendix (Figure A.65–A.67). Additionally, the four rank-1 approximation kernels were further used to convolve a more complex test signal, which includes a circle and two tilted and adjacent squares. The corresponding images can be seen in Figure 6.6–6.10, and the additional material examples are included in the appendix (Figure A.68–A.73). For the guided approximation, the kernel corresponding to $k = 1$ was chosen as a representative example, since this represents a trade-off between center ($k = 0$) or falloff approximation ($k = 2$), as outlined in Section 5.3.

These simple artificial test signals make it possible to easily compare the various kernels visually, and quickly estimate their approximation quality and examine their individual properties. Please note that these convolutions were performed using kernels with importance sampling and 200 samples along each dimension. This high sample count is impractical for real-time frame rates but was chosen to allow close examination of the different kernel approximations without distortions due to possible sampling artifacts. The final rendering examples, included in the next section, however, represent actual real-time examples for which a low sample count of 17 was used.

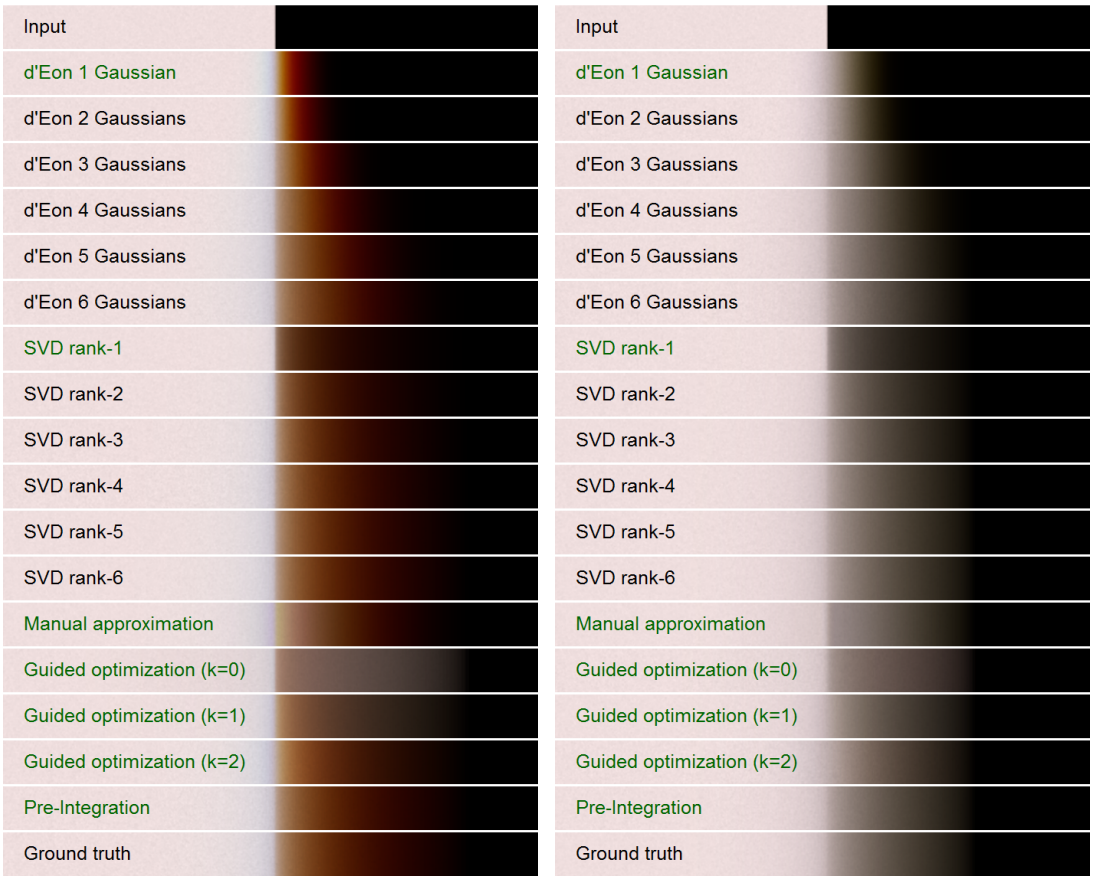
Although kernel-space RMS errors are not suited to assess the final approximation quality, as illustrated via the rank-1 SVD-based kernel example, the errors of the four rank-1 approximation

kernels for the materials used in the final rendering examples, are included in Table 6.3, for the purpose of completeness. This table can be used to illustrate that the (not normalized) rank-1 SVD-based approach has the lowest RMS error in kernel space in all cases.

6.3 Test Renderings

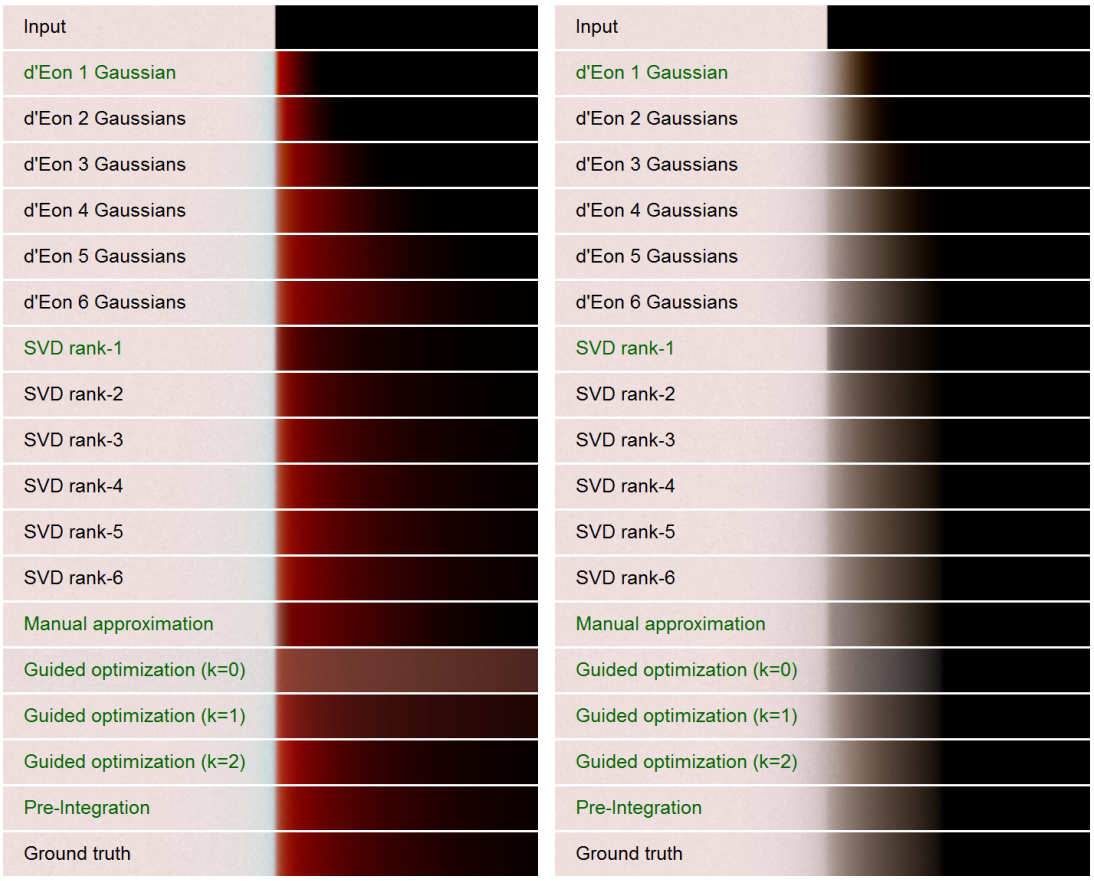
In order to test the different kernels for practical real-time rendering scenarios, a series of scenes using corresponding materials were rendered. The final renderings are shown in Figure 6.11 to 6.16, which include smaller cut-outs showing different kernel models for comparison. Difference images and corresponding image-space RMS errors are shown in Figure 6.17–6.22 and Table 6.4, respectively.

Please note that these renderings are merely a proof of concept, since some objects and the material kernel used for rendering do not match, in specific cases. Therefore, the Fruits and the Plant scene used the same single Apple material kernel, and the soap was rendered using the Wholemilk material kernel. In order to ensure real-time frame rates, these renderings used kernels with 17 samples (importance-sampling), as outlined in Section 12. Image-space RMS errors, for the corresponding cutouts, are included in Table 6.4.



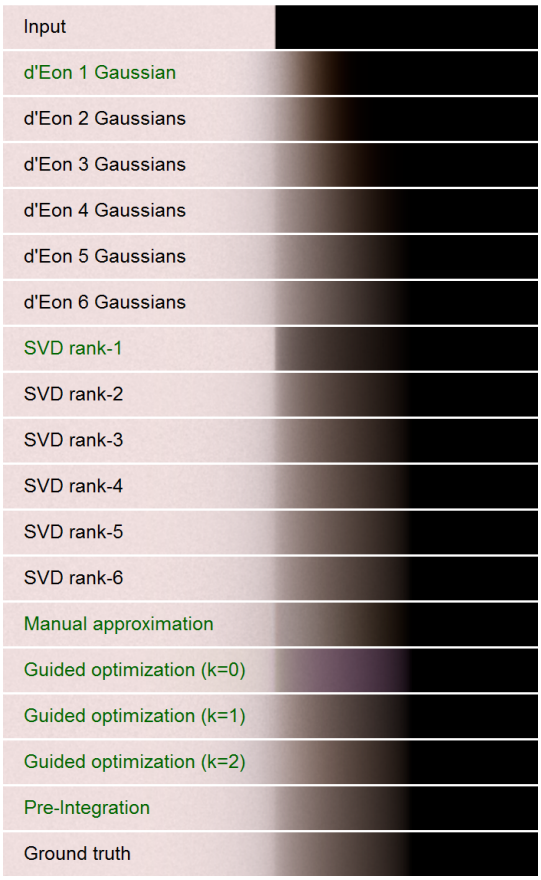
(a) Skin1 (b) Apple

Figure 6.3: Overview of the simple 1D test signal convolutions using different kernels for material Skin1 and Apple.



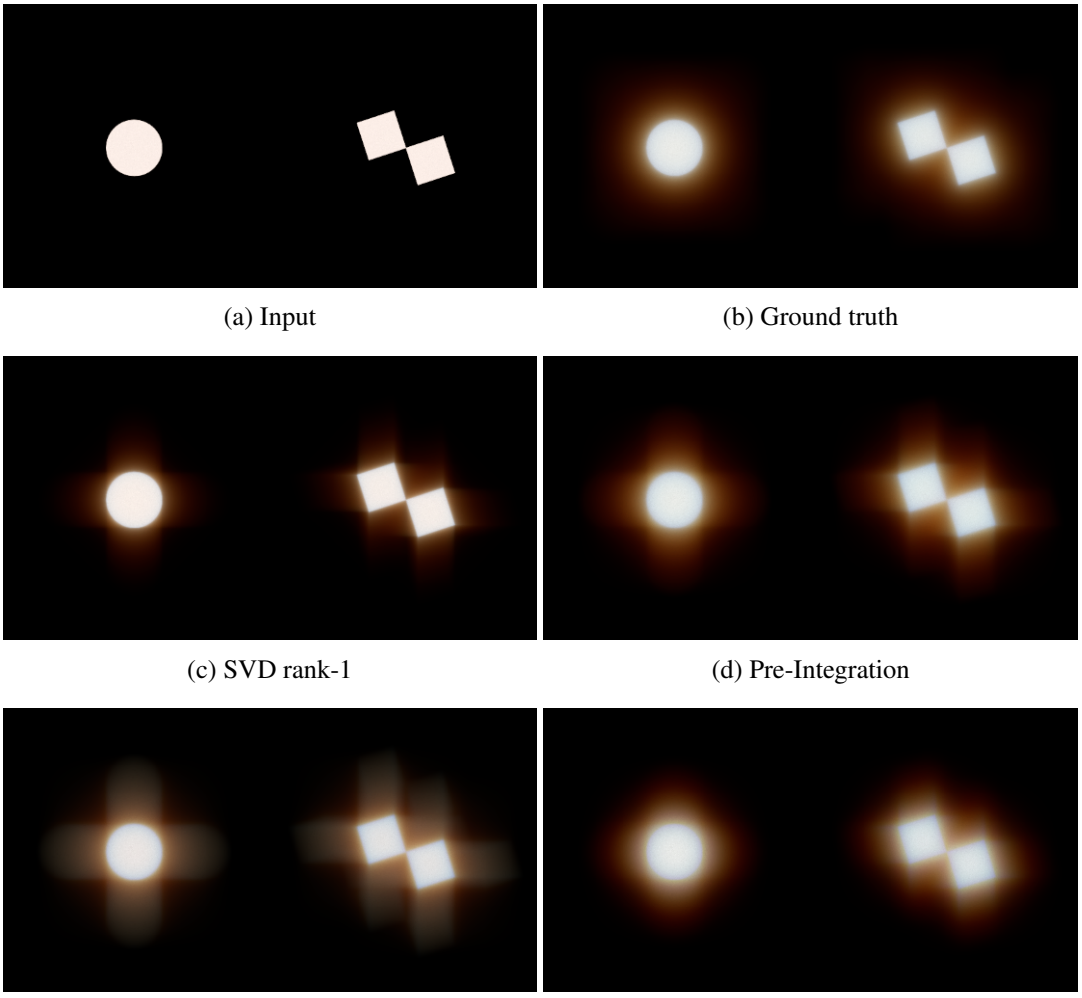
(a) Ketchup (b) Wholemilk

Figure 6.4: Overview of the simple 1D test signal convolutions using different kernels for material Ketchup and Wholemilk.



(a) Marble

Figure 6.5: Overview of the simple 1D test signal convolutions using different kernels for material Marble.

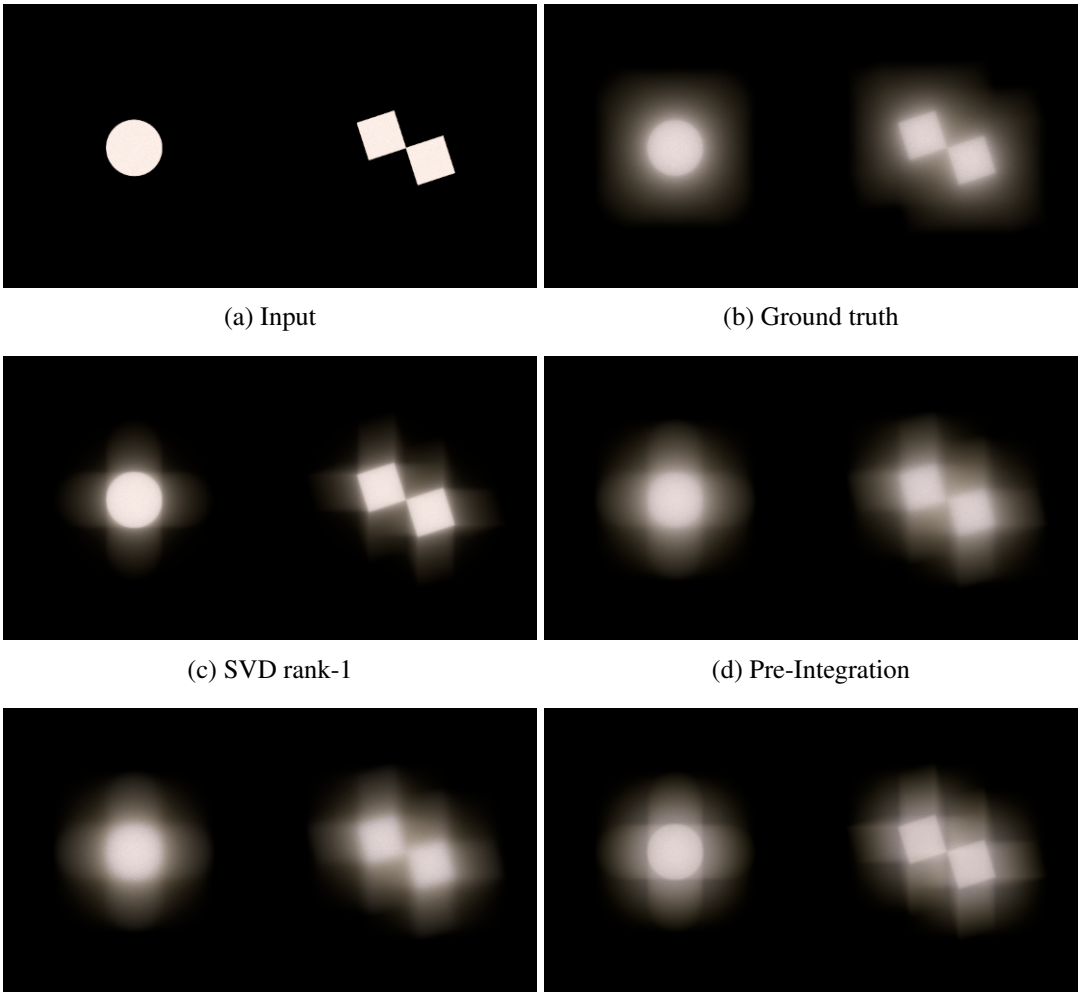


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1) (f) Manual approximation

Figure 6.6: Convolution of an artificial test signal using different kernels for material Skin1.

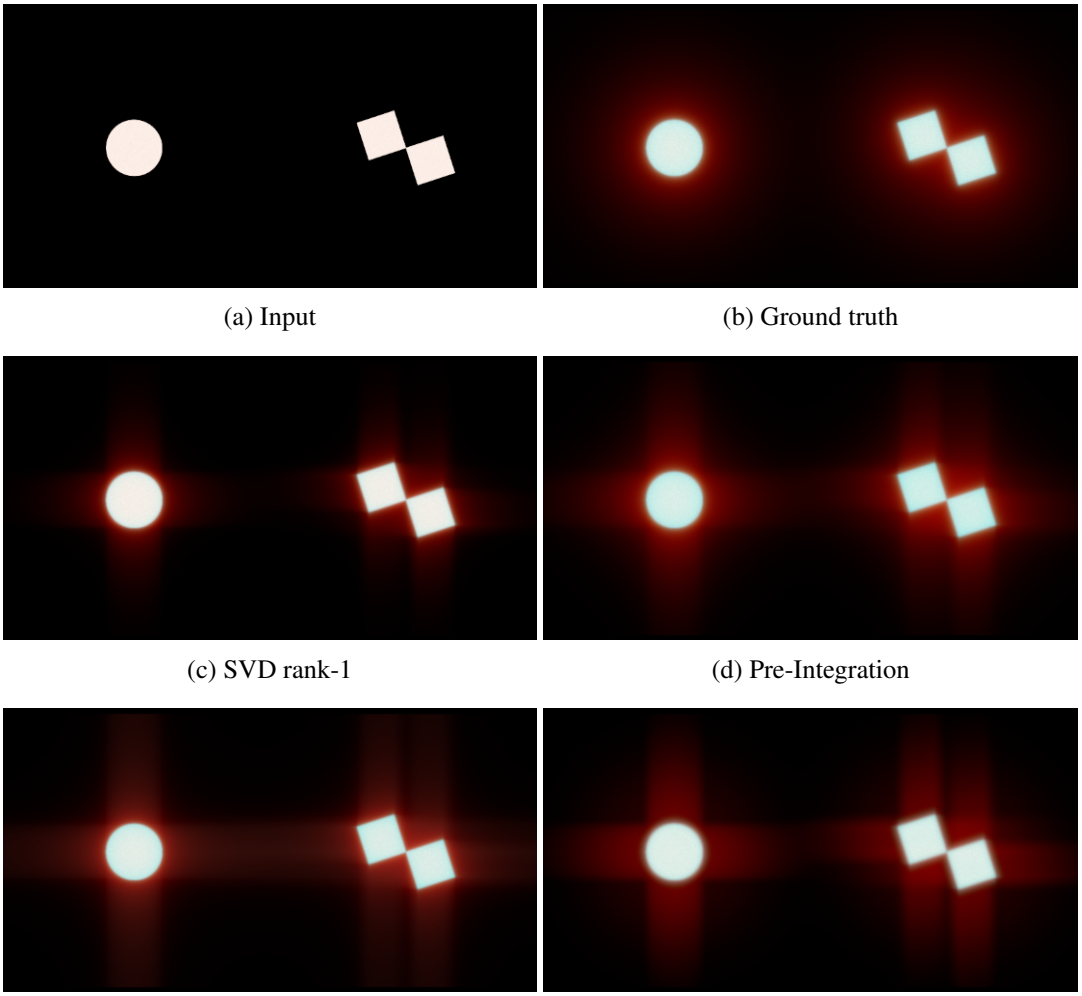


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1) (f) Manual approximation

Figure 6.7: Convolution of an artificial test signal using different kernels for material Apple.

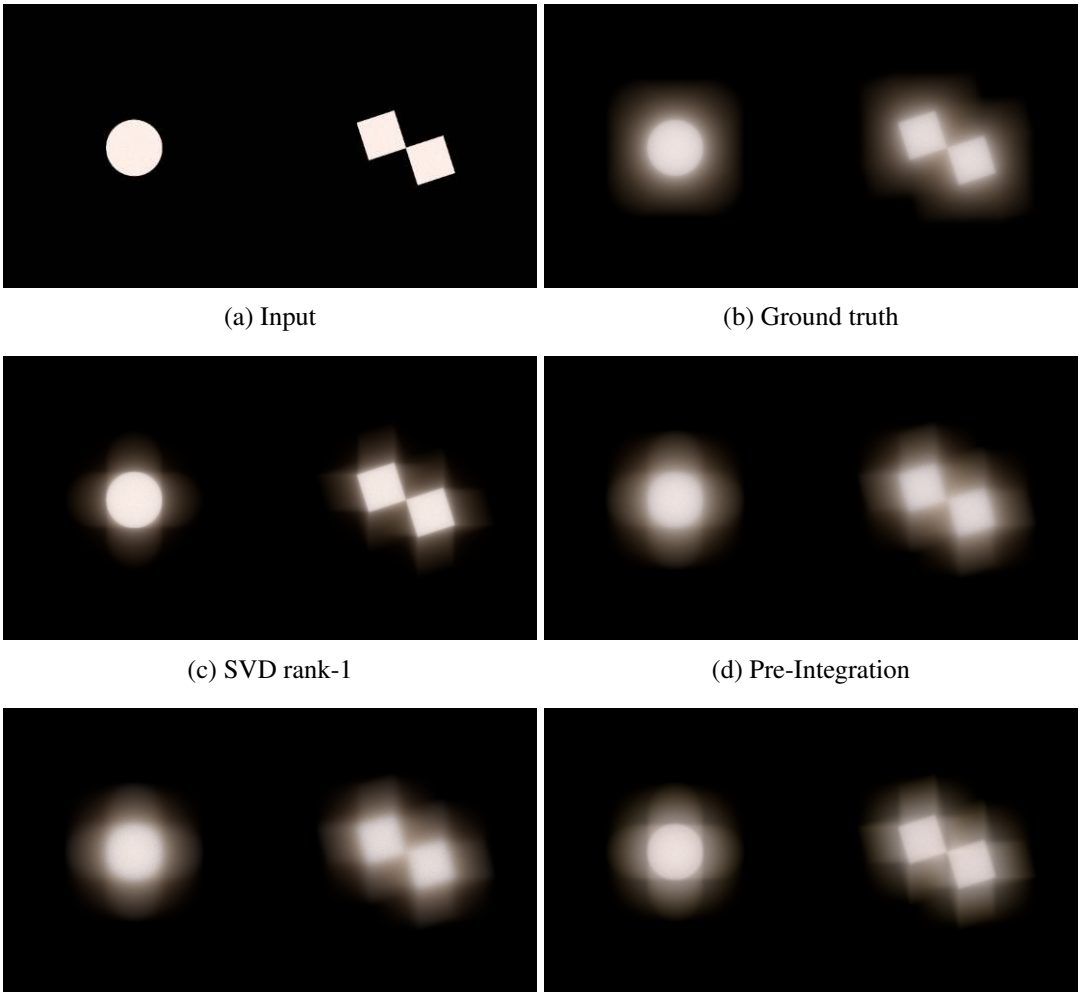


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1) (f) Manual approximation

Figure 6.8: Convolution of an artificial test signal using different kernels for material Ketchup.

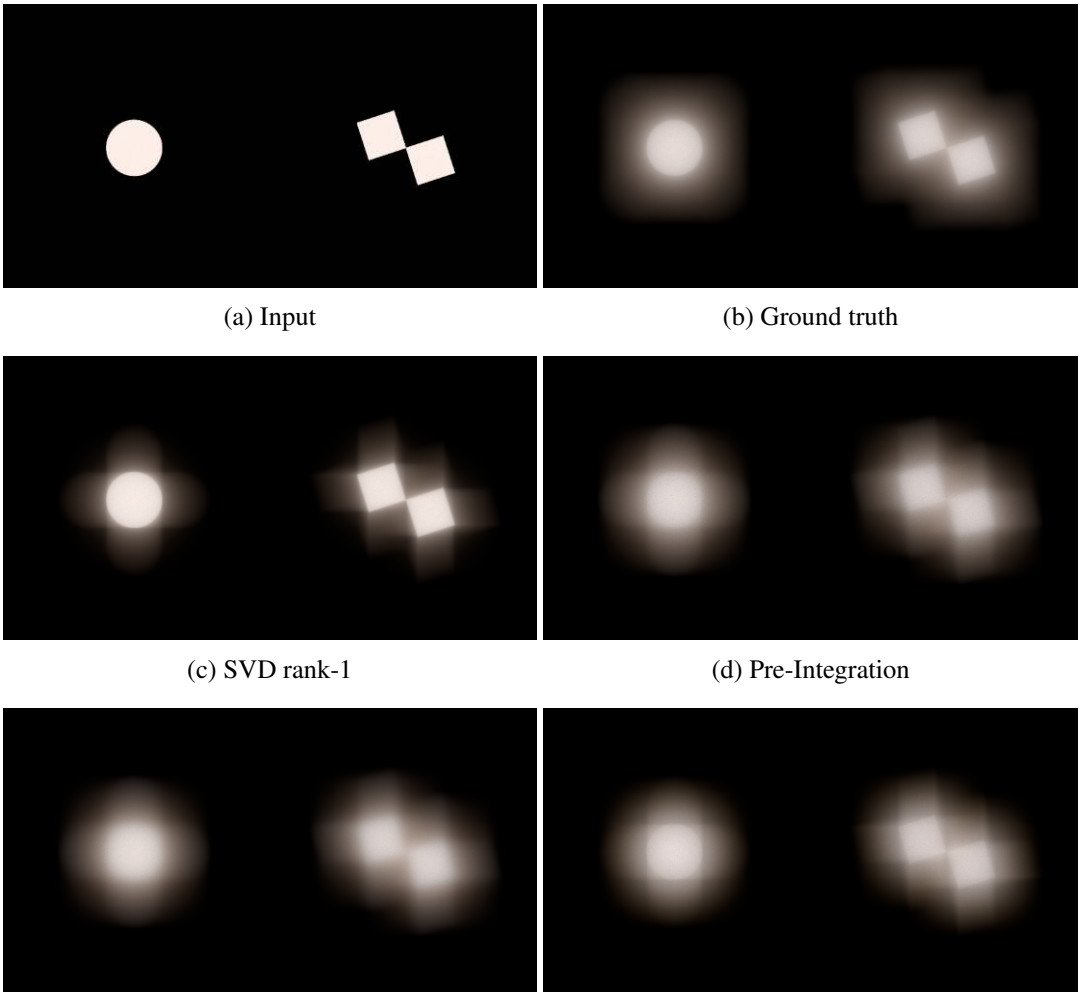


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1) (f) Manual approximation

Figure 6.9: Convolution of an artificial test signal using different kernels for material Wholemilk.



(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1) (f) Manual approximation

Figure 6.10: Convolution of an artificial test signal using different kernels for material Marble.

	SVD rank-1				Pre-Integration			
	R	G	B	Σ_{RGB}	R	G	B	Σ_{RGB}
Apple	0.057	0.058	0.066	0.181	0.120	0.122	0.142	0.384
Marble	0.038	0.041	0.045	0.124	0.079	0.088	0.096	0.263
Ketchup	0.061	0.161	0.157	0.379	0.139	0.370	0.354	0.863
Skin1	0.035	0.062	0.095	0.192	0.074	0.142	0.226	0.442
Wholemilk	0.036	0.040	0.046	0.123	0.074	0.084	0.099	0.258
	Guided optimization (k=1)				Manual approximation			
	R	G	B	Σ_{RGB}	R	G	B	Σ_{RGB}
Apple	0.137	0.141	0.162	0.440	0.106	0.107	0.118	0.330
Marble	0.090	0.099	0.110	0.299	0.077	0.074	0.075	0.226
Ketchup	0.141	0.167	0.160	0.469	0.151	0.683	1.000	1.833
Skin1	0.084	0.166	0.230	0.479	0.093	0.202	0.261	0.557
Wholemilk	0.084	0.097	0.115	0.296	0.070	0.081	0.091	0.243

Table 6.3: This table shows the kernel-space RMS errors for the materials used for final rendering. Please note that the RGB errors are normalized, which means that the highest per-channel error is mapped to one.

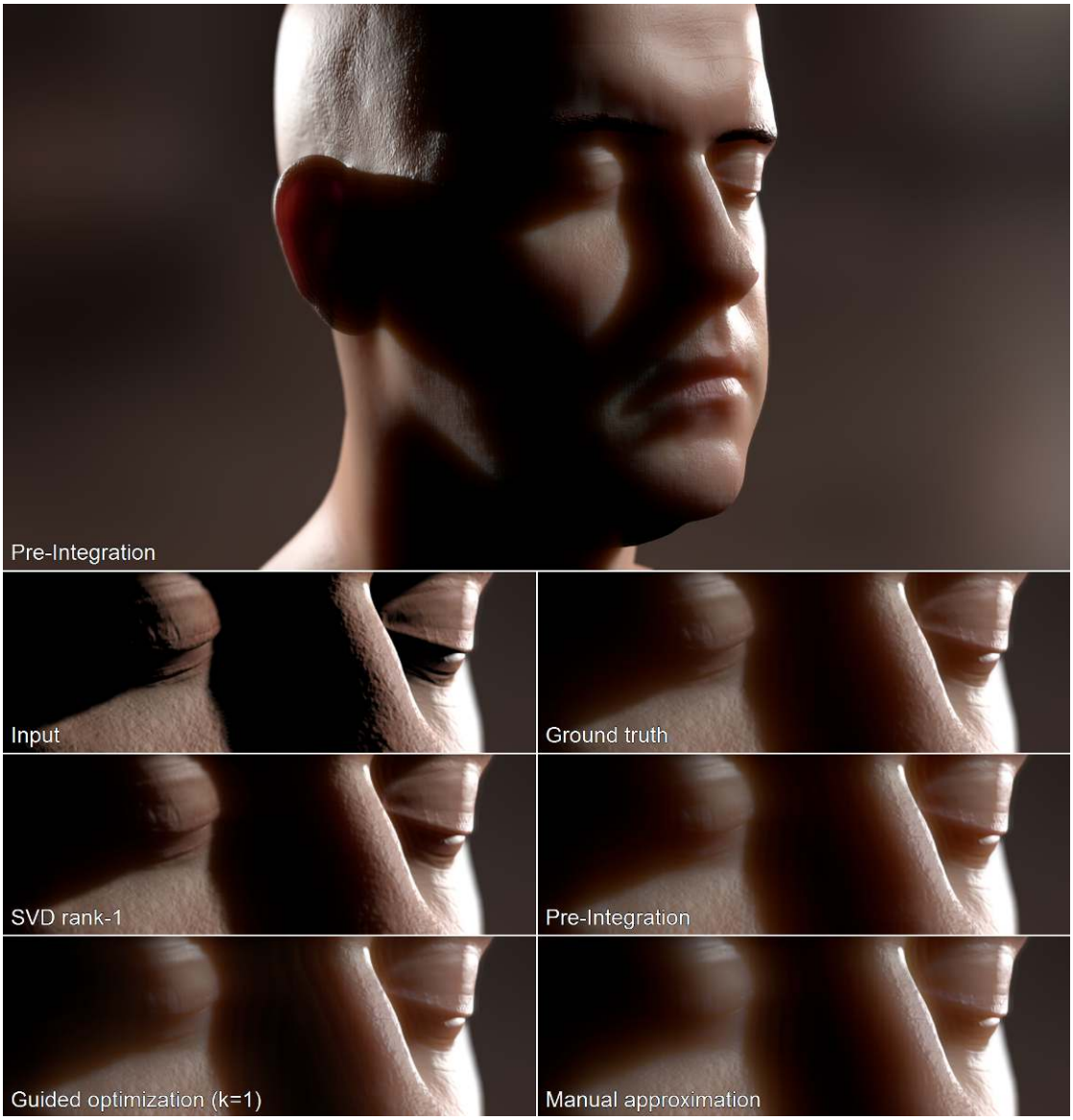


Figure 6.11: Renderings of the Head scene using different kernels. The cut-outs below show that the pre-integration based kernel is quite close to the ground truth, while the guided optimization and the manual approximation have a slightly different falloff in the shadow region.

Figure 6.12: Renderings of the Dragon scene using different kernels. The comparison in the cut-outs shows that most approximations blur the surface details of marble more than the ground truth. The guided optimization kernel shows the strongest blur, in comparison to the pre-integration and manual approximation approximations.

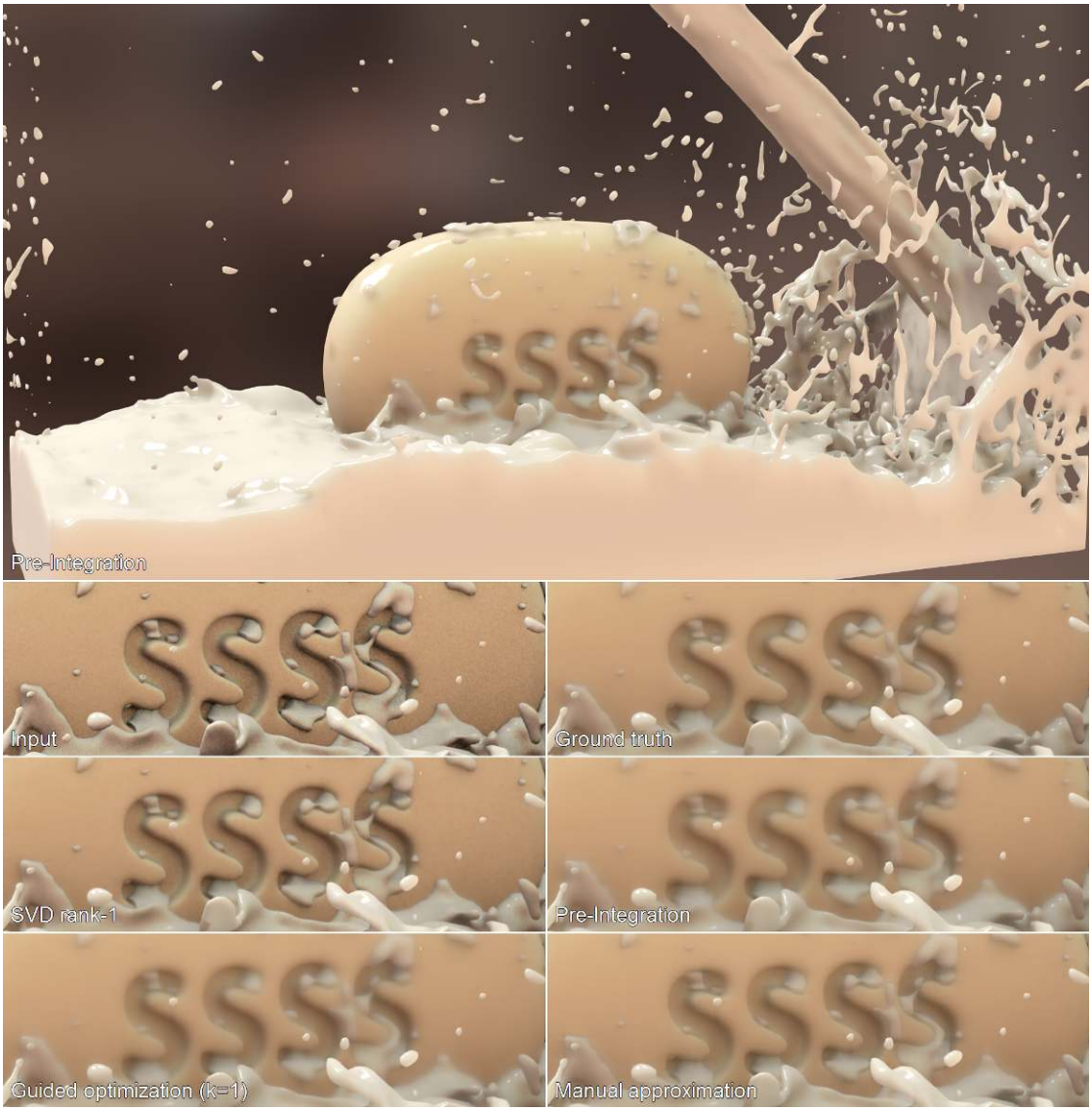


Figure 6.13: Renderings of the Milk scene using different kernels. In this particular case the kernel differences are best observed on the engraved SSSS letters.

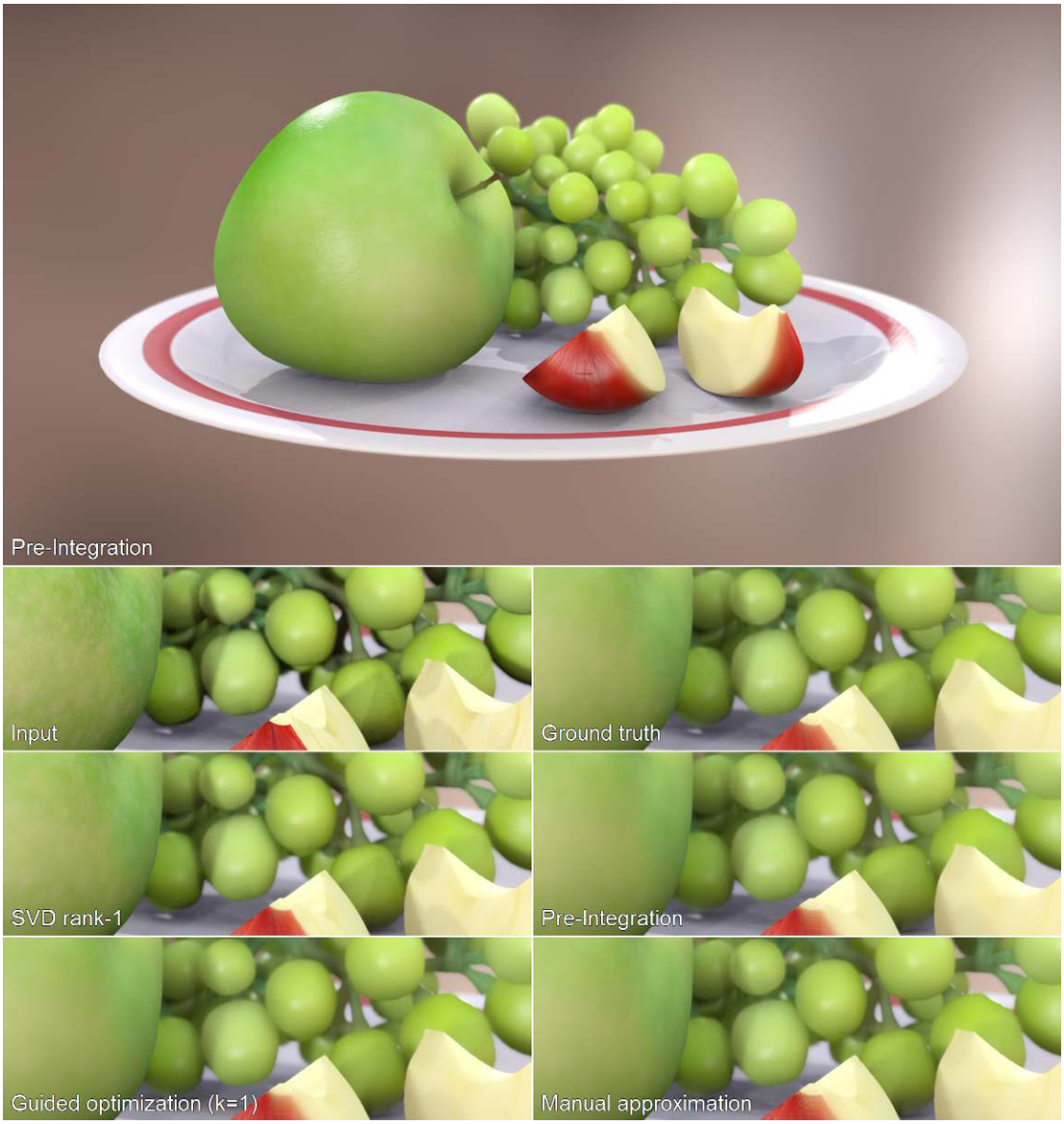


Figure 6.14: Renderings of the Fruit scene using different kernels. In this proof-of-concept scene the illumination is quite uniform and the surface details are less pronounced. This makes the differences between the individual kernels quite subtle and harder to observe. Therefore, the kernels look quite similar with some differences visible in falloff regions.

Figure 6.15: Renderings of the Plant scene using different kernels. In this particular scene larger surface areas are missing and the kernels may be best observed on individual plant leafs shown in the cut-outs.

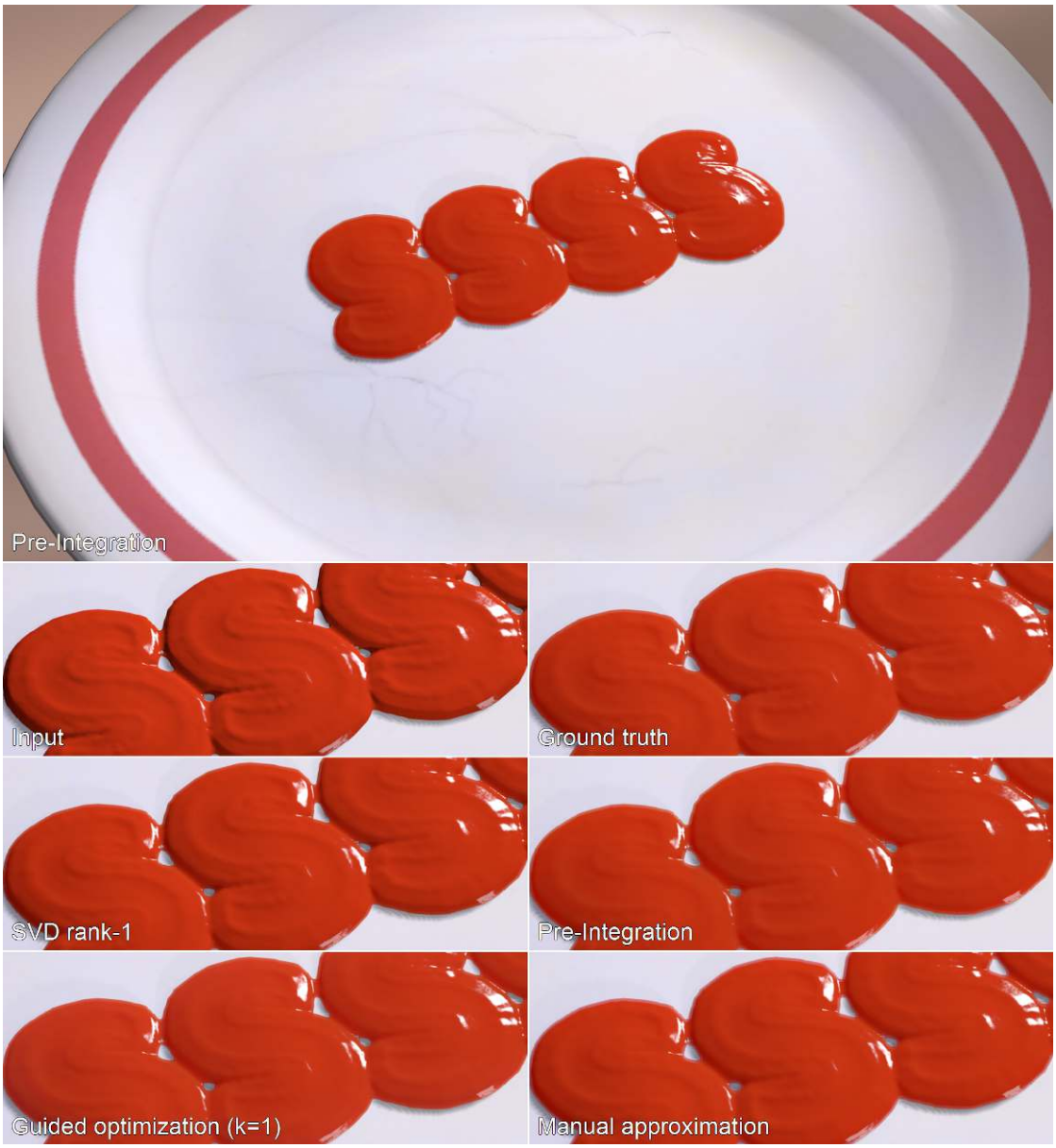
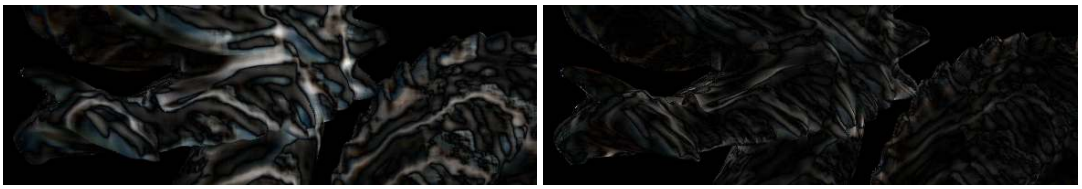
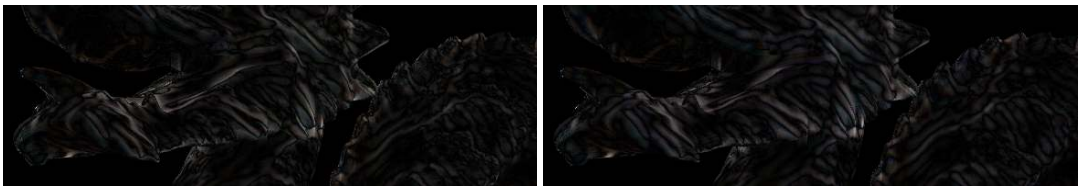


Figure 6.16: Renderings of the Ketchup scene using different kernels. This scene includes uniform illumination and very few surface details, which makes differences between the individual kernels harder to observe. Therefore, the kernel approximations look quite similar and close to the ground truth.



(a) SVD rank-1 (b) Pre-Integration



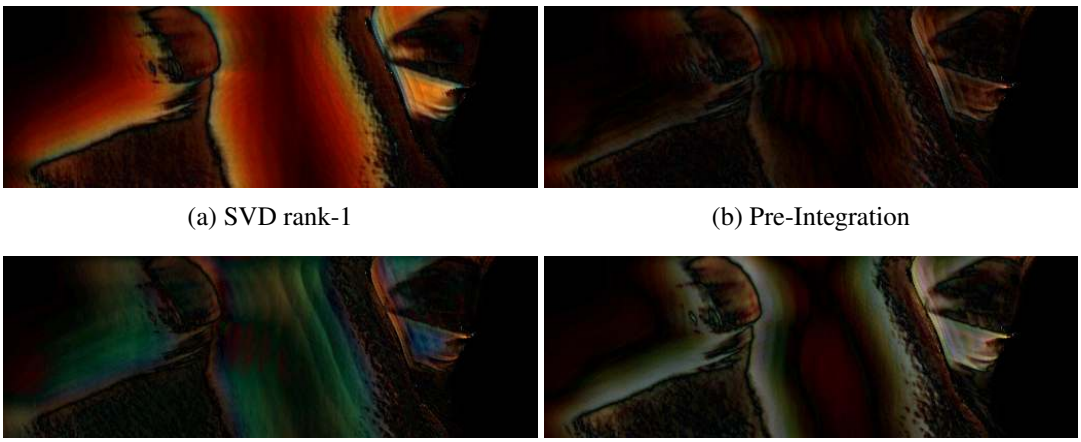
(c) Guided optimization (k=1) (d) Manual approximation

Figure 6.17: Difference images for the different approximation kernels used for the Dragon scene cutout renderings.

(a) SVD rank-1 (b) Pre-Integration

(c) Guided optimization (k=1) (d) Manual approximation

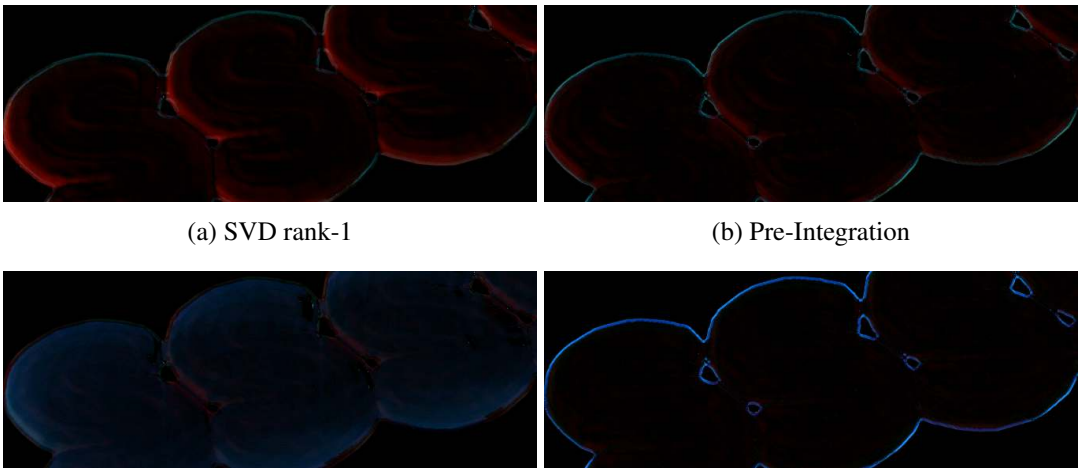
Figure 6.18: Difference images for the different approximation kernels used for the Fruit scene cutout renderings.



(a) SVD rank-1 (b) Pre-Integration

(c) Guided optimization (k=1) (d) Manual approximation

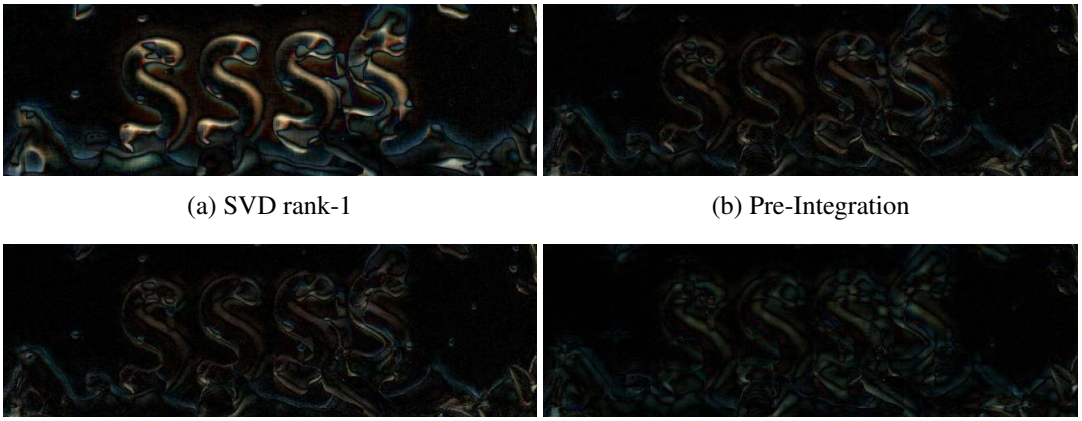
Figure 6.19: Difference images for the different approximation kernels used for the Head scene cutout renderings.



(a) SVD rank-1 (b) Pre-Integration

(c) Guided optimization (k=1) (d) Manual approximation

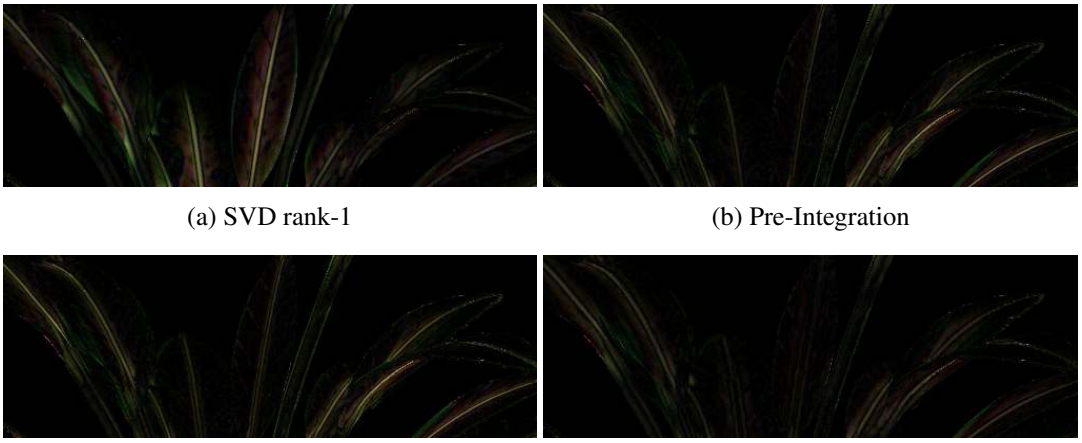
Figure 6.20: Difference images for the different approximation kernels used for the Ketchup scene cutout renderings.



(a) SVD rank-1 (b) Pre-Integration

(c) Guided optimization (k=1) (d) Manual approximation

Figure 6.21: Difference images for the different approximation kernels used for the Milk scene cutout renderings.



(a) SVD rank-1 (b) Pre-Integration

(c) Guided optimization (k=1) (d) Manual approximation

Figure 6.22: Difference images for the different approximation kernels used for the Plant scene cutout renderings.

	SVD rank-1				Pre-Integration			
	R	G	B	Σ_{RGB}	R	G	B	Σ_{RGB}
Dragon	0.676	0.684	0.677	2.037	0.318	0.314	0.304	0.937
Fruit	0.257	0.339	0.247	0.843	0.148	0.198	0.140	0.486
Head	1.000	0.455	0.198	1.653	0.320	0.177	0.115	0.611
Ketchup	0.425	0.138	0.140	0.702	0.246	0.193	0.206	0.644
Milk	0.461	0.448	0.384	1.292	0.208	0.211	0.186	0.605
Plant	0.245	0.227	0.147	0.618	0.193	0.184	0.115	0.492
	Guided optimization (k=1)				Manual approximation			
	R	G	B	Σ_{RGB}	R	G	B	Σ_{RGB}
Dragon	0.332	0.315	0.308	0.955	0.325	0.307	0.323	0.955
Fruit	0.141	0.187	0.128	0.456	0.133	0.184	0.133	0.450
Head	0.298	0.358	0.252	0.908	0.524	0.473	0.354	1.351
Ketchup	0.155	0.267	0.494	0.915	0.124	0.257	0.510	0.891
Milk	0.210	0.211	0.192	0.613	0.139	0.179	0.159	0.478
Plant	0.203	0.193	0.111	0.506	0.147	0.134	0.094	0.376

Table 6.4: This table shows the image-space RMS errors of the different kernel approximations used for the rendering cutouts included in Figure 6.11–6.16. Please note that the RGB errors are normalized, which means that the highest per-channel error is mapped to one. The corresponding difference images can be seen in Figure 6.17–6.22.

6.4 Discussion

In this section, the different kernel approximation models are compared and discussed based on the previously presented results. The simple test-signal convolutions shown in Figure 6.3–6.10 enable a quick visual overview for the different approximation kernels, and the more advanced renderings (Figure 6.11–6.16) are accompanied by difference images (Figure 6.17–6.16) and corresponding image-space RMS errors in Table 6.4.

SVD Model: The SVD-based approach offers a deterministic solution, where the approximation quality directly corresponds to the chosen rank for the approximation kernel. While this model is, in principle, optimal in terms of the kernel-space RMS error, the lack of energy conservation, especially at low ranks, destroys this property, as the SSSS algorithm normalized all kernels prior to convolution.

The artificial test-signal convolutions in Figure 6.3–6.10 indicate that SVD-based rank-1 approximations are of low quality for all materials. The corresponding kernels poorly approximate the falloff of the ground truth and therefore underestimate the far-range scattering, which is a significant feature of SSS. This is, furthermore, supported by the test renderings (Figure 6.11 to 6.16) and the corresponding difference images (Figure 6.17–6.22), as well as by the image-space RMS errors (Table 6.4), for which the SVD-based rank-1 approximations has the highest errors in most cases. Consequently, the SVD-based rank-1 approximation kernels are less suited for SSS rendering based on a separable kernel.

However, it is worth noting that the approximation quality rapidly increases with higher ranks. Figure 6.3–6.5 indicate that the rank 3–5 solutions are already quite close approximations, and that the rank-6 kernel is almost indistinguishable from the ground truth. This is furthermore illustrated by example renderings of the human head shown in Figure 6.23, where this scalable approach of higher rank SVD approximation is compared to a similar state-of-the-art model by d'Eon et al. [5, 6], which also supports scalability in terms of convolution passes. This model uses a Gaussian Mixture, where the number of Gaussian determines the approximation quality and the number of required convolution passes. The image-space RMS error comparison in Table 6.5 indicates that the SVD-based approximation scheme represents a saving of roughly one rank (two 1D convolutions) in comparison to the d'Eon approach of the same quality. The image-space RMS errors, as well as the corresponding difference images shown in Figure 6.24 and 6.25), support that the SVD-based approach, compared to the Gaussian Mixture model, has a favorable quality progression with respect to the required convolution passes and converges more quickly to the ground truth. However, this increase in approximation quality via higher-rank approximations requires more convolution passes for rendering and, therefore, more computation time than a separable (rank-1) approach.

The only separable (rank-1) approximation provided by d'Eon et al. [5,6] is the single Gaussian. The approximation of the ground-truth kernel using only a single Gaussians is expectedly quite difficult and provides low-quality approximations, which is also the case for the rank-1 SVD approach. Therefore, more suitable models for rank-1 approximations are discussed below.

	SVD				d'Eon			
	R	G	B	Σ_{RGB}	R	G	B	Σ_{RGB}
rank-1	0.910	0.391	0.180	1.481	1.000	0.449	0.229	1.679
rank-2	0.518	0.230	0.104	0.852	0.874	0.380	0.152	1.406
rank-3	0.274	0.111	0.048	0.432	0.613	0.244	0.092	0.949
rank-4	0.166	0.070	0.036	0.272	0.386	0.127	0.045	0.557
rank-5	0.089	0.034	0.017	0.141	0.230	0.068	0.032	0.330
rank-6	0.055	0.024	0.010	0.088	0.147	0.040	0.026	0.214

Table 6.5: This table shows image-space RMS errors corresponding to the SVD–d'Eon comparison shown in Figurer 6.23. Please note that the RGB errors are normalized, meaning that the highest error is mapped to 1. The corresponding difference images can be seen in Figure 6.24–6.25.

Pre-Integration Model: The analytic pre-integration based model is, like the SVD-based approach, deterministic, but only provides a single level (rank-1) of approximation quality. It furthermore produces the exact same convolution result as the ground-truth kernel for additively separable irradiance input signals, as shown in Figure 6.3–6.5.

However, the results for more complex irradiance signals (Figure 6.7–6.9) and especially the final renderings (Figure 6.11–6.16) indicate that, even for arbitrary irradiance signals, the quality of the analytic pre-integration kernel is sufficient to approximate SSS in a visually plausible way. The difference images shown in 6.17–6.22 as well as the image-space RMS errors (Table 6.4) show that the approximation quality is quite similar to the guided optimization and the manual approximation models in most cases, while the SVD-based approach is of lower quality.

If the pre-integration approach does not produce satisfactory results, or a certain non-physical appearance is desired, the following two models may be applied to compute a customized rank-1 approximation.

Guided Optimization Model: The guided optimization model enables the computation of separable and energy-conserving approximation kernels via the minimization of a parametrized objective function which is used to guide the optimization, as outlined in Section 5.3. This model is able to produce results similar to the analytic pre-integration model, as illustrated by the rendering examples shown in Figure 6.11–6.16, but provides additional degrees of freedom, as shown in the simple test signal convolutions included in 6.3–6.5.

Using the parametrized objective function, it is possible to guide the optimization to approximate either far or near scattering. It is therefore possible to customize the kernel for the specific application at hand, as illustrated in Section 5.3, by selecting the appropriate value for parameter k. Figure 6.3–6.5 show corresponding kernels optimized for a closer approximation of the near scattering ($k = 0$) and kernels customized for far scattering ($k = 2$). The results indicate that

the parameter k basically controls the approximation-quality trade-off between the center region and the falloff. In case of a close falloff approximation ($k = 2$), the center region is overly blurred, while for a close near-scattering approximation ($k = 0$), the falloff is of poor quality.

For the rendering examples, the kernel corresponding to the parameter $k = 1$ was chosen as it represents the trade-off between falloff and center approximation. Although the renderings shown in Figure 6.11–6.16 suggest that the guided optimization approach looks quite similar to the pre-integration and manual approximation models, the difference images (Figure 6.17–6.22) show the more subtle differences, and Table 6.4 shows slightly increased errors, for most cases.

One drawback of the guided approximation approach is that the optimization of the kernels can be quite difficult and time consuming. The optimization time for the guided approximation kernels ranges from ~1–19 minutes, depending on the material and parameter configuration. For efficient optimization of usable solutions, a soft 1-norm constraint was used, which resulted in a maximal constraint error of ~25%, depending on material and parameter configuration. Furthermore, this model only provides quite limited control over the kernel approximation. In case more specific control over the kernel shape is required, the following model may be more suited.

Manual Approximation Model: This rank-1 model based on a Gaussian mixture provides a few intuitive parameters suitable for manual control, and can be interpreted as a manual optimization in image space, as opposed to the guided optimization approach which operates in kernel space.

The parameters may be adjusted by an artist to obtain either physically based approximations of the ground-truth model or non-physically based kernels that produce a desired effect. As this model only uses two 1D Gaussians, it may be difficult to closely approximate the ground-truth kernels. However, the test-signal convolution results, shown in Figure 6.3–6.9, and also the final renderings, shown in Figure 6.11–6.16, indicate that even this simple model is able to approximate SSS in a visually plausible way. This is furthermore supported by the difference images (Figure 6.17–6.22) and the image-space RMS errors included in Table 6.4, which show that this model provides similar, if not lower, RMS errors in comparison to the other models. It is however worth nothing that these kernels are highly adapted to the given scene and its corresponding features. On one hand, the manual adjustment of the model parameters can be tricky and may be considered as a drawback by some users, but on the other hand this simple model may be a viable tool for artists to achieve a desires SSS effect.

The example renderings shown in Figure 6.11–6.16, in combination with the corresponding difference images (Figure 6.17–6.22), as well as the image-space RMS errors (Table 6.4), suggest that, apart from a few outliers, all models, expect the rank-1 SVD-based approach, offer comparable approximation quality. It is, however, worth noticing that the RMS errors condense a whole 2D image into one single value, and, subsequently, cannot capture regional differences, therefore, additional difference images are provided. Furthermore, please note that the image-space difference is dependant on the rendered scene, and that the count of different scenes is quite moderate. As the results do not show a single best solution for all example cases, the individual properties and characteristics of each model play an important role in the decision which

model to use in a particular situation. This is further outlined and discussed in the next and final chapter.

An overview of the different models is provided in Table 5.1, which conveniently summarizes their properties. It lists different properties corresponding to visual quality, number of required 1D convolutions and the separability for each kernel model. The attribute 'Closed-form solution' also indicates if the kernel computation requires automatic optimization (✗) or not (✓).

The models for SSSS rendering, presented in this work, are manifold and offer different approaches to compute separable (rank-1) approximation kernels, which enable fast SSS rendering. This makes real-time SSS rendering for arbitrary materials possible, as long as an appropriate approximation can be computed. The proof-of-concept test renderings shown in Figure 6.11 to 6.16 indicate that the models are applicable for a variety of materials and show that they produce plausible results for arbitrary geometry and illumination scenarios. Furthermore, it is possible to render SSS based on these models using importance sampling with as low as 17 samples, as outlined in Section 12. This enables real-time frame rates, where SSS is computed in less than 1 ms. However, as the models and the underlying algorithm both use approximations, this approach has some limitations discussed in the following section.

Limitations

The most obvious limitation is the approximation itself. Apart from using higher rank SVD-based approximations, the kernels are limited to rank-1 in order to enable fast convolution. Ground-truth kernels, however, are in general of higher rank, and therefore rank-1 models include an approximation error. The error increases with the rank of the ground-truth kernel and depends on the energy distribution among the corresponding singular values. However, due to the particular shape of diffuse reflectance profiles, only a few singular values are of significance, which aids the proposed low-rank approximation models. Furthermore, as SSS can be a very subtle effect, and many people may not be aware of it, the approximation error may not be noticeable especially if the ground-truth comparison is not provided.

An additional limitation of the rank-1 approximation is that it is in general not radially symmetric, as outlined in Sections 4.3 and 5.1. This property may be exposed by small features, which may act similar to a Dirac peak and reveal the radially asymmetric shape of the kernel. This asymmetry usually has the form of a more or less pronounced cross aligned with the kernel's main axis. It becomes more and more visible as features in the irradiance signal become smaller than the kernel size. An example of the corresponding artifacts is shown in Figure 6.26.

This image also illustrates artifacts due to low sampling rates in combination with a large on-screen kernel size. In this particular case, the convolution was performed using importance sampling with only 17 samples. As a result, banding artifacts may be visible in regions were the kernel falloff is clearly visible, due to large kernel size and harsh illumination. However, the results previously shown in this chapter indicate that for many rendering scenarios such artefacts are less noticeable.

Another limitation, not directly related to our proposed models but worth noticing, stems from the translucency approach used by the SSSS method. This approach is based on simple shadow mapping and therefore may overestimate the thickness of non-convex objects and sub-

sequently reduce their translucency for corresponding regions. This can be especially noticeable under harsh back-illumination from a single light source, hence illumination from different directions using multiple light sources may hide such artifacts. An example of such artifacts is shown in Figure 6.27. In this particular case, milk droplets cast shadows on an edge where the translucency component of SSS is clearly visible. Due to the non-convex shape of the mesh, the thickness in the shadowed regions is overestimated, and, therefore, the translucency is wrongly attenuated. Although shadowed regions, per definition, exhibit no light that could be transmitted through the material, it would be expected that at least some light from illuminated neighbouring regions would scatter into the shadows on the backside.

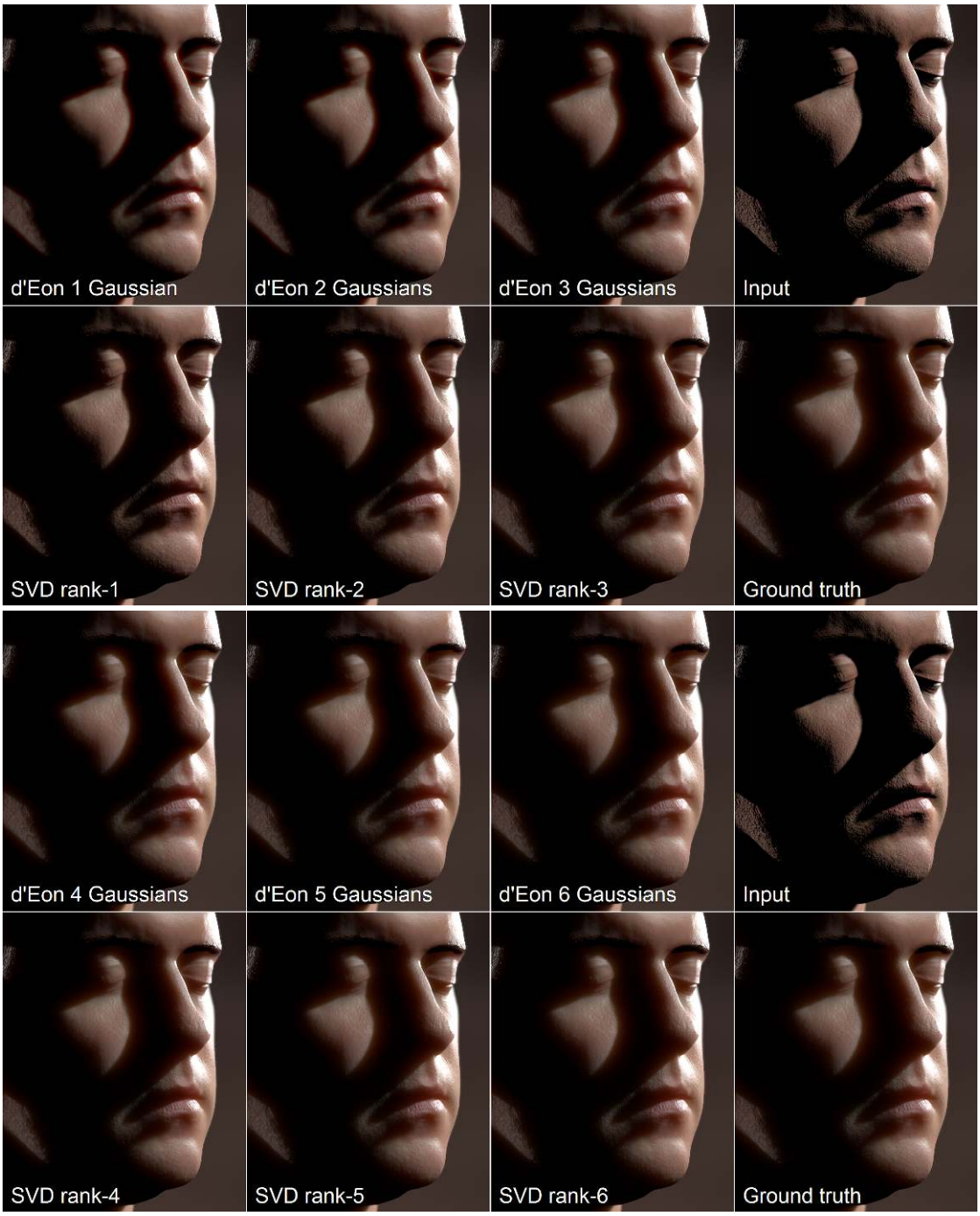
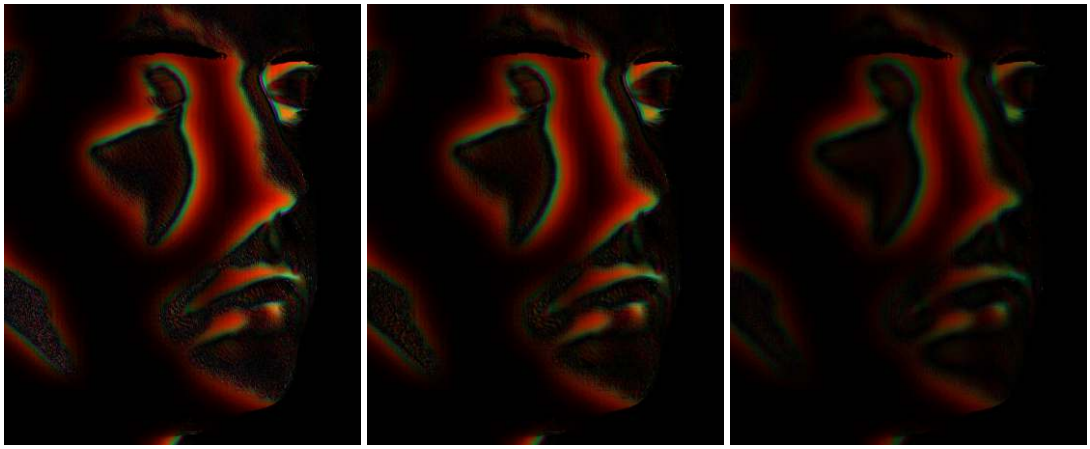


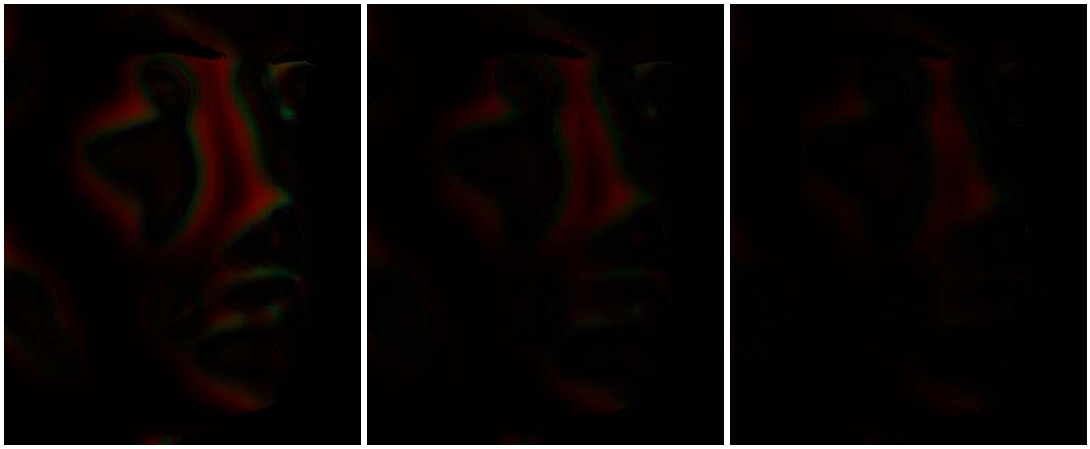
Figure 6.23: Comparison of the SVD model versus the Gaussian mixture model from d'Eon et al. [5, 6] using the Skin1 material applied to the Head scene (17 samples). The corresponding image-space RMS errors are included in Table 6.5, and the difference images are shown in Figure 6.24 and 6.25.



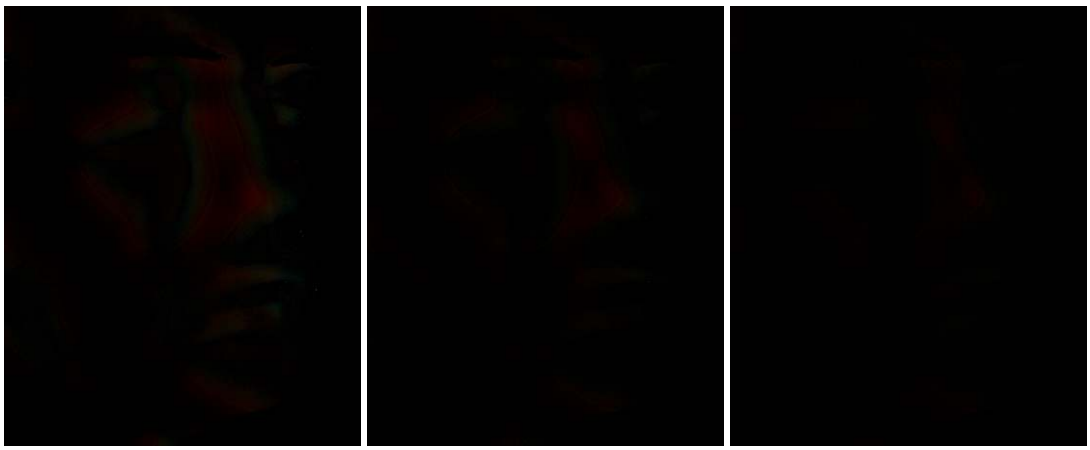
(a) d'Eon 1 Gaussian (b) d'Eon 2 Gaussians (c) d'Eon 3 Gaussians

(d) SVD rank-1 (e) SVD rank-2 (f) SVD rank-3

Figure 6.24: Difference images of the SVD–d'Eon (rank 1-3) comparison shown in Figure 6.23, with corresponding image-space RMS errors included in Table 6.5.



(a) d'Eon 4 Gaussians (b) d'Eon 5 Gaussians (c) d'Eon 6 Gaussians



(d) SVD rank-4 (e) SVD rank-5 (f) SVD rank-6

Figure 6.25: Difference images of the SVD–d'Eon (rank 4-6) comparison shown in Figure 6.23, with corresponding image-space RMS errors included in Table 6.5.

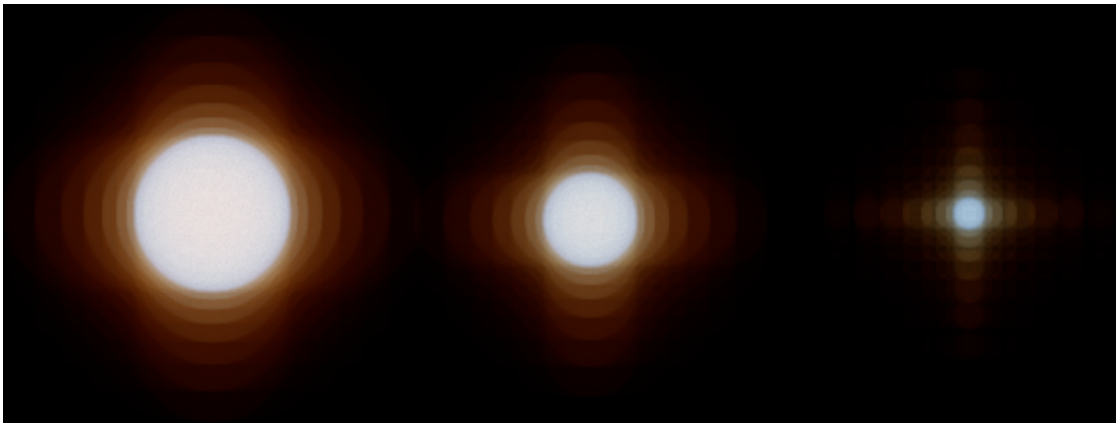


Figure 6.26: This figure shows example artifacts due to the radially asymmetric shape of a separable (rank-1) kernel. In addition, banding artifacts are visible due to harsh illumination combined with a large on-screen kernel size and a low sample count of 17. In this particular case, the analytic pre-integration kernel for the Skin1 material was used.

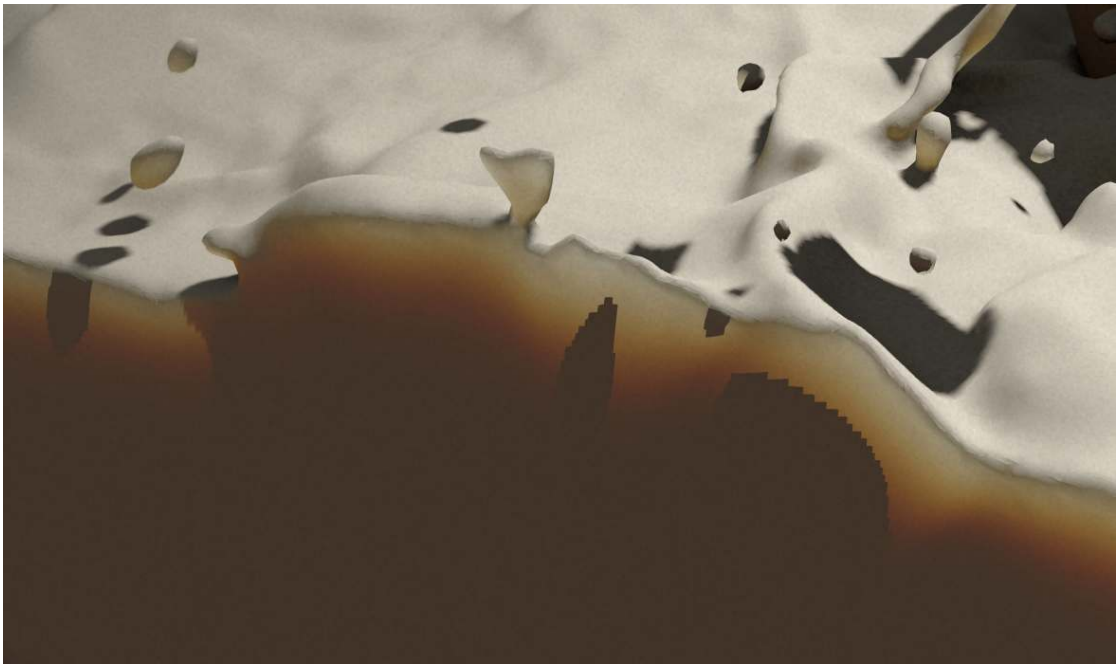


Figure 6.27: This figure shows examples of translucency artifacts in case of non-convex objects. These artifacts stem from the translucency approach based on shadow mapping, which is used by the SSSS algorithm.

CHAPTER 7

Conclusion

The rendering of SSS for arbitrary materials in real time is a challenging task. While offline rendering algorithms can afford complex simulations, real-time methods need to compute visually plausible images in just a few milliseconds. This lead to various real-time approaches that use different approximations to render SSS. One commonly established scheme is to filter surface irradiance in screen space, which has the beneficial property of being almost independent from scene complexity. In this context, rendering speed and quality are mainly determined by the used filter. One of the most simple filters is a single 2D Gaussian, which is radially symmetric and separable. This separability enables fast convolution, but the use of only a single 2D Gaussian results in poor rendering quality. Therefore, the state of the art is to perform multiple convolutions based on a 2D Gaussian mixture approximation of the diffuse reflectance profile, which delivers high quality results at the price of increased rendering time.

The basic idea of Separable Subsurface Scattering is to use a separable, but not necessarily radially symmetric, filter which mimics the ground truth as close as possible. The models proposed in this thesis build upon this idea and represent different separable approximations. The SVD-based approach offers a rank-1 kernel with inferior quality compared to the other models. However, the results show that higher rank SVD-based approximation kernels have a favorable cost-quality progression compared to the state of the art. The analytic pre-integration based model offers a deterministic solution, fast computation and is exact for special-case irradiance signals, while providing much better quality than the SVD-based rank-1 solution. In case more control over the approximation quality is desired, the guided optimization approach offers a parametrized model to compute kernels which may be adapted for near or far scattering. A drawback of this approach is that the optimization is more difficult and time consuming. The last model is based on the manual optimization of a mixture of two 1D Gaussians, which enables total artistic control by providing an intuitive interpretation of the parameters. The results show that even this simple model is able to provide visually plausible approximation kernels.

The conclusion, based on the results presented in the previous section, is to use higher-rank SVD-based solutions whenever time constraints allow for multiple (2+) convolutions, which will eventually be the case, as the speed of modern GPUs increases rapidly. In case only two

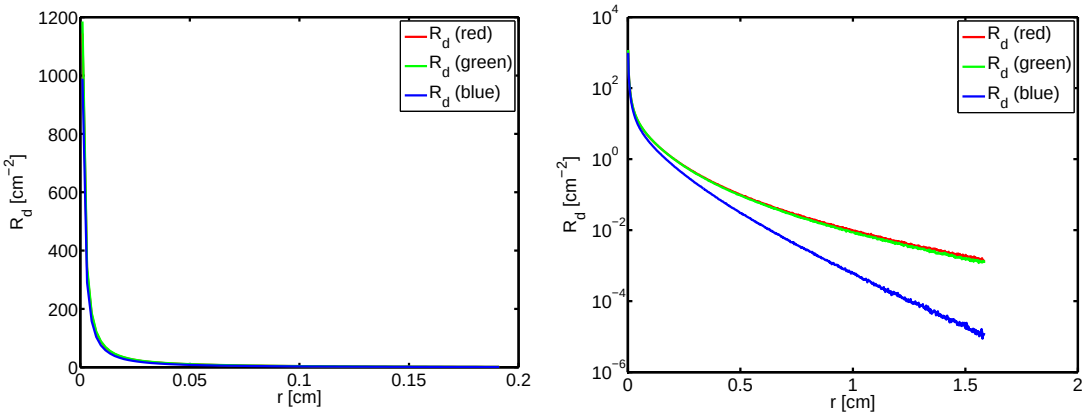
1D convolutions are affordable, a separable (rank-1) approximation provides a good trade-off between quality and rendering speed. As all models, except the SVD-based rank-1 model, deliver quite similar results, and there is no single best solution for all cases, the reason to favour one model over the other is highly determined by their individual properties. In case fast and automatic physically based kernel approximation is required, the pre-integrated model is a viable solution. However, if more artistic control or a non-physical effect is desired, the manual approximation approach delivers good quality, while providing easy and interactive control over the approximation. This is further supported by the fact that this model is already used in production. Since the guided optimization approach, only delivers similar and not clearly superior results in comparison to the previous models, but is more difficult to use and needs more time for kernel generation, this approach seems less attractive, but may be viable if the application scenario requires automatic computation of a specific approximation.

This approach for real-time rendering of SSS inherits a few limitations from the proposed kernel models and the corresponding SSSS algorithm. The fact that the proposed models represent an approximation of a ground-truth kernel is a limiting factor for the final SSS rendering quality. Rank-1 kernels are only capable of approximating the ground truth within their rank-1 constraint, and the quality depends on the energy distribution among the singular values. Higher quality approximations are only possible by using kernels of higher rank which require multiple (2+) convolutions. In order to ensure real-time performance, it is also necessary to use a small sample count. This may introduce banding artifacts and reveal the radial asymmetry of separable kernels, in case of small features in combination with a large on-screen kernel size. The simplified translucency approach used for SSSS rendering may also introduce artifacts for non-convex objects.

Despite these limitations, different renderings of artificial test signals and proof-of-concept scenes showed that all four models, with the exception of rank-1 SVD-based approximations, enable plausible approximation of SSS for various materials. Moreover, since the SSSS algorithm is in essence a post-processing step, this SSS rendering approach may be easily integrated into existing rendering pipelines. The proposed models, in combination with the SSSS algorithm, enable real-time rendering of SSS for arbitrary materials in fully dynamic scenes.

APPENDIX A

Appendix



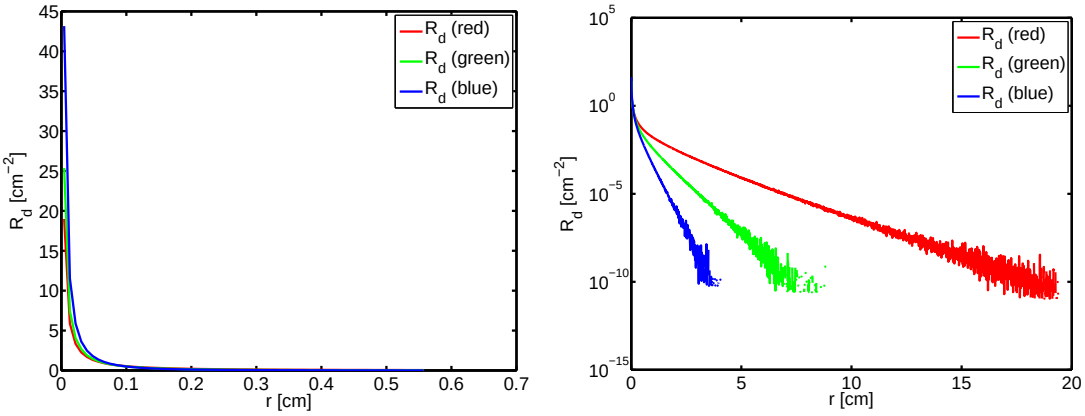
(a) Linear

(b) Logarithmic

Figure A.1: Plots of the diffuse reflectance profile for material Apple. The reason behind occasional noise is explained in Section 6.1.

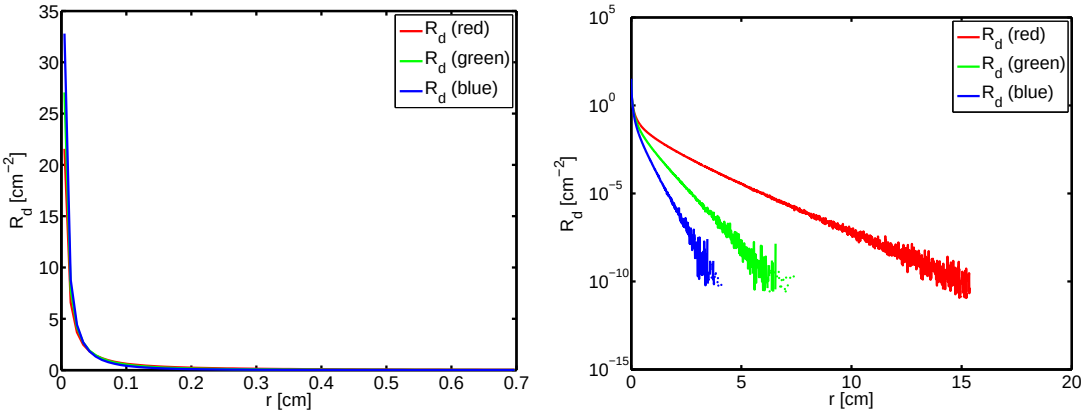
Material	$\sqrt{\tau_n}$ [RGB]	$\sqrt{\tau_f}$ [RGB]	w
Apple	[0.0495, 0.0455, 0.0372]	[1.1, 1.08, 0.926]	0.368
Ketchup	[0.0907, 0.0331, 0.0331]	[1.5, 0.118, 0.0966]	0.665
Marble	[0.0371, 0.0207, 0.015]	[0.663, 0.616, 0.558]	0.234
Skin1	[0.429, 0.211, 0.03]	[1.15, 0.69, 0.471]	0.533
Wholemilk	[0.0846, 0.0846, 0.0583]	[1.5, 1.39, 1.17]	0.407

Table A.1: Model parameters for the manual approximation kernels.



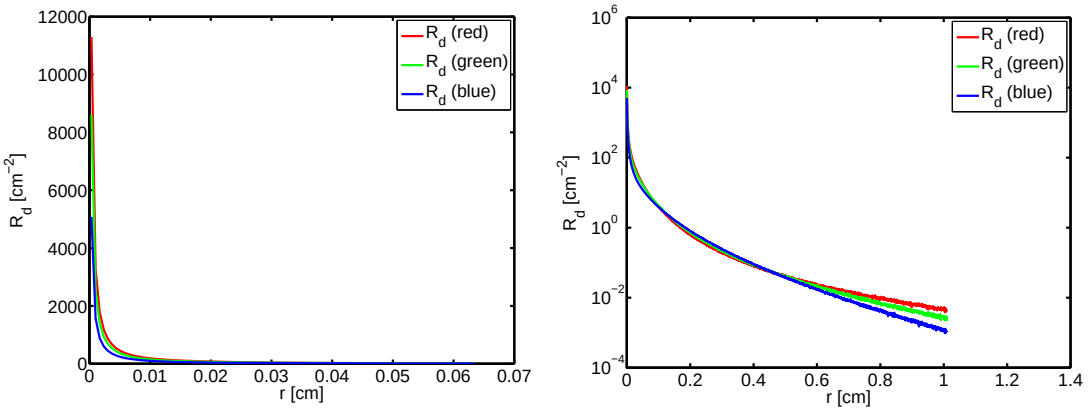
(a) Linear (b) Logarithmic

Figure A.2: Plots of the diffuse reflectance profile for material Chicken1. The reason behind occasional noise is explained in Section 6.1.



(a) Linear (b) Logarithmic

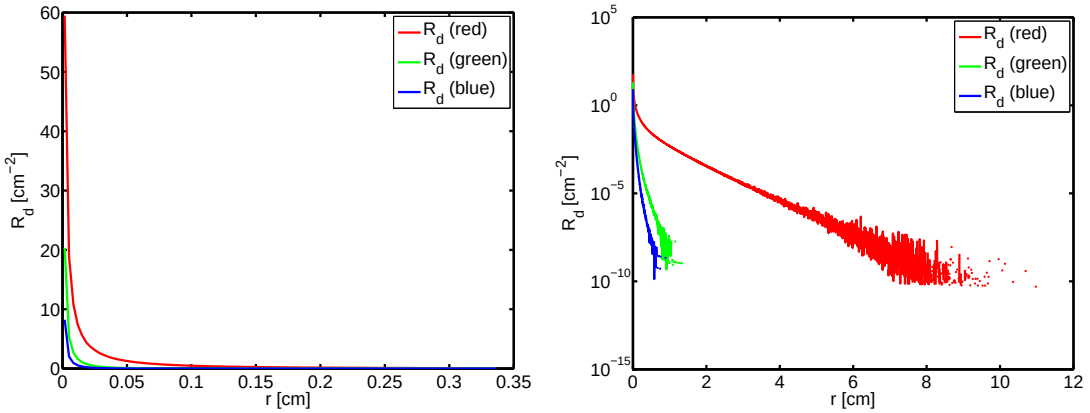
Figure A.3: Plots of the diffuse reflectance profile for material Chicken2. The reason behind occasional noise is explained in Section 6.1.



(a) Linear

(b) Logarithmic

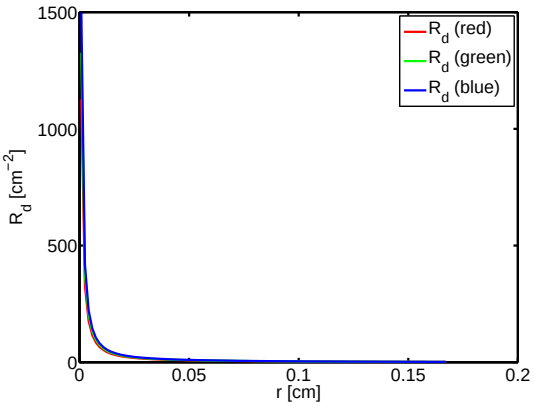
Figure A.4: Plots of the diffuse reflectance profile for material Cream. The reason behind occasional noise is explained in Section 6.1.



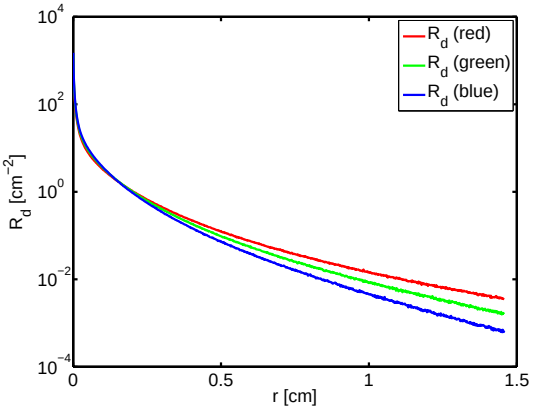
(a) Linear

(b) Logarithmic

Figure A.5: Plots of the diffuse reflectance profile for material Ketchup. The reason behind occasional noise is explained in Section 6.1.

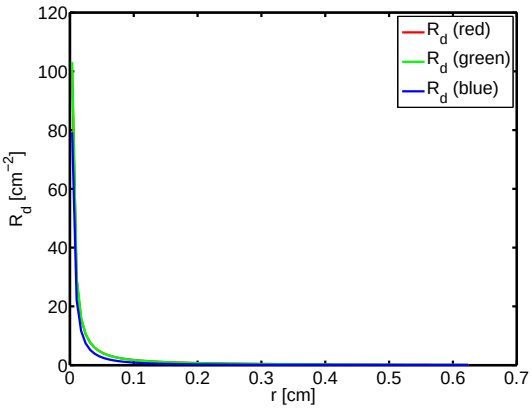


(a) Linear

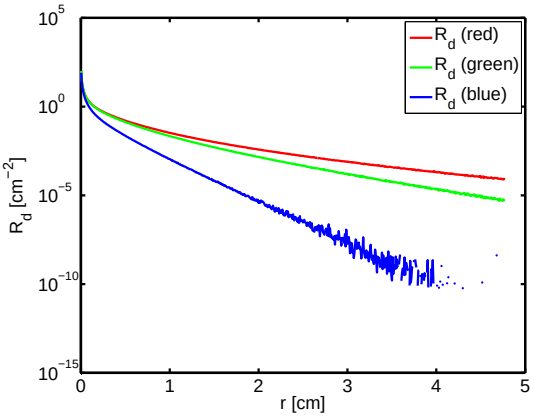


(b) Logarithmic

Figure A.6: Plots of the diffuse reflectance profile for material Marble. The reason behind occasional noise is explained in Section 6.1.

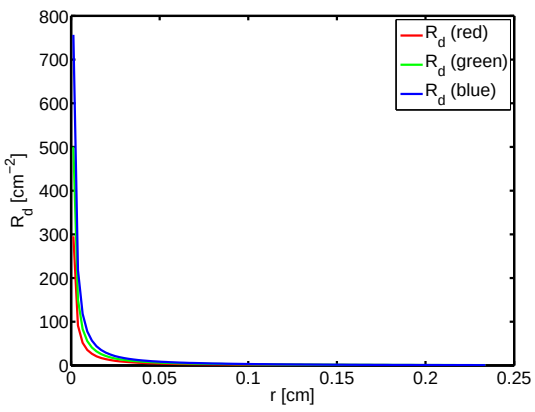


(a) Linear

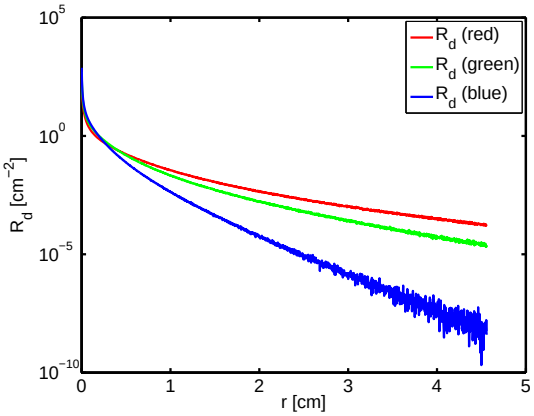


(b) Logarithmic

Figure A.7: Plots of the diffuse reflectance profile for material Potato. The reason behind occasional noise is explained in Section 6.1.

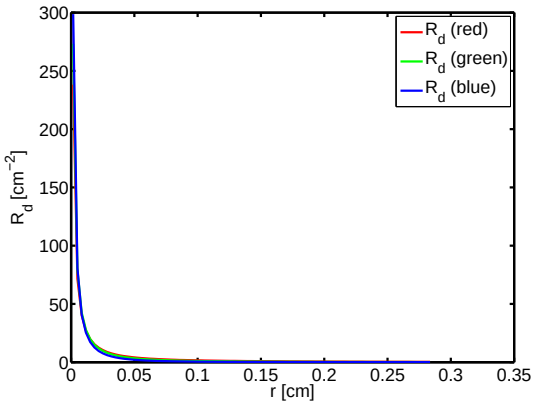


(a) Linear

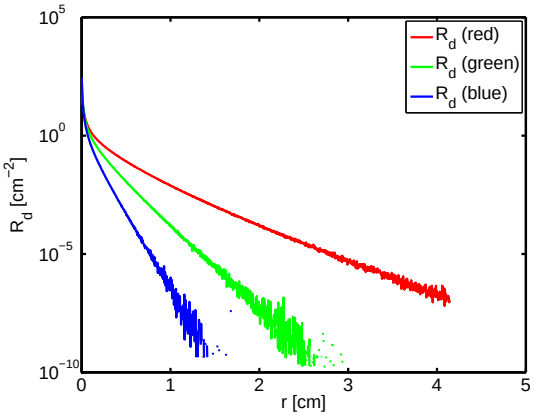


(b) Logarithmic

Figure A.8: Plots of the diffuse reflectance profile for material Skimmilk. The reason behind occasional noise is explained in Section 6.1.

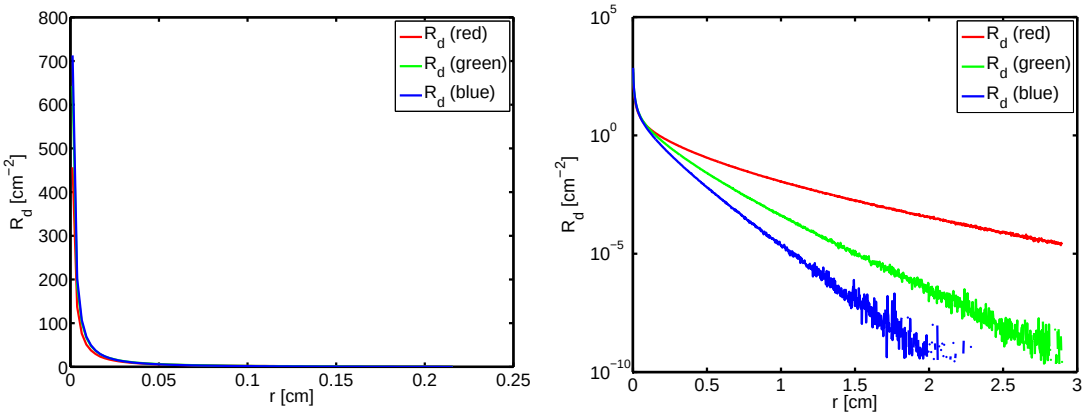


(a) Linear



(b) Logarithmic

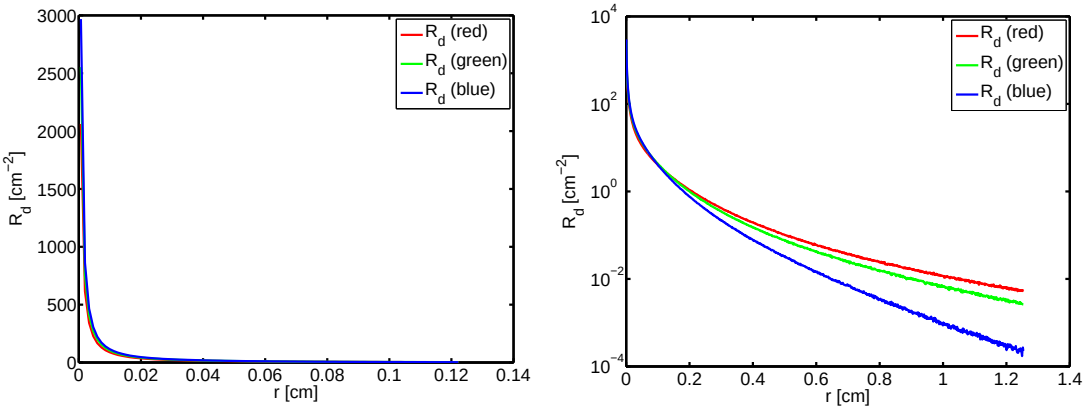
Figure A.9: Plots of the diffuse reflectance profile for material Skin1. The reason behind occasional noise is explained in Section 6.1.



(a) Linear

(b) Logarithmic

Figure A.10: Plots of the diffuse reflectance profile for material Skin2. The reason behind occasional noise is explained in Section 6.1.



(a) Linear

(b) Logarithmic

Figure A.11: Plots of the diffuse reflectance profile for material Wholemilk. The reason behind occasional noise is explained in Section 6.1.

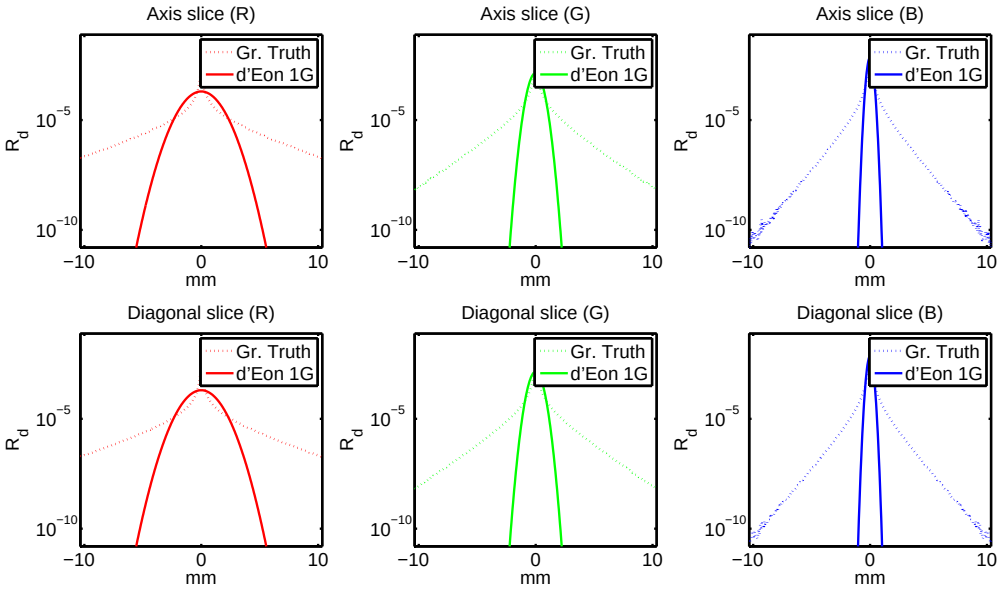


Figure A.12: Plot of the d'Eon 1 Gaussian kernel for material Skin1.

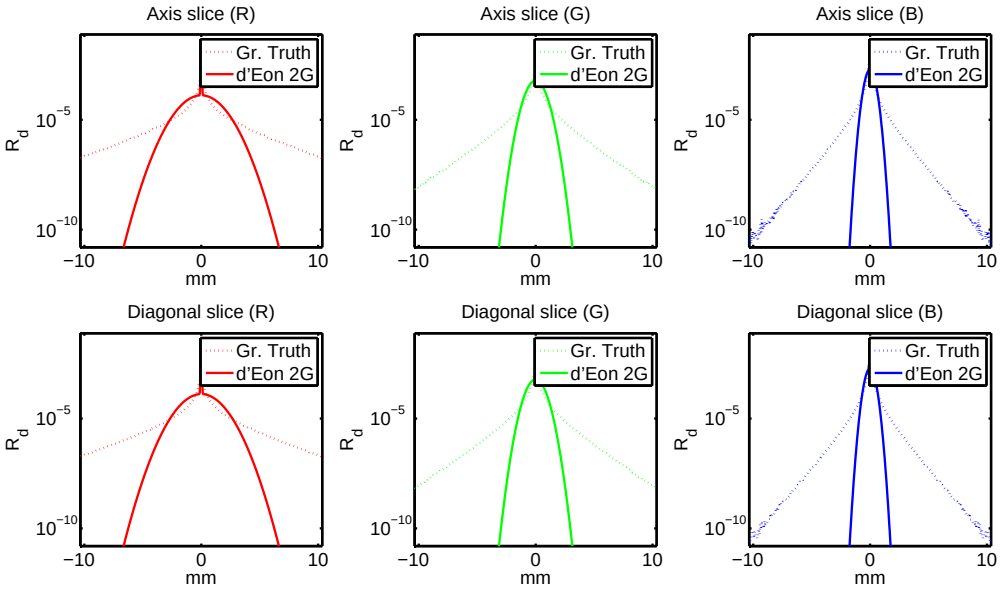


Figure A.13: Plot of the d'Eon 2 Gaussians kernel for material Skin1.

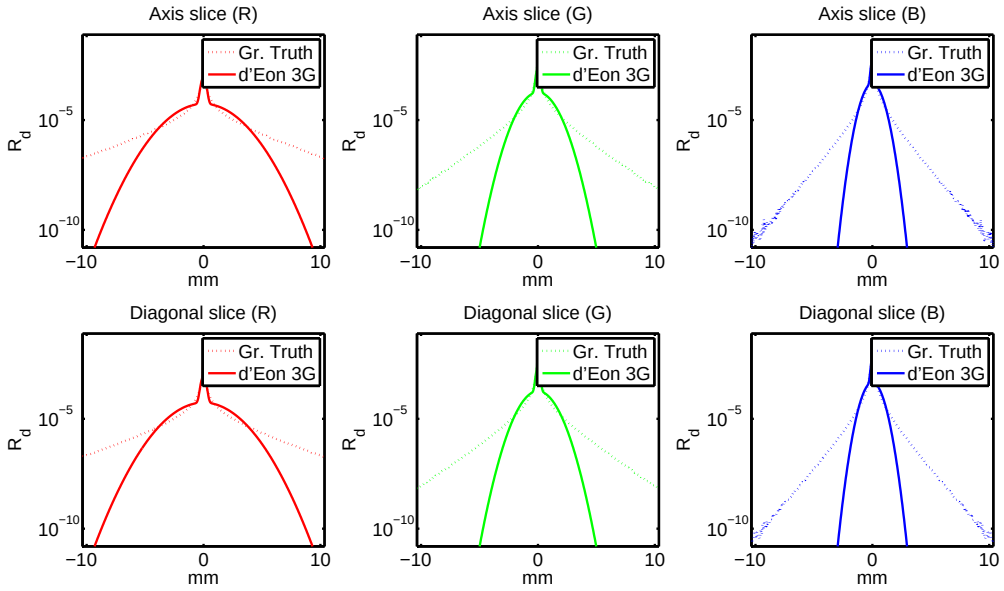


Figure A.14: Plot of the d'Eon 3 Gaussians kernel for material Skin1.

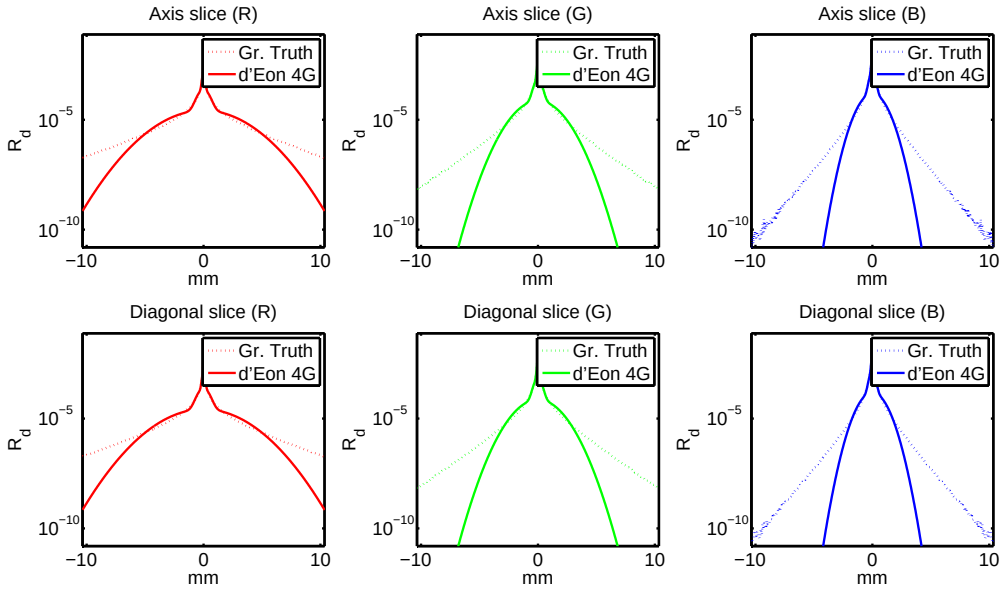


Figure A.15: Plot of the d'Eon 4 Gaussians kernel for material Skin1.

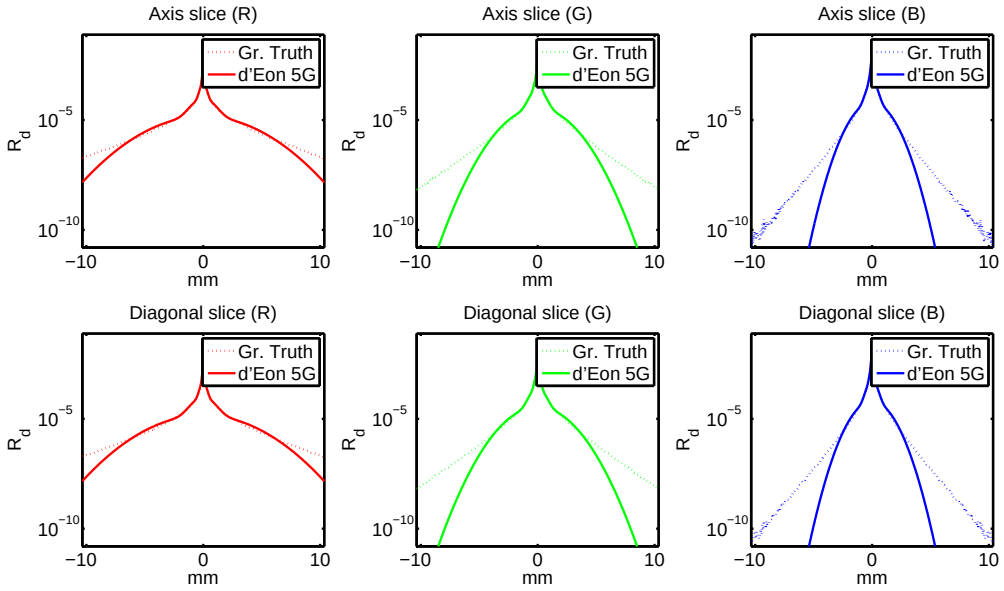


Figure A.16: Plot of the d'Eon 5 Gaussians kernel for material Skin1.

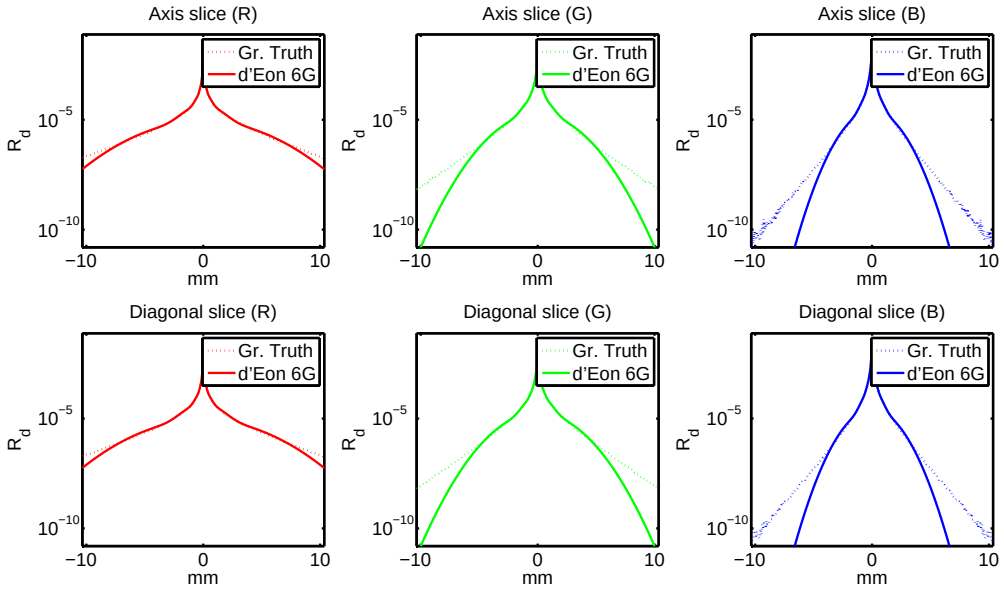


Figure A.17: Plot of the d'Eon 6 Gaussians kernel for material Skin1.

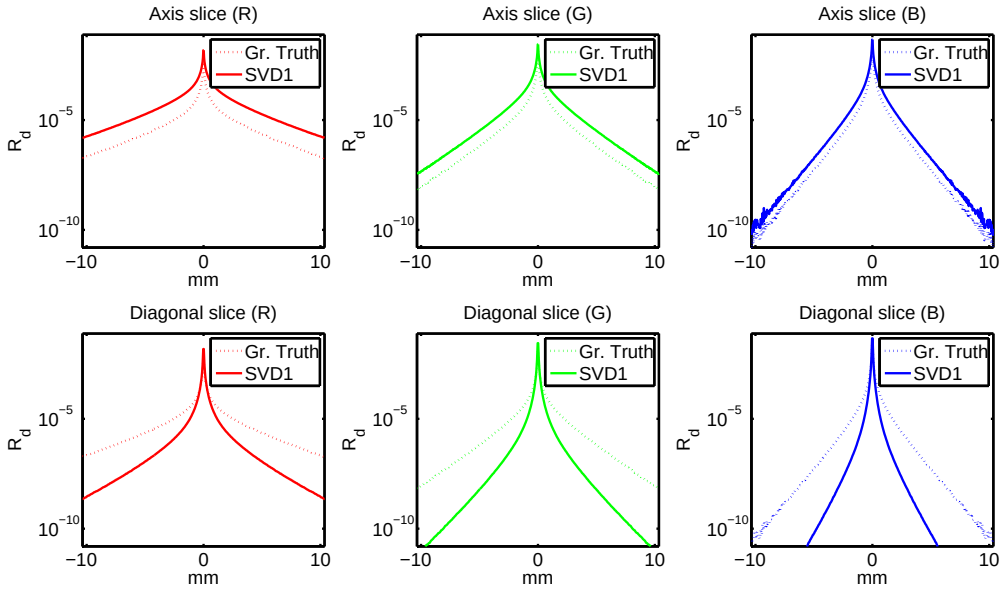


Figure A.18: Plot of the SVD rank-1 kernel for material Skin1.

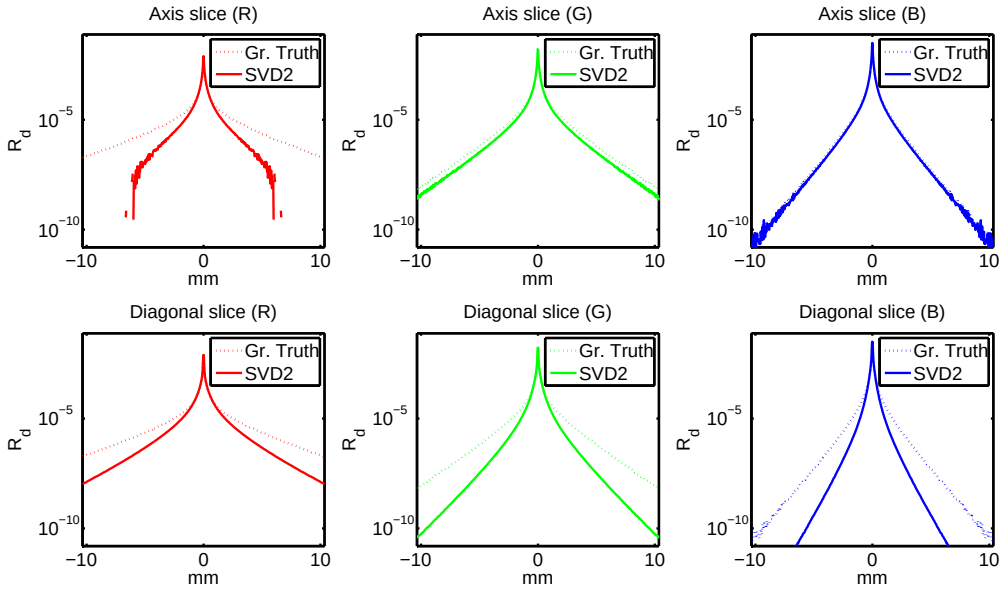


Figure A.19: Plot of the SVD rank-2 kernel for material Skin1.

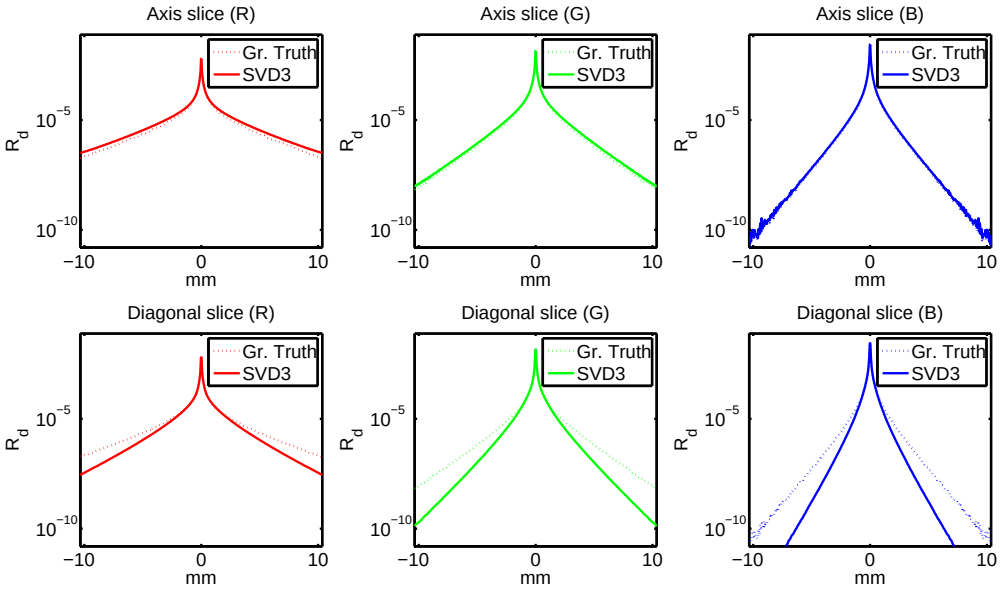


Figure A.20: Plot of the SVD rank-3 kernel for material Skin1.

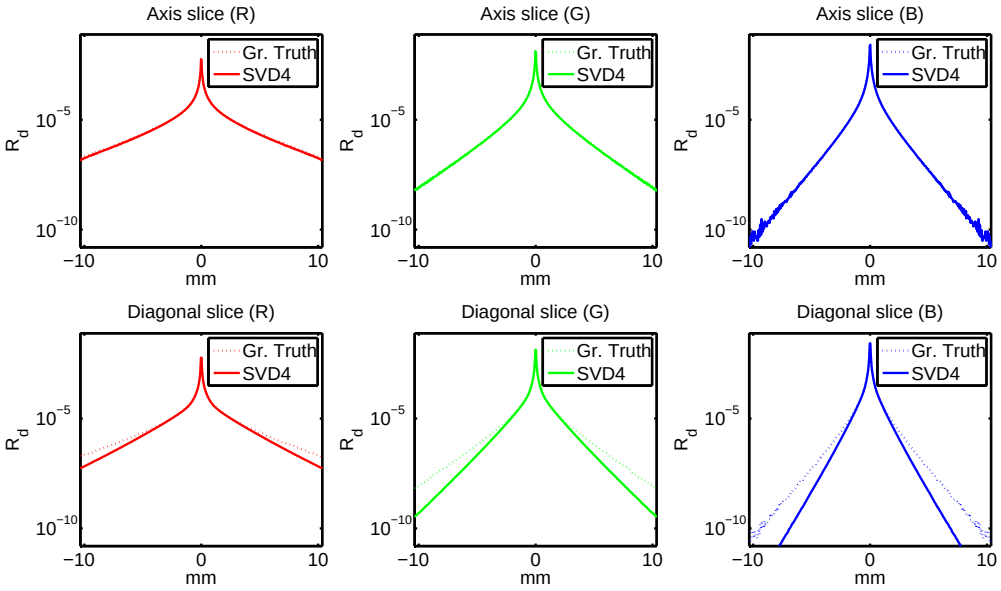


Figure A.21: Plot of the SVD rank-4 kernel for material Skin1.

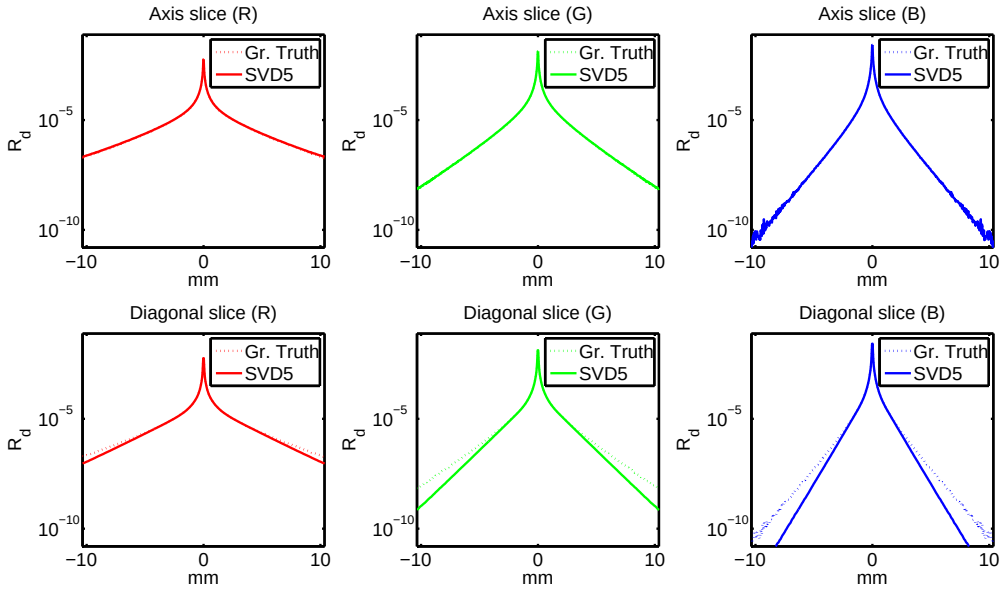


Figure A.22: Plot of the SVD rank-5 kernel for material Skin1.

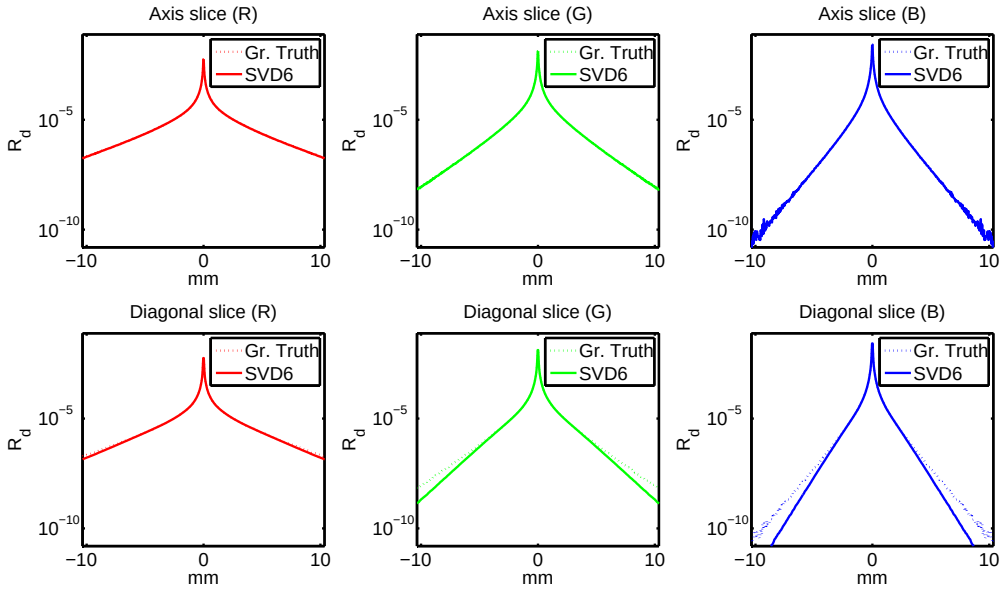


Figure A.23: Plot of the SVD rank-6 kernel for material Skin1.

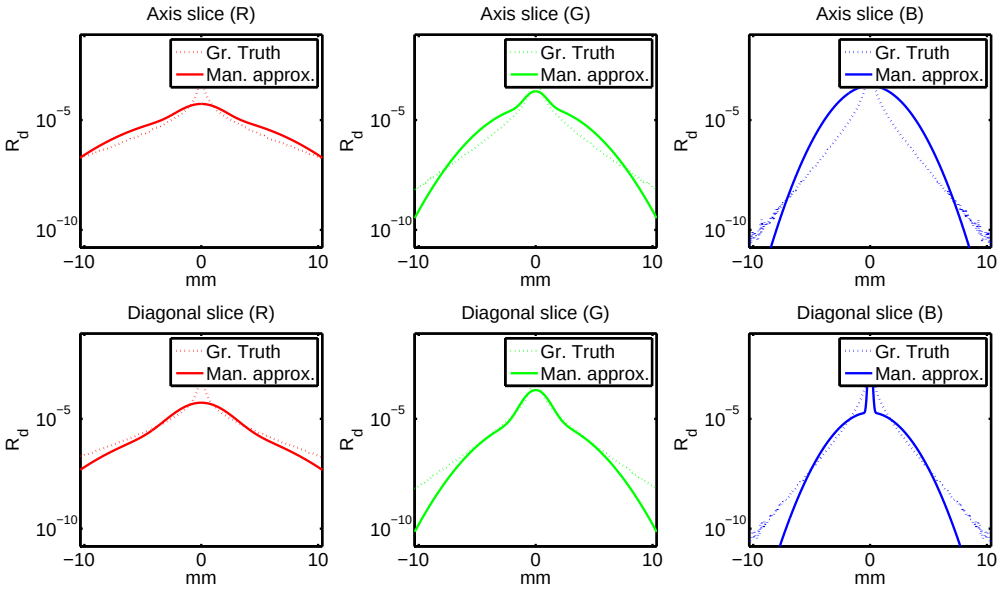


Figure A.24: Plot of the Manual approximation kernel for material Skin1.

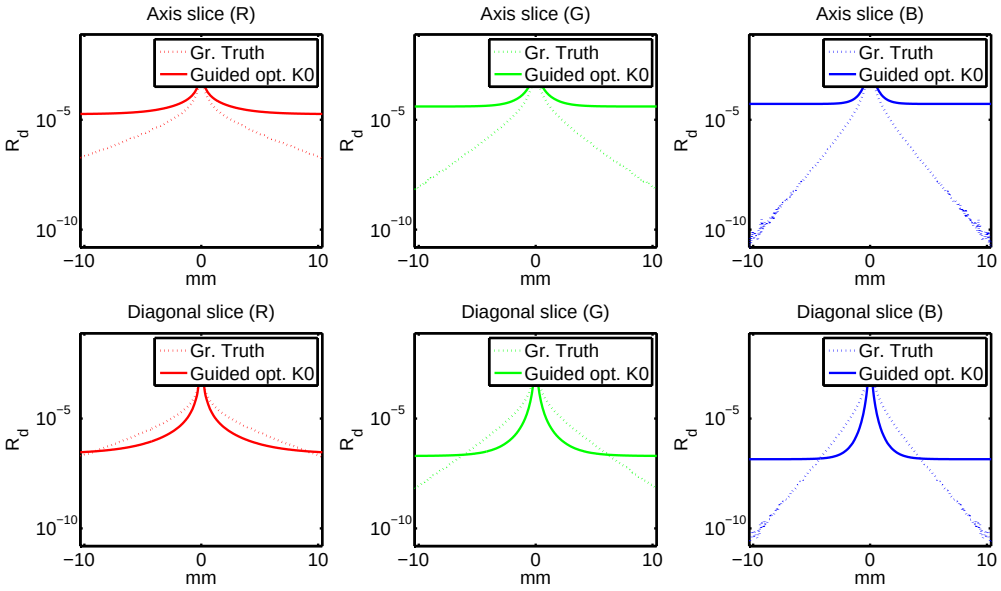


Figure A.25: Plot of the Guided optimization (k=0) kernel for material Skin1.

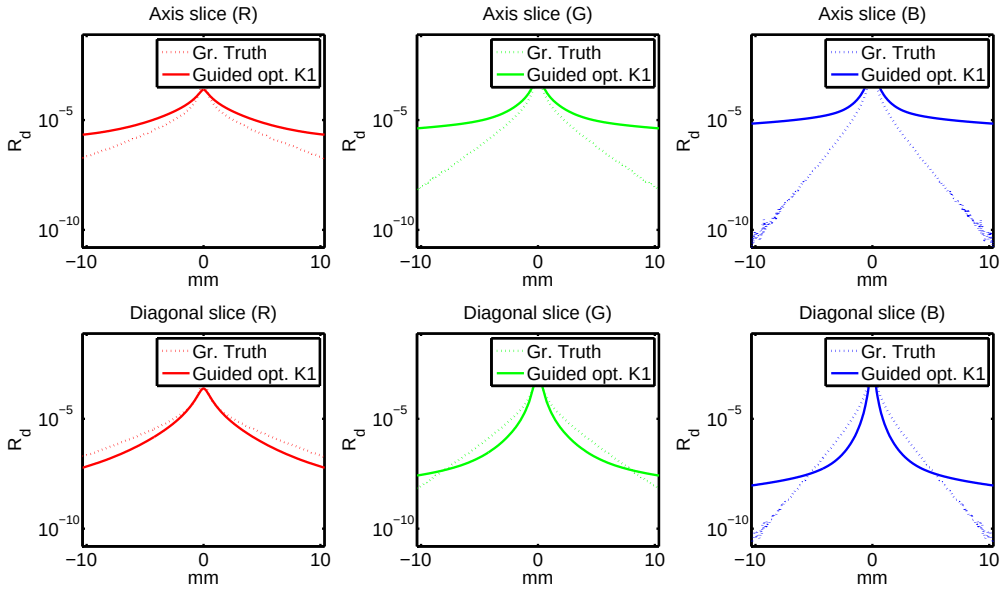


Figure A.26: Plot of the Guided optimization (k=1) kernel for material Skin1.

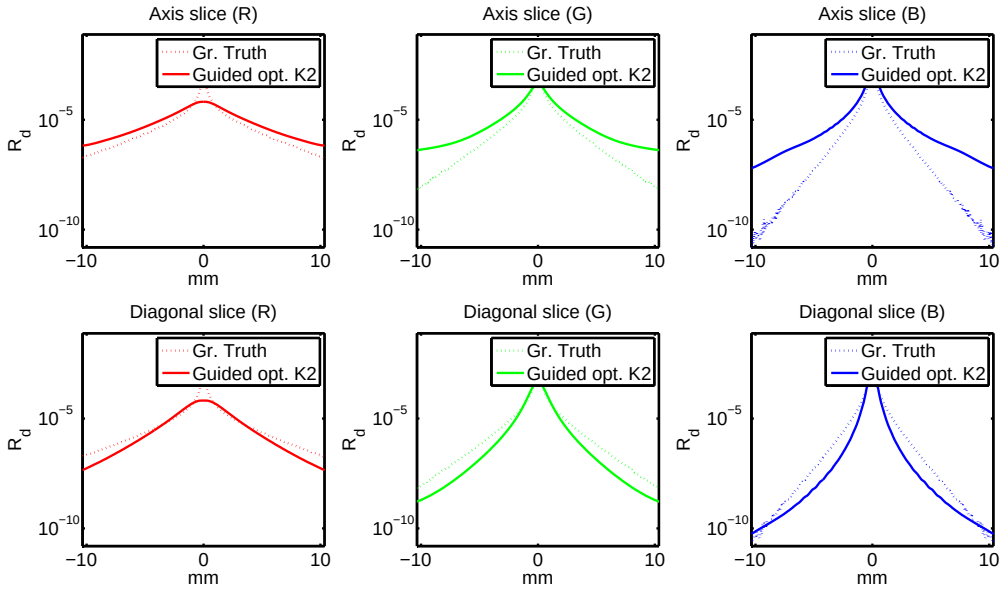


Figure A.27: Plot of the Guided optimization (k=2) kernel for material Skin1.

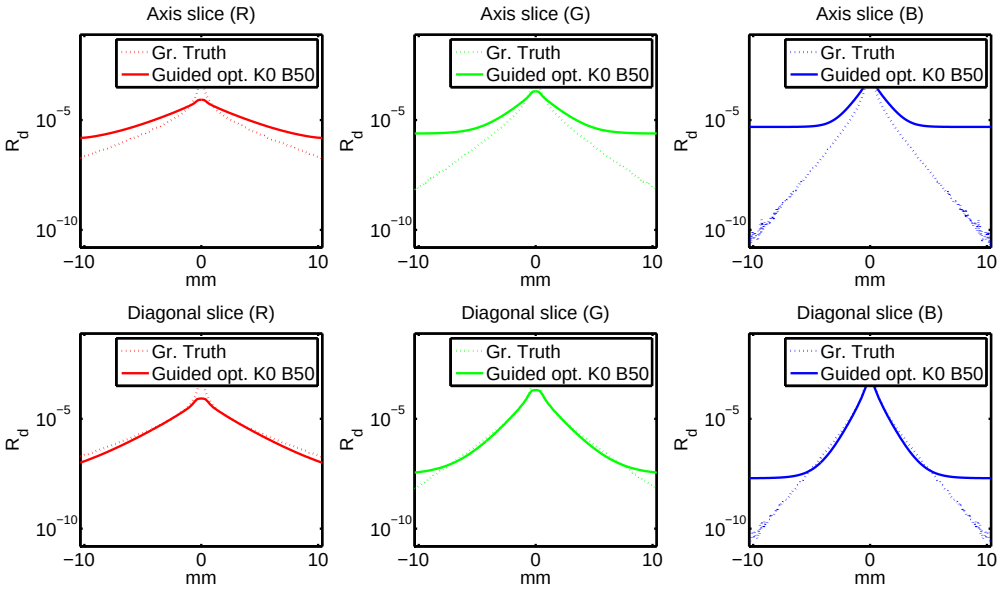


Figure A.28: Plot of the Guided optimization (k=0, b=50) kernel for material Skin1.

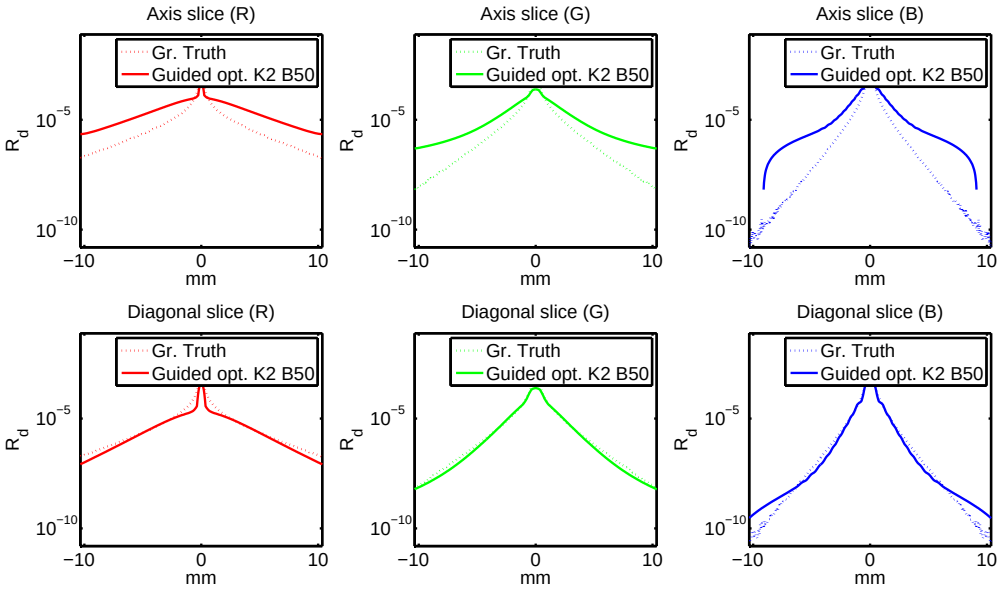


Figure A.29: Plot of the Guided optimization (k=2, b=50) kernel for material Skin1.

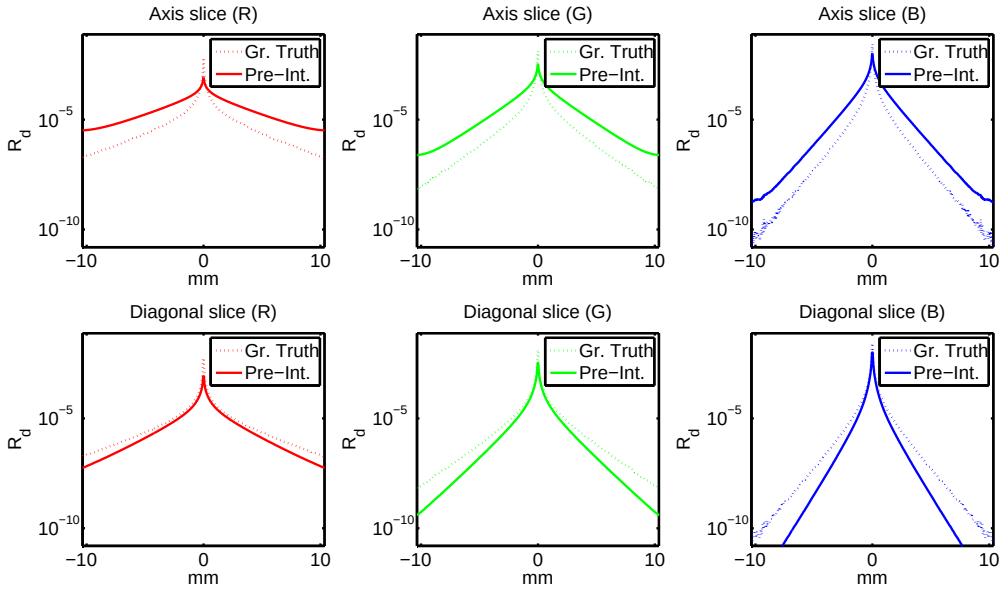


Figure A.30: Plot of the Pre-Integration kernel for material Skin1.

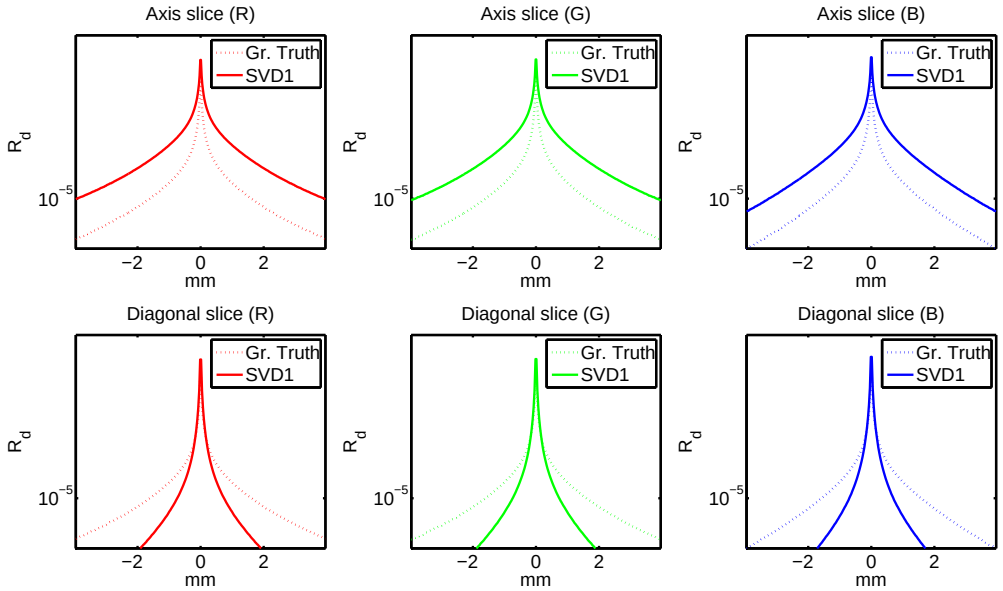


Figure A.31: Plot of the SVD rank-1 kernel for material Apple.

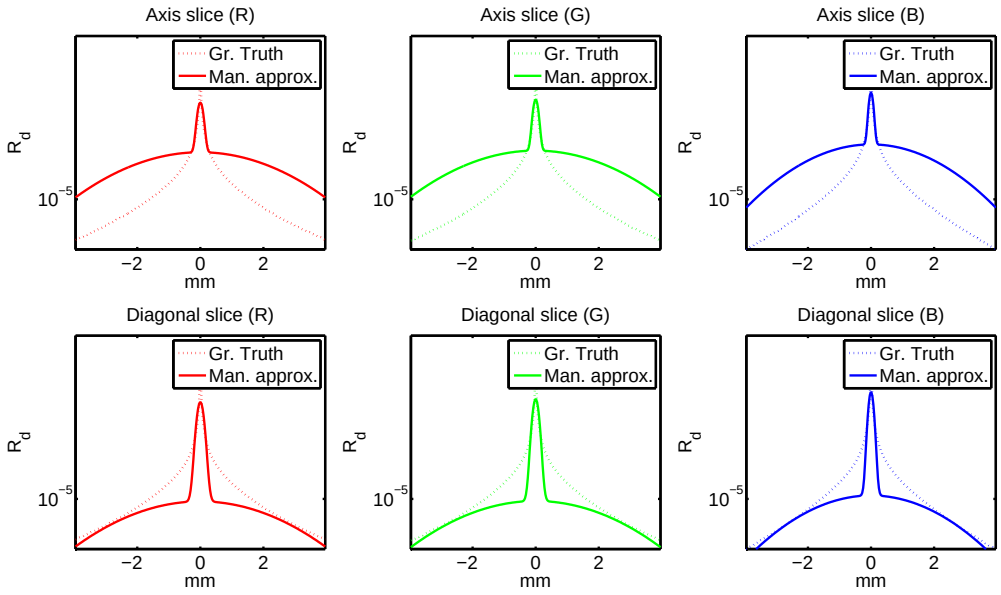


Figure A.32: Plot of the Manual approximation kernel for material Apple.

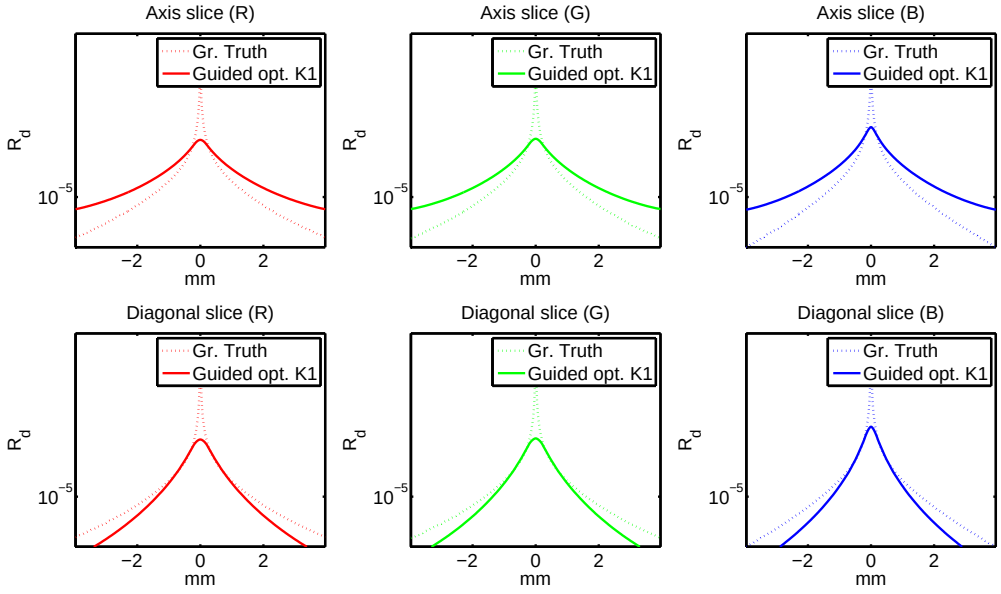


Figure A.33: Plot of the Guided optimization (k=1) kernel for material Apple.

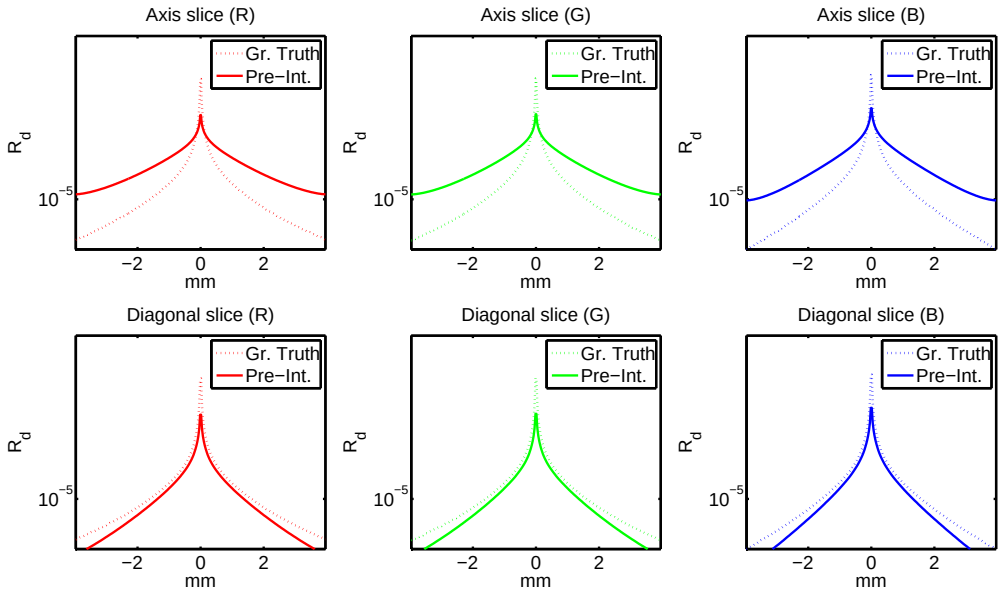


Figure A.34: Plot of the Pre-Integration kernel for material Apple.

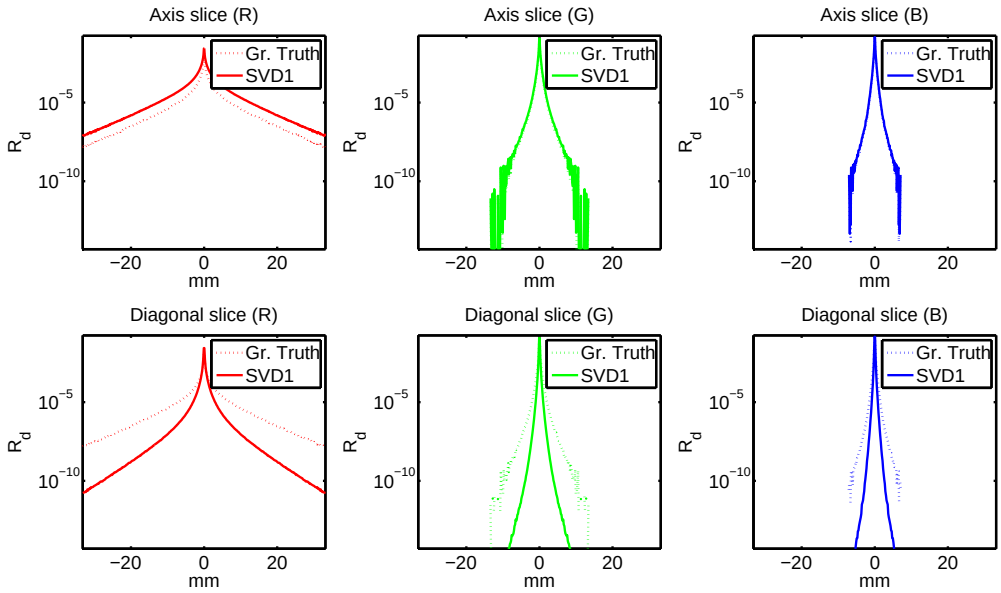


Figure A.35: Plot of the SVD rank-1 kernel for material Ketchup.

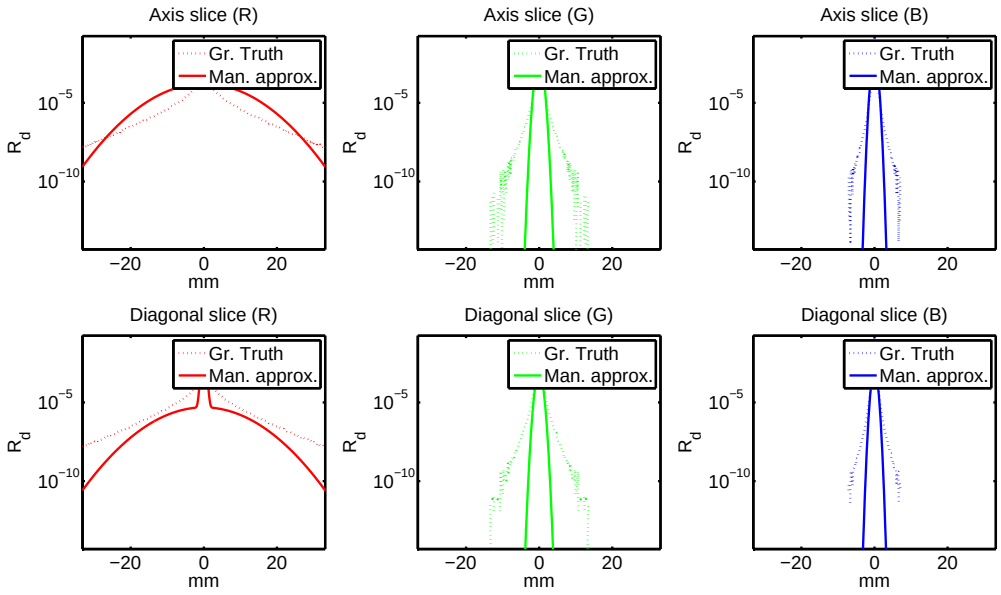


Figure A.36: Plot of the Manual approximation kernel for material Ketchup.

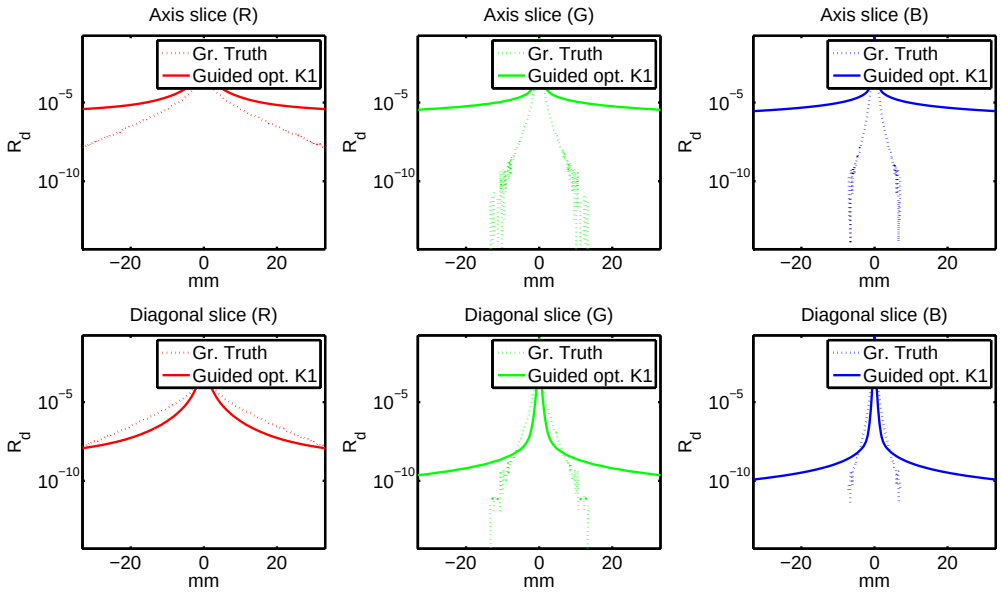


Figure A.37: Plot of the Guided optimization (k=1) kernel for material Ketchup.

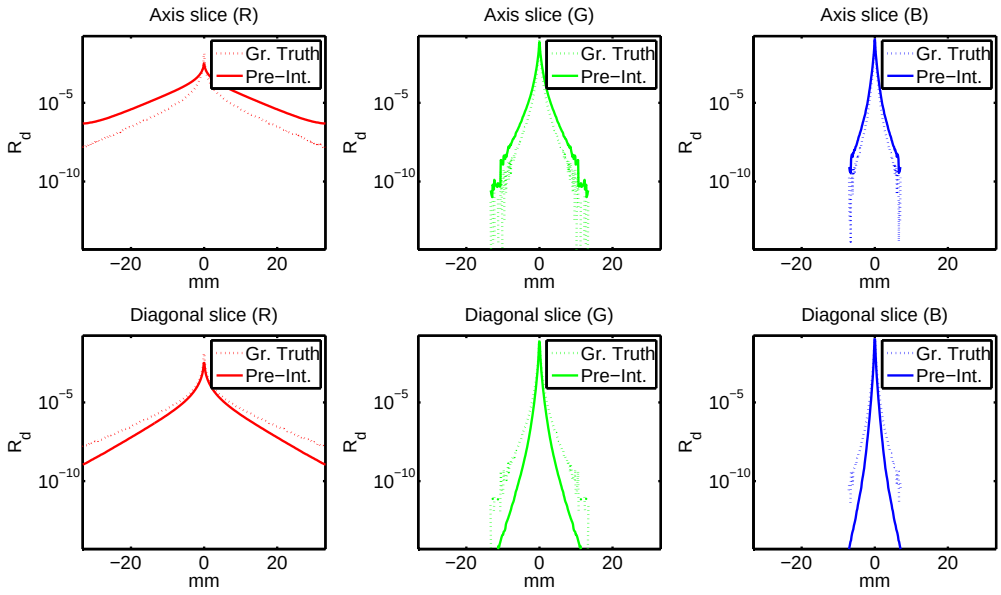


Figure A.38: Plot of the Pre-Integration kernel for material Ketchup.

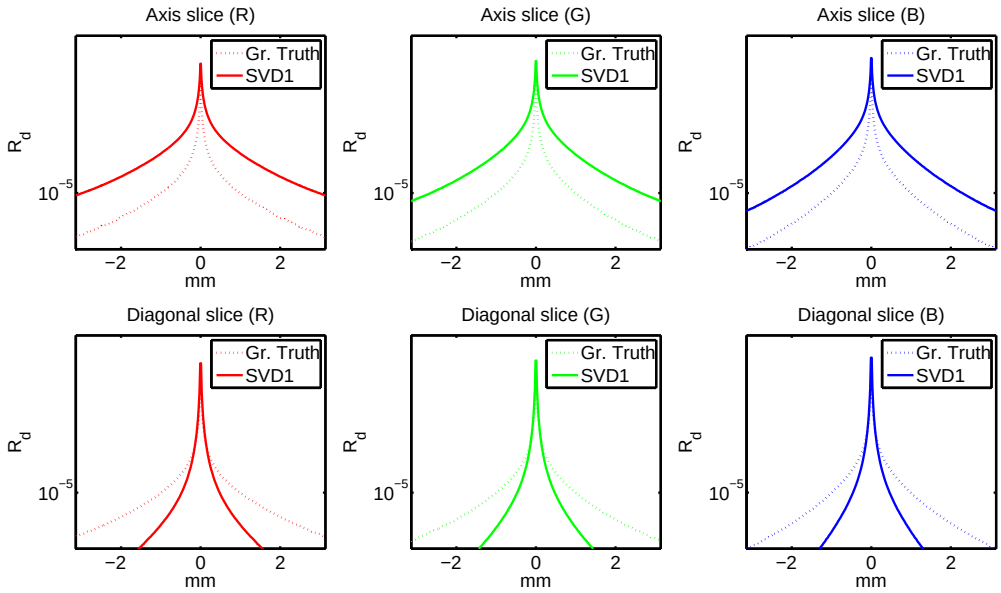


Figure A.39: Plot of the SVD rank-1 kernel for material Wholemilk.

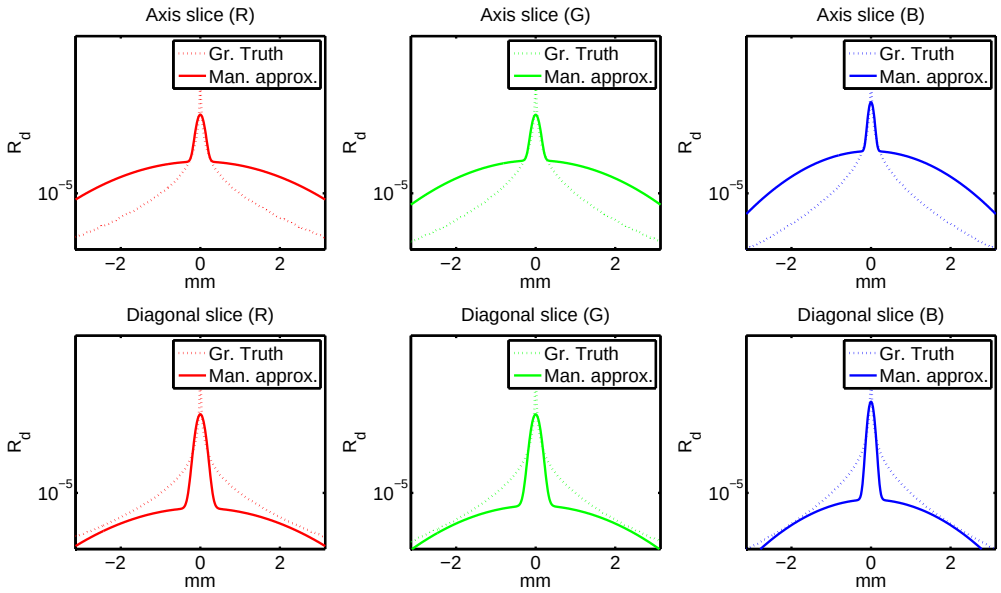


Figure A.40: Plot of the Manual approximation kernel for material Wholemilk.

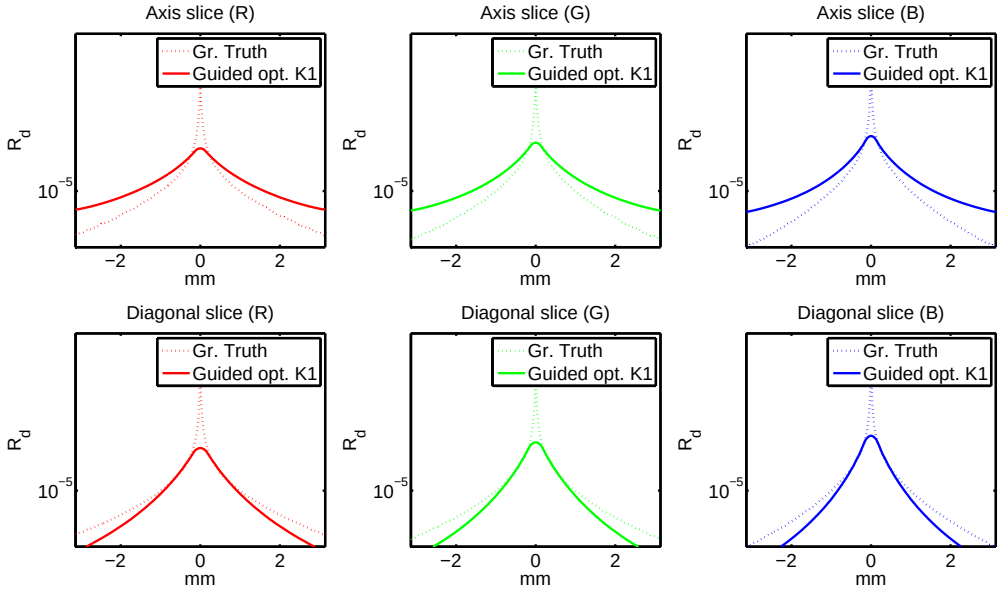


Figure A.41: Plot of the Guided optimization (k=1) kernel for material Wholemilk.

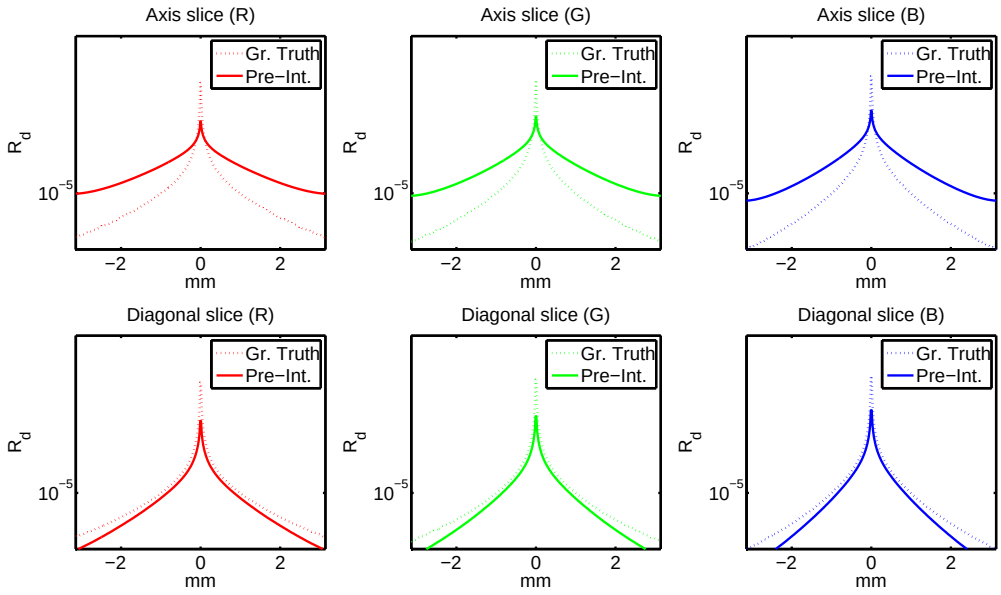


Figure A.42: Plot of the Pre-Integration kernel for material Wholemilk.

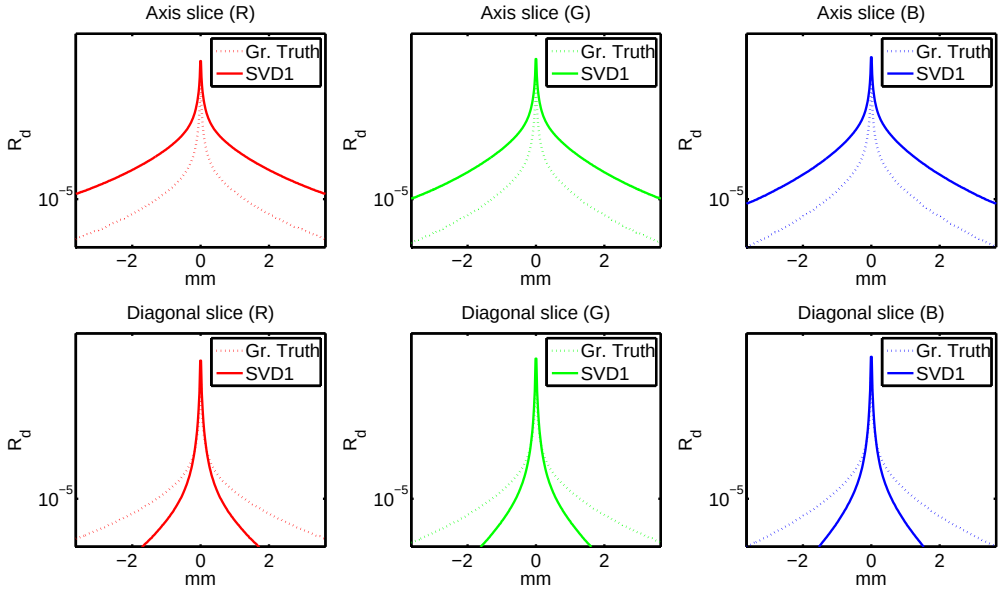


Figure A.43: Plot of the SVD rank-1 kernel for material Marble.

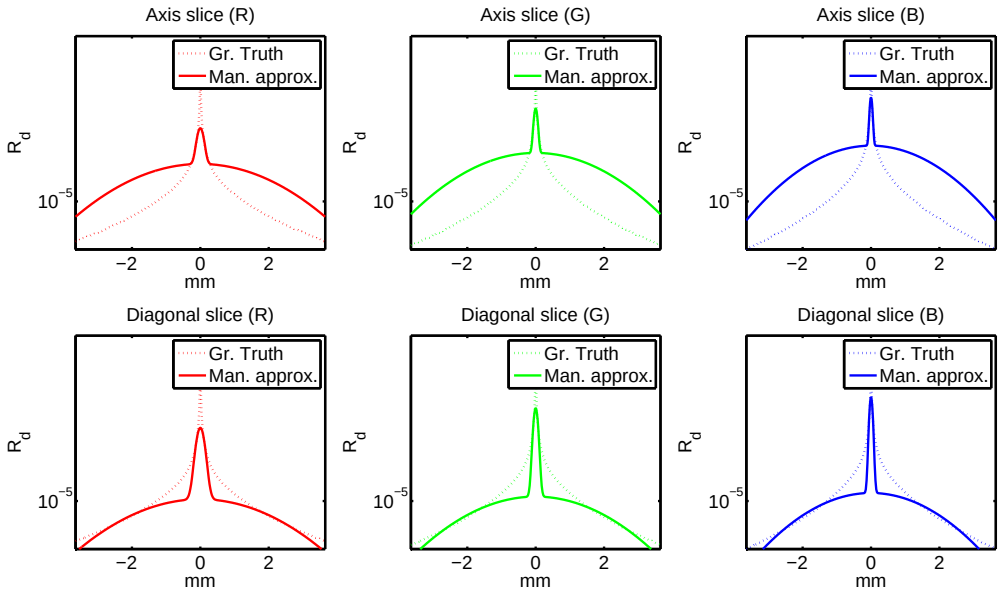


Figure A.44: Plot of the Manual approximation kernel for material Marble.

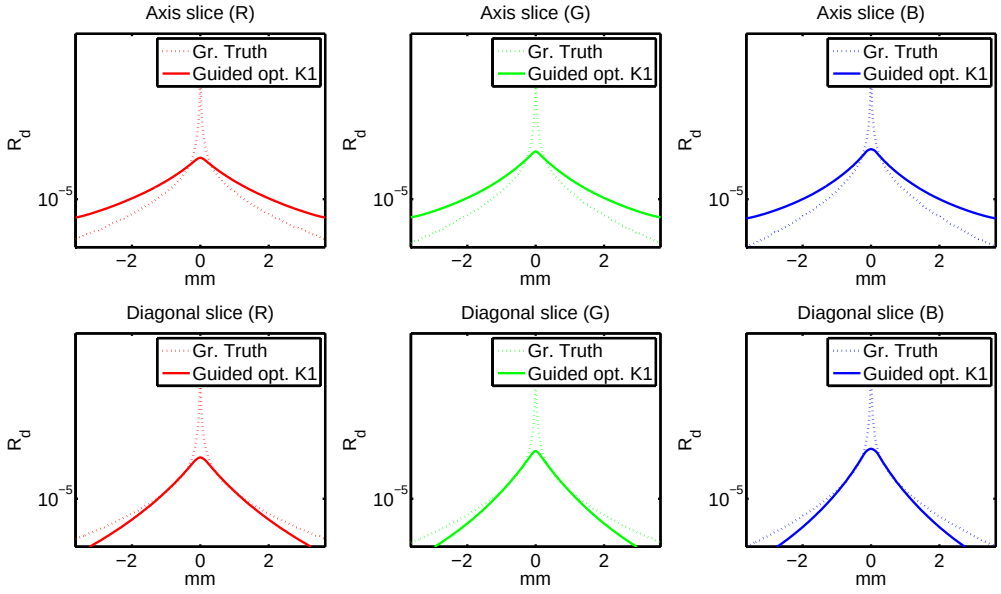


Figure A.45: Plot of the Guided optimization (k=1) kernel for material Marble.

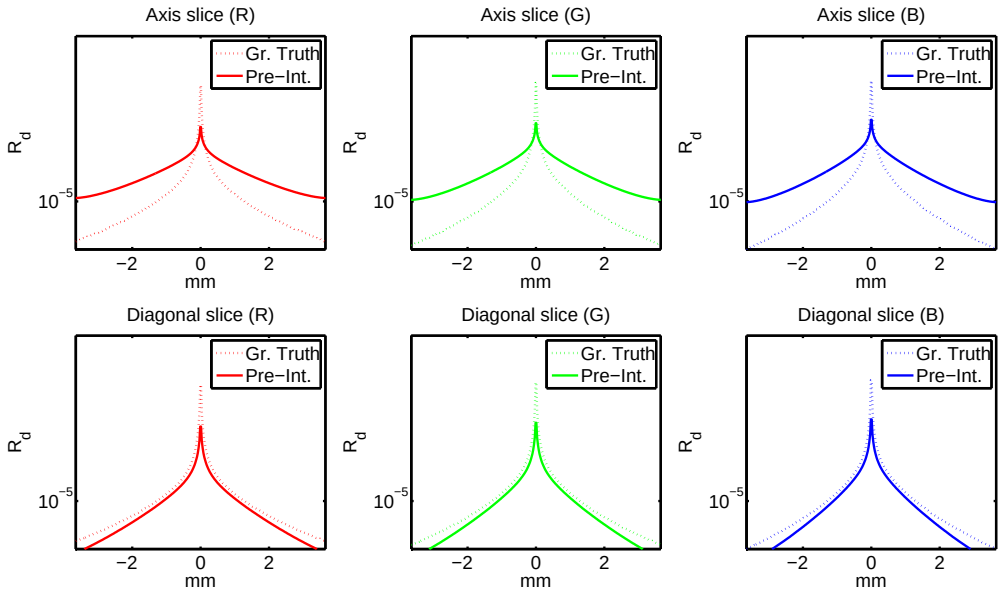


Figure A.46: Plot of the Pre-Integration kernel for material Marble.

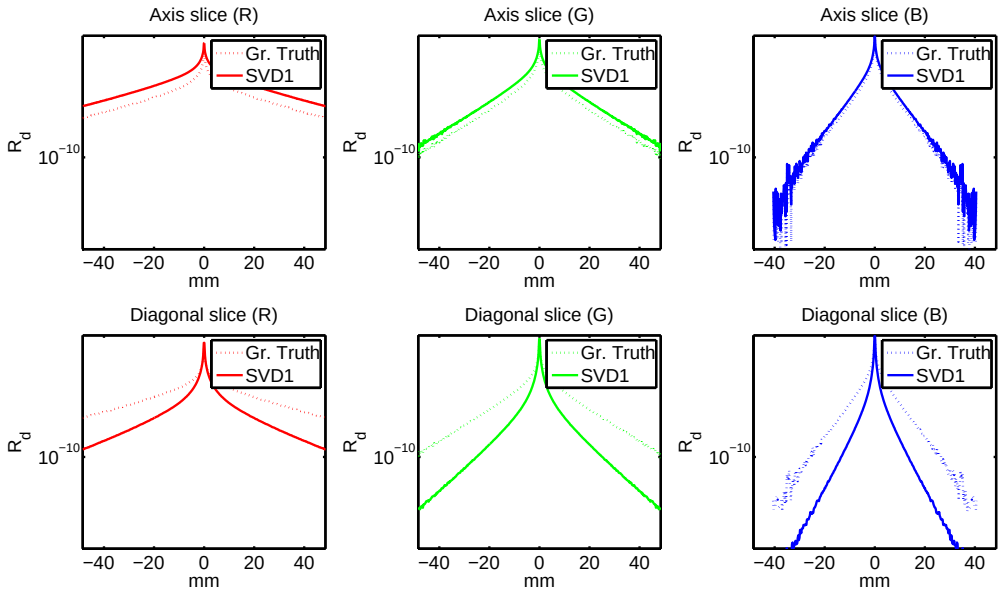


Figure A.47: Plot of the SVD rank-1 kernel for material Chicken1.

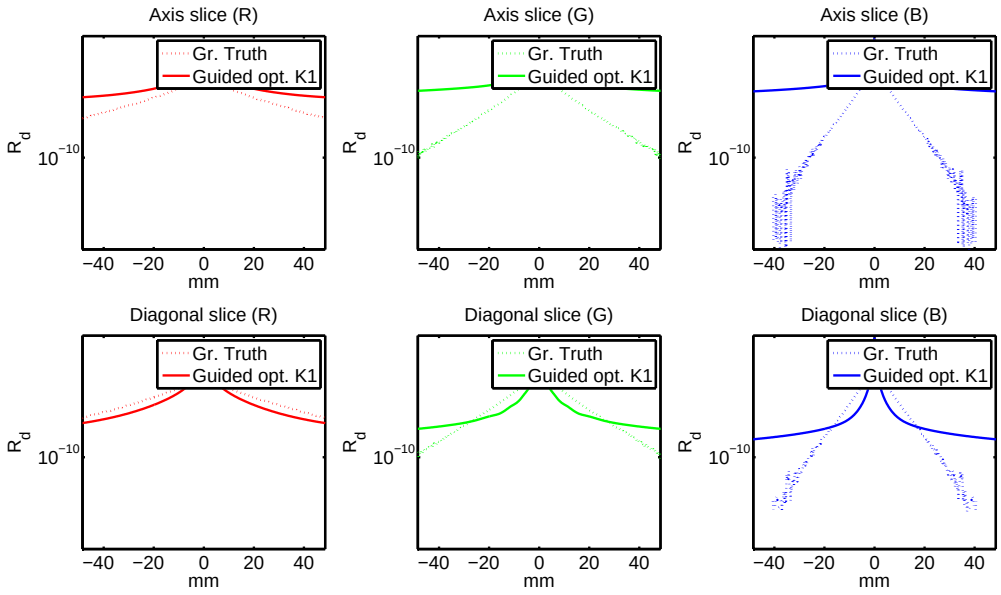


Figure A.48: Plot of the Guided optimization (k=1) kernel for material Chicken1.

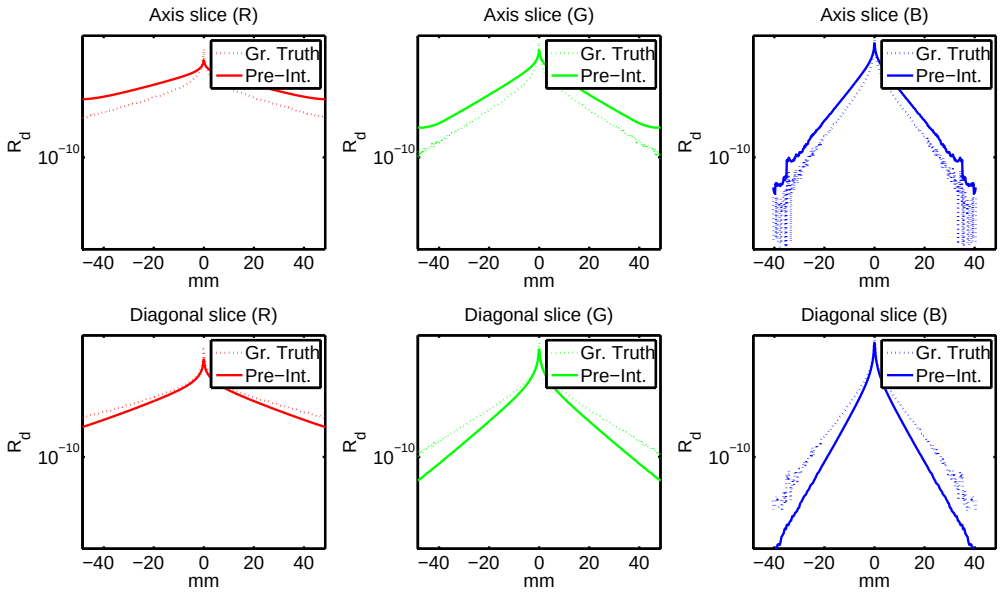


Figure A.49: Plot of the Pre-Integration kernel for material Chicken1.

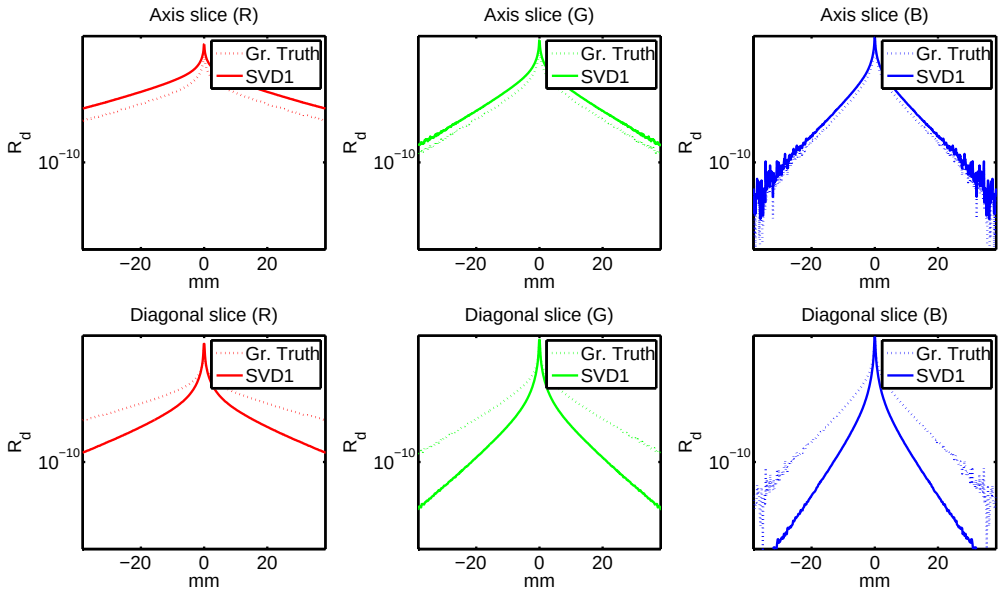


Figure A.50: Plot of the SVD rank-1 kernel for material Chicken2.

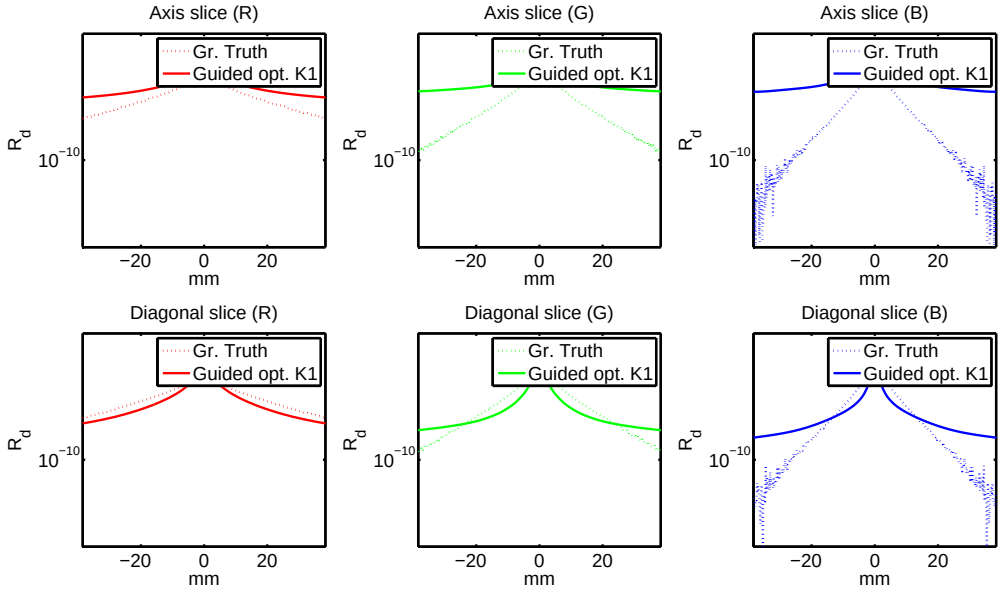


Figure A.51: Plot of the Guided optimization (k=1) kernel for material Chicken2.

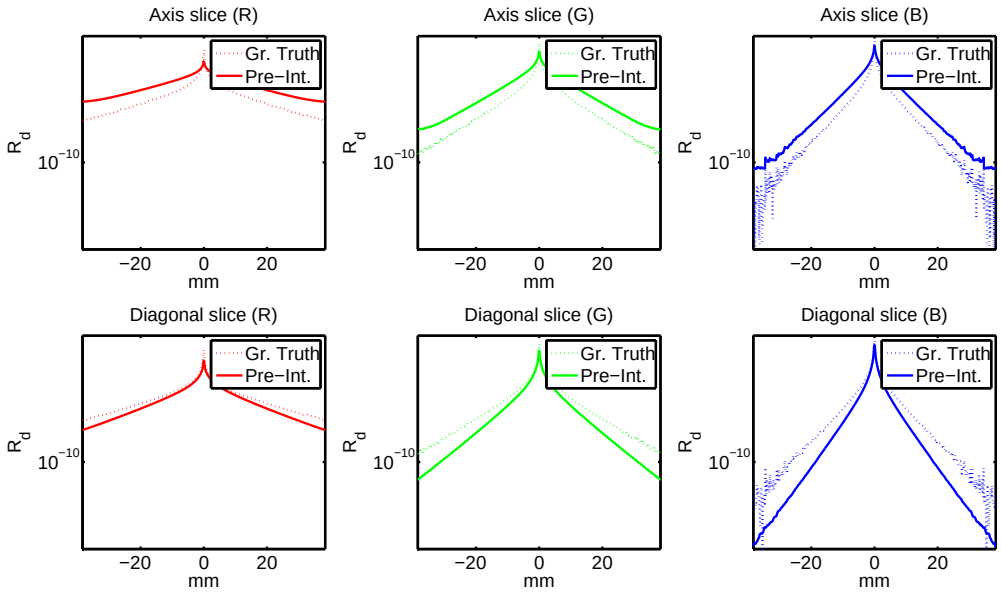


Figure A.52: Plot of the Pre-Integration kernel for material Chicken2.

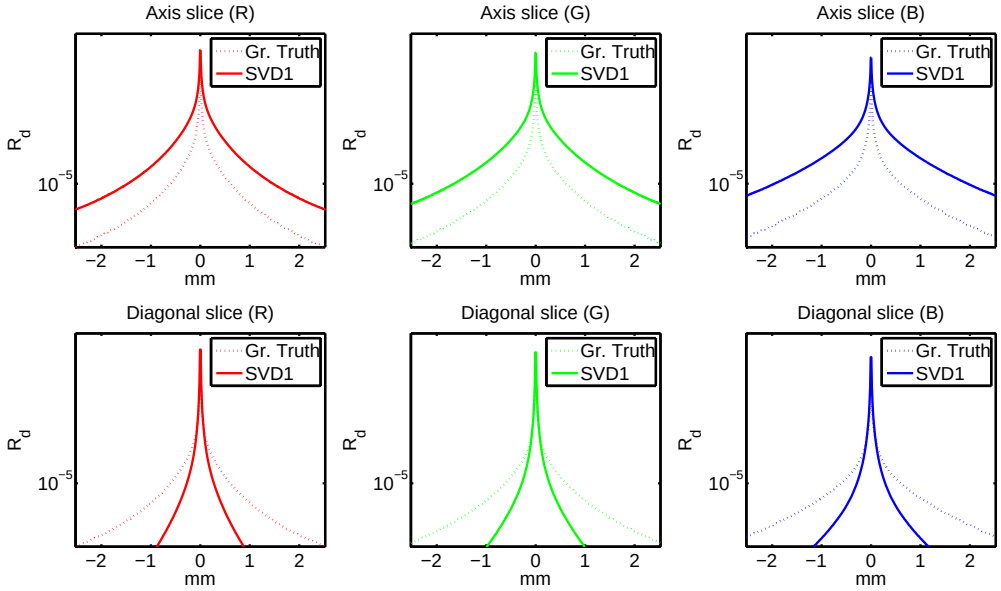


Figure A.53: Plot of the SVD rank-1 kernel for material Cream.

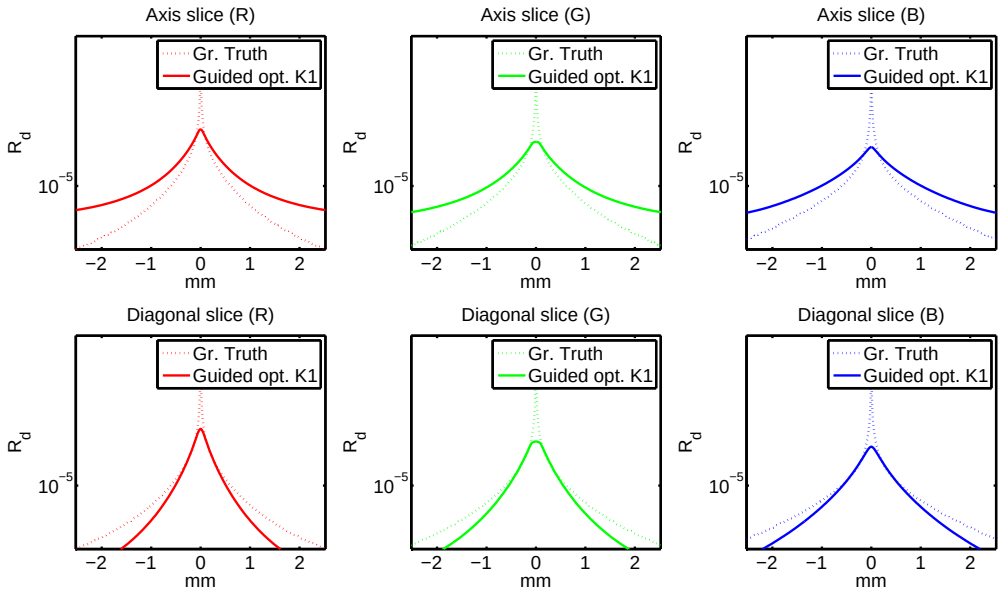


Figure A.54: Plot of the Guided optimization (k=1) kernel for material Cream.

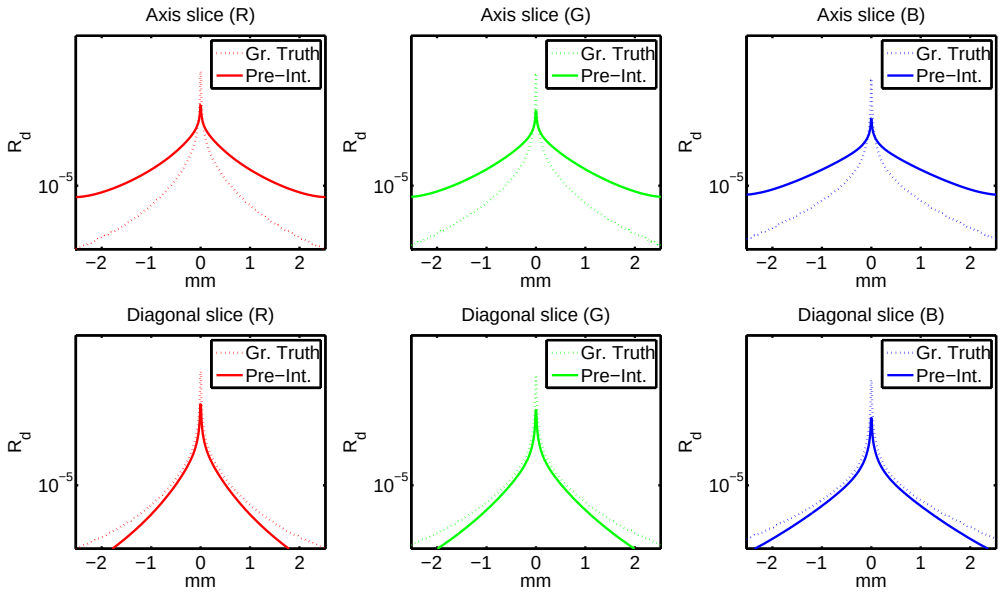


Figure A.55: Plot of the Pre-Integration kernel for material Cream.

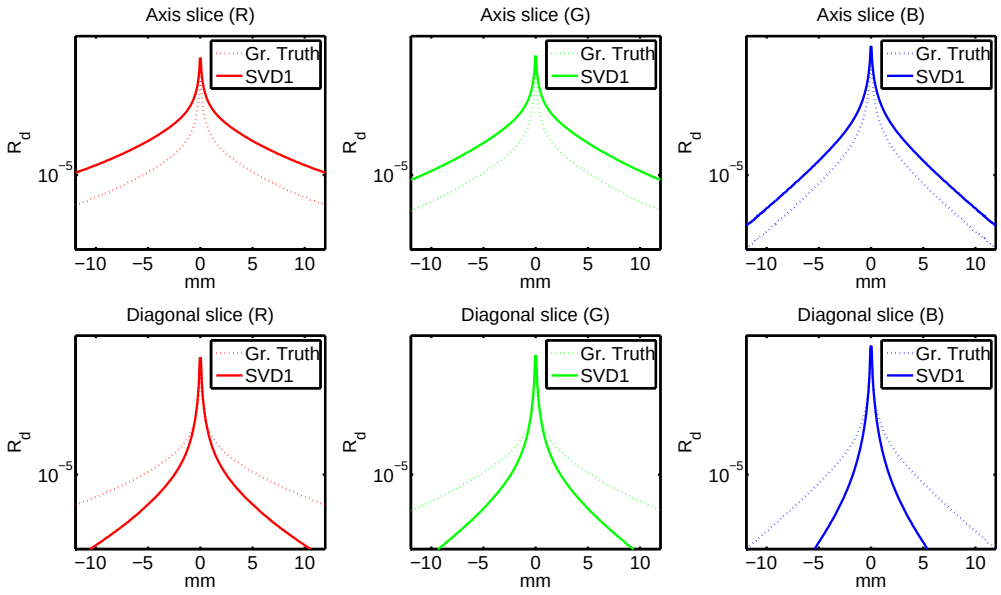


Figure A.56: Plot of the SVD rank-1 kernel for material Potato.

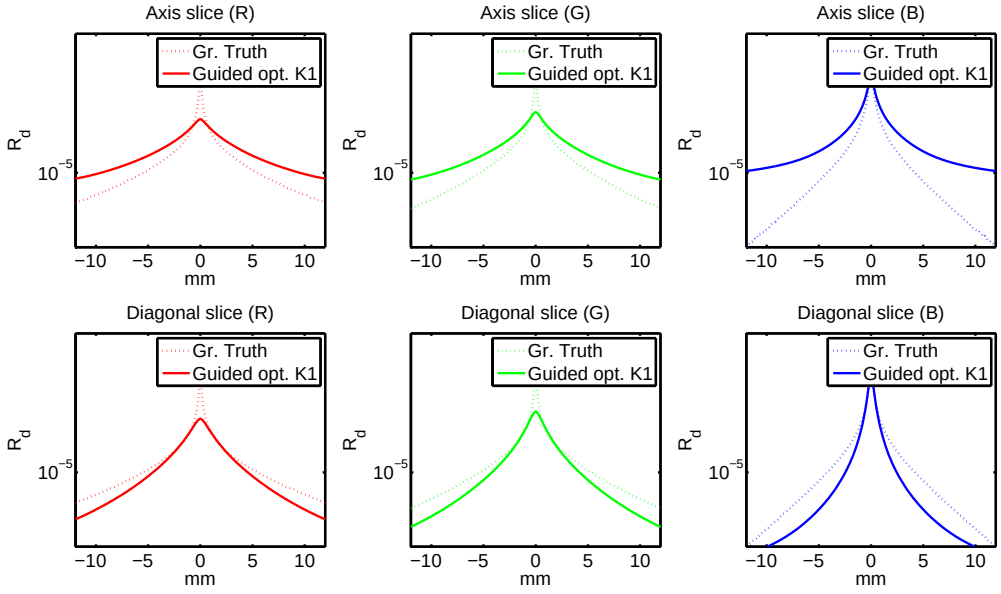


Figure A.57: Plot of the Guided optimization (k=1) kernel for material Potato.

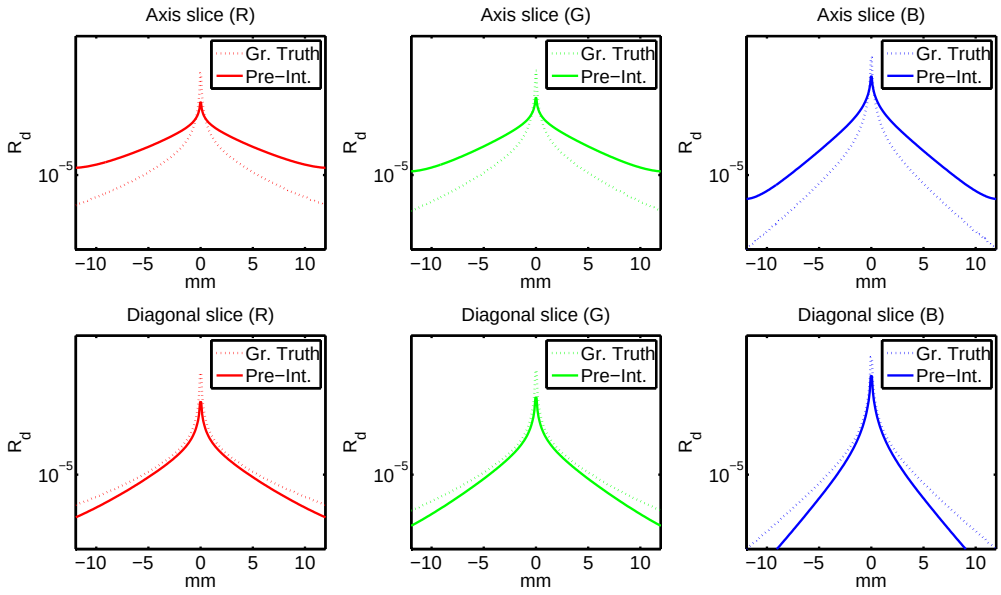


Figure A.58: Plot of the Pre-Integration kernel for material Potato.

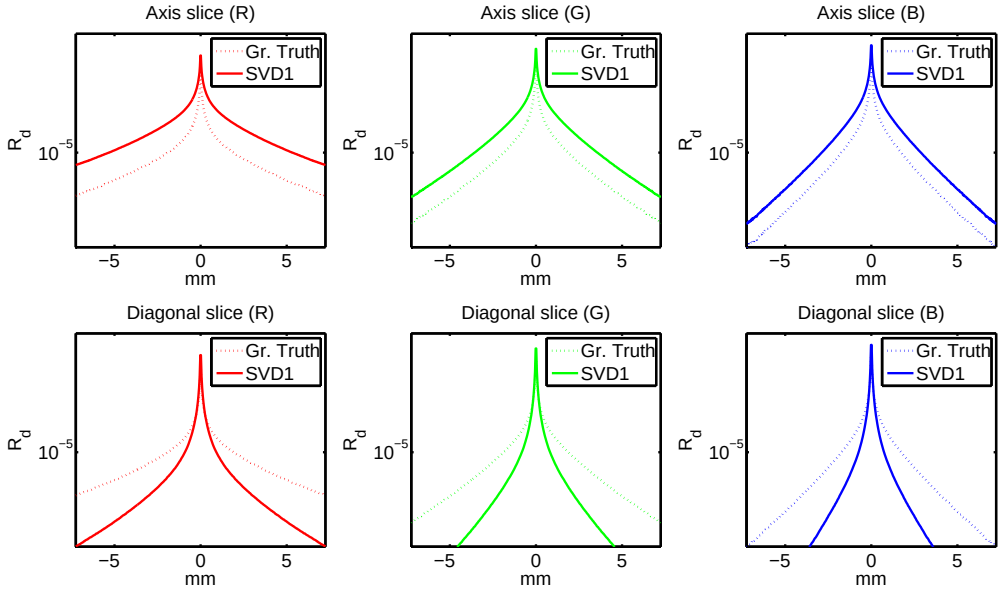


Figure A.59: Plot of the SVD rank-1 kernel for material Skin2.

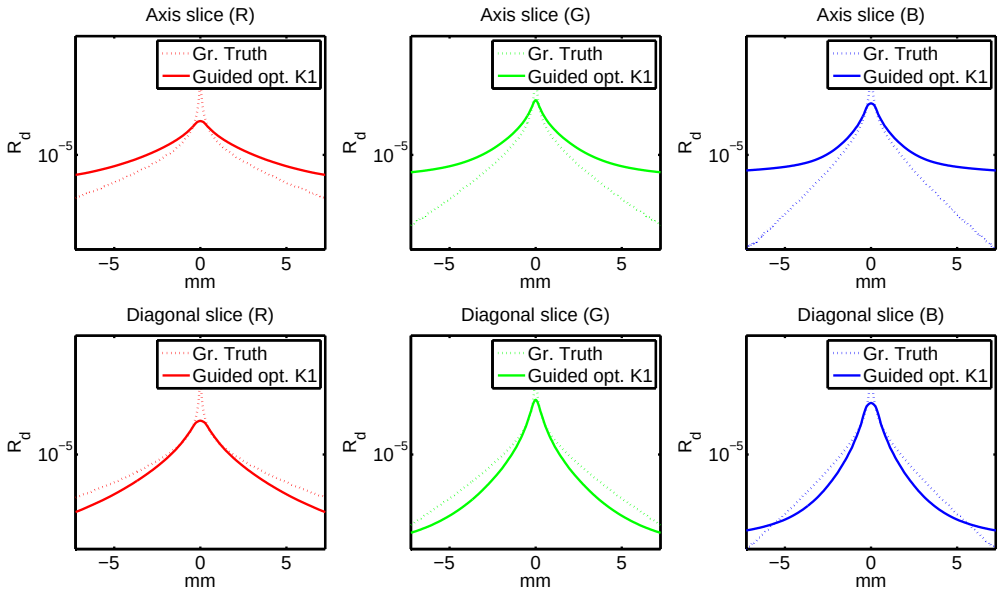


Figure A.60: Plot of the Guided optimization (k=1) kernel for material Skin2.

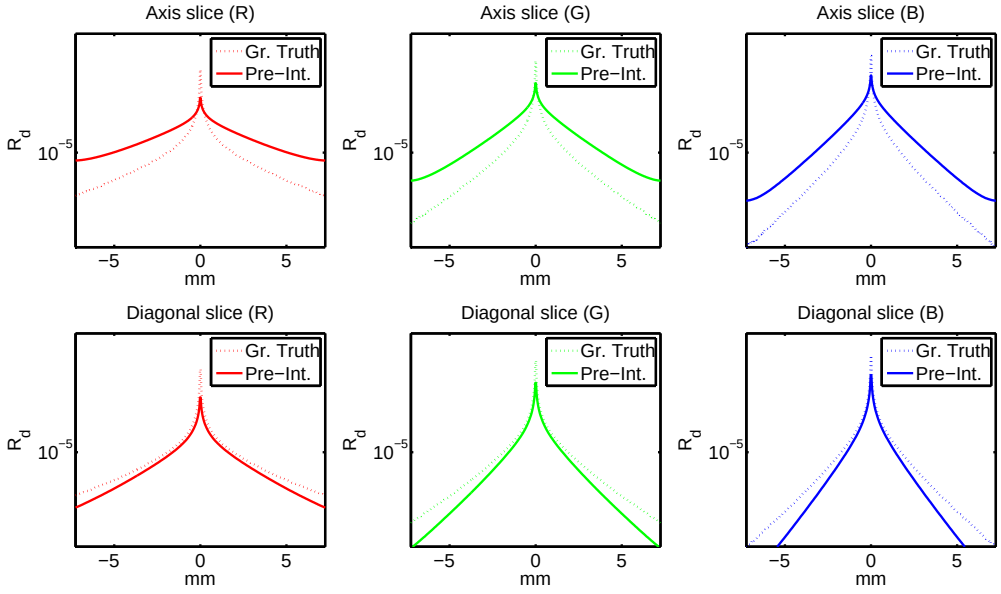


Figure A.61: Plot of the Pre-Integration kernel for material Skin2.

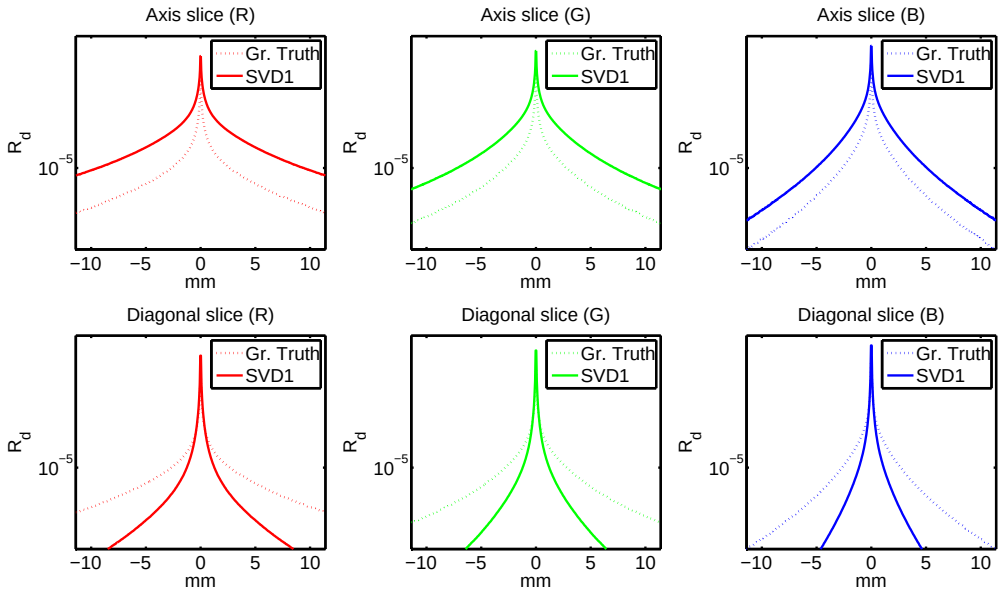


Figure A.62: Plot of the SVD rank-1 kernel for material Skimmilk.

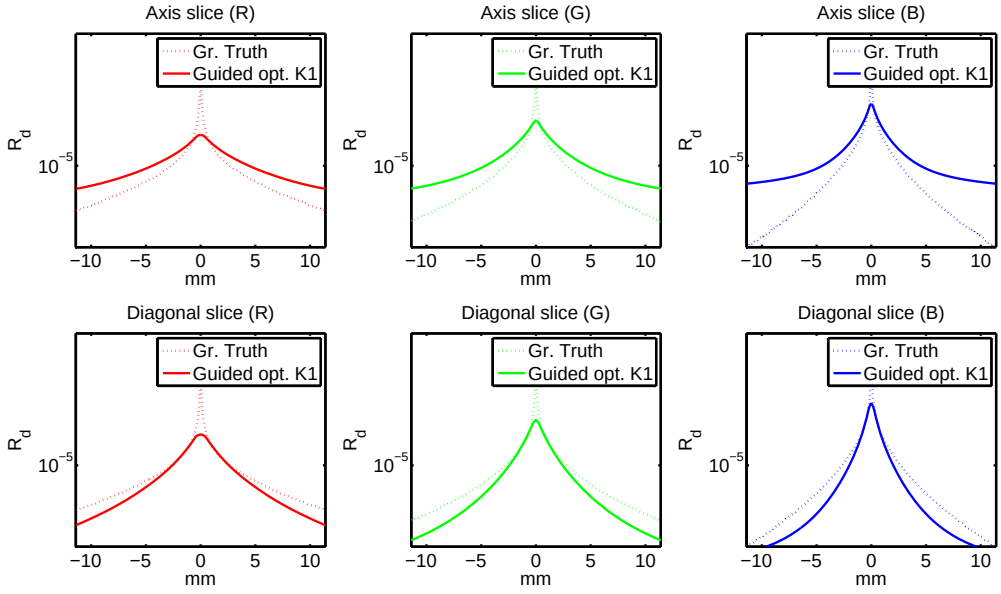


Figure A.63: Plot of the Guided optimization (k=1) kernel for material Skimmilk.

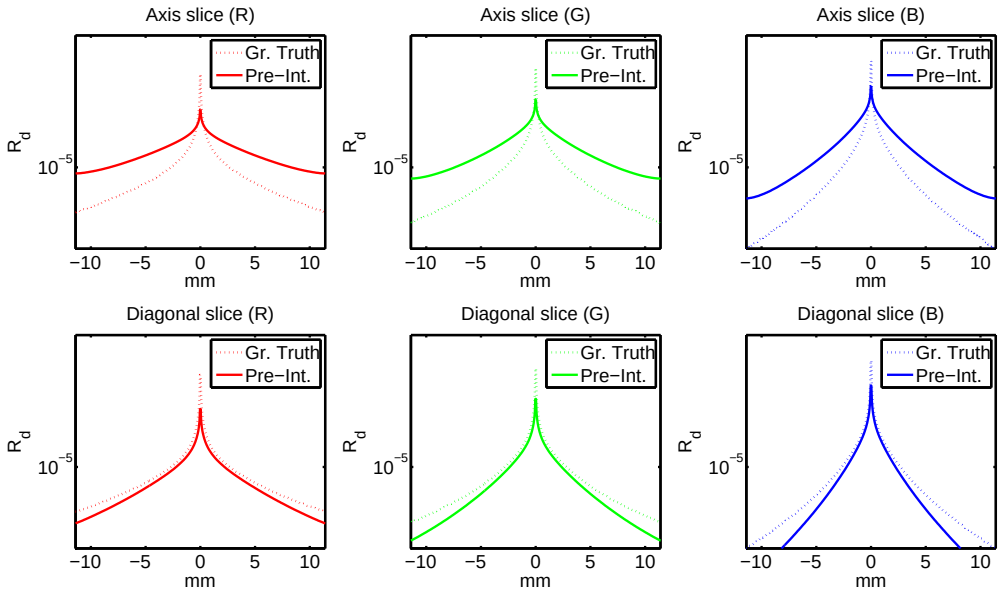
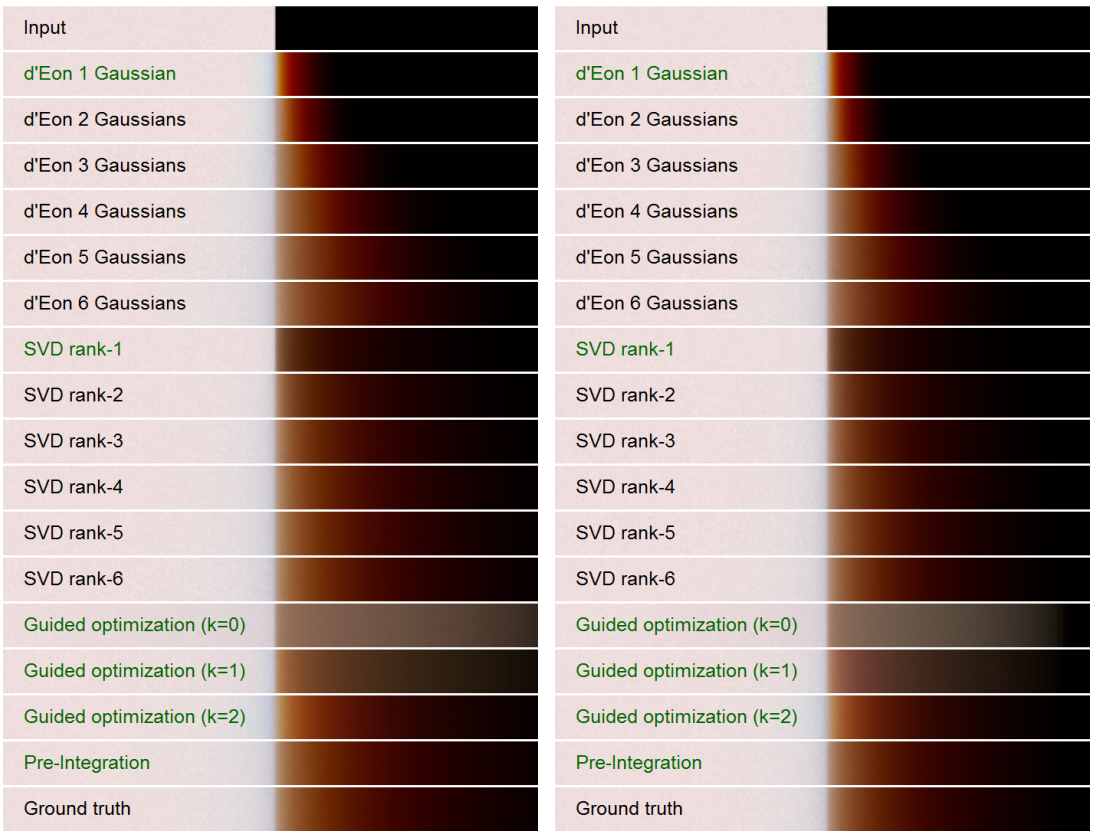
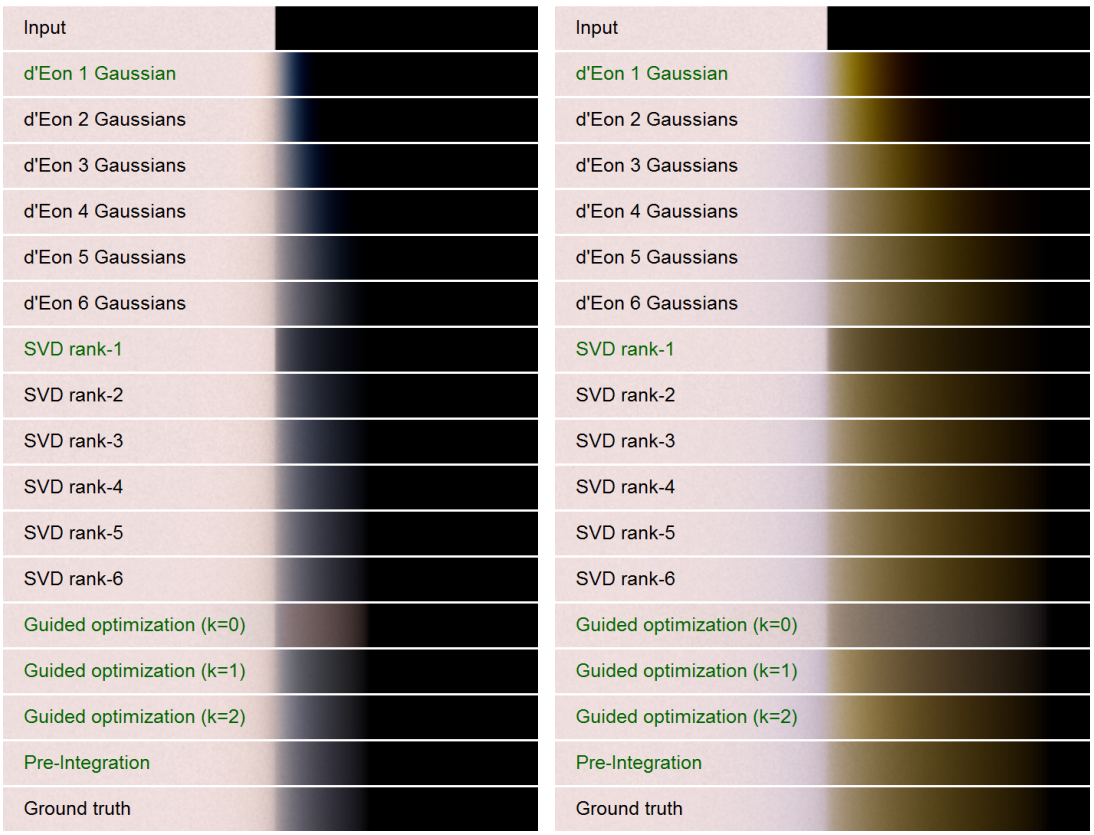


Figure A.64: Plot of the Pre-Integration kernel for material Skimmilk.



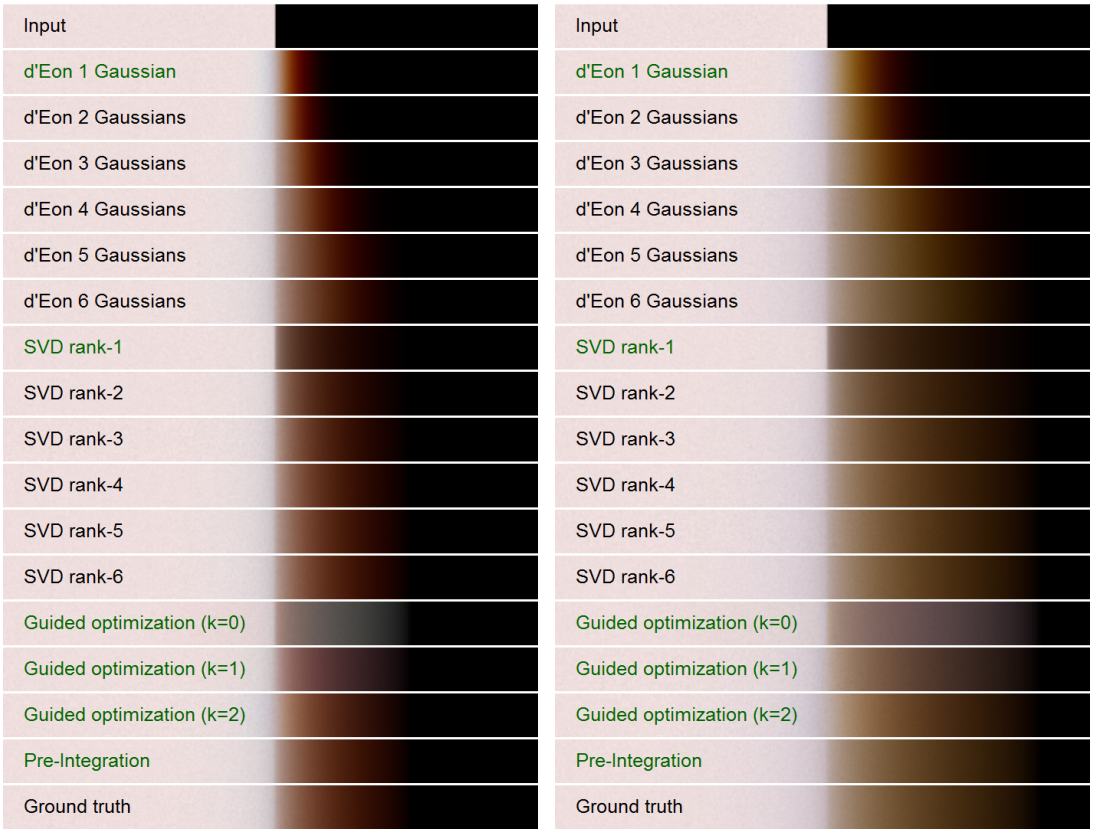
(a) Chicken1 (b) Chicken2

Figure A.65: Overview of the simple 1D test signal convolutions using different kernels for material Chicken1 and Chicken2.



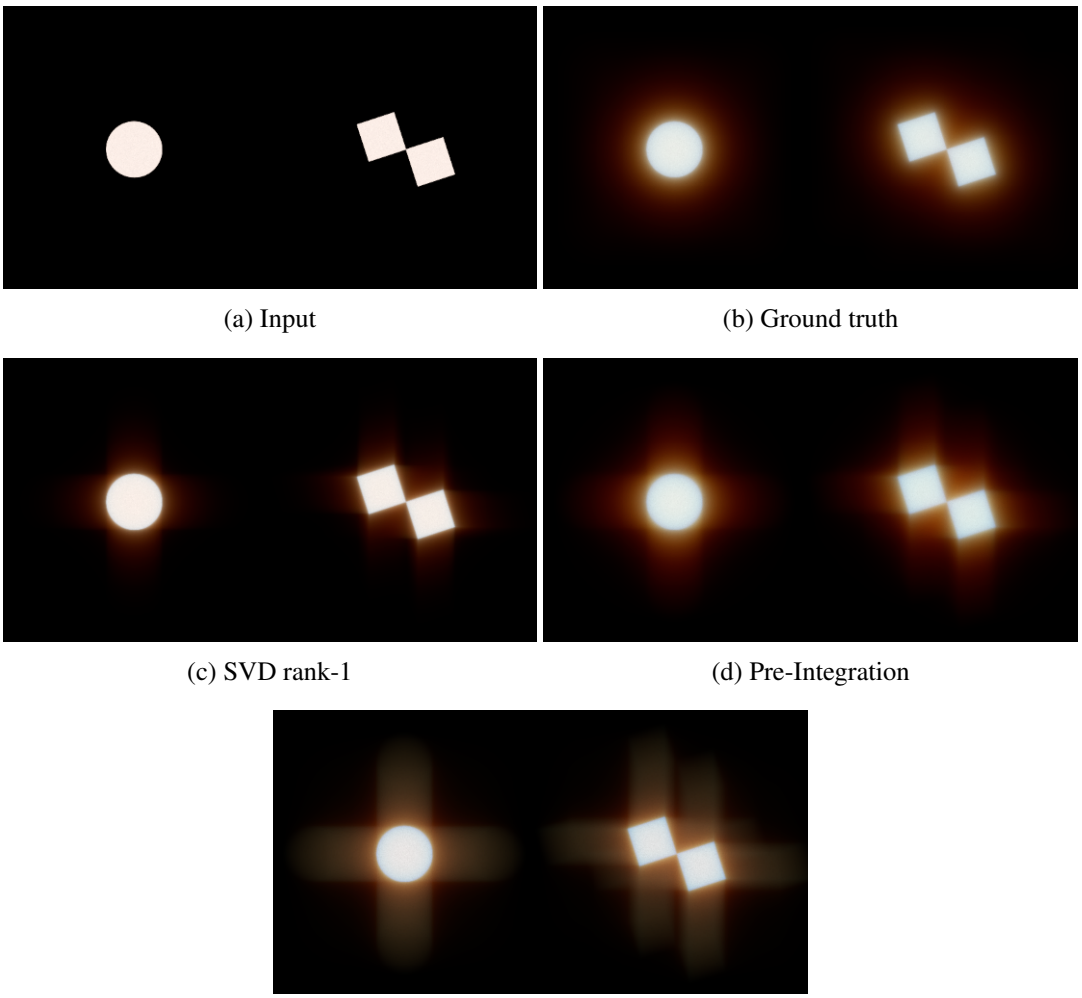
(a) Cream (b) Potato

Figure A.66: Overview of the simple 1D test signal convolutions using different kernels for material Cream and Potato.



(a) Skin2 (b) Skimmilk

Figure A.67: Overview of the simple 1D test signal convolutions using different kernels for material Skin2 and Skimmilk.

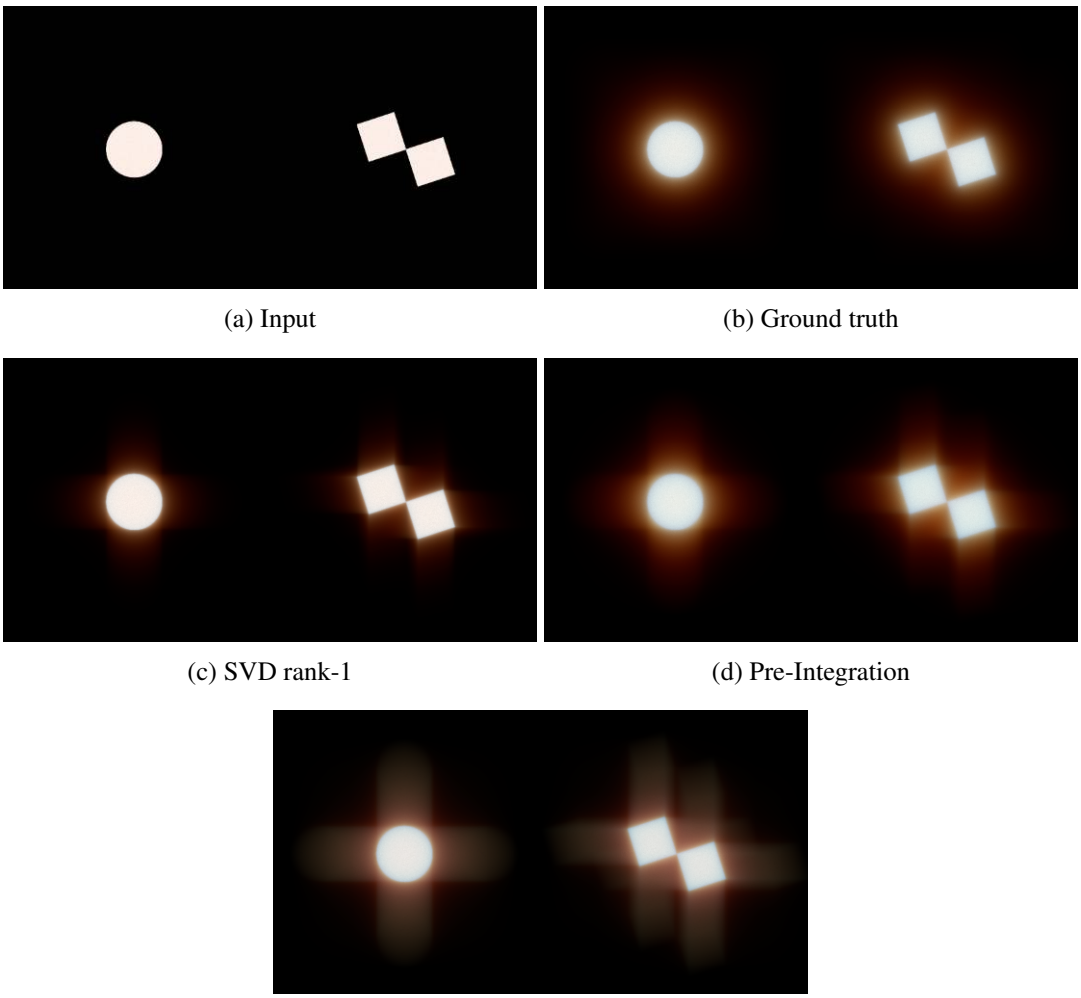


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.68: Convolution of an artificial test signal using different kernels for material Chicken1.

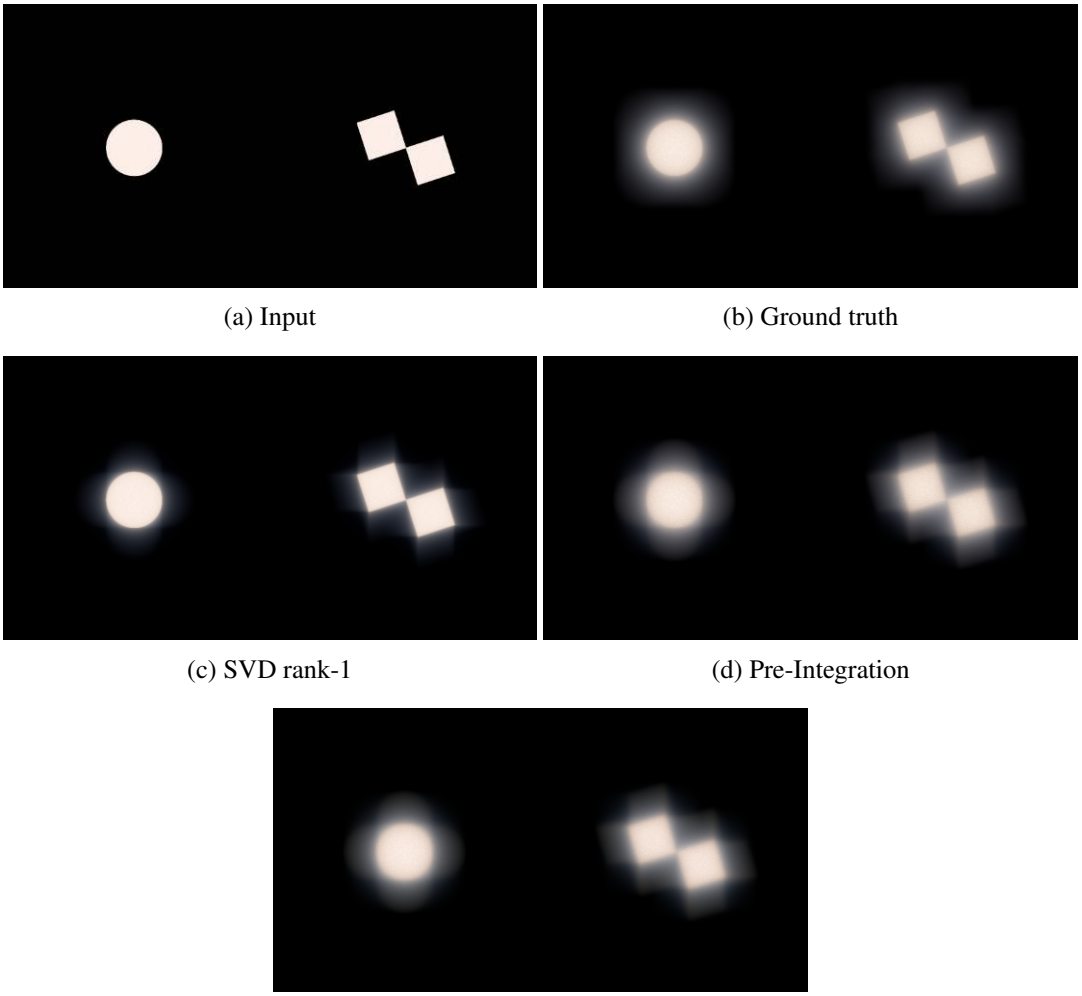


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.69: Convolution of an artificial test signal using different kernels for material Chicken2.

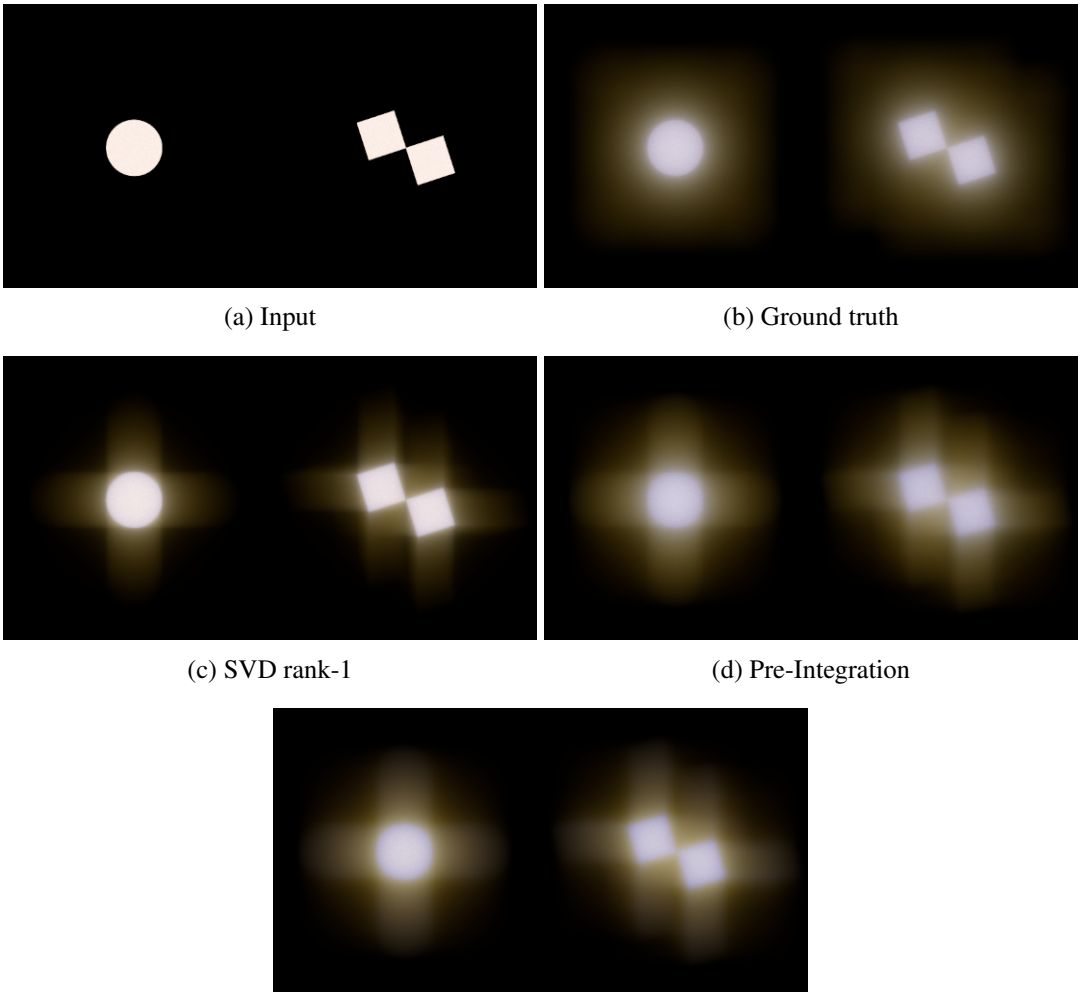


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.70: Convolution of an artificial test signal using different kernels for material Cream.

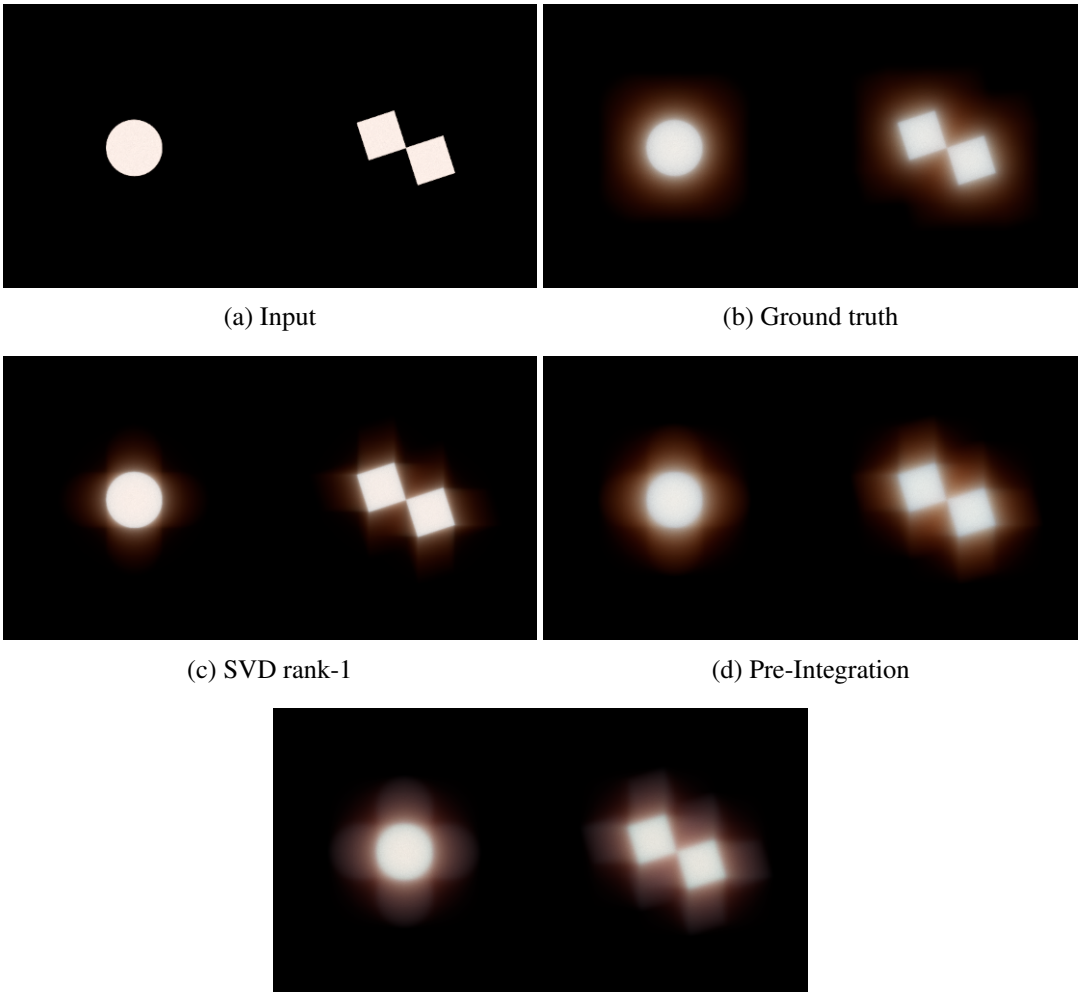


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.71: Convolution of an artificial test signal using different kernels for material Potato.

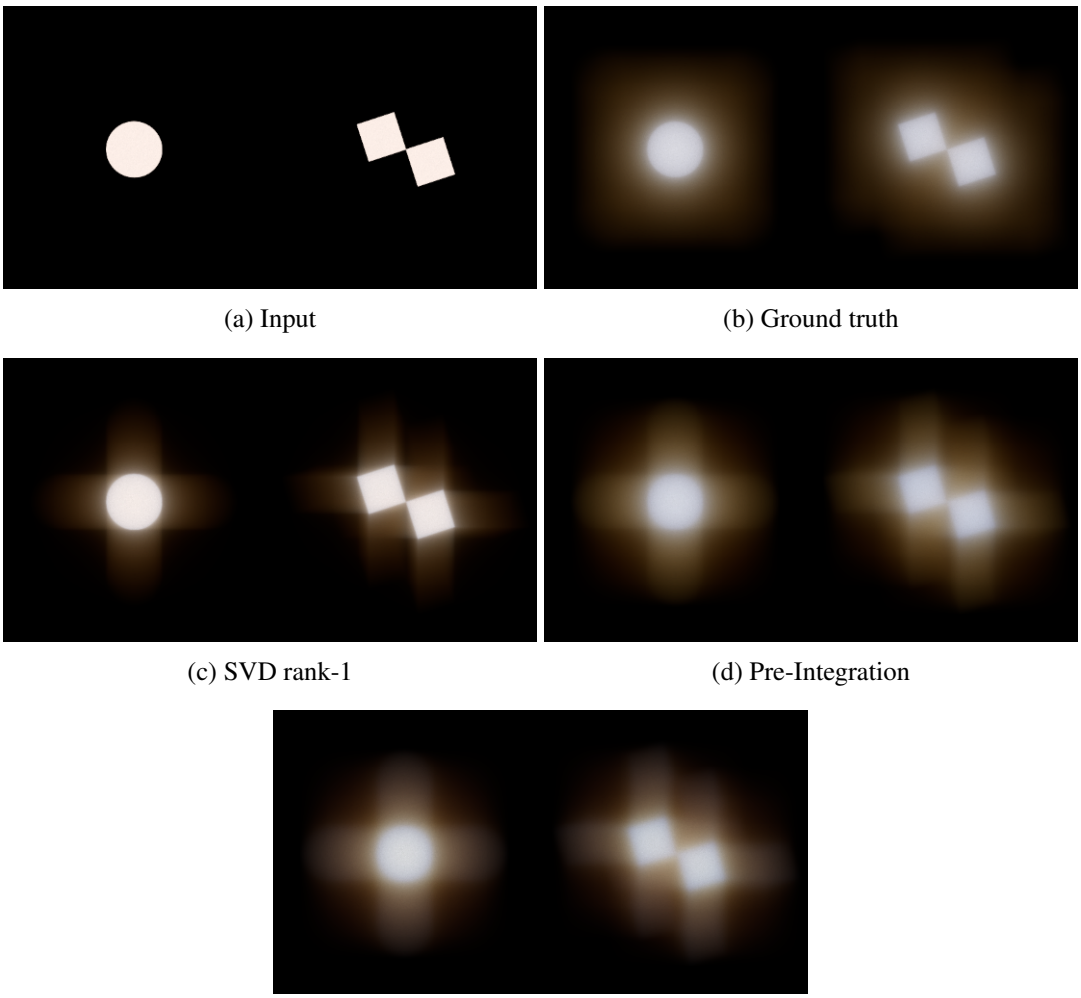


(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.72: Convolution of an artificial test signal using different kernels for material Skin2.



(a) Input (b) Ground truth

(c) SVD rank-1 (d) Pre-Integration

(e) Guided optimization (k=1)

Figure A.73: Convolution of an artificial test signal using different kernels for material Skim-milk.

Scene	Object		Info
Human Head		Source:	Infinite-Realities (included in the original SSSS demo)
		Author:	Lee Perry-Smith
		Link:	`http://www.ir-ltd.net`
Marble Dragon		Source:	Stanford University Computer Graphics Laboratory
		Link:	`http://graphics.stanford.edu/data/3Dscanrep/`
Fruits	Green apple	Source:	Blend Swap
		Author:	metalix
		Link:	`http://www.blendswap.com/blends/view/25355`
	Red apple slices	Source:	CadNav
		Link:	`http://www.cadnav.com/3d-models/model-8246.html`
	Grapes	Source:	Blend Swap
		Author:	PickleJones
		Link:	`http://www.blendswap.com/blends/view/52078`
	Plate	Source:	Blend Swap
		Author:	longrender
		Link:	`http://www.blendswap.com/blends/view/1279`
Plant		Source:	Blend Swap
		Author:	betomo16
		Link:	`http://www.blendswap.com/blends/view/69457`
Ketchup	Plate	Source:	Blend Swap
		Author:	longrender
		Link:	`http://www.blendswap.com/blends/view/1279`
Milk	Fluid	Author:	Károly Zsolnai

Table A.2: List of model authors and sources of various objects used for the test rendering scenes.

Bibliography

[1] Blender Online Community. Blender - a 3d modelling and rendering package, 2014. URL: `http://www.blender.org`.

[2] G. Borshukov and J. P. Lewis. Realistic human face rendering for 'the matrix reloaded'. In *ACM SIGGRAPH 2003 Sketches & Applications*, 2003.

[3] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960.

[4] Visual Computing Lab ISTI CNR. Meshlab. http://meshlab.sourceforge.net/.

[5] E. d'Eon and D. Luebke. Advanced techniques for realistic real-time skin rendering. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 14, pages 293–347. Addison Wesley, 2007.

[6] E. d'Eon, D. Luebke, and E. Enderton. Efficient rendering of human skin. In *Proceedings of Eurographics Symposium on Rendering*, pages 147–157, 2007.

[7] Eugene D'Eon and Geoffrey Irving. A quantized-diffusion model for rendering translucent materials. *ACM Trans. Graph.*, 30(4), 2011.

[8] C. Donner and H. W. Jensen. Light diffusion in multi-layered translucent materials. *ACM Transactions on Graphics*, 24(3):1032–1039, 2005.

[9] Craig Donner and Henrik Wann Jensen. Rendering translucent materials using photon diffusion. In *Proceedings of the Eurographics Symposium on Rendering*, pages 243–252, 2007.

[10] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[11] Josh Evans. King Tut: April 2012, 2012. Accessed July 28, 2014. URL: `http://kingtutdigital.blogspot.co.at/2012_04_01_archive.html`.

[12] David R. Gosselin, Pedro V. Sander, and Jason L. Mitchell. Real-time texture-space skin rendering. *ShaderX3: Advanced Rendering Techniques in DirectX and OpenGL, Engel W., Charles River Media, Inc.*, 2004.

[13] Simon Green. Real-time approximations to subsurface scattering. In Randima Fernando, editor, *GPU Gems*, chapter 16, pages 263–278. Addison-Wesley, 2004.

[14] Ralf Habel, Per H. Christensen, and Wojciech Jarosz. Photon beam diffusion: A hybrid monte carlo method for subsurface scattering. *Computer Graphics Forum (Proceedings of EGSR 2013)*, 32(4), 2013.

[15] Tom Haber. Tom Haber's Homepage: Subsurface Scattering, 2006. Accessed July 28, 2014. URL: `http://research.edm.uhasselt.be/thaber/subsurface.php`.

[16] John Hable, George Borshukov, and Jim Hejl. Fast skin shading. In Wolfgang Engel, editor, *Shader X7*, chapter 2.4, pages 161–173. Charles River Media, 2009.

[17] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the galaxy. *Annales d'Astrophysique*, 93:70–83, January 1941.

[18] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[19] H. W. Jensen and J. Buhler. A rapid hierarchical rendering technique for translucent materials. *ACM Transactions on Graphics*, 21(3):576–581, 2002.

[20] Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 21–30, London, UK, UK, 1996. Springer-Verlag.

[21] H.W. Jensen, S.R. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. In *Proceedings of ACM SIGGRAPH 2001*, pages 511–518, 2001.

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

[23] Jorge Jimenez. Separable SSS, 2012. Accessed August 14, 2014. URL: `http://www.iryoku.com/separable-sss-released`.

[24] Jorge Jimenez and Diego Gutierrez. *GPU Pro: Advanced Rendering Techniques*, chapter Screen-Space Subsurface Scattering, pages 335–351. AK Peters Ltd., 2010.

[25] Jorge Jimenez and Diego Gutierrez. Separable Subsurface Scattering, 2011. Accessed August 14, 2014. URL: `https://github.com/iryoku/separable-sss`.

[26] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[27] Christopher Kulla and Marcos Fajardo. Importance sampling techniques for path tracing in participating media. *Comp. Graph. Forum*, 31(4):1519–1528, June 2012.

[28] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Compugraphics '93*, pages 145–153, 1993.

[29] Dongping Li, Xin Sun, Zhong Ren, Stephen Lin, Yiying Tong, Baining Guo, and Kun Zhou. Transcut: Interactive rendering of translucent cutouts. *Visualization and Computer Graphics, IEEE Transactions on*, 19(3):484–494, 2013.

[30] T. Mertens, J. Kautz, P. Bekaert, F. V. Reeth, and H. P. Seidel. Efficient rendering of local subsurface scattering. *Computer Graphics Forum*, 24(1):41–50, 2005.

[31] Morten Mikkelsen. Skin rendering by pseudo-separable cross bilateral filtering. Technical report, Naughty Dog Inc., August 2010.

[32] Matthew Muldoon and Jonathan Acosta. Blend Swap, 2014. Accessed August 17, 2014. URL: `http://www.blendswap.com/`.

[33] Eric Penner and George Borshukov. *GPU Pro 2*, chapter Pre-Integrated Skin Shading, pages 41–55. AK Peters Ltd., 2011.

[34] M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann. Morgan Kaufmann/Elsevier, 2010.

[35] Musawir A. Shah, Jaakko Konttinen, and Sumanta Pattanaik. Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE Computer Graphics and Applications*, 29:66–78, 2009.

[36] Jos Stam. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop*, pages 41–50, 1995.

[37] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[38] Jiaping Wang, Shuang Zhao, Xin Tong, Stephen Lin, Zhouchen Lin, Yue Dong, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of heterogeneous translucent materials using the diffusion equation. *ACM Trans. Graph.*, 27(1):9:1–9:18, March 2008.

[39] 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.

[40] Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes. In *Computer Graphics Forum*, volume 29, pages 497–506. Wiley Online Library, 2010.