Rendering Filters for Controlling Detail and Creating Effects

Christopher Robert DeCoro

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

By the Department of

Computer Science

Advisor: Szymon Rusinkiewicz

June, 2009

Abstract

This thesis demonstrates the way in which various methods for controlling detail and creating effects in computer graphics may be unified under the general theme of the *rendering filter*. Generally stated, such a filter is a passive, stateless operator that acts upon a decomposition of terms in the rendering equation. In the first Part, we present background that motivates this concept, and provides an understanding of the way in which the rendering filter follows logically from existing use in other domains. First, in Chapter 1, we discuss the general and historical use of the term "filter," especially as a useful metaphor that encapsulates various similar operations. We present examples of filters in photography, electronics, imaging and geometry processing. In Chapter 2, we provide background specific to rendering in graphics, examine the process of rendering as inherently related to filtering, and define the rendering filter itself.

In the second Part, we see the application of these concepts by three specific examples of rendering filters. In addition to demonstrating the utility of the methods themselves, we show how these distinct algorithms are unified by the underlying rendering filter framework. In Chapter 3, we show various ways in which artists use "abstract," or otherwise unrealistic shadows to achieve compositional purposes; to allow similar control in real-time graphics, we present the *stylized shadow filter*. This filter accepts an intuitive set of controls, with which it converts an accurate shadow into a stylized shadow, and simulates many of the same artistic effects. Importantly, we present an algorithmic framework sufficiently efficient for real-time, interactive rendering. In Chapter 4, we present the *subtractive shadow filter*, which derives from the principle that rather than adding unshadowed light contributions from multiple lights in the scene to produce a shadowed result, we rather subtract extraneous light from an unshadowed rendering. By doing so, we enable user control over shadow level-of-detail in a progressive and flexible manner, so as to achieve a time/quality tradeoff over rendering, analogous to existing filtering techniques for other domains. In Chapter 5, we focus on improving the quality of realistic light transport through the *path-density filter*, which removes outlier points that degrade the quality of the scene. Their removal leads to significant improvement in rendering quality while retaining plausibility of the result. These particular filters serve as specific examples of rendering filter in practice, but by no means constitute the limit of what may be achieved according to this framework. It is hoped that by introducing these concepts we may stimulate the subsequent development of many additional methods.

Acknowledgments

The author was supported in the 2007-2008 school year by an AMD/ATI Technologies Research Fellowship; and he would like to thank them, and in particular his supervisor at ATI, Natalya Tatarchuk, additionally for the invaluable experience of a research internship in Summer 2006. Partial support for this work was also provided by the PICASSO Fellowship program at Princeton University, and by NSF Grants CCF-0347427 and IIS-0511965.

Datasets, meshes and images have been provided by the following, and the author gratefully acknowledges their contributions: the Blender Foundation / Netherlands Art Institute (Elephant's Dream still images), Weather Underground (Trenton Airport temperature measurements), Markus Bernet (dandelion image), Colin Burnett (brick wall image), Matt Pharr and Greg Humphreys (killeroo scene), 3D Cafe (octopus model), the Digital Michelangelo Project (David model) and the Stanford Computer Graphics Lab (bunny and dragon models), AIM Shape (heptoroid, trellis, filigree, Neptune and hippo model), Marko Dabrovic (Sponza Atrium model), Paul Debevec (Grace Cathedral, Eucalyptus Grove, St. Peter's Basilica environment maps), Cyberware (horse model), and DeEspona (floor and wall mirror models).

Finally, the motivational environment created by many members of the Princeton Computer Science Department has been invaluable, as has all of their input and feedback over these last six years. The author would like to especially thank the members of his thesis committee, Tom Funkhouser, Brian Kernighan, Fei-Fei Li, and Tim Weyrich, as well as Connelly Barnes, Benedict Brown, Michael Burns, Forrester Cole, Eden Chlamtac, Adam Finkelstein, Alexey Golovinsky, Diego Nehab, Joshua Podolak, Phillip Shilane and my advisor, Szymon Rusinkiewicz.

This product of my time at Princeton is dedicated to the friends whom I have made here; each of whom, in their own way, has in turn made me into that which I am:

Christian Bienia, Zafer Barutçuoğlu, Leona Qi, Kareen Rozen, Kira Hohensee, Annie Nalbandian, Berk Kapıcıŏglu, Elizabeth Ortiz and Juliane Prade.

Should ne'er again our paths may cross,
To all, fore'er am I in debt.

Contents

Part I

Foundations and Motivations of Rendering Filters

Chapter 1

The Filter as Convenient Metaphor

In this thesis we present three novel computer graphics algorithms that enable both control over rendered detail and the creation of stylistic effects. While these may appear at first glance to be unconnected, we suggest that they can all be considered specific examples of *rendering filters*, and that this concept provides a coherent framework that unifies a wide range of computer graphics algorithms. Our primary goal is to introduce the metaphor of the rendering filter by recognizing the connection between these algorithms and the general concept of *filtering* as traditionally employed in many fields. We may then apply many of the same techniques and extensive background to the rendering filter.

Many of the metaphors used in computer graphics – cameras, filters, lights, etc. – have natural correlates in physical photography, and therefore a long history on which to draw upon for understanding. Resultingly, it is useful to first consider the development of these terms in photography and depiction, as we do in the next section. We then continue our discussion with the generalization of the term to analog electronics and signal processing in Section 1.2, and to digital signal processing in Section 1.3, during the course of which we provide a discussion of the mathematical basis of these methods. This leads to the further application of filtering to computer graphics applications in Section 1.4. Subsequently, the next chapter will discuss the use of filtering specific to rendering.

1.1 Filters in traditional photography

The term camera as used in photography derives from the *camera obscura* (Latin for *veiled room*), in which a small opening (or *aperture*) is made in the side of an otherwise darkened room, causing an image of the view through the aperture to be projected on the opposite wall. Well-known since the Renaissance, the camera obscura is diagrammed in the notes of Leonardo da Vinci (see [51], pp. 180-183 for an accessible edition). The first widespread method to permanently preserve the resulting image was introduced by Louis Daguerre in 1839. The *daguerreotype*, as images produced by his method came to be known, consists of a polished silver plate coated with silver iodide. The photosensitive silver iodide tends to form atomic silver when exposed to light; this effect is more pronounced in regions with greater light. By *development* of this plate with mercury vapor, an image becomes visible [29].

As photography became more widely used, other sources produced alternate photographic plates to those of Daguerre. Plates made of varying compositions of photosensitive materials correspondingly display varying sensitivity to light. The result, therefore, was that two photographs taken of the same scene at the same time, but with different makes of photographic plates, would show differences in color and relative level of exposure. To compensate, most manufacturers of photographic plates would also supply "filters" – transparent colored discs that attach to the lens – specially adapted to their plates (see [98] pg. 335). Similarly, filters allow the photographer to compensate for the ambient colors in the environment. For example, the left image of Figure 1.1 was taken under incandescent tungsten lighting, which has an overall amber-red hue. Were the photographer to use a #80A Cooling Filter over the lens, the filter would disproportionately remove red frequencies so as to correct the hue [28]. The inverse of the #80A is the #85 Filter, which makes a scene taken in bluish daylight appear to be in artificial light. These numbers given are part of the *Wratten number* standard, promoted by Eastman KODAK Company; in addition to their Wratten number, filters are also specified in terms of their density (60% in the figure). The use of a standard numbering allowed a photographer to easily select and use the particular filter for the task. In many ways, then, this allowed the concept of a photographic filter to have a natural relation with that of a liquid filter – particular filters have specific properties in terms of the objects they block, and those that they allow to pass through.

Original, Tungsten Lighting Color-Corrected with #80A Filter

Figure 1.1: **Color Correction Using Photographic Filters.** Uncorrected tungsten lighting lends an artificial amber-red tone to objects in the photograph that would be perceived as white by a viewer in the scene. The use of a #80A Cooling Filter (60% density), which is specifically designed for this purpose, corrects to display more neutral tones. (Note specifically that the image has been altered using the #80A Filter in Adobe Photoshop CS3).

The traditional definition of filter, the only one which early photographers would have been familiar with, is that of an object used to free a fluid from particles held in suspension; the term originates from an alternate spelling of "felt," which was often employed for such purposes. The usage in photography is the first known generalization to an alternative definition, according to *The Oxford English Dictionary*, around 1900: to separate specific frequency bands of light from a wider spectrum [68].

As the art of photography developed, practitioners found use of filters for stylistic reasons, in addition to correctness. An essential aspect of the cinematographer's role in movie production is setting the tone and hue of the images so as to achieve a particular look and feel [69]. This may have a significant impact on the perception of the images by the audience, and their reaction to it. Images with dominant red hues are considered "warmer", and evoke a different response from those "cooler" images with dominant blue or green hues. Particular photographic filters would be used and changed as needed to reflect the director's intentions.

As a specific example of this, consider the still frames from the short movie *Elephant's Dream*[1] shown in Figure 1.2. The movie focuses around two characters: the boy and old

[1] *Elephant's Dream* was produced by the Blender Foundation and the Netherlands Art Institute, and has been released under a Creative Commons License that permits its use here.

Red Colors / Warmer Mood Green Colors / Colder Mood

Figure 1.2: **Stylistic Filtering for Compositional Purposes.** In the left scene, while the focus of the plot is on the boy, the color hue is red (evoking a warmer mood), reflecting the character. When (approximately a second later) the old man becomes dominant, the overall hue switches to his own, colder, greens. In this way, filtering has been used to convey additional information from the cinematographer to the viewer.

man shown in the images. The director makes strong use of hue to differentiate the characters, and switches back and forth to set the mood of the scene, according to the character in the focus of the plot at that moment. The frames shown are taken a second apart. While the boy is talking (left) the reddish hue is designed to evoke warmth and calm, reflecting the character. The old man grows angry (right), and in response presses a button so as to explicitly change the dominant hue to his own colder greens, representing a change of focus on the characters, as well as a change of mood in the scene. While this film makes the connection between hue and plot explicit, this concept is present in subtler forms in a wide range of films [80]. We demonstrate similar stylistic filters for compositional effect later in this work.

Another common physical filter is the use of the *soft-focus* lens, in which the camera forms blurred, or "softer," images. Originally, soft focus was the result of imperfections in the surface of the lens that prevent light rays from converging at a focus point. However, in certain photographic applications, this may be desirable, such as in removing imperfections and creating a more surrealistic image. Photographic filters have been developed to achieve this effect artificially; for example, the popular Dallmeyer-Bergheim stigmatic lens has been used for adjustable soft focus since the late 19th Century [98]. In addition to commercial filters, this effect can be approximated with such home-made solutions as petroleum jelly coated across the camera lens [34].

Contrasting with the soft-focus, the *unsharp mask* filter increases apparent sharpness in the image. The filter derives its name from a blurred version of the image that is used to partially block light when the image is developed – the eponymous mask. The use of the unsharp mask blocks similar low-frequency features more so than high frequencies, causing an increase of apparent sharpness [33].

Since the development of digital photography, it became possible to replicate these effects in the computer, as well as achieving additional effects that are difficult or impossible to produce with traditional physical filters. Our work in this thesis revolves around extending filter concepts to other problem domains. To understand such filtering methods, it is important to understand the mathematical foundations of filtering as developed in the field of signal processing, which we discuss in the next section.

1.2 Filters in signal processing

Much of the formal concepts applied to filter design were developed in the context of electrical and electronic engineering, and we proceed with a discussion thereof. Within several decades after the use of "filter" as an object that removes certain frequencies of light, a similar definition became applied to electronic circuits. In particular, filtering in both the electrical and photographic contexts involves operations in frequency space. Specifically, an electronic filter is defined as a passive circuit that attenuates signals according to their frequency bands [68].

The properties of such circuits can be formally studied according to a branch of mathematics known as spectral, or Fourier, analysis. In general, an electromagnetic signal, such as electrical current or light, can be viewed as possessing a magnitude that varies continuously in time, usually of an oscillatory nature. As shown by the mathematician Joseph Fourier in 1807, any such continuous signal can be represented exactly by the (possibly infinite) sum of regularly oscillating sine and cosine waves of varying amplitudes (for an overview see [93]). For example, the irregular signal $f(t)$ in Figure 1.3 (top left) can be expressed as a linear combination of simpler, regular cosine waves.

The projection of a function onto sine and cosine bases is known as the Fourier transform $\mathcal{F}[\cdot]$, which transforms a function $f(t)$ in the time domain to a function $F(\omega)$

in the frequency domain. While the example of Figure 1.3 is decomposed into a discrete number of cosine waves, a general function will consist of a continuous spectra, or combination of cosine and sine waves. The analytic form of the transform is given as:

$$F(\omega) = \mathcal{F}[f] = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \omega} dt, \tag{1.1}$$

yielding a frequency-space representation $F(\omega)$ equivalent to $f(t)$, but instead expressing the function in terms of frequency ω rather than t. The transform is invertible by the related Inverse Fourier transform $\mathcal{F}^{-1}$, allowing processing directly in frequency space, with subsequent recovery of a modified signal in the temporal space.

This form allows for simplification of many filtering operations. Let us denote the result of applying the filter K to a function $f(t)$ as $K[f](t)$. For example, the low-pass filter $K^{\text{low}}_{\omega_0}[\cdot]$, which removes all frequencies above ω_0 in its input, can be conveniently represented as a multiplication in frequency space. Given the *filter kernel* function

$$K^{\text{low}}_{\omega_0}(\omega) = \begin{cases} 1 & \text{if } |2\pi\omega| \leq \omega_0, \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

the Fourier transform of the filtered function is represented as

$$F'(\omega) = K_{\omega_0}(\omega) F(\omega). \tag{1.3}$$

Applying the inverse transform back to the temporal domain produces the filtered result. We illustrate this in Figure 1.3, in which the function $f(t)$ has been transformed to the equivalent $F(\omega)$ (bottom left, represented as 4 pairs of Dirac delta distributions[2]). Application of $K^{\text{low}}_{2\pi}$ removes all but the two low frequency components less than or equal to 2π, yielding a low-pass filtered result.

The filtering operation can also be performed directly in the temporal domain through the use of the *convolution* operator, where for functions $f(t)$ and $k(t)$,

$$(k * f)(t) = \int_{-\infty}^{\infty} k(t-u) f(u) du \tag{1.4}$$

[2]The Dirac delta $\delta(t)$, or *impulse function*, is zero everywhere except at $t = 0$, where $\delta(0)$ is infinitely large such that the total integral $\int_{-\infty}^{\infty} \delta(t)dt = 1$. This definition has the consequence that $\int_{-\infty}^{\infty} \delta(t-x) f(t) dt = f(x)$, so that the integration of $\delta(t-x)$ with a *test function* $f(t)$ selects the value $f(x)$ [92].

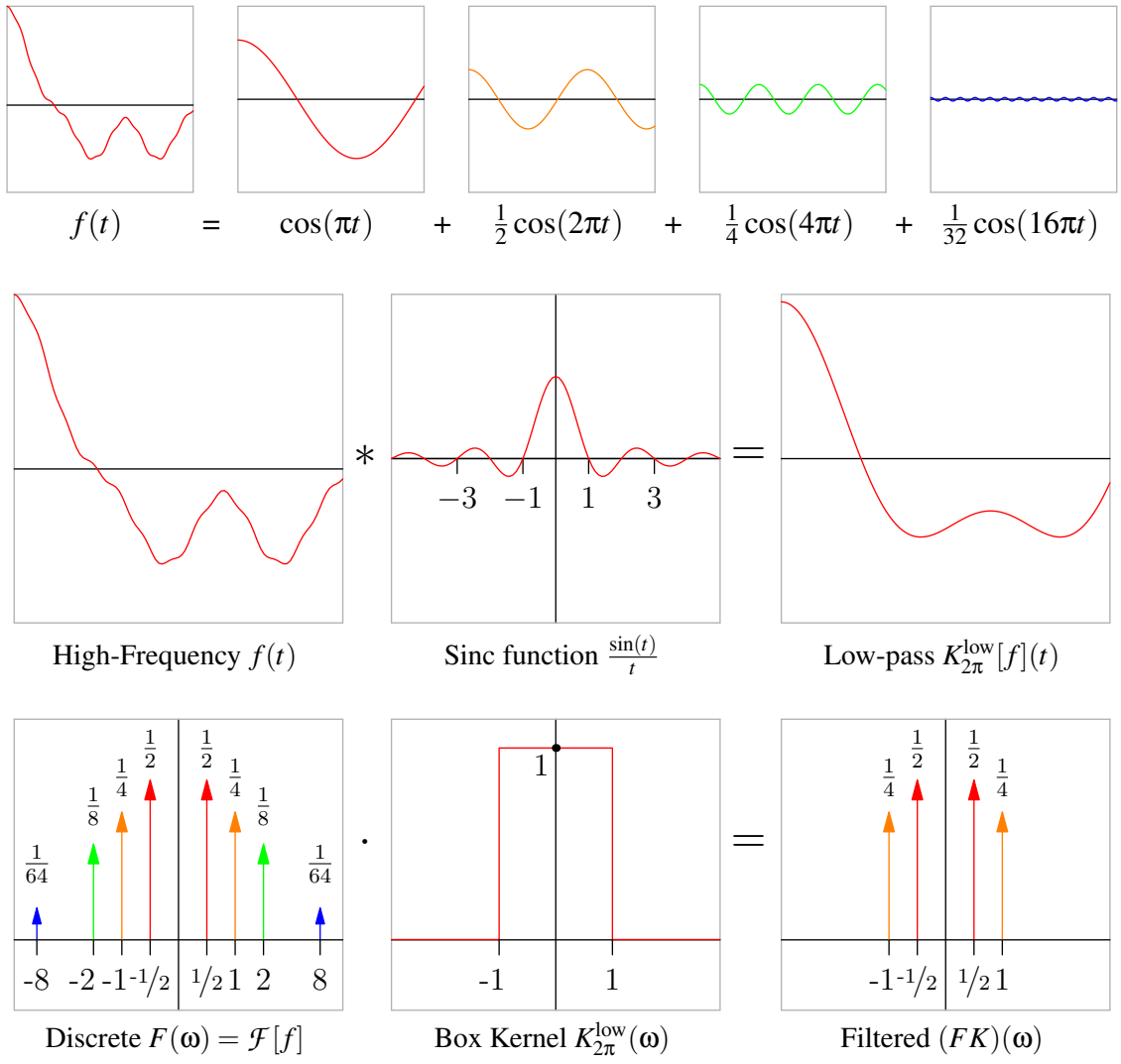


Figure 1.3: **Low-pass Filtering in Time and Frequency.** The example function previously given is equivalently represented in frequency space as a sum of delta distributions, shown bottom-left with their offsets and scales, one pair for each frequency. By multiplying with a rectangular filter kernel, the high frequencies are removed. Note the equivalence between the multiplication in frequency space and the convolution in time.

The convolution bears a special relationship with the Fourier transform operator, in that convolution of two functions in temporal space is equal to multiplication in Fourier space (and vice versa).

$$(k * f)(t) = \mathcal{F}[k](\omega)\mathcal{F}[f](\omega) = (KF)(\omega). \tag{1.5}$$

The convolution $k * f$ is equivalent to $K[f]$ when $k = K[\delta](t)$, in which case k is termed the *impulse-response function* of K. While it is possible to evaluate (1.4) directly (that is to say, in the time domain) the reasons for using frequency space are twofold. First, complex integral operators can be expressed in greatly simplified form in frequency space, leading to ease of filter design and analysis. For example, note from Figure 1.3 that the low-pass filter in (1.2) has a compact form in frequency space, but the transcendental, infinitely-supported $\sin(t)/t = \text{sinc}(t)$ in temporal space. Second, for computational purposes a simple multiplication in frequency space is significantly more convenient than evaluating a convolution. With the use of an optimization known as the *Fast Fourier Transform*, discussed in the next section, we can efficiently compute the spectra of functions, so as to perform filtering operations.

With these filters in hand, we can directly apply common operations to simple signals, such as sound and radio waves, whose information content is contained directly in the signal frequency. While sound is not an electromagnetic wave, per se, but rather a wave of kinetic motion of mass-bearing objects, such waves are converted into electromagnetic signals through use of an electroacoustic transducer (such as a microphone), and can then be filtered by electronic circuits.

However, any perfect high- or low-pass filter is unrealizable in practice due to an infinite impulse response. The pair of $K^{\text{low}}(\omega)$ and $\text{sinc}(t)$ is one such example, which can be shown to generalize to any filters that perfectly block specific frequencies. Therefore, we must be satisfied with a filter that *attenuates* frequencies outside a frequency band, rather than removes them entirely. One common example is the Gaussian $G_\sigma(t) = e^{-\frac{t^2}{2\sigma^2}}$, the Fourier transform of which is also a Gaussian, $G_{2\pi\sigma}$. While it has infinite support both in temporal and frequency space, over 99% of its energy is within three standard deviations of the origin, and therefore is often truncated at that point without noticeable effect. In the next section, we look at other practical issues that arise in the implementation of real-world systems.

1.3 Signal Processing in a Digital System

Analog circuits suffer from the problem of partial information loss due to the attenuation of the analog signal. This is especially a problem as circuit complexity increases, as errors in early stages propagate and increase over the system as a whole. In contrast, digital circuits suffer from information loss only at the conversions between analog to digital representations at the input and outputs to the circuit. This allows for arbitrarily complex circuits, and digital signal processing has proved especially useful.

First, however, one must establish how a continuous analog signal can be represented in the context of a discrete digital system. According to the Sampling Theorem [84], a continuous signal can be *exactly* represented by a discrete sampling under the following condition:

If $f(t)$ contains no frequencies higher than B cycles per second, it is completely determined by giving its ordinates at a series of points spaced $\frac{B}{2}$ seconds apart.

The frequency B is known as the *band-limit*, and the quantity $B/2$ is the minimum sampling rate required for reconstruction, often known as the *Nyquist rate*. It follows that for a window of length W seconds, reconstruction requires at least $2 \cdot WB$ samples. Note that *any* such independent samples, if known accurately, are suitable for reconstruction; the restriction to regular samplings is not required. However, for our purposes we assume regularity.

We can consider the discrete sampled function as $\text{Ш}_T(t)f(t)$, the multiplication of $f(t)$ with the *impulse-train* function $\text{Ш}_T(t) = T\sum_{i=-\infty}^{\infty}\delta(t-iT)$, consisting of an infinite series of impulse functions spaced T seconds apart. The parameter T is the inverse of sampling rate. We see directly that because both the Ш function and $\text{Ш}f$ are discontinuous, their Fourier decompositions have infinitely high frequencies.

The Fourier transform of an impulse train is in fact also an impulse train, in particular $\mathcal{F}[\text{Ш}_T] = \text{Ш}_{(1/T)}$. Therefore, according to (1.5), $\text{Ш}_T(t)f(t) = \text{Ш}_{(1/T)}(\omega) * F(\omega)$. This can be visualized as a sequence of *images* or copies of the spectrum $F(\omega)$, one for each impulse function in the train, such as in Figure 1.4. The images may overlap (we discuss the implications of overlapping images in a moment).

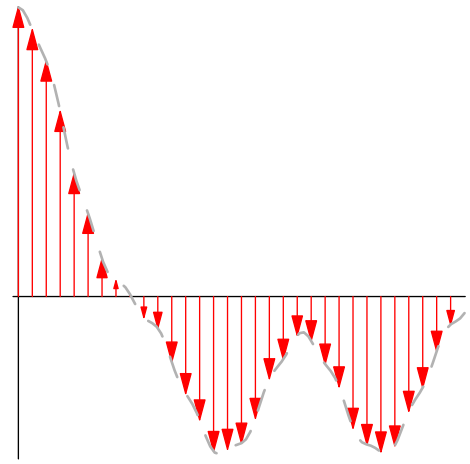
The intuition of reconstruction is that we will filter the sampled representation to remove the frequencies above the band-limit that are induced by the discontinuities (those

images in addition to the primary image), and in doing so reconstruct the original function. To demonstrate, consider our running example, which we know is band-limited to $B = 16$ cycles/second, by construction. Therefore our minimum sampling is 32 samples/second. The sampled $\text{Ш}_{2B}f$ (Figure 1.4a) can also be visualized in frequency space (c) as a infinite series of F (which we have already seen consists of four pairs of delta distributions). Because the function is sampled at $2B$, the images are spaced $2B$ apart, and non-overlapping. By filtering the spectrum to limit frequencies to the function band-limit (shaded region) we are left with specifically $F(\omega)$, from which we can directly find $f(t)$ via the inverse Fourier transform.

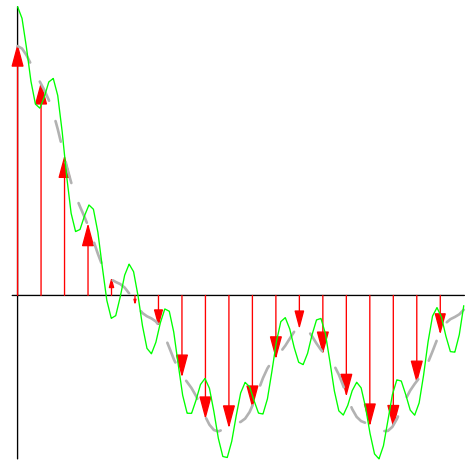
Contrast this, however, to sampling with a lower sampling rate; in particular, we visualize $\text{Ш}_{19}f$ and its spectrum in Figure 1.4 (b & d). The result is a spectrum with overlapping images; in particular, the band-limit region now contains an additional two pairs of delta distributions. In temporal space, these produce two additional cosines terms, leading to errors in reconstruction. This phenomenon is termed *aliasing*, in which two frequencies are indistinguishable "aliases" of one another in a particular sampling.

Two problems now present themselves when applying the theoretical guarantees of the Sampling Theorem to real-world problems. First, as we have seen previously, a perfect low-pass filter is unrealizable in practice, implying the need to choose a realizable, albeit imperfect filter from the options available. The choice of reconstruction filter will have subtle, yet noticeable, impact on the reconstructed result, especially in its behavior with regards to high frequencies from spectrum images. We consider the trade-offs shortly. As these filters are unable to remove entirely such high frequencies, but merely attenuate them, there is the additional implication that systems should sample at rates somewhat higher than that specified by the Sampling Theorem.

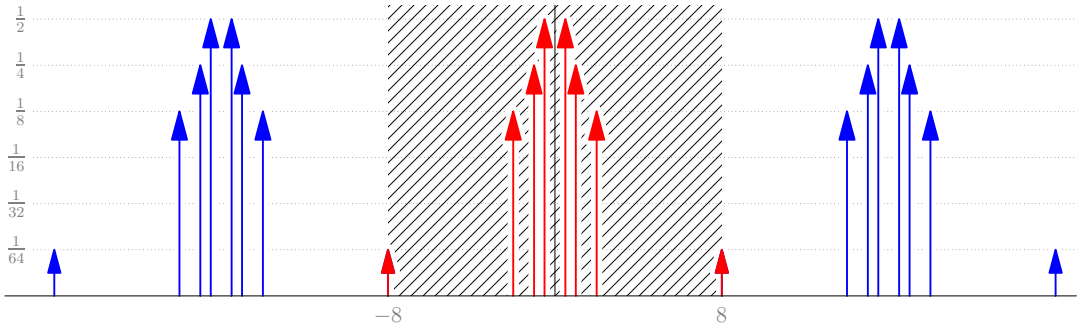
This leads to the second problem: in many cases the functions in question have significantly high frequencies, rendering such sampling impractical; or indeed, are discontinuous and therefore have infinite frequencies, in which case perfect sampling is impossible. This is a particular problem in rendering, in which discontinuities (especially prevalent due to occluded visibility) and perspective may lead to arbitrarily high detail concentrated in a fixed region. We therefore consider the effect that a choice of filter kernel has on such functions, keeping in mind that many such effects will appear in computer graphics problems.



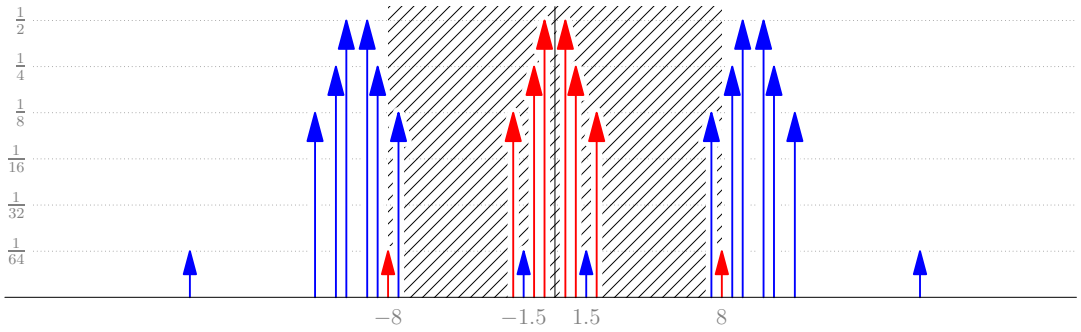
(a) Well-sampled Function, $\text{Ш}_{32} f$



(b) Under-sampled Function, $\text{Ш}_{19} f$

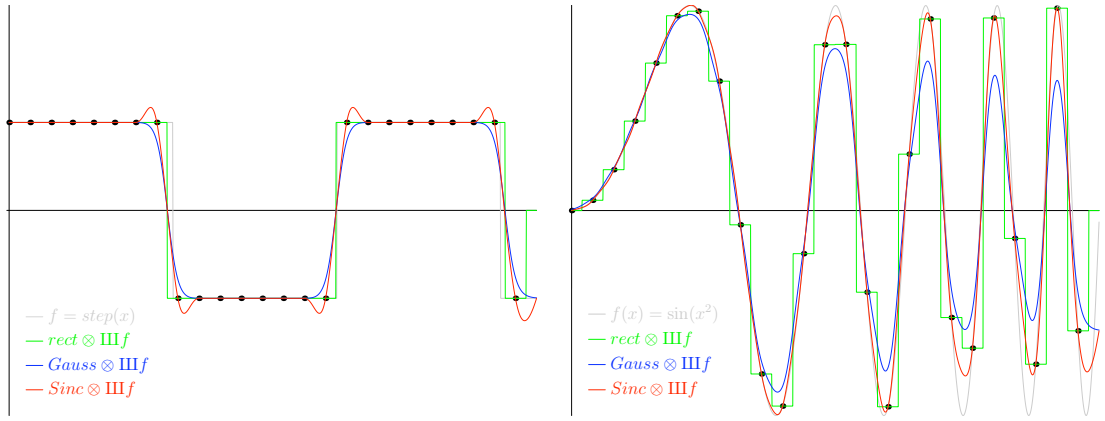


(c) Spectrum $\text{Ш}_{1/32} F$ of (a)



(d) Spectrum $\text{Ш}_{1/19} F$ of (b)

Figure 1.4: **Sampling and Reconstruction.** When the band-limited function $f(t)$, with highest frequency $B = 16$, is sampled using $\text{Ш}_{2B=32}$ (a), the resulting frequency spectrum (c) avoids overlap between F (in red) and its images due to sampling (blue). Applying a low-pass filter for the shaded region, F is isolated, and produces the original function f when converting to temporal space. By contrast, sampling using Ш_{19} (b) has insufficient separation of the images (d); filtering for the low-pass region will include two additional cosine terms, $\frac{1}{32}\cos(3\pi x) + \frac{1}{4}\cos(15\pi x)$, causing incorrect reconstruction (b, green line).



(a) Reconstruction of Step Function (b) Reconstruction of $\sin\left(x^2\right)$

Figure 1.5: **Varying Reconstruction Filters on High-Frequency Functions.** Two high-frequency functions are reconstructed using varying kernels, from the 25-point sampling as shown. While convolution with the (time-domain) box function matches the sharp transition of the step (a), it introduces discretization artifact in the continuous sine (b). Alternatively, the Gaussian better matches (b), at the cost of a smoothing across the transition in (a). The windowed Sinc function interpolates exactly the samples in (b), and with a sharper transition in (a), but with the cost of ringing.

Arguably the simplest filter to implement is the time-space box filter (frequency-space sinc), which amounts to an averaging across a neighborhood of elements. Because of this simplicity, it is often implemented in practice. In Figure 1.5, we show the effect of reconstructing an alternating step function and a sine function of increasing frequency, using the box filter, the Gaussian filter, and frequency-space sinc. As expected, the box filter introduces sharp discontinuities in the reconstruction due to the high-frequencies of its spectrum; this is suitable for reconstructing the discontinuous step function, but not for the sine. Instead, we use a continuous reconstruction kernel with bounded influence at higher frequencies, such as the Gaussian, which preserves the continuous nature of the sine function. However, this gives the negative effects of "smoothing across" the otherwise sharp discontinuity in the step functions. In many application domains, including computer graphics, discontinuities are a key element of the viewer's perception [32]. Therefore, a filter that poorly preserves discontinuities will have a detrimental effect on the perceived result.

Finally, we show the result of filtering with a *windowed sinc* filter: a sinc modulated with a Gaussian to give it bounded influence. This most closely resembles the ideal frequency-space box filter, but without requiring infinite inputs. As can be seen, the

result provides significantly sharper interpolation of the step and the sine function than does the Gaussian. However, it does so at the cost of "ringing" artifacts just before and after significant discontinuities in the sampling. Known as *Gibbs' phenomenon*, this is a result of any truncated Fourier approximation of a sharp discontinuity [59]. We note that these artifacts may be reduced by increased sampling, but not eliminated; rather, they are inherent limitations of *linear* filters (those represented by a convolution). Because the signals we encounter in Part II contain such discontinuities, we will later discuss several non-linear filters to address this situation.

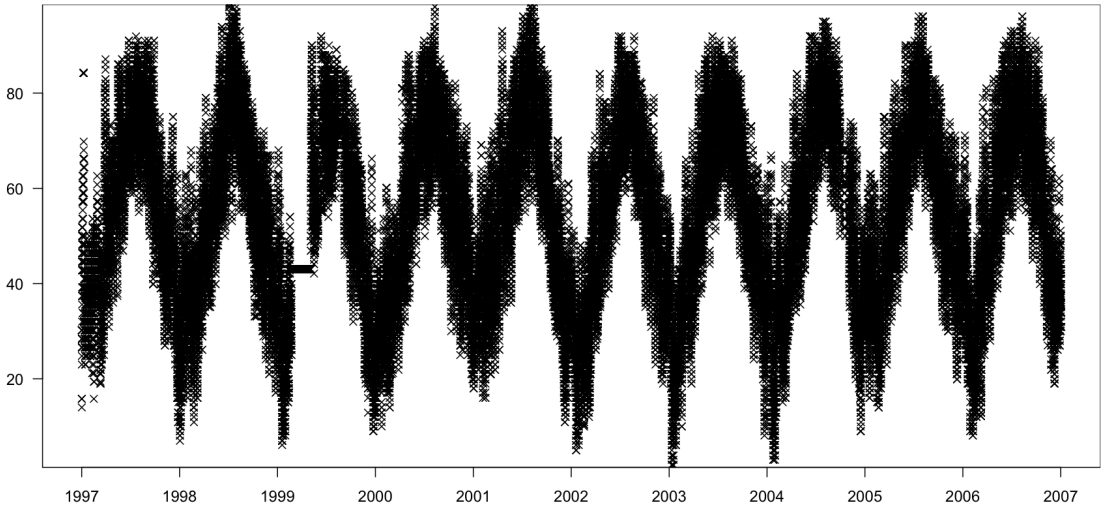
Once a continuous function has been reconstructed from the discrete samples, we may then of course compute the value for any t. Additionally, we may apply an alternate impulse train Ш_T, distinct from the original. This operation is known as *resampling*, and has important usage in operations such as data compression and image resizing, as well as a key component of the subtractive shadow filter presented in Chapter 4.

To see an example of filtering in practice, we use a dataset of temperature readings shown in Figure 1.6, which consists of hourly temperature measurements (not averages) in the Princeton area for the 10 years from 1998 through 2007, comprising about 88 thousand measurements[3].

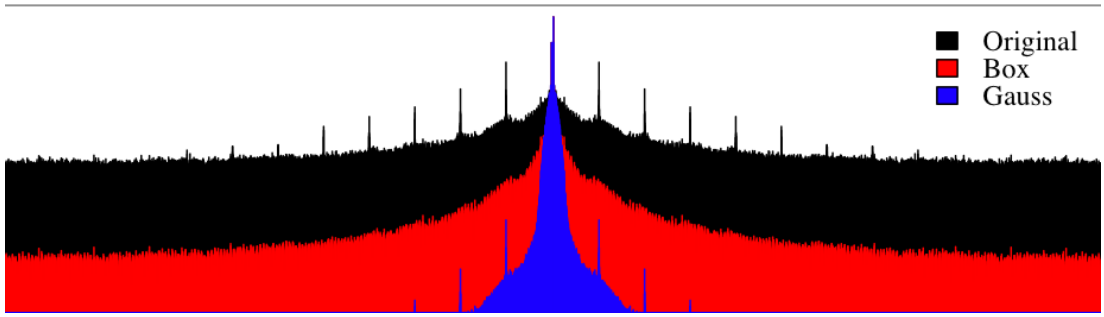
First, as we can see the data is quite noisy, due to periodic fluctuations on multiple scales, and imprecisions of measurement - or even outright errors. Suppose we wished to compute a running average of the weekly temperature; that is, at any point sample we compute the average of the nearest 7 days worth of temperature readings. We can frame this as a filtering problem, in which the input function is convolved with the rectangle function with a width corresponding to 7 days (168 hours).

As we have discussed at length, the Fourier transform of a box function is a sinc; therefore if our *temporal* filter is a box, convolution will result in a frequency-space multiplication with a sinc function. As the sinc function has infinitely high frequencies, this will lead to high-frequencies in the result, as in Figure 1.6(b). By instead choosing

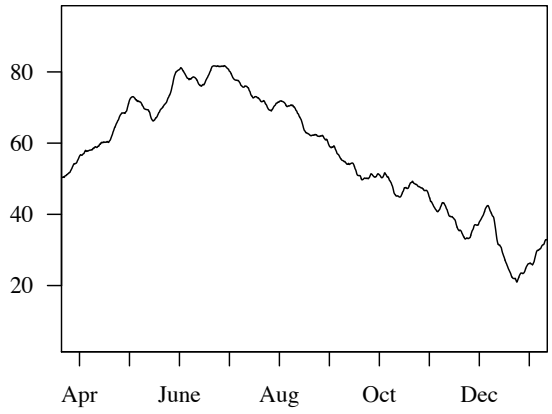
[3]This dataset was measured at the Weather Monitoring Station located at Trenton-Mercer Airport, as provided by Weather Underground (wunderground.com). In most cases measurements are taken hourly; in the case of multiple measurements per hour, all but the first are discarded. In limited cases (about 6.6% of the total measurements) temperature data has been marked as unavailable; we report such measurements as unchanged from the previous, a method known as *zero-order hold* [60]. Certain clear outliers exist in the data, particularly in temperatures known to be above record highs. We have intentionally not made an effort to filter these manually; a discussion of automated outlier detection and correction is given in Chapter 5.



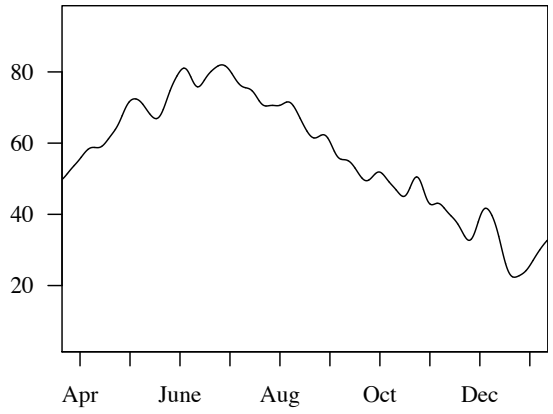
(a) Observed Hourly Temperature at Trenton-Mercer Regional Airport, in Fahrenheit



(b) Original and Filtered Frequency Spectra



(c) Avg. Weekly Temp. (Box)



(d) Avg. Weekly Temp. (Gaussian)

Figure 1.6: **Digital Filtering of Temperature Data.** Raw hourly measurements of temperature have significant noise and variation, making interpretation difficult (a). In the function spectrum, we see that significant energy is collected in high frequency components (b, black). Filtering allows for easier interpretation of statistics, such as viewing data as a weekly average (c and d). A naive weekly average (box filter) fails to remove high frequencies (b, red; and c), compared to a Gaussian filter kernel chosen for its spectral properties (b, blue; and d).

a Gaussian function with variance equal to that of the box, and which is smooth both in time and frequency, we can compute a running average with the same basic shape but with reduction in the high frequency content.

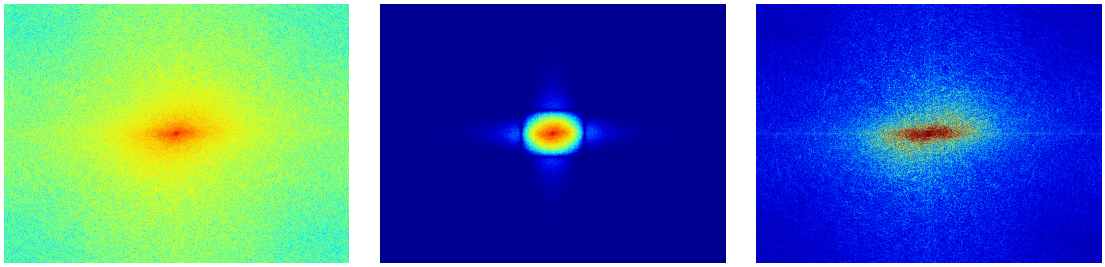
Note that we have approached the selection of a filter kernel from a different perspective than before: whereas previously, we were selecting a kernel for its frequency properties and finding the best matching temporal function, in this case we are selecting a filter first in temporal space, then considering the implications this choice will have on the resulting filtered spectrum. This running average could be resampled to provide either finer grained statistics consistent with the data, or zoomed out (weeks, etc.). The filtered representation better represents the continuous variations and trends occurring over the course of the week, removing the high-frequency changes due to random variations. In the next section, we will show how these filters on 1D signals can be generalized to handle higher dimensional signals that appear in computer graphics applications.

1.4 Higher-dimensional Computer Graphics Filters

As can be observed in the illustrations in Figure 1.6, the application of a low-pass filter to a signal serves to visually smooth the output. We can readily transfer these concepts to smoothing or noise removal in images. Rather than assume a one-dimensional signal $f(t)$ as a function of time, we will treat an image $I(x,y)$ as a two-dimensional function of spatial positions x and y. As with 1-dimensional functions, we will again represent an image using a sampled representation, now using the 2D function $\text{Ш}_T(x,y) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\delta(x-iT)\delta(y-iT)$. The DFT is *separable*: the 2D transform is equivalent to performing the 1D transform on each row of the image, and subsequently transforming each column of the row-transformed intermediate result.

We may thus perform the same sorts of filtering operations on 2D images as we have previously seen on 1D signals, as shown in Figure 1.7. By filtering with a Gaussian kernel $G(x)$, all but the lower frequencies are attenuated, smoothing the reconstructed image. Alternatively, in certain applications, it can be useful to accentuate the higher frequencies to make details stand out. We can accomplish this with a filter kernel that amplifies higher frequencies, while keeping low frequencies as-is. For example, the filter

(a) Original Image (b) Smoothed Image (c) Sharpened Image

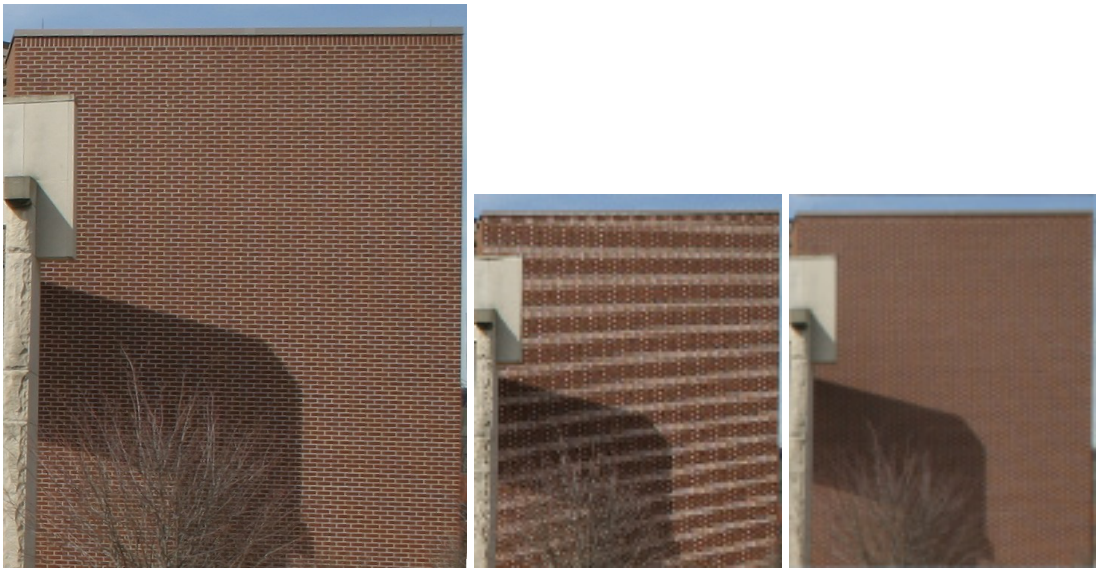


(d) Original Spectrum (e) Low-pass Spectrum (f) High-pass Spectrum

Figure 1.7: **Frequency-Space Filters for Images.** Using the 2D DFT, we apply similar frequency-space filters to those of 1D function. The spectra (d,e,f) are shown on a logarithmic scale, with (f) further scaled by 10. By removing the high frequencies from the Fourier spectra (d), through multiplication with a Gaussian kernel, we produce a smoothed filtered result (b). Alternatively, we sharpen the image (c) with a filter that boosts the high frequencies relative to the low frequencies (f).

$2 - G(x)$, which ranges from 1 at the lowest frequencies, and converges to 2, produces such an effect.

Low-pass filtering in particular has a particular importance in image resizing. Scaling an image from size $w \times h$ to a new size $w' \times h'$ can be thought of as reconstructing a continuous function from the sampled input, then resampling at the new size. When increasing size, a sharpening filter can be useful to preserve edges in the resulting image. When decreasing size, the criteria of the sampling theorem must be taken into account; otherwise, high-frequency image content will be expressed as aliasing in the down-sampled image. This can be seen in downsampling the image in Figure 1.7a to a smaller image according to a naïve direct sampling, as in Figure 1.7b. By filtering the input image according to a kernel chosen to match the sampling rate, the aliasing artifacts can be avoided (Figure 1.7c).



(a) Original Image (b) Naïve Scaling (c) Filtered Scaling

Figure 1.8: **Image Resampling.** While direct subsampling of a high-frequency image (a) leads to artificial low-frequency aliasing artifacts (b), analogous to those seen in Figure 1.4b, sampling a low-pass filtered image removes the artifact (c).

It follows from the sampling theorem that low frequency information can be sampled or represented at a lower rate. This is implicit in the representation given by the DFT, in which the few low-frequency components have greater influence on the reconstructed result that than larger number of high-frequency components. If an image has significant energy in low-frequencies, space efficiency can be achieved by storing only the highest-energy components. Figure 1.9 demonstrates this effect, visualizing the result of discarding a progressively increasing percentage of low-energy frequencies. Because the image spectra (shown previously in Figure 1.7d) has greatest energy concentrated in low frequencies, discarding high-frequencies has a relatively insignificant effect on the reconstructed image quality. We will employ a similar strategy in Chapter 4 to accelerate rendering of scenes with soft shadows, by computing the low-frequency at a lower sampling rate, while computing the high-frequency illumination at full detail.

An aspect of this representation worth noting is its *progressive* nature: a low-quality, low-storage representation of the image can be progressively enhanced to a high-quality image (even to the full quality of the original) by inclusion of additional frequencies in the representation. This can be viewed as an example of a method for *variable level-of-detail* (LOD); such methods allow for a tradeoff between quality/detail and some

5% of Original, .033 RMS | 1% of Original, .047 RMS | 0.1% of Original, .076 RMS

Figure 1.9: **Compression via Filtering.** By discarding all but the strongest 5% of frequencies, we achieve significant storage savings with relatively minor decrease in image quality (compared to original previously shown in 1.7a). By discarding additional frequencies, we achieve progressively greater storage efficiency, albeit at the cost of image quality.

efficiency parameter, such as time or storage cost. Transforms such as the Fourier, and related operations such as the Cosine Transform, are key components in level-of-detail algorithms used in image and video compression. A contribution of this thesis is to adapt this sort of LOD algorithms to other graphics fields: Chapter 4 presents a system to establish a quality-speed tradeoff in real-time graphics, while Chapter 5 demonstrates the effect of filtering to allow for a progressive enhancement of quality in realistic rendering.

As we showed in Figure 1.5, linear filtering has certain limitations with regards to sharp transitions: specifically, that one faces a tradeoff between smoothing and ringing. For certain applications, we employ *non-linear* filters, two of which we discuss: the median filter and the bilateral filter.

For the former, consider the image shown in Figure 1.10a. Random pixels of a photographic image have been corrupted by being set to a random value; this introduces high-frequency energy localized to that sample. However, standard low-pass filtering will have difficulty removing this peak because of its large magnitude, instead blurring it to neighboring pixels. As we see in Chapter 5, the filtered value of the corrupted pixel is an example of a *non-robust estimator*, such that corruption of a single data point may induce arbitrary error in that statistic as an estimate of the true parameter.

Standard filters, such as the Gaussian, compute their outputs according to a linear combination of the input elements. Recall Equation (1.4), in which the filter kernel K is a function only of $x-t$, rather than the input value $f(x)$. The *median filter* K_w^{med} solves this problem by using the function values themselves to determine their contribution to

(a) Corrupted Image (b) Gaussian Filtering (c) Median Filtering

Figure 1.10: **Median Filtering.** An image of which random pixels have been corrupted thus contains localized high-frequency information (a). Because of the linear nature of Gaussian low-pass filtering, the noise is insufficiently attenuated before removing important edge detail in the correct portions of the image (b). The non-linear median filter, in particular the variant using MAD as described in the text, is able to correct only those areas in error (c).

the output. For width w, the filtered value $K_w^{\text{med}}[f](i,j)$ is equal to the median of pixels in the $w \times w$ box centered at (i,j). A variant replaces a value $f(i,j)$ with the median m if and only if the distance between $f(i,j)$ and m is greater than the *median of absolute deviations* (MAD) of the neighboring values, where

$$\text{mad}(X) = \text{median}(|X_n - \text{median}(X_n)|). \tag{1.6}$$

This avoids modifying any values unless $f(i,j)$ is clearly inconsistent with its neighborhood. It is this variant that we display in Figure 1.10c; which performs effectively at removing the noise present in the corrupted image. Median filters do have significant limitations, and the filtering method of Chapter 5 was developed to address these.

Another common type of non-linear filter is the *bilateral filter* [94], which generalizes linear filtering with a Gaussian kernel with an additional term explicitly taking function values into account. This allows the bilateral filter to better preserve sharp transitions in its input. The bilateral filter is defined as following; compare to the linear filter in (1.4):

$$K_{\sigma,\tau}^{\text{bi}}[f](t) = \int_{-\infty}^{\infty} k_\sigma(t-u) s_\tau(f(t) - f(u)) f(u) du. \tag{1.7}$$

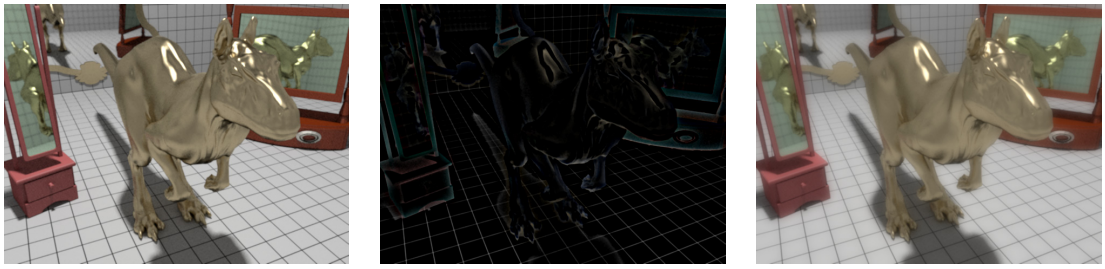
In this formulation, $k(\cdot)$ is the filter kernel as before, termed here the *spatial filter*, and $s(\cdot)$ is a function of the difference in image values, termed the *range filter*. Assuming that both k and s are Gaussian kernels with standard deviations σ and τ respectively, edges are preserved by setting τ to a finite value. In Figure 1.11, we show the result of smoothing an image with the bilateral filter (see Figure 1.7 for the original and linear-filtered images).

Figure 1.11: **Bilateral filtering.** The use of a range filter preserves edges compared to Figure 1.7b

As with median filtering, bilateral filtering also has its limitations. We delve into the specific limitations of bilateral filtering later; however, several concerns are common to both filtering methods. As they are not convolutions, the convolution theorem does not apply, and they cannot be implemented with the Fast Fourier Transform; nor is either filter separable, leading to an exponential increase in time complexity with increasing dimension. As a result, straightforward implementations of either will be very slow for significant spatial widths. However, a number of significant optimizations have been recently presented for both that significantly improve their running time. A key optimization is filtering in the *joint domain* to compute the bilateral filter [18]. In this method, rather than viewing the image as a dense sampling of a 2D function, $f(x,y)$ for all x and y, it is a sparse sampling of the 3D points $(x,y,f(x,y))$. This 3D function is resampled onto a 3D grid, which is a separable procedure. The output filtered value $K[f](i,j)$ is equal to the value of the joint 3D *bilateral grid* $\rho(i,j,f(i,j))$. A similar joint-domain filter is used in Chapter 5.

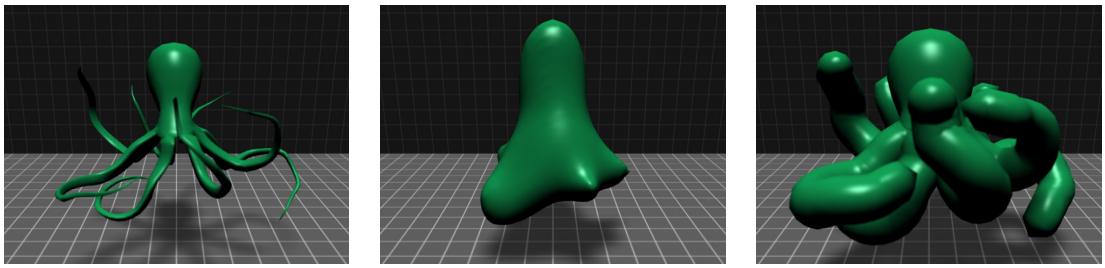
Other operations can be applied for a wide variety of effects, either for increased realism, or for any arbitrary range of effects. We term filters that are not realistic, but instead intended to convey a certain perceptual effect, as *stylistic filters*.

For example, the *bloom filter* intentionally takes high-energy pixel values and spreads their energy, in such a way as to simulate a similar effect seen in actual photographs. This effect, in which light appears to produce a glare around bright objects and spread to their surroundings, is a result of imperfect focus in the lens. While it is therefore "incorrect" in the sense of being a measurement artifact, the use of this stylistic effect results in an image more closely aligned with the viewer's expectations. Such a filter is implemented



Original Rendering | Added Bloom | Final Image

Figure 1.12: **Bloom Filter.** By blurring the high-intensity energy with a long-tailed filter, and adding this bloomed image back into the original, we produce a stylized result which nevertheless more accurately represents the viewers expectations of bright objects.



Original Mesh | Smoothing Filter | Inflation Filter

Figure 1.13: **Mesh Filters.** As with images, we may apply stylistic filters to polygonal meshes to achieve a range of effects. For example the octopus mesh (a) has been filtered with a smoothing filter that reduces high-frequency detail (b), as well as an inflation filter that produces an offset surface. Noting that the filter naturally affects the shadow, as well as the object, we show in Chapter 3 a rendering filter that performs such operations directly, and in a more efficient and user-configurable manner.

using a kernel with a longer "tail." An example is shown in Figure 1.12. Other stylistic filters can be applied for alternate effect; for example, to artificially increase the contrast of the object as in histogram equalization or for other visualization or artistic purposes. We will see the result of stylistic filters specifically for shadows in Chapter 3.

Filters can be further extended to even higher dimensions, and generalized to datasets represented by more complex functions. One extension of the concepts previously shown would be to 3D datasets, or *volumes*, in which a fixed region of space is mapped to a density or color. These types of datasets see frequent use in computer graphics to represent solid objects, and many of the same filtering methods – including smoothing, sharpening, compression and level-of-detail – have application to volumes. Arguably more common for the representation of solid objects, however (and on which we will

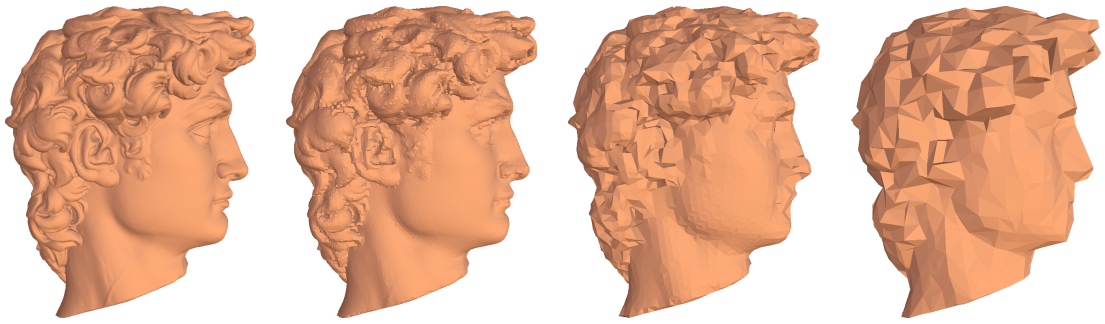


Figure 1.14: **Level-of-detail of Triangle Meshes.** Just as with images, geometric representations such as triangle meshes may be filtered to allow for variable level-of-detail. In this example, the high-resolution David model is reduced in detail by vertex clustering [26], allowing a tradeoff between both rendering time and storage size on the one hand, and mesh resolution on the other. In Chapter 4, we present a rendering filter that allows a similar time/quality tradeoff for shadows, and in Chapter 5, for global illumination.

focus in the remainder of this chapter) are *boundary representations*, such as *polygonal meshes*, in which an object is represented through its surface.

We may view a polygonal mesh as a function mapping all points on a 2D surface into 3D space. We may perform filtering operations on either the 2D function or upon its embedding in $\mathbb{R}^3$. Many of the filters that we have previously seen have been implemented on meshes. For example, in Figure 1.13, an input mesh (a) has been filtered with a low-pass smoothing filter defined over the embedding in space (b) and with a stylistic *inflation* filter relative to the parametric surface orientation (c). Additionally, a large amount of research has been dedicated to level-of-detail in polygonal meshes (see [56] for an extended overview). In Figure 1.14 for example, we demonstrate a high-resolution mesh that has been reduced in detail by a *vertex clustering* filter [54].

As we have shown, filtering is a ubiquitous concept in computer graphics. While it encompasses many wide and varying sort of operations, across disparate applications, these all share common themes that make the notion of a "filter" a useful construct. We conclude the chapter in the next section with a discussion of these shared themes.

1.5 Applying the Metaphor

The notion of a photographic filter, or of a computational filter, is particularly useful in that it has a natural correlate to the physical filters that one encounters in the real world, and its related concept of "filtering." Therefore, it serves as a metaphor: a construct that enables transfer of existing knowledge and intuition from one domain into another. We propose, then, the following definition of filter within the context of computation:

A filter is a passive, stateless operator with parameters independent of data size.

- **Operator.** A transformation of a given input to an output; it is a function that operates on another function (the input), and whose output is independent of the original.
- **Passive.** All operations are performed on the input, as provided; this is in contrast to an "active" process, which interacts with or otherwise has feedback to its environment to generate new data.
- **Stateless.** Any operations take into account only the input itself, rather than past inputs, and on some set of *parameters*. Repeated application of the filter to identical data produces identical results.
- **Independent of Data Size.** These parameters are themselves constant with respect to the size of the data; the input data set may be increased arbitrarily for a fixed definition of the filter parameters.

This has important implications, one in particular is that so long as the input and outputs of multiple filters are of the same or of compatible formats, they may readily be composited in a *filter chain*. As well, the lack of data-size dependent parameters and of filter state allows for an arbitrarily large amount of data. This is consistent with the intuitive notion of a filter as in common use. For example, the following real-world pipeline consists of a system with multiple filters:

$$\text{Tap water} \rightarrow \boxed{\text{purification filter} \rightarrow \text{heating element} \rightarrow \text{coffee grinds} \rightarrow \text{paper filter}} \rightarrow \text{Coffee}$$

In this system, the concept of an input that "passes through" to the output, and which is modified along its route by the four filters in the system, is intuitively evident. Individual filters could be swapped out, replaced, or added, and this would have an understandable impact on the remainder of the system. Further, one does not have to know about

the details of say, the water purifier, to understand how to change the coffee grinds so as to produce a different result. The concept of the filter is essential to this intuitive understanding.

We present in this thesis the concept of the *rendering filter*, which operates on specific data acquired in the course of rendering a computer graphics image, and indicate why it is useful to think of the operations presented in Part II specifically as filters rather than as arbitrary computer graphics algorithms. Specifically, the view of them as rendering filters makes it clear how they can be employed in filter chains, or as an intermediate part in a much larger rendering pipeline that allows to "pass through" those components. This modularity allows for greater encapsulation and interoperability between a range of computer graphics-related algorithms, in the same way that a photographic filter – or indeed, a physical filter such as one for water – allows in physical systems.

Chapter 2

Background and Foundations

Rendering, or image synthesis, in the context of Computer Graphics is the mathematical simulation of the imaging process as takes place in the camera. By formalizing the problem we can develop algorithmic solutions; in doing so in this chapter, we will build up an understanding of existing filtering methods that play a significant role in graphics, yet also understand their limitations. We do so in order to establish our concept of the rendering filter, and compare and contrast this novel formulation from existing filters as presented both here and previously in Chapter 1. Subsequently, we demonstrate specific examples of rendering filters in Part II.

First, we will describe how the rendering problem is one of simulating a *lightfield* [31], which is represented by the 7-dimensional *plenoptic function*, $p(x,t,\omega,\lambda)$ [1]. This function yields the *radiance* of the light with wavelength λ incident to a (three-dimensional) point x, from a (two-dimensional) direction ω, at a particular time t. We proceed now with a definition of these concepts.

2.1 Rendering Fundamentals

Electromagnetic radiation (EMR) consists of oscillating, periodic waves of energy that propagate through space. Generated from many sources, we may characterize such waves as having a particular wavelength; those with wavelengths from about 400-750 nm are detected by optical receivers in our eyes – we term this range the *visible light spectrum*.

In this section, we present a model of visible light as used in computer graphics, and discuss the relevance of filtering methods in this context.

A complete simulation of the wave model of light would require a representation of the wave properties of amplitude, wavelength, frequency, phase, and orientation. While use of the wave model would allow for simulation of a wide range of visual phenomena, multiple waves combine in a non-linear fashion, and interact in a complex manner with objects whose size is on the order of the wavelength. While these lead to the visual phenomena of *interference* and *diffraction*, computer graphics generally assumes the simplified *geometric optics* model of light propagation, and the loss of these effects is often imperceptible. In this model, light originates at a point and travels an infinite distance along a straight line, which we may model mathematically as a ray, $x_0 + t\omega$ for all non-negative t. The point x_0 is the origin of the light ray, and the vector ω is its direction of travel. While light rays are considered to have a frequency λ, this exists solely as a quantity of information carried by the ray, rather than an influence on its behavior.

The following assumptions are also commonly made in computer graphics; while not inherent limitations to the geometric optics model, their use allows simplification of the problem without significant loss of realism.

- *Non-polarized light.* Transverse-wave orientation is disregarded through the assumption that light consists of a uniform distribution of all orientations simultaneously. Such an assumption prevents the simulation of effects due to *polarization*, light with a bias towards particular orientations.

- *Infinite speed.* A reasonable approximation in the simulation of terrestrial scenes, we assume that light travels infinitely fast. The resulting consequence is that a lightfield will have reached an equilibrium state for any instant in time.

- *Time independence.* The notion of time in the simulation is further simplified by assuming that any light energy in the system at time t was emitted exactly at t. This is linked with the previous condition, and prevents simulation of *phosphorescence*: the absorption and storage of energy for later emission.

- *Non-interacting Wavelengths.* The transport of light at any given wavelength is assumed to be independent of the presence of light at alternative wavelengths. This specifically prevents the simulation of *florescence*, in which energy absorbed at high (often ultra-violet) frequencies is subsequently emitted at lower frequencies.

As we have stated, light waves (or EMR in general) propagate radiant energy through space. The system as a whole is termed a *lightfield*, which, following from the previous assumptions, is characterized by the rate of emission of radiant energy at time t, or *radiant flux* $\Phi(t)$. If energy is measured in the SI unit of joules, flux is given in *watts*, or joules per second.

From the definition of a ray, a light source at the 3D point x_0 emitting Φ watts of flux directionally emits light as a function of the unit vector ω. The differential quantity of watts per unit of direction is the *radiant intensity* $I(\omega) = \frac{d\Phi}{d\omega}$, and represents the directional distribution of a light source. The unit of direction is the portion of the unit sphere subtended by a set of directions, or *solid angle*, and measured in *steradians*. Analogous to the radian measurement in plane angles, a given solid angle may be as large as 4π steradians.

A sphere of radius r and area $A = 4\pi r^2$ centered at x_0 will receive a total of Φ watts of flux across its surface. We refer to the flux density at a point on the surface as its *irradiance*, $E = \frac{d\Phi}{dA}$, whose standard units are watts per square meter. For any point x on the sphere, we may derive the direction vector from the light source to that point as $\omega = \frac{x - x_0}{r}$. Note that as r increases, while the radiant intensity $I(\omega)$, remains constant, the area of the surface increases by $4\pi r^2$, such that the irradiance $E(x)$ at the points corresponding to ω on two such distinct circles will decrease accordingly.

Therefore, we may see that the amount of flux arriving at a surface point is dependent on the *projected area* of that surface onto the unit sphere about the light source. Alternatively stated, all surfaces with equivalent projections onto the unit sphere about a light source will receive equal flux. This quantity is that of *radiance* $L = \frac{d\Phi}{d\omega dA^{\perp}}$, the flux per unit solid angle ω per projected area $A^{\perp}$. It is this quantity that is invariant along a light ray, and which we will use to describe the lightfield.

Given this, we may now define a lightfield as the radiance $p(x, \omega, t, \lambda)$ of light with wavelength λ arriving at a point x from incident direction ω at time t. The function $p(\cdot)$ is termed the *plenoptic* function, and is 7-dimensional. The light falling on a particular point is equal to the integral over all other dimensions, and evaluating this integral is fundamental to computer graphics. Note that in this context λ is strictly a quantity of information, and does not imply any wave-like phenomena.

Naturally, a high-dimensional integral such as this poses challenges for numerical evaluation. In a rendering algorithm, we compute only a portion of this function, simulating that portion that is measured by a real-world camera. A camera measures the light incident to a photosensitive plate, often referred to as the *camera plane*, through an opening, or *aperture*, directed towards the objects of interest in the scene. Further, because the camera is closed except for the aperture, only rays that pass through the aperture will affect the photosensitive elements of the film. This reduces significantly the region of interest in the plenoptic function. In the extreme, let us assume that the aperture consists of a single point, a_0, such that any ray of light entering the camera must pass through a_0. This is a so-called *pinhole* camera. Thus when computing the radiance on a point c on the camera plane we need only evaluate p for $\omega = c - a_0$.

$$L^{\text{pinhole}}(c,t,\lambda) = p(a_0, c-a_0, t, \lambda) \tag{2.1}$$

In the more general case, the aperture represents a small but finite solid angle, rather than a single point. In this case the radiance at c requires an integral over the aperture A. A photosensitive element will display a non-uniform response to light incident to points across the aperture; strongest in the center and falling off towards the edges. We can represent this in the integral with the *sensor response function* W. Further, in addition to having a small, but finite size, the aperture is open for some period of time T, during which the photosensitive element is exposed to light, with sensitivity S. Reconstructing radiance at a point c and time t_c in this case then becomes:

$$L^{\text{finite}}(c,t_c,\lambda_c) = \int_A \int_T W(a-c)S(t-t_c)p(a,c-a,t,\lambda)\, da\, dt. \tag{2.2}$$

Note the similarity of the above to the convolution equation (1.4). Reconstruction of the radiance on the camera plane can be viewed as the application of the filters W and S to the plenoptic function p.

The camera is non-uniformly sensitive to light of varying wavelength λ, and this non-uniform sensitivity is important to its perception of color. The color of a lightfield is defined by its *spectral power distribution* (SPD), $p(\lambda)$ for fixed x, ω and t. A color (as opposed to black and white) photodetector has multiple sets of photosensitive elements, each with their own distinct response sensitivity to portions of the SPD. In computer graphics it is common to integrate the SPD against the sensitivity curves of the human

eye (or similar), leading to a description of color as a triplet of red, green and blue (corresponding to the eye's long, medium and short wavelength sensors, respectively). Projection of the radiance p onto a basis also takes the form of a filter, according to the radiance-response function R:

$$L_{\lambda_c}(c,t_c) = \int_A \int_T \int_\Lambda W(a-c)S(t-t_c)R_{\lambda_c}(\lambda)p(a,c-a,t)\,da\,dt\,d\lambda. \tag{2.3}$$

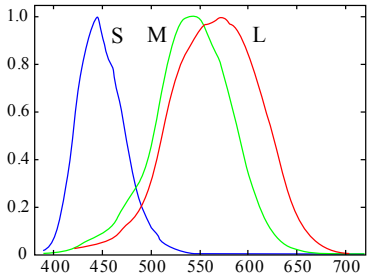


Figure 2.1: **Cone Responses**

To reconstruct a color image, L_{λ_c} is evaluated for all distinct λ_c components in the film with their distinct sensitivity curves. For example, the responses for the photoreceptors in the human eye are shown in Figure 2.1 (Figure and data from [88]). Now that we have identified the rendering problem as a one of filtering the plenoptic function, we demonstrate in the next section evaluation of the plenoptic function. This leads to application of the filtering techniques shown in Chapter 1 to problems encountered in rendering.

2.2 The Light Transport Equation

If we assume that light is emitted or reflected only at the surface of objects, calculating $p(a,c-a)$ is therefore is reduced to determining the object along the ray $a+(a-c)t$ and computing the outgoing radiance thereof (the sum of both emitted and reflected radiance). We denote by $r(x,\omega)$ the first point of intersection between the ray $x+t\omega$ and the scene. At such a point $r(\cdot)$, light may be directed back towards the camera aperture either by emission, or by reflection. The presence of reflected light at $r(x,\omega)$ requires that radiance was emitted or reflected towards $r(x,\omega)$ from some other point in the scene, establishing a recursive definition of exitant radiance. At any point x_0 in the scene, we may express this relationship according to the *light-transport equation* (LTE), also known as the *rendering equation* due to its importance in this field [48].

$$L(x_0 \to \omega_0) = L_e(x_0 \to \omega_0) + \int_\Omega f(\omega_1 \to x_0 \to \omega_0)L(\omega_1 \to x_0)d\omega_1 \tag{2.4}$$

$$L(\omega_1 \to x_0) = L(r(x_0,\omega_1) \to -\omega_1) \tag{2.5}$$

Equation (2.4) states that the outgoing radiance $L(x_0 \to \omega_0)$ from point x_0 towards ω_o is equal to the incident radiance $L(\omega_1 \to x_0)$ from all directions ω_1 as reflected towards ω_o, plus the radiance emitted L_e. As shown by (2.5), the incident radiance $L(\omega_1 \to x_0)$, is itself equivalent to the exitant radiance from the first point of intersection along the ray $x_0 + t\omega_1$. Therefore (2.4) is a recursively defined equation. At a point x on an object, the ratio of reflected radiance toward ω_0 relative to incident radiance from ω_1 is given by $f(\omega_1 \to x \to \omega_0)$, the *reflectivity* of the object[1].

Evaluation of the whole requires recursive evaluation of the individual terms. We may rewrite (2.4) as an expansion of the recursive terms of the integral. Before we do so, let us make some auxiliary definitions. In particular, let the *path vertex* x_n as a function of all ω_i such that $i = 0 \ldots n$, as follows:

$$x_n(\omega_n \cdots \omega_0) = r\left(x_{n-1}(\omega_{n-1} \cdots \omega_0), \omega_n\right), \tag{2.6}$$

that is, x_n is the n-th surface point on the *path* generated by following the successive rays given by (and thus is a function of) the directions $\omega_0 \cdots \omega_n$. Similarly, we define terms for the n-th reflectance term and emission term.

$$L_{e_n} = L_e(x_n \to \omega_n) \tag{2.7}$$

$$f_n = f(\omega_{n+1} \to x_n \to \omega_n) \tag{2.8}$$

We use these to define the camera radiance L as the *path integral* over all directions.

$$L_0 = L_{e_0} + \int_\Omega \cdots \int f_0 L_{e_1} + f_0 f_1 L_{e_2} + f_0 f_1 f_2 L_{e_3} + \cdots \, d\omega_1 \omega_2 \omega_3 \cdots \tag{2.9}$$

Individual radiance samples are then filtered to reconstruct a 2D image, as in (2.3). In a direct implementation, each of the individual *rendering terms* (L_{e_n}, f_n, etc.) are kept only so long as to compute a single value of L, and then discarded. However, it has frequently been shown that in many cases it is useful to filter these terms directly, leading to an intermediate between image filters and geometry filters, for which we suggest the term *rendering filters*. While similar filters might also be represented with

[1] In many cases, reflectivity is specified according to a *bidirectional-reflectance distribution function* (BRDF), which is the ratio of reflected radiance to incident *irradiance* [64]. For the BRDF f_r and surface normal n_x, in the LTE as given in (2.4) the reflectivity $f(\omega_1 \to x \to \omega_0)$ is equal to the BRDF $f_r(\omega_1 \to x \to \omega_0)(n_x \cdot \omega_1)$, where the multiplicative term accounts for the conversion from irradiance to radiance.

a combination of image- and geometry filters, the rendering filter allows for a greater flexibility and ease-of-use. We show an example decomposition into rendering terms in Figure 2.2. Note the differences in frequency content of each term; the reflectance f_0 has the highest detail, resulting from a spatially-varying texture on the surface, while indirect illumination $f_0 f_1 L_{e_2}$ has smoother changes in areas away from geometric discontinuities. By processing these terms individually, rendering filters may better take advantage of their distinct properties. An overview of many such filters are given in Section 2.4.

We can now partition the set of rendering algorithms into two general classes: those of object-space algorithms, known also as *rasterization*, and those of image-space algorithms, best represented by *ray tracing*. A rasterization algorithm starts with a basic geometric object such as a triangle or line segment, for which the corresponding pixels in the output image are located and evaluated. A ray-tracing system proceeds in the opposite direction, that is, given each output pixel, the system identifies the corresponding objects visible at that pixel and computes the color accordingly.

Observe that this distinction primarily amounts to a different order of evaluation of the rendering terms of (2.9). As we will see, rasterization methods evaluate in parallel a large number of lower-order terms across different rendering samples. In contrast, ray-tracing methods are able to selectively evaluate terms of arbitrarily-high order per sample, but at the cost of a slower evaluation overall. Both have their complementary advantages and disadvantages, and have been the subject of extensive research to address their particular shortcomings relative to the other.

2.3 Sampling via Rasterization Methods

The primary advantage of rasterization is the ability to exploit the coherency inherent in an object-space evaluation of rendering terms. Many values can be computed sparsely, and subsequently interpolated; all this occurs in a parallel fashion. To explain, we first give an overview of rasterization. For each triangle, a rasterization algorithm:

1. Projects the vertices to the screen with a linear transformation
2. Interpolates the projected vertices to determine screen pixels
3. Interpolates values from each vertex over the pixels to compute color

(a) Reflectance Only, f_0

(b) Direct Illumination, $f_0 L_{e_1}$

(c) Indirect Illumination $f_0 f_1 L_{e_2}$

(d) Global Illumination

Figure 2.2: **Rendering Terms.** Tracing rays from the camera position, the rendering algorithm can directly determine (a) the reflectance and emissive terms L_{e_0} and f_0. By tracing additional rays to evaluate the lighting (from the environment map shown subsequently in Figure 2.6a) the rendering algorithm can compute (b) the direct illumination term $f_0 L_{e_1}$, and recursively evaluate the indirect illumination term $f_0 f_1 L_{e_2}$. Note that (b) has sharper contrast between light and dark regions than (b), which has softened subsequent to the additional filtering of illumination by f_1. These are combined to form (d) the global illumination image $L_{e_0} + f_0 L_{e_1} + f_0 f_1 L_{e_2}$.

4. Conditionally updates the value of the screen pixel

Each stage can operate largely in parallel: every triangle, vertex, and pixel computation is independent of each other. The only data hazard occurs in the final update of the pixel to the screen (a memory region known as the *frame buffer* in rasterization terminology). Even this, however, occurs in a structured and localized manner that can be implemented efficiently with simple atomic operations. Under our categorization of rendering algorithms by their order of evaluation of rendering terms, a rasterization algorithm evaluates common low-order terms for nearby, correlated pixels.

Consider the "one-bounce" term $f_0 L_{e_1}$ in (2.9), which is light arriving directly from an emissive object to a given point and then to the eye. We refer to this as *direct illumination*, and it can be computed by summing $f(\omega_i \to x \to \omega_{eye}) L(\omega_i \to x)$ for all lighting directions ω_i. In this case, x could either be a vertex position, in which case the result is linearly interpolated across the face (thus the lighting is evaluated in Step 2 and interpolated in Step 3), or x could be an individual pixel position (where all lighting is computed in Step 3).

Several challenges arise. First, the algorithm must consider whether or not the pixel has already been written to by a triangle closer to the viewer. The solution to this here is *depth-buffered hidden surface removal*, also known as *z-buffering* [15]. In addition to the frame buffer storing the color of each pixel rendered so far, it also stores the distance d from the camera to the interpolated triangle at that point. Before updating a screen pixel, the depth of the incoming pixel is tested against the existing depth. If d is closer to the viewer, the color is written to the pixel, and the depth is updated.

Second, note that the algorithm does not consider the possibility that x has an occluded view of the light source direction ω. This subset of the direct illumination is termed *local illumination*. A solution to the complete direct illumination is to use a visibility algorithm. We can rewrite the rendering equation as an integral over area,

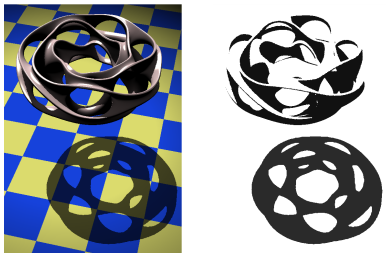
$$L(x_1 \to x_0) = L_e(x_1 \to x_0) + \int_A f(x_2 \to x_1 \to x_0) L(x_2 \to x_1) V(x_2 \leftrightarrow x_1) dx_2, \quad (2.10)$$

where the pairwise visibility $V(x_2 \leftrightarrow x_1)$ is 1 if and only if the space between x_1 and x_2 is unobstructed by any other surface, and 0 otherwise. In terms of the ray casting operator,

$$V(x_2 \leftrightarrow x_1) = \begin{cases} 1 & \text{if } r(x_1, (x_2 - x_1)) = x_2, \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Note also that the reflectance f has a slightly different meaning in this context than in (2.4); here f indicates the fraction of radiance reflected through x_1 from x_2 to x_0, and as such must take into account the distance between x_1 and x_2, by dividing by $||x_2 - x_1||$.

We now consider the visibility function V as an additional rendering term, which we compute by means of a *visibility-determination* algorithm. A direct visualization of this term is shown in Figure 2.3. Assuming that we are limiting our discussion to direct illumination, this is the most difficult term to compute, and we restrict our attention to its computation for the remainder of this section. The most commonly used algorithms are divided into two classes: shadow volumes, an object-based technique; and shadow maps, which is a sampled, image-based technique (note that both are used in the larger context of the object-space rasterization algorithm).



Rendering $V(x \leftrightarrow c)$

Figure 2.3: **Binary Visibility.**

The *shadow volume algorithm* is so named because it explicitly represents the volume of space in which objects are shadowed (presented in [101], current form results from [40]). For a point light source L, the shadow volume consists of all front-facing triangles, the projection of all back facing triangles out to the infinite plane away from the light source, and the triangles connecting them. The algorithm extracts this volume by searching for all silhouette edges in the model that separate a front-facing from a back-facing triangle. We then trace a line from the light source to the silhouette edge and out to infinity, bounding the shadow volume. We show a diagram in Figure 2.4.

Once the shadow volumes are extracted, rendering proceeds in 4 steps. Each step expects that z-buffering is enabled, so that a write will not occur to any pixel such that the existing depth value is closer to the viewer than the incoming value.

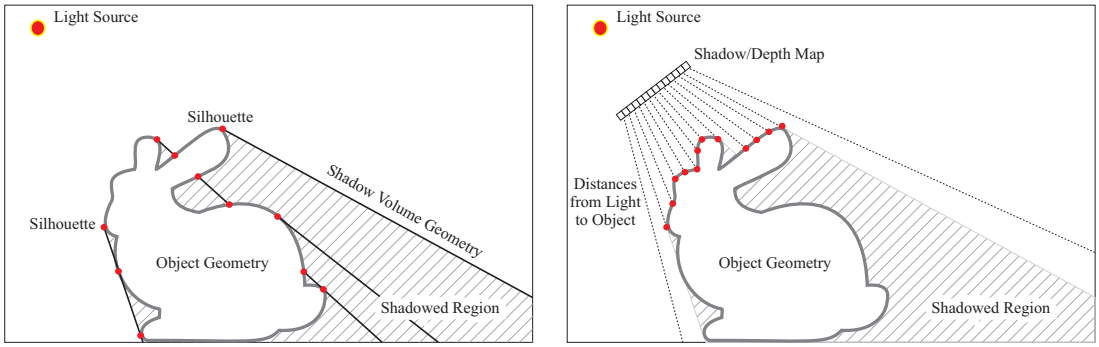
1. Render the scene while writing only depth values (not color)
2. Render all shadow volume triangles that face the viewer into a *stencil buffer*, which increments a per-pixel value, rather than writing a color.
3. Render all shadow volume triangles facing away from the viewer into the stencil buffer, this time decrementing values per pixel
4. Re-render the scene with full lighting at pixels for which the stencil value is 0.

The stencil buffer contains a representation of the visibility function V between each pixel and the light. If any pixel belongs to a surface point that is behind a viewer-facing shadow volume triangle (i.e. it is inside the shadow volume), it will have a non-zero stencil value, and will not be lit in the final pass. If, however, the pixel is outside the shadow volume, it will have a value of 0 in the stencil, and lit normally. Multiple lights are generally supported by repeating Steps 2-4 for each light, and accumulating the result. However, in Chapter 4 we will see how a subtractive approach makes the task more amenable to filtering operations.

While we have implemented stenciled shadow volumes for the implementations in Chapters 3 and 4, they are dependent on high-quality geometry and the computational ability to process potentially very large geometric datasets per-frame. In particular, triangles meshes must be both closed (no holes exist in the surface) and be orientable (possessing consistently defined front and back sides). Otherwise, the shadow volume itself is undefined. In many common applications (especially games) this is not the case, and shadow volumes are often supplanted by the alternative shadow map algorithm.

The *shadow map algorithm* derives its name from its use of an additional texture map that contains the distances of all lit pixels to the light source [21]. This is generated by first setting the camera to the light source position and orientation, and rendering the corresponding view of the scene with depth-writing only (the color buffers are not set). When rendering the scene itself, the algorithm proceeds with the following steps.

1. Project each pixel p into the coordinate frame of the light L
2. Identify the shadow map pixel $S(p)$ corresponding to p
3. Compute the distance between p and L
4. If p is further from L than $S(p)$ ignore, otherwise draw lit pixel



(a) Shadow Volumes (b) Shadow Map

Figure 2.4: **Alternate Methods for Rasterized Shadows.** The shadow volume representation (a) explicitly represents the volume of space occluded by the object, by identifying silhouette points and tracing boundaries from the light out to infinity. This geometry is used during rasterization to prevent lighting of objects behind it. By contrast, the shadow map (b) tabulates the measured distances of all points on the surface to the light. When rasterizing, points are illuminated only if they match the expected distance given by the shadow map.

The shadow map S contains the distance of the nearest surface to the light, for the rays extending from the origin of the light through each pixel in S. Alternatively, the shadow map indicates all surfaces that are visible to the light, and therefore illuminated by it. When rendering the scene, at each pixel a test is performed to determine if that pixel is visible; if not, the pixel is ignored.

The significant advantage of this algorithm is that it is independent of any specific knowledge of the geometric representation, and of any assumptions on that geometry. All it requires is that the geometry can be rendered consistently from multiple viewpoints (that of the true camera and of the light). As a downside, however, this algorithm is dependent on a high sampling rate in the shadow map to account for changes in the geometry. Because any finite sampling will be unable to account for all points, some approximations must be made when checking a distance against the shadow map. This has led to a wide range of attempts at solving this problem (see for example [87, 102]), and the general algorithm is considered robust enough that it is commonly used in production.

There also exist hybrids of these methods to leverage the strengths of both. For example, one may use an approximate shadow volume geometry or extracted silhouettes to

quickly limit the region of the frame that may potentially be in shadow, and subsequently use a shadow map-like approach to compute the final shadowed result [16, 17].

A limitation of both of these algorithms is that they consider only light emitted from a single point. This is inherent to the nature of the rasterization framework, in which a low-order rendering terms are evaluated in parallel across all pixels. It is certainly possible to represent a single surface light source as the union of many point lights, and render them all accordingly, but this starts to become prohibitive quickly. Certain special purpose algorithms for *soft shadowing* have been developed as adaptations of these two algorithms. The penumbra wedge algorithm [6, 7], for example is a generalization of shadow volumes in which a more complex volume that contains both umbra (the hard shadow, equivalent to that extracted by standard shadow volumes) and penumbra (the soft shadow) is extracted from the mesh. Most methods, however, tend to use shadow maps as a foundation, and bear a resemblance to filtering methods. We observe that the soft shadows generated from an area light can be thought of as having low frequency compared to those high-frequency shadows generated from a point light. Therefore, algorithms have been developed that filter either the visibility buffer itself, or use a filtered shadow map in order to compute partial visibility (see for example [8, 13, 27, 30, 36]).

While many of these algorithms are able to make significant progress in evaluating the direct illumination and visibility terms, rasterization in general is significantly limited in its ability to extend to further terms. Computing the general *global illumination* of all rendering terms would require evaluation of high-order rendering terms for the majority of scene elements, which becomes unfeasible even with large-scale parallelization. In the next section, we will discuss the use of ray tracing algorithms to better compute global illumination at the expense of slower computation of direct illumination.

2.4 Sampling via Ray Tracing Methods

While the advantage of the rasterization framework was found in its highly structured and coherent evaluation of rendering terms, ray tracing draws its strengths exactly from breaking this assumption. Specifically, ray tracing allows for a random-order evaluation of arbitrary rendering terms.

As we have established, the value of a discrete pixel $I(x)$ on the image plane is evaluated through a resampling of the continuous LTE $L_o(x, x-c)$ for camera center c and all x on the camera aperture. While any structured attempt to evaluate all such rendering terms (as rasterization does for low-order terms) will be prohibitive, it is possible to make an progressive approximation of the integral by evaluating *random* terms. We first note that the definite integral of a function $f(x)$ is proportional to the *expected value* $E[f(X)]$ for a uniformly distributed random variable X. The *strong law of large numbers* states further that this is equal to the average of an infinite sequence of random $f(X_i)$,

$$\int_a^b f(x)dx = \frac{1}{b-a}E[f(X)] = \lim_{n\to\infty} \frac{1}{n(b-a)} \sum_{i=1}^{n} f(X_i), \tag{2.12}$$

where X is a uniformly distributed random variable, and X_i is the i-th independent sample from X. From the definition of expectation as a limit, we arrive at a method for evaluating the definite integral of $f(x)$; namely, continuing to draw random samples and average their values. This is known as *Monte-Carlo integration*, after the casino. (For an introduction to the relevant probability theory, see [81].)

While any uniformly random sequence X_i converges to the correct estimate of the integral in the limit, the choice of X_i has important consequences for finite samplings. Considering the regular samplings in Section 1.3, we saw that a regular sampling with a rate below the Nyquist rate produces aliasing. Random numbers are defined by their (near-)uniform energy across all frequencies; a random sampling therefore has the effect of moving error into the high frequencies, converting aliasing into noise. While true random numbers can only guarantee uniform density in the limit, *quasirandom sequences* have the high-frequencies of true-random numbers, but in fact are not random at all, as the density of any subset is guaranteed to be uniform [65].

Monte-Carlo methods allow us to progressively compute an approximation I_n of an integral I where the expected error is proportional to $\frac{1}{\sqrt{n}}Stddev[I]$. Importantly, note that this error bound is independent of the number of dimensions. Instead, it is a function only of the number of samples n and the variance of I. Adding additional samples decreases the approximation error, as shown in Figure 2.5. This dimensionality independence is especially important when drawing samples in infinite-dimensional path space, as in (2.9), where standard quadrature methods would require time exponential in the number of

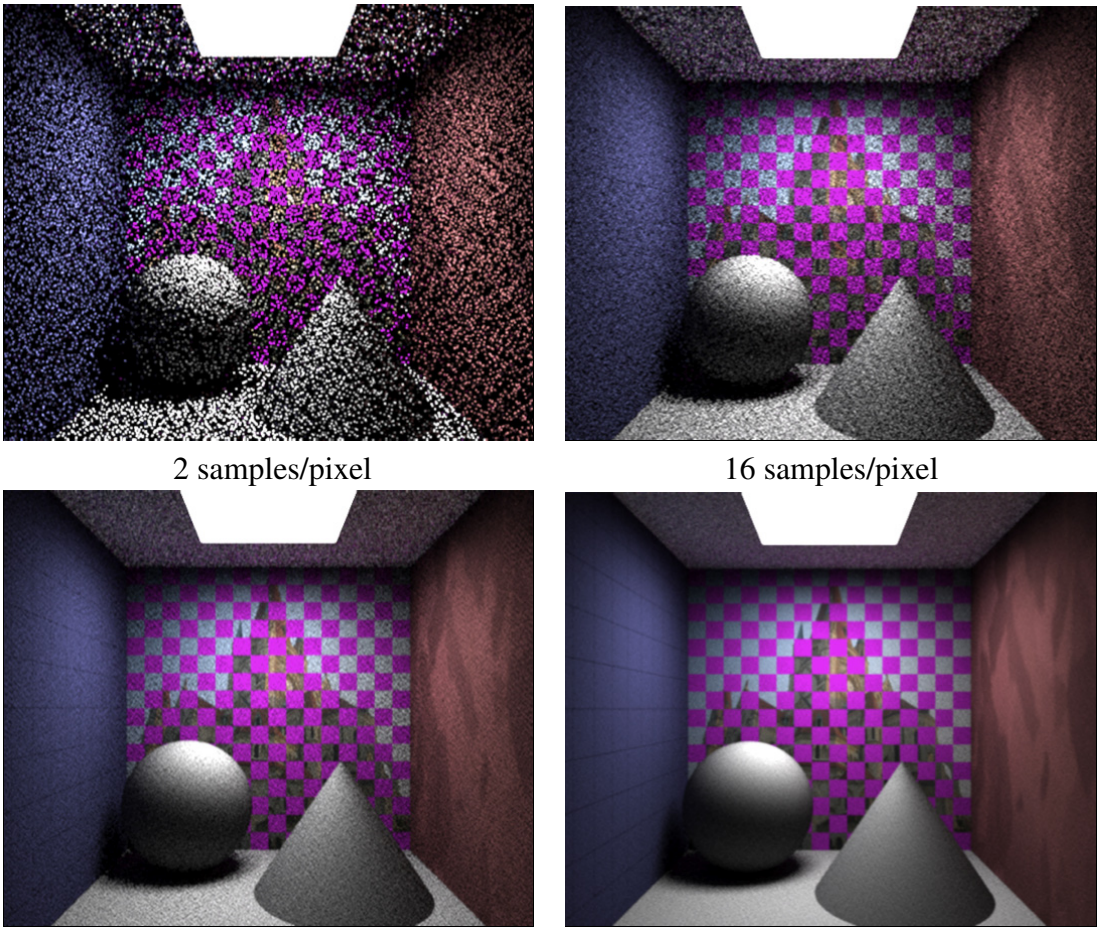
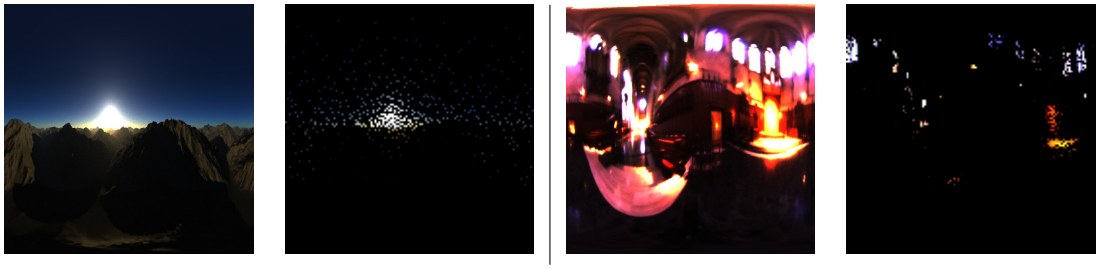


Figure 2.5: **Progressive Monte-Carlo Rendering.** As the number of samples increases, the quality of the approximation progressively increases, as well.

dimensions. (See [96] for a more detailed introduction to Monte Carlo methods as used in rendering).

Observing that each level of recursion in the LTE represents the application of a filter (the reflectance) to incident illumination, which represents the computationally intensive part of the rendering process, significant work has gone into *prefiltering* an *a priori* fixed illumination or reflectance of a convenient form. This prefiltered representation is stored and may be recalled as-needed during the rendering process.

Of these, *prefiltered illumination* algorithms assumed that the incident radiance $L(\omega \rightarrow x)$ (or some portion thereof) is held fixed. If the reflectance is also held to be fixed, the illumination and reflectance can be premultiplied, stored in an efficient basis under a



(a) Mountain Environment and Sampling (b) Cathedral Environment and Sampling

Figure 2.6: **Environment Importance Sampling.** When the lighting L_e is fixed, the complete illumination may be approximated by choosing *a priori* a set of lighting directions with high energy, as shown in the figures. Further, methods such as structured importance sampling and wavelet importance sampling prefilter the radiance within each region, assigning to the point at the center of each region the integrated contribution of all points.

convenient representation, and accessed as needed. For diffuse reflectance, the color of an illuminated surface is equivalent to the incident irradiance from its environment, and the function $f(\cdot)$ acts as a low-pass filter. Prefiltering fL thus results in a low-frequency result that can be stored using few coefficients. It is shown to be possible to reconstruct high-quality results of the incident irradiance function, parametrized by surface normal, with only the first 9 spherical harmonics (the spherical analog to the Fourier decomposition for Euclidean functions) [78]. Reflectance functions more complex than the diffuse require additional parametric dimensions, and similar methods have been applied to store this parametrization efficiently using a spherical harmonic representation [79].

For slightly more complex reflectance functions that can be parametrized in 2 dimensions, such as the commonly used phenomenological Phong model [75], a common approach is to create a prefiltered texture map parametrized by reflected direction. This map can be created by convolving a lobe from the Phong model with the incident reflectance directly in the spherical domain, through the use of the spherical harmonic basis. Alternatively, [41] presents method for approximating a spherical function in a rectangular domain using the *dual-parabolic* mapping, allowing the use standard Fourier-based filters. These have been extended to more general reflectance functions; in [49] a method is presented for a more general type of BRDF lobe, while in [14] the authors demonstrate a sparse sampling of arbitrary isotropic (3D) reflectance, and interpolate values to extend to the entire domain. Many of these representations require that the reflectance is fixed, as well as the illumination. Methods using summed-area tables [22]

preprocess a fixed illumination in such a form as to allow for rapid reconvolution with more arbitrary reflectance functions [42].

In order to allow for varying visibility, importance sampling methods such as structured importance sampling [2] and wavelet importance sampling [19] combine the preprocessed, partially-filtered approach of summed-area tables with the selection of a set of lighting directions for subsequent visibility testing. The directions preferentially selected as the regions of highest energy (see Figure 2.6); within the Voronoi region of each direction the radiance is preintegrated. Wavelet importance sampling extends previous methods by providing a multiresolution sampling that can be evaluated at multiple levels of detail. We will see more use of these in Chapter 4.

The previous class of prefiltering methods make the assumption that lighting (and possibly reflectance) is fixed, and precomputes partial results accordingly. An alternative approach is to pre-compute a composite filter of the $f_0 \cdot f_1 \cdots f_n$ term, and apply that to varying lighting environments. These methods, known under the term *precomputed radiance transfer* assume a constant geometry (with exceptions, to be discussed). Combined with a fixed form of reflectance, the algorithms are able to relight complex scenes with global illumination rapidly according to changing lighting conditions.

The largest variations among these methods are in their choice of basis representation, with certain ramifications thereof. The implicitly low-frequency spherical harmonic representations of the original PRT methods have been replaced by alternative bases with the goal of supporting both high- and low-frequency transport and illumination terms (so called *all-frequency*). For example, the methods of [62, 63, 90] use the Haar wavelet basis, a multiresolution basis that allows for varying scales of coarse-to-fine detail. In [11], the higher-order Daubechies 4-tap wavelet [23] is used for reflectance editing. While similar to the low/high frequency detail representable by the Fourier expansion, wavelet representations differ in that they allow for localized features, in contrast to the global effect of the trigonometric functions in Fourier representations (an introduction to wavelets in computer graphics can be found in [89]). Alternatively, another method uses a variable-width Gaussian basis for similar effect [35].

While many PRT methods expect that the sources of light are infinitely far from the receiver (so that $L(\omega \rightarrow x)$ for fixed ω and varying x is equivalent), this is increasingly generalized, for example to deal with medium-range changes in incident illumination [4].

Other methods have addressed the problem of partially deforming geometry, such as by using a subset of the spherical harmonic basis more amenable to rotations [86].

Turning now from the prefiltering methods, which take an *a priori* approach towards identifying the filtered rendering terms, other methods take an approach that combines partial evaluation of rendering terms resulting from ray tracing. For example, it is observed that the incident irradiance term $L(\omega \to x)$ often changes smoothly across the surface of visible objects. The high-frequency detail observed by the viewer is mostly a result of high-frequencies in f_0. In Figure 2.2, for example, while the scene (d) appears high-frequency, much of this detail results from the surface texture reflectance (a). The illumination itself contains mostly lower frequencies, which can be sampled at a lower rate and interpolated. If the LTE is sampled to accurately compute incident irradiance at a small sampling of points, these values can then be filtered to interpolate irradiance at all visible points. This method is termed *irradiance caching* [100], and is particularly useful for diffuse scenes, as the diffuse reflectance function acts as a low-pass filter. The additional rendering terms of depth and surface orientation are also used in the filtering step to avoid smoothing over boundaries in these terms. This method has seen many extensions and optimizations, for example, derivative information may be used to provide for better interpolation [99]. We use a similar method for soft shadowing in Chapter 4; in particular, we note that the change in illumination due to a soft shadow varies slowly over the image, allowing for shadowing computation at fewer points.

The method of photon mapping (originally presented in [44, 45], later given in extended form [46]) can be viewed as a spatial filtering of radiant flux in world space. Using "forward" ray tracing (that is, proceeding from the light, as opposed to the traditional "reverse" ray-tracing from the eye) the algorithm builds an estimate of incident radiant flux at a large number of points in the scene, storing energy as photons in the eponymous map. This very sparse estimate, however, may be interpolated spatially to compute an estimate of radiant flux at all points in the scene. This has shown to be particularly useful for reducing overall computation, as energy from nearby paths can be shared, rather than recomputed for each path. The path-outlier filter presented in Chapter 5 bears many similarities to photon mapping, especially in the use of shared information across paths, and the use of a density estimate. Where it differs is in generalization to higher-dimensional densities, especially to the joint density between both position and flux.

Filter Type	Filtered Terms	Assumptions/Limitations
Summed-Area Environment	L_e	Fixed, directional L_e, constant V
Structured Importance Sampling	L_e	Directional L_e, requires additional computation for correct V
Prefiltered Environment	$f_0 L_{e_1}$	Directional L_e, limited f, both of which are fixed, constant V
Precomputed Radiance Transport	$f_0 f_1 \cdots$	Generally implies static geometry, V and f; medium to far L_e
Irradiance Caching	$f_1 L_{e_2} + f_2 f_1 L_{e_3} \cdots$ n_{x_0}, x_0	f is diffuse, n_x is the surface normal term at x
Photon Mapping	$f_0 \cdots f_n \Phi$	Filters incident flux Φ after arbitrary n bounces through scene
Anisotropic Diffusion	L_0, n_{x_0}, x_0	
Bilateral Filter, α-trimmed Mean	L_0	
Stylized Shadow Filter	V_0	See Chapter 3
Subtractive Shadow Filter	$f_0 L_{e_1}(1 - V_0)$	See Chapter 4
Path-density Filter	$\rho(x, L)$	ρ is joint density, see Chapter 5

Table 2.1: **Rendering Filters Compared.** A number of filters are listed with the corresponding rendering terms on which they act, for purposes of comparison. We also state a number of limitations and assumptions inherent to several of the filters. The last three are the filters presented in Part II.

Finally, other methods apply filtering operations as a post-process on fully-evaluated L_0 in such a way as to reduce noise. Rather than reconstructing a pixel using the mean of all samples, which is prone to corruption by large values resulting from variance, the alpha-trimmed mean rejects those inconsistent with the remainder of the data [53]. The trimmed mean is referred to as a *robust estimator*, and such estimators are the focus of Chapter 5. Others use a method based on the bilateral filter as a robust estimator to remove effects from both atypical outliers and common noise [91, 103]. The non-linear energy-preserving filter of Rushmeier and Ward [82] operates by spreading the energy of very large samples across a wide range of pixels; the greater the energy, the greater the range. Such outlying values are identified by computing the effect of that sample on the variance of each pixel estimator, relative to some number of existing samples. The method of anisotropic diffusion [57] (which can be thought of as a generalization of bilateral filtering; see [9] on this point) operates similarly; high-energy samples are diffused across pixels in a region with similar distributions of sample values. We show a comparison of these rendering filters in Table 2.1.

2.5 Conclusion

Filtering methods are a powerful class of algorithms that see wide use in disparate fields. As we have shown, they play an important part in the rendering process. We find it useful to consider the rendering filter as presented here as being a distinct entity from either image-based or geometry-based filters, instead acting on the intermediate representation of the partially-evaluated rendering terms. By doing so, a rendering filter may leverage the strengths of both and combine them in novel ways that provide greater flexibility and efficiency than accomplished by either or both image and geometric filters acting independently. In Part II of this thesis, we present a series of rendering filters that have proven themselves useful in addressing certain common problems in computer graphics, and their exploitation of this novel intermediate representation is critical to their usefulness.

Part II

Applications of Rendering Filters

In Part I of this thesis, we have discussed our notion of the rendering filter and presented the theoretical foundation on which it is based. In discussing the development of the term, we have shown specific examples of filters in other domains, and have indicated its utility as a mental construct for addressing common problems experienced in computer graphics. In this Part, we now present specific, novel examples of rendering filters, based on the groundwork previously shown.

The first two filters focus on rendering of shadows, primarily in a rasterization context. Shadows play a key role in scene visualization, perception, and establishing mood. In Chapter 3, we present a series of *stylized shadowing filters* for creative and compositional control over the appearance of shadows. As discussed in the chapter, in many cases a "correct" shadow is inconsistent with either or both of the viewers' expectations or the presenters' intentions. The filters presented therein are analogous to several of the filters presented previously in the context of image and geometry filtering. However, as rendering filters, they may accomplish these effects in such a way as to be efficient for real-time, interactive feedback. In this context, rendering filters are an efficient and convenient formulation for presenters to achieve their compositional goals.

In Chapter 4, rather than a focus on stylistic effects, we demonstrate with the *subtractive shadowing filter* the use of filtering for efficiency. As we discussed in the previous chapter, shadowing poses a computational diffuculty for rasterization algorithms, as they require a variable evaluation of rendering terms at each pixel. This filter takes a step toward addressing this problem by enabling a form of continuous level-of-detail for shadow rendering Using a filtering approach, this combines high-quality and high-frequency prefiltered illumination with plausible low-frequency soft shadows. These methods rely on results derived previously from Fourier theory regarding the frequency content of signals. By separating high- and low-frequency rendering terms and filtering them independently, we leverage the strengths of methods individually tailored to each.

Finally, we turn from rasterization in order to apply our rendering filter framework to a common problem in ray-traced rendering, that of noise due to statistical outliers. Chapter 5 presents the *path-density filter* to allow for rejection of extreme values inconsistent with the empirical distribution of rendering terms; which, if not discarded, cause significant perceptual artifacts in the final renderings. By the use of this filter, we are able to achieve a significantly more perceptually accurate results, and do so in a manner which cannot be replicated with an image-based filter.

Chapter 3

Stylistic Visibility Filters for Creative Artistic Control

(a) Accurate Shadow (b) Shadow in Fine Art (c) Stylized Shadow

Figure 3.1: Traditional computer graphics algorithms produce "accurate" shadows (left). Artists often deliberately render abstract shadows, such as the shadow with reduced contour detail in this painting by John Vanderlyn, 1818 (middle). Our system offers controls for creation of stylized shadows (right).

While much research has focused on rendering physically-correct shadows, a "correct" shadow often exhibits unnecessary detail that distracts from the primary subject of the scene. Artists often prefer to have creative control over the rendered appearance of the shadow. This chapter presents a rendering filter that offers control over stylized shadows, based on four intuitive parameters – *inflation*, *brightness*, *softness*, and *abstraction* – that together support a broad range of effects. The filter can easily be incorporated into existing rendering pipelines, and is independent of scene geometry or shadowing method.

3.1 Introduction

Shadows play a significant role in our perception of the world. They provide cues for light direction and atmospheric conditions (whether sunny, cloudy, dusty, etc.), as well as position and shape cues inherent in projections of an object that augment that of the viewing transformation. Works of art and animation commonly include shadows to provide such cues, which are particularly important when the viewer lacks 3D cues such as stereo and parallax. In cel animation and other media where a character and background may be rendered in different styles, shadows play the crucial role of anchoring the character in the scene by showing where the character's feet meet the ground. Finally, shadows offer a mood cue for dramatic effect (cheerful, spooky, etc.), with a well-developed vocabulary for cinematographers [3].

Artists typically avoid the use of physically accurate ("correct") shadows for several reasons. Even in situations in which they are easy to produce, such as in computer graphics applications using global illumination solvers, the resulting shadows may be too detailed and therefore distract from the subject. Fortunately, people tend not to notice or care whether shadows are physically correct. This provides the artist a large design space ranging from "plausible" to obviously abstract. See Figures 3.1b and 3.2 for examples from fine art.

This chapter describes a method for stylizing shadows rendered in 3D scenes (Figure 3.1c). To control the visual characteristics of the resulting shadow, we introduce four parameters that in combination provide a broad range of stylization effects:

1. *Inflation i* controls the size of the shadow, relative to the original, such that increased i gives the effect of a shadow emanating from a larger version of the shadow-casting object.
2. *Brightness b* is the intensity of the shadow region when fully occluded (or the effect of indirect illumination).
3. *Softness s* indicates the width of the transition region from fully occluded to fully visible, simulating the effect of an area light.
4. *Abstraction* α is a measure of shadow's accuracy; lower values yield more detailed, accurate shadows, whereas larger values produce rounder, simplified shadows.

We present an image-based algorithm that uses these parameters to produce stylized shadows from a shadow matte (the $V(\cdot)$ visibility rendering term) as computed by standard techniques. Our implementation, which makes use of efficient Monte Carlo

Original Inflation Brightness Softness Abstraction

sampling techniques implemented on graphics hardware, provides interactive control of the light, camera, and shadow parameters, affording the artist a fluid environment for exploring this design space. Furthermore, because the algorithm and controls are based in rendering space, the design space itself affords more intuitive navigation for artists than would emerge from more conventional cinematic controls attached to lights in 3D. Another benefit of the approach is that it works for general visibility-determination methods, from shadow volumes [21] and shadow maps [101] used in a scanline-rasterization pipeline, to raytraced visibility used in an offline system.

Applications for this filter include computer-generated imagery for movies, cartoons, games, industrial and architectural design – essentially any context in which an artist is involved in the composition of the scene. For such applications, artists in practice already use a variety of ad-hoc methods for abstracting shadows, for example by blurring either shadow maps or the resulting mattes, or by rendering shadows from pre-simplified or stand-in geometry. Thus the main contribution of this work is to provide a set of intuitive controls that work well in concert, together with an efficient algorithm that implements these controls such that they may be explored in an interactive setting.

3.2 Related Work

In art and cinematography, the interplay of light and shadow has a long-standing tradition for dramatic effect. In both live action and computer generated film, artists have employed a broad set of tools to compose abstract representations of shadows [10, 55]. For example, a simple trick that has been used is to replace the "true" shadow casting geometry with a simpler form that is meant to suggest that geometry. In live action the lighting director may make use of a "gobo" (or "cookie", or "blocker") – a card with cutout shapes attached to the light and thereby casting shadows into the scene.

Where shadows have been used in traditional cel animation, they have been hand drawn, typically as light, blobby shapes that serve to anchor the characters in the scene

Figure 3.2: **Artistic Stylized Shadows.** Artists make use of shadow stylization for compositional purposes, as seen here (from the Metropolitan Museum of Art in New York). Note the use of highly simplified shadows in the left images, used to allow shadow cueing without adding distracting detail from the foreground object; in the bottom-left, only a simple, softened circle is used. In the right images, distinct umbra and penumbra are shown, with varying stylizations in each.

without distracting from the action. Petrovic et. al [73] developed a method for semi-automatic creation of shadow mattes for cel animation, based on the hand drawn artwork in the scene. While not directly concerned with abstraction, the shadows resulting from their system tend to be abstract for two reasons. First, they are based on abstract characters. Second, they are produced by casting lights from simple geometry created by "inflating" the hand drawn artwork. Central to our process for abstracting shadows is an inflation algorithm, which though different from that of Petrovic enjoys the property of creating smooth shadow mattes.

In computer cinematography, the lighting director applies a spectrum of tricks to control shadows through lighting for various artistic goals [10], for example making many or all lights respect the shadow map of the "key" light (to simplify cast shadows), "cheating" the shadow map away from the key light (to position the shadow for better dramatic composition), or inserting invisible, often simplified, geometry into the scene (for the sole purpose of creating shadows that would not otherwise appear). A recent system based on deep frame buffers allows artists to *interactively* explore the broad space of effects available from complex CG lights [71]. The method proposed here might well complement such a system, as it focuses specifically on interactive control of the abstraction of shadows. Finally, in computer cinematography a variety of image processing techniques are commonly used to adjust shadows in the final compositing stage, for example blurring the shadow matte to create softer shadows. Our framework also supports such effects, but in such a way that they can respond to geometric properties of the scene. For example, we support blurring less near occluders to simulate a narrowing penumbra.

Researchers have described various methods for controlling lighting parameters by directly manipulating the resulting shadows. For example, the method of Poulin and Fournier [76] allowed a designer to transform shadow volumes in wireframe view, while that of Pellacini et. al [70] let the artist drag shadows directly in an interactively rendered image. A number of researchers have developed techniques for optimization of lighting parameters in order to achieve various design goals (including shadows), e.g. [50, 83, 37, 52]. Together, these systems provide an array of intuitive controls for positioning and shaping shadows, but are not directly concerned with stylization or abstraction. While not considering stylization, the previous work of [25] has also considered post-render processing of the visibility buffer, as in our approach. This is done mostly for increased

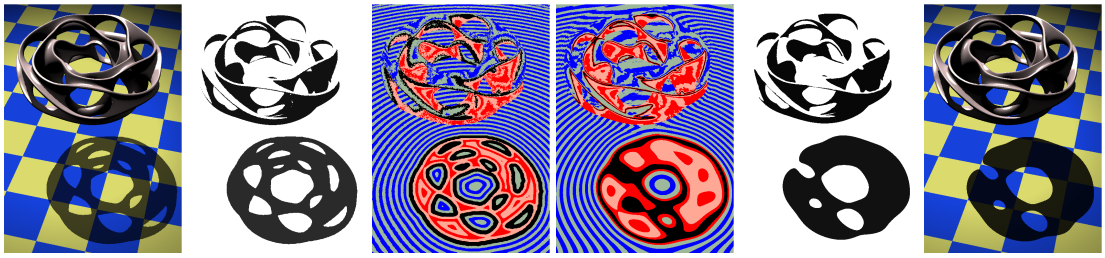


Figure 3.3: **Overview.** The accurate shadow (left) is filtered to produce a stylized shadow as follows: (1) using the visibility computation resulting from a conventional shadow algorithm, (2) a signed distance transform to the shadow boundary is found – red and blue stripes denoting negative and positive isovalues; (3) a blur filter is applied, followed by (4) a transfer function, such as a simple threshold, yielding a shadow matte which is used in (5) lighting the scene.

performance (visibility buffers are rendered at low resolution, and then resampled), but this work also demonstrates that the resampling step can be used to soften shadows.

3.3 Algorithm Description

For a scene with l directional lights, the standard method for computing the direct, shadowed illumination of a point x is:

$$L(x \to \vec{\omega}) = \sum_l \rho(\vec{\omega}_l \to x \to \vec{\omega}) S_l(x) L(\vec{\omega}_l \to x), \tag{3.1}$$

such that the exitant radiance $L(x \to \vec{\omega})$ from x towards $\vec{\omega}_o$ is computed from the reflectance ρ, incident radiance $L(\vec{\omega}_l)$, and a shadowing function $S_l : x \to [0,1]$. This scale factor denotes the proportion of lighting received from the l-th light, and in real-time applications is commonly the binary visibility $V(x \leftrightarrow \vec{\omega}_l)$ of the light relative to the point x. $V(\cdot)$ is commonly called the *shadow matte*.

We propose an alternative formulation of $S_l(x)$ for providing stylistic control over the appearance of shadows, by defining an operator on $V_l(x)$. Our algorithm applies the control parameters in five steps for each light l (shown in Figure 3.3):

1. Render the visibility buffer $V_l(x)$ corresponding to the l-th light.
2. Compute a signed L_p-averaged distance transform $D(V_l)$
3. Filter D with a Gaussian G, producing $G * D(V_l)$

4. Apply a transfer function f, yielding $S_l(x) = f(G * D(V_l))$.
5. Light the scene with $S_l(x)$ according to Eq.3.1

3.3.1 Control Parameters

Inflation. We first consider the inflation parameter i. By increasing the value of this parameter, we approximate inflating the original mesh by inflating the shadow represented in the matte, or rather, creating an offset curve at distance i from the original shadow. We compute a distance transform D applied to V, such that $D(V(x))$ is equal to the distance (according to a metric we define shortly) from x to the original shadow contour. This contour is intuitively the boundary between shadowed and unshadowed regions as indicated by V. Therefore the isocontour $D(V(x)) = i$ is equivalent to an inflation at distance i of the hard shadow, which we can represent by applying a threshold transfer function $f_i(D) = \text{threshold}_i(D)$. Note that once computed, this representation allows interactive modification of i, as we only require recomputing f from $D(V)$, rather than recomputing $D(V)$ itself. To allow for deflation, as well as inflation, we consider D as a signed transform by negating $D(V(x))$ for all points x such that $V(x) = 0$ (points that are shadowed in the original shadow matte).

Were we to use the standard Euclidean L_2 metric, however, offset curves would display cusps and other sharp artifacts due to the discontinuous nature of the distance field in the area of the medial axis. We would prefer a distance transform that is smooth, and thus we adapt an existing method proposed in the literature for inflation of meshes.

Previous work produced a surface representation of layered concentric shells, suitable for producing inflated or deflated representations [72]. Unlike the naïve approach of displacement along the normal (corresponding to the L_2 metric) shells generated with this method are guaranteed to be free from cusps and other visually unpleasant artifacts. The method is based on the *L_p-averaged distance metric* defined over R^3 relative to a surface. The parameter p allows for a trade-off between smoothness (with small values of p) and approximation to the Euclidean distance metric, which it approaches as p goes to infinity. While [72] addressed surfaces in R^3, we limit our use of this metric to curves defined in R^2, producing:

$$D_p(V(x)) = \left(\int_C \frac{1}{|x-y|^p} dy \right)^{-1/p} \Bigg/ \left(\int_C dy \right)^{-1/p} \tag{3.2}$$

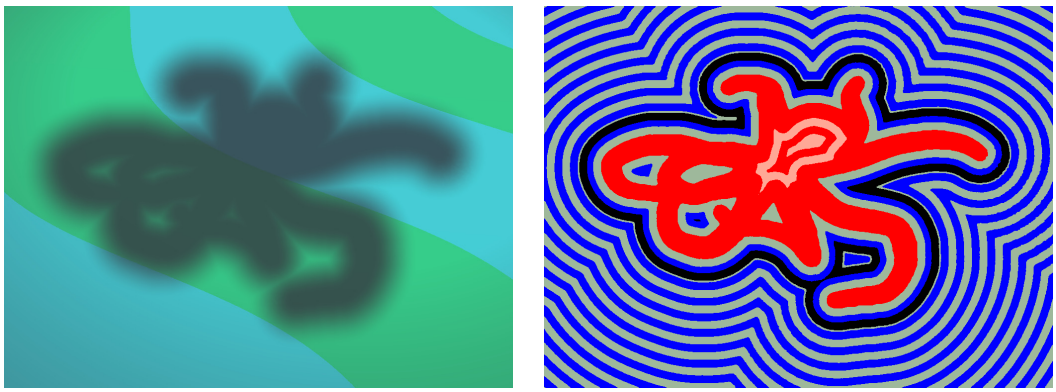
in which C is the shadow contour in V, and where the denominator is a normalizing term for the integral. In practice, we have found that $p = 8$ provides a practical trade-off between smoothness and accuracy, and is used for the results in this paper. Unlike L_2, this metric produces a distance transform D that is free from discontinuities and medial axis effects. Furthermore, the average transform exhibits better temporal coherence than the the Euclidean transform, since it is not as sensitive to changes in the Voronoi structure of the sample points. In Figure 3.4 we show a comparison of the isocurves for the Euclidean L_2 transform and the L_8-averaged transform. Figure 3.5 then demonstrates the effect of varying the inflation parameter. Note that as the shadow is inflated, it maintains a smooth, visually appealing shape.

Brightness. We will use the brightness term, b, to represent the effect of ambient lighting in the shadowed region. Implicitly, if there exists some light reaching the area fully in shadow, it has reached it through indirect means. Brightness represents the "darkest" a shadow can be, or in our framework the minimum of the transfer function f and therefore the lowest visibility. Note that, without loss of generality, we do not assign a parameter for the maximum value of f, which is always 1; an object outside of the shadow is fully visible to the light. To attenuate the lighting at that object, one would either inflate the shadow, or correspondingly decrease the power of the light.

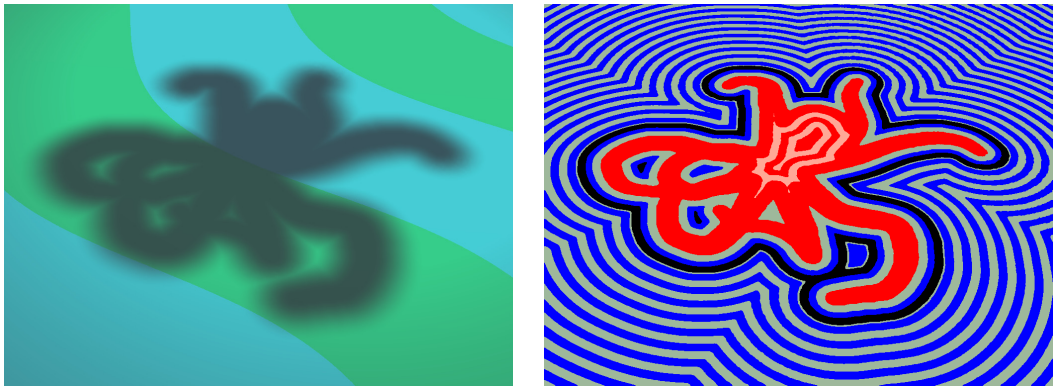
We can use the brightness parameter to combine multiple lights, which generalizes to multiple regions of the same shadow, such as a highly stylized umbra and penumbra. This effect can be achieved using a low brightness shadow for the umbra, and an inflated or softened shadow with higher brightness. When visibilities are computed separately, and used to modulate lighting that is accumulated in the final rendering, this produces effects reminiscent of shadows (see Figure 3.6 as compared to works in the right column of Figure 3.2).

Softness. After applying the inflation and brightness parameters, we are still left with hard shadows, possibly inflated or deflated from the original. The softness parameter is used to add continuous variation in intensity to the shadow rendering.

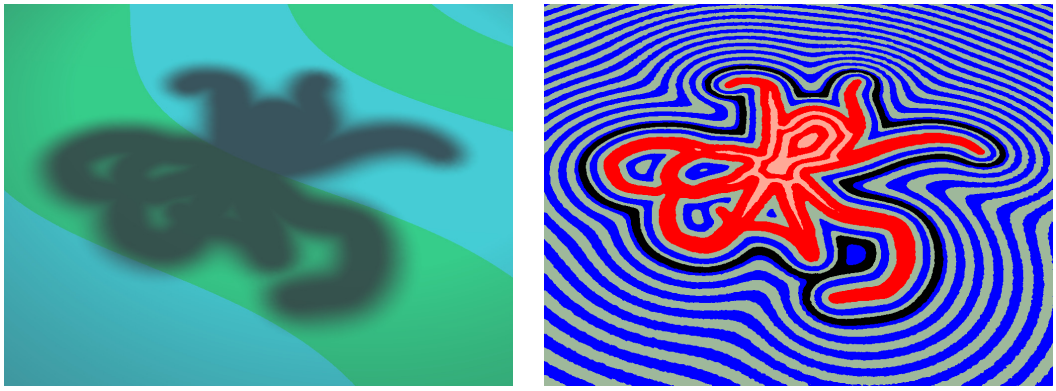
In physically-based rendering, the softness of a shadow is proportional to the area of the light source; rather than viewing $V(x)$ as a binary-valued function in which a light is either visible or occluded relative to a point, visibility instead varies continuously across the penumbra region. We can consider then, two separate shadow contours: one deflated from what would be the location of the hard shadow, delineating the umbra from the penumbra, one inflated, delineating the outermost boundary of the penumbra.



Euclidean Distance, Screen Space

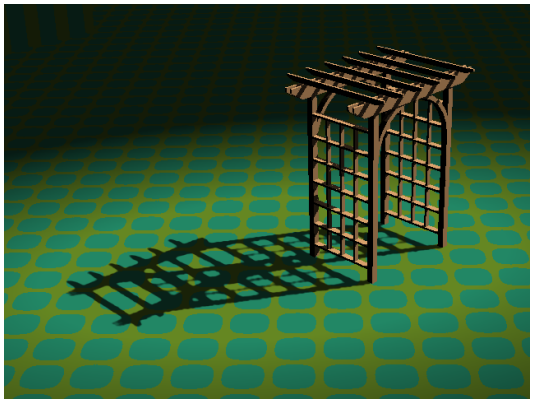


Euclidean Distance, World Space

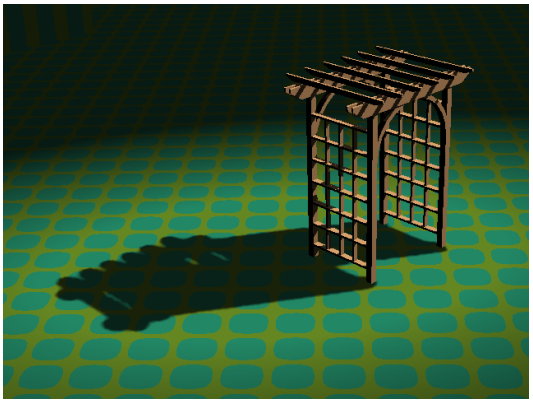


L_8-Averaged Distance, World Space

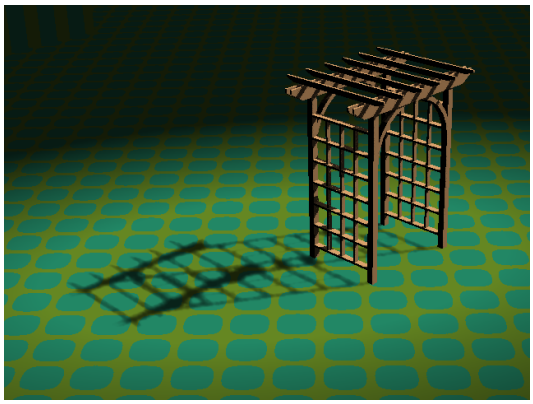
Figure 3.4: **Distance Transforms.** Distance transform of the original hard shadow matte (red region) of the Octopus scene (shown in Figure 3.9). The black line represents the inflated shadow contour. The Euclidean metric in screen space displays discontinuities in the soft shadow contours, and does not account for perspective foreshortening or the orientation of the ground plane. While the Euclidean metric in world space corrects the latter, unpleasant discontinuity artifact is clearly noticeable. The L_8-averaged metric in world-space corrects both problems.



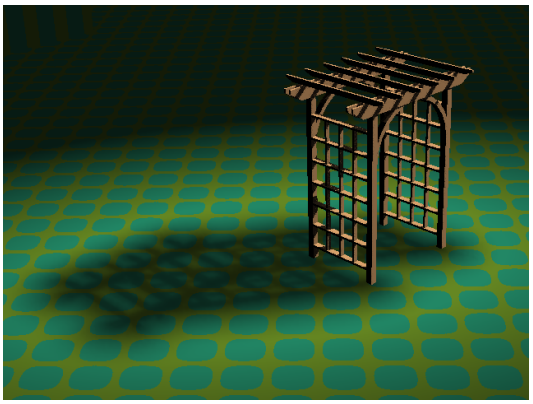
Hard Shadow



Inflation
$i = 20$, $s = 5$



Deflation
$i = -10$, $s = 5$



Inflation and High Softness
$i = 20$, $s = 50$

Figure 3.5: **Inflation.** From the original shadow matte, we can produce alternate shadows of various inflations, either using a constant value of i, or varying i according to the approximate occluder distance.

Figure 3.6: **Umbra and Penumbra**

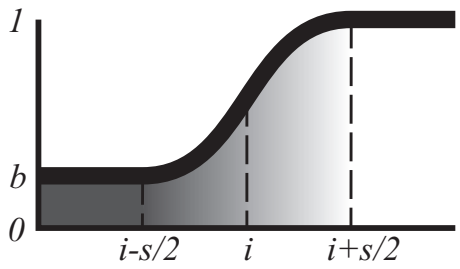


Figure 3.7: **Transfer Function**

Using our distance transform representation, we can extract these contours directly, and because of our use of the smooth L-averaged metric, the shadow intensity will vary smoothly between the two contours. Therefore, we can define softness as the width of the transition from $f(D) = b$ to fully visible $f(D) = 1$.

Given these parameters, we can now specify our complete transfer function. We assume a monotonic falloff from the center of the shadow outwards, and prefer that the function be at least C^1 continuous to avoid visual discontinuities. We choose the following, where smoothstep is a C^2 Hermite polynomial as commonly defined in shader languages (see Figure 3.7 for an illustration).

$$f(D) \quad = \quad (1-b)^{-1}\,\text{smoothstep}(D, i-s/2, i+s/2)+b. \tag{3.3}$$

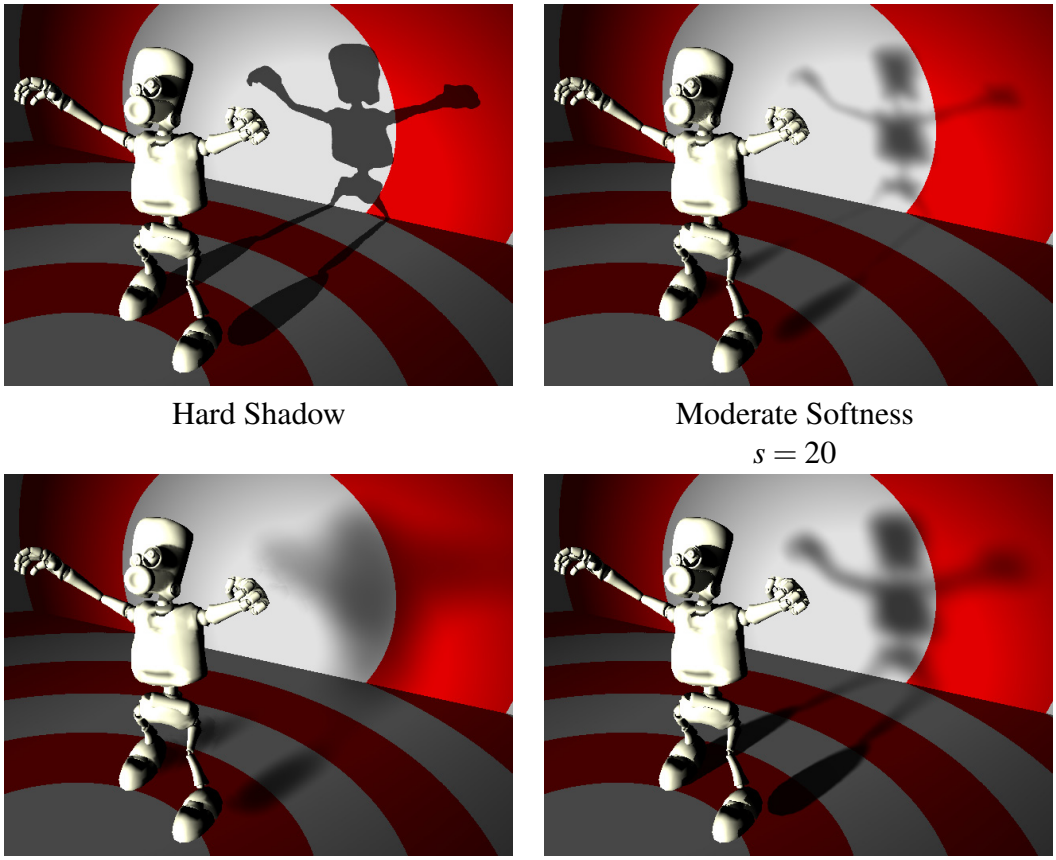
Abstraction. We define that the original hard shadows are least abstract, and the tendency towards a perfect circle to be greater abstraction. Therefore, we will define the abstraction parameter α as a limit on the curvature detail of the visibility isocurves. As the abstraction parameter increases, the isocurves lose sharp detail and become rounder.

We implement abstraction by convolving the distance function with a Gaussian kernel of standard deviation α. Under appropriate conditions (away from the medial axis) this has the effect of applying a low-pass filter to the isophote (contour normal) curvature, in which nearly all of the high-frequency curvature detail has been attenuated.

Figure 3.9 shows the result of filtering the L_8-averaged isocurves seen previously in Figure 3.4, causing the previously sharply-curving shadows to become smoother and more rounded and leading to a "blobby" or cartoon appearance. However, note that they still maintain their hard transition from light to dark and do not appear blurred, thus mimicking the shadows frequently seen in traditional cartoons.

3.3.2 Higher-order Rendering Terms

As described so far the algorithm proceeds entirely in screen space without considering any world-space geometric information for visible points x. However, the use of additional rendering terms provide more intuitive shadow behavior, although it is conceivable that another implementation might strictly use image-space information for stylistic effect. The three additional terms our system uses per pixel are the world-space position, normal, and approximate distance to occluder. As a practical matter, these can be computed at rasterization time and stored, and thus add trivial overhead.



Hard Shadow

Moderate Softness
$s = 20$

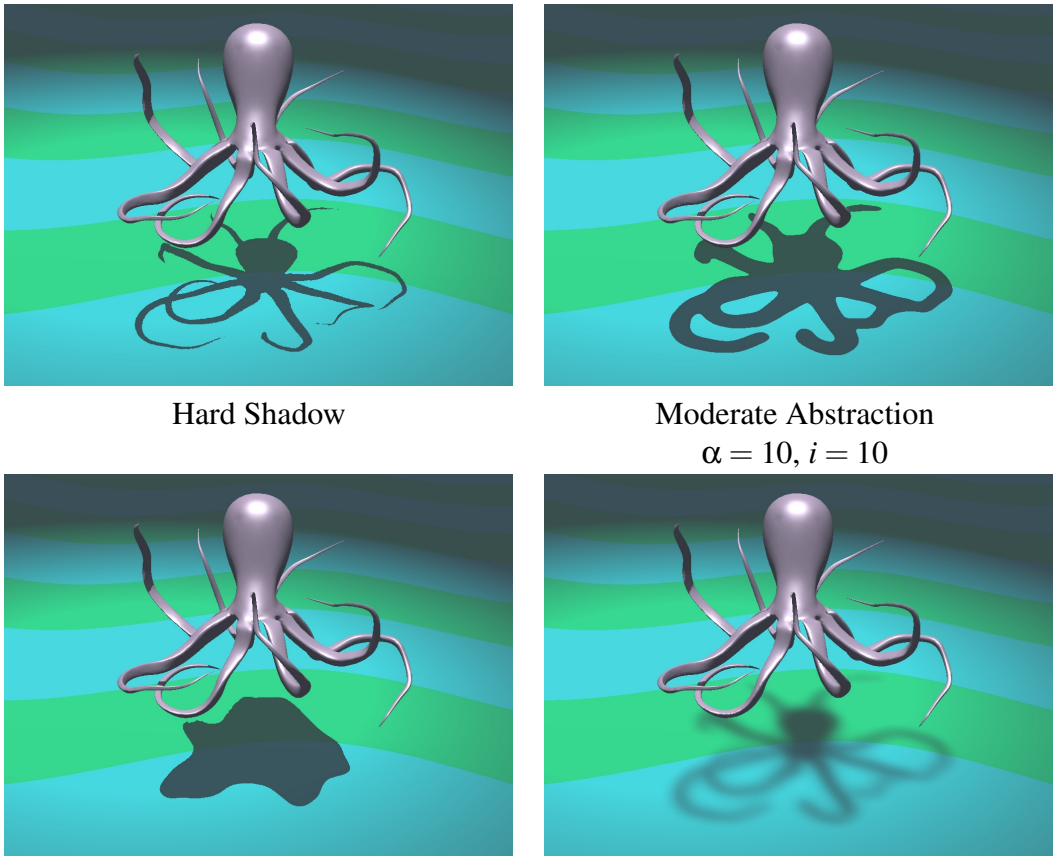
High Softness and Abstraction
$\alpha = 50, s = 50$

Softness Varying With Distance
$\alpha = 10, s = 20d^2$

Figure 3.8: **Softness.** We can approximate the effect of a soft area light by changing the softness parameter. The bottom-left figure uses geometric information to harden shadows near shadow casters as discussed in Section 3.3.2

World-space Distance Metric. In our computation of the L_8-averaged distance transform $D(V)$, we interpret distances in world space, rather than screen space, by using the stored 3D position of each pixel. This allows for important effects such as proper foreshortening of shadows away from the camera. This effect is clearly noted in Figure 3.4. Additionally, this prevents the shadowing from areas nearby in screen-space, yet distant in world-space, from bleeding across each other. As can be seen in the illustrations, especially Figures 3.8 and 3.9, an unshadowed foreground object may cross a shadowed background without artifact on either object.

Normal Discontinuities. To prevent artifacts resulting from sampling across discontinuities, we apply an angular threshold based on the normal when computing the distance transform and blur. If the angle between the normal of x and any point y is greater than



Hard Shadow

Moderate Abstraction
$\alpha = 10$, $i = 10$

High Abstraction
$\alpha = 70$, $i = 10$

Abstraction and Softness
$\alpha = 20$, $s = 20$

Figure 3.9: **Abstraction.** By applying a Gaussian filter to the distance transform, we are able to reduce the high detail in the contours, allowing the later thresholding step to produce rounder, more abstract shadows. Compare these to the results achieved using the analogous mesh filters in Figure 1.13.

45°, we ignore y for the purpose of computing the distance metric or blur. This has some intuitive, if not more rigorous, basis as lighting is a function of normal, and so different normals will lead to different illumination. Importantly, it works well in practice, and we have used this for all illustrations.

Non-constant Stylization Parameters: In physically realistic rendering, we note that certain properties of shadows vary relative to scene geometry, such as an increase in softness and brightness of a shadow at greater distances from the occluder. Therefore, we allow our stylization parameters to vary as functions of such geometric information.

Our method would easily allow for functions of arbitrary parameters, though we focus our implementation on functions of the approximate distance of a point x to its occluder,

which we denote $d(x)$. To motive our desire to allow the shadow parameters to vary with $d(x)$, we observe that to mimic realistic shadowing, softness should increase with $d(x)$, as the projected area of the occluder decreases relative to the projected area of the light. Brightness should also increase with $d(x)$, as more ambient illumination reaches x. This combined effect is often referred to as “hardening" of the shadow near occluders, and is an important perceptual cue. We demonstrate its use in Figures 3.8 and 3.11. We have found that quadratic functions of $d(x)$ allow suitable control; more complex functions could be used as needed.

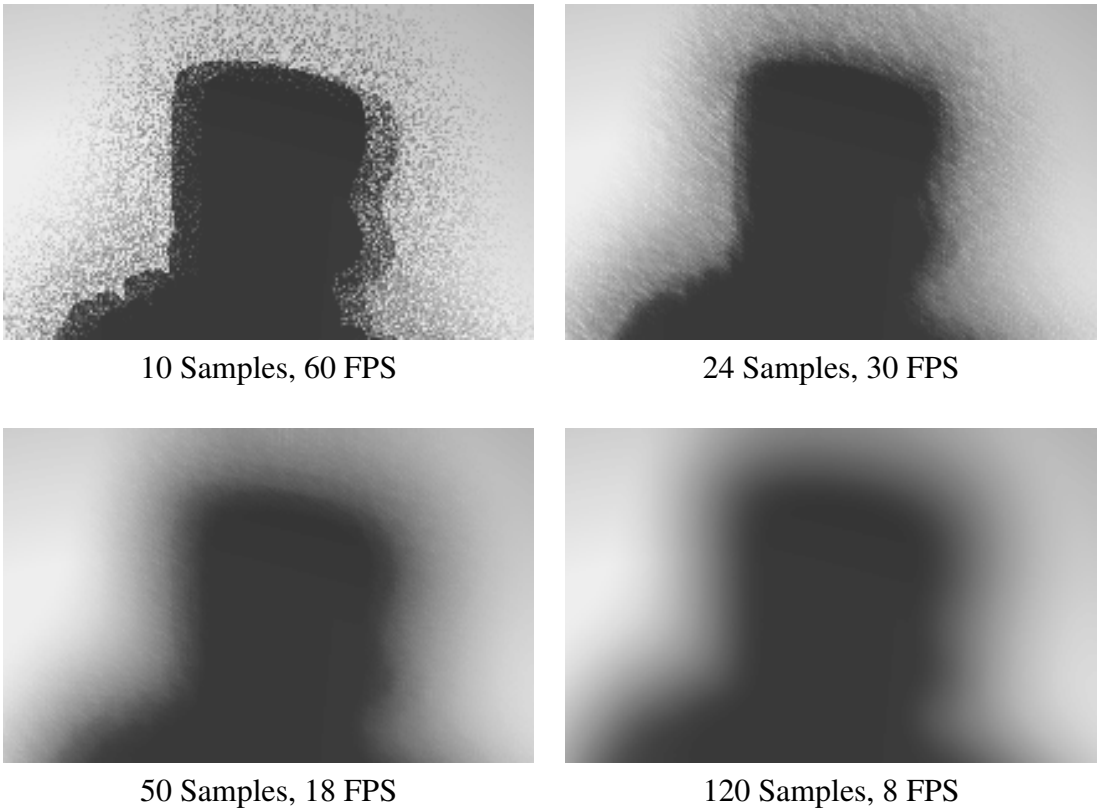
The determination of accurate distances from points to occluders is not a simple problem; however, we have found that for the purpose of setting shadowing parameters, only rough approximations are required to achieve a wide variety of useful stylistic effects. While the algorithm to compute $d(x)$ is orthogonal to our method as a whole, for the figures shown here we use a simple distance between the shaded point and a coarse occluder geometry represented by a small number (1 to 2) of points or lines that is computed in the shader on rasterization. We stress that other implementations could use more accurate distances, such as those computed with raytracing.

3.3.3 Monte-Carlo Filtering

Both the averaged distance transform (Step 2) and the Gaussian blur (Step 3) require integration over a potentially large domain of the shadow matte. In order to implement these efficiently, we perform the algorithm in a GPU fragment shader using a probabilistic Monte Carlo approach. Note that while rendering efficiency is a secondary concern to stylistic control, it is nevertheless useful in rapidly configuring parameters in order to develop an artistic style. We use a slightly different approach in each step.

Distance Transform. The L-averaged distance is defined on the distance from a point x to a contour C; however, using a uniform spatial sampling over the entire shadow matte V, the probability of finding a pixel $y \in C$ is low. Furthermore, the distance transform must be defined a significant distance from the shadow boundary, in order to allow for shadows with high inflation or softness, requiring sampling over a large region. Clearly, a uniform sampling of V would be inefficient.

To address this problem, we selectively sample only over pixels on shadow boundaries. Currently, we implement this by transferring the visibility buffer V from the GPU back to the CPU, detecting the edges on the CPU, and sending back to the GPU a list of



10 Samples, 60 FPS | 24 Samples, 30 FPS

50 Samples, 18 FPS | 120 Samples, 8 FPS

Figure 3.10: **Performance vs. Quality.** As we take fewer samples, we can see that the quality of the image relative to the reference (measured as root-mean squared error per pixel) decreases; however, the frame rate increases significantly as well.

coordinates of pixels in C. When rendering a pixel $x \in D(V)$, the GPU shader performs Monte Carlo integration by sampling a random set of pixels $y \in C$, and using them to compute the L_8-averaged distance to x.

Gaussian Blur. The Gaussian blur may also need to sample over a large portion of the image; fortunately, however, its effect is limited by the falloff of the Gaussian filter kernel. We combine Monte Carlo integration with a windowed approach: each pixel x will randomly sample a disk of pixels of radius 3α around x, where α is the abstraction parameter and filter standard deviation. We have found that both importance sampling the Gaussian kernel, and using a quasi-random Halton distribution [77], instead of a uniform distribution has a noticeable effect on reducing sampling noise. Additionally, while combined 1-D filters cannot be used to reduce the number of operations, as the function contains discontinuities in world-space and therefore the kernel cannot be separated, we can perform two 2-d convolutions of $\sqrt{1/2}$ the width to achieve the same result as a

larger kernel, with significantly fewer computations. Because of these optimizations, the Gaussian blur causes only minor overhead.

Sampling. By using Monte Carlo integration, our approach allows for a continuous time-quality trade-off; while the performance is good even at high quality, fewer samples can be used for slightly less accurate results when previewing shadows. Further, we only need to resample the distance transform (the slowest-converging and therefore most time-intensive portion of our algorithm) when the camera or light changes. Inflation, brightness and softness can be modified without refiltering, allowing for fast ($>$ 100 fps for many scenes) modification. Abstraction requires only a recomputation of the rapidly-converging Gaussian blur, and which can be updated in real-time ($>$ 50 fps). Additionally, we can automatically reduce the number of samples while moving camera or light to allow for rapid preview.

We provide performance figures for different sample counts in Figure 3.10, from an implementation of our algorithm running on a 3Ghz Intel Core CPU, and a GeForce 8800 GPU. This is taken from the "worst-case" situation for our algorithm: moving the light or camera with high softness and no abstraction, which would otherwise remove the noise introduced by the distance transform.

3.4 Discussion

We have presented a system for flexible shadowing that allows a wide array of stylized effects. As seen in Figure 3.11, the style of the shadows may be adapted according to the control of the artist, and may be used to produce widely different styles of rendering. The most significant shortcoming of our work is the algorithm for computing the L-averaged distance transform, which is currently the slowest part of our algorithm. Note that, interestingly, the CPU-GPU communication is not the bottleneck; rather, it is the large number of samples required for low-variance distance computation. This suggests that one solution is to use biased methods that trades accuracy for reduction in noise.

We note that many of the choices we presented for our system could be generalized in alternate implementations. For example, we made the assumption that the shadow softness always exhibits a monotonic fall-off from the center, but this need not always be the case. The isocurve renderings in this paper were created by substituting an appropriate stepped transfer function; other artistic effects could be created similarly. Additionally, control parameters could be made functions of the surface material properties, or of

the viewing angle to approximate specular reflectance. However, we believe that this implementation provides intuitive control for a majority of applications.

Our implementation contains certain limitations that would need to be addressed for use in a production environment. First, per-object controls over shadow parameters would be required. Moreover, even with support for multiple objects, static parameters for an entire scene might not allow for flexible artistic control; parameters that achieve the desired result for a given combination of light direction and camera angle may be inappropriate for others. Therefore, we would suggest keyframing of the stylization parameters as a reasonable solution, effectively making the parameters a function of time. Practical implementations might also control the parameters as functions of other variables, such as light and camera direction, or as non-linear functions. Finally, although we consider the parameters discussed as the most intuitive, one could consider adding additional parameters, such as anisotropic elongation of the shadow away from the light or additional possibilities for non-photorealistic rendering of the shadows (e.g., outlining, hatching, or stippling).

Another disadvantage of the current work is the lack of coherent support for shadows that are already soft, resulting from area light sources. While the simplest adaptation might be to sample the light and apply our stylization transfer function for each sample independently, this would not be computationally efficient. Instead, we believe that it is possible to express the result of the stylization computation in terms of the area-light shadow intensities themselves, as well as their gradients. We discuss other potential future work in Chapter 6

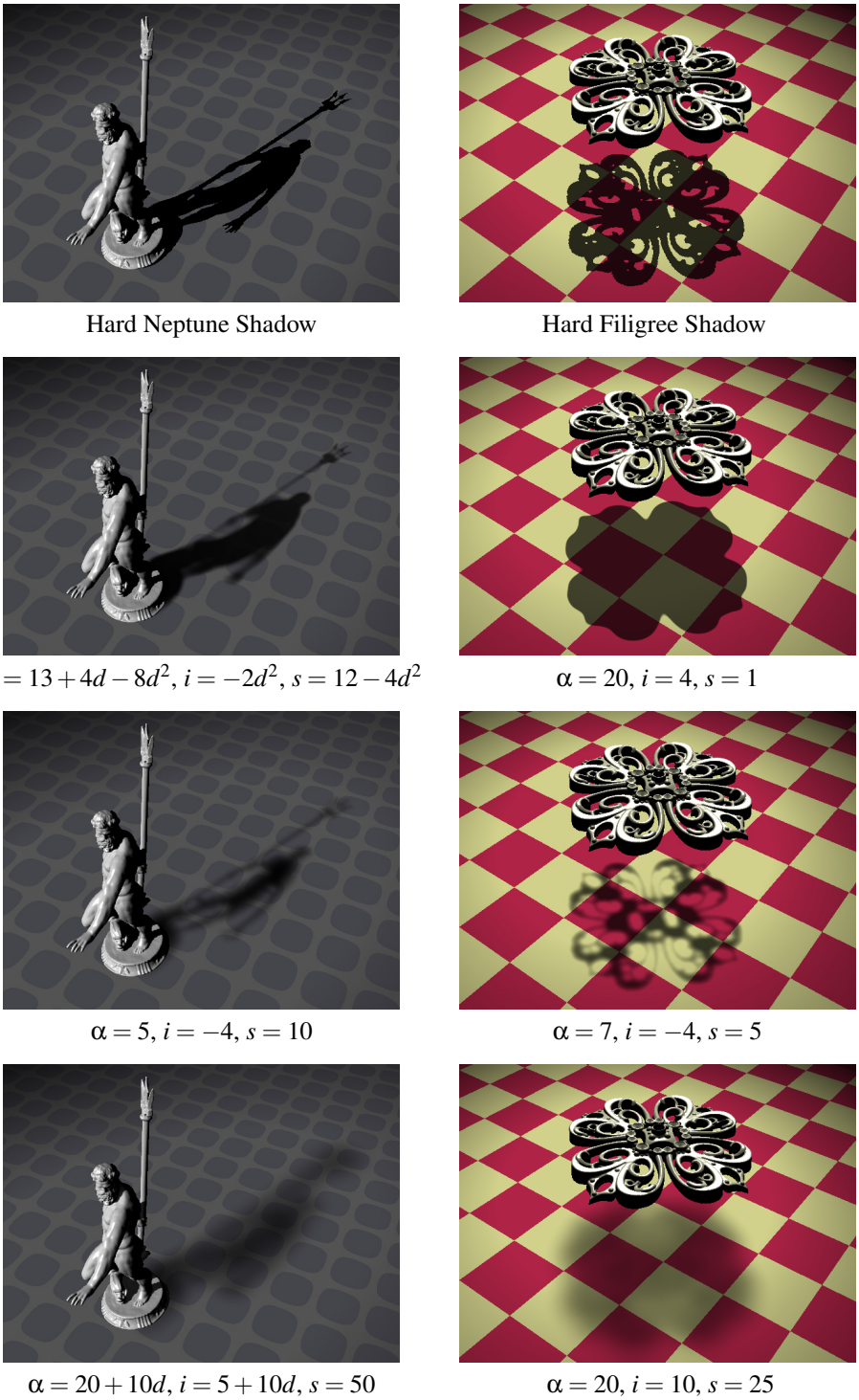


Figure 3.11: **Various control parameters.** The first row shows the hard, unabstracted shadow. The second row shows abstracted but hard shadows. Note despite abstraction, critical detail (such as the trident) is preserved. In the third row the shadow is shrunk and softened while the character of the shape is preserved. In the final row the shadow is blurred and lightened to remove detail, while maintaining a sense of the light position.

Chapter 4

Subtractive Shadow Filters for Flexible Illumination

The subtractive shadow filter presented in this chapter is a method for progressive shadow level-of-detail, analagous to the image compression and geometric simplification methods demonstrated in Chapter 1 (in particular Figures 1.9 and 1.14). Recall that these methods rely on a principle of the Fourier theory presented in the same chapter, the sampling theorem, which states that low-frequency signals can be represented with fewer samples than their high-frequency counterparts. The utility of these methods derives from the fact that, in these particular applications, low-frequency information has the predominant impact on perception, such that removal of some high-frequency detail is often unnoticeable or insignificant. Given that the the rendering process can be viewed as a sample-based reconstruction of the light transport equation, as we saw in Chapter 2, it is reasonable to consider using reduced samplings to reconstruct low-frequency rendering terms. In this chapter, we will employ such a method so as to improve the performance of real-time rendering with cast shadows.

The rendering filter framework suggests that one should consider the relevant rendering terms to be used in a given reconstruction, identify those of varying high- and low-frequency content, and sample them independently. Limiting our discussion to the reconstruction of the one-bounce direct illumination $f_0 L_{e_1} V_0 = L_0$, we see directly that final illumination L_0 incident to the camera plane contains significant high frequencies (Figure 4.1d), and so is unsuitable for reconstruction using reduced sampling rates. Fol-

lowing the filtering principle of decomposition into frequency bands, we instead compute:

$$L_0 = f_0 L_{e_1} - f_0 L_{e_1}(1 - V_0), \tag{4.1}$$

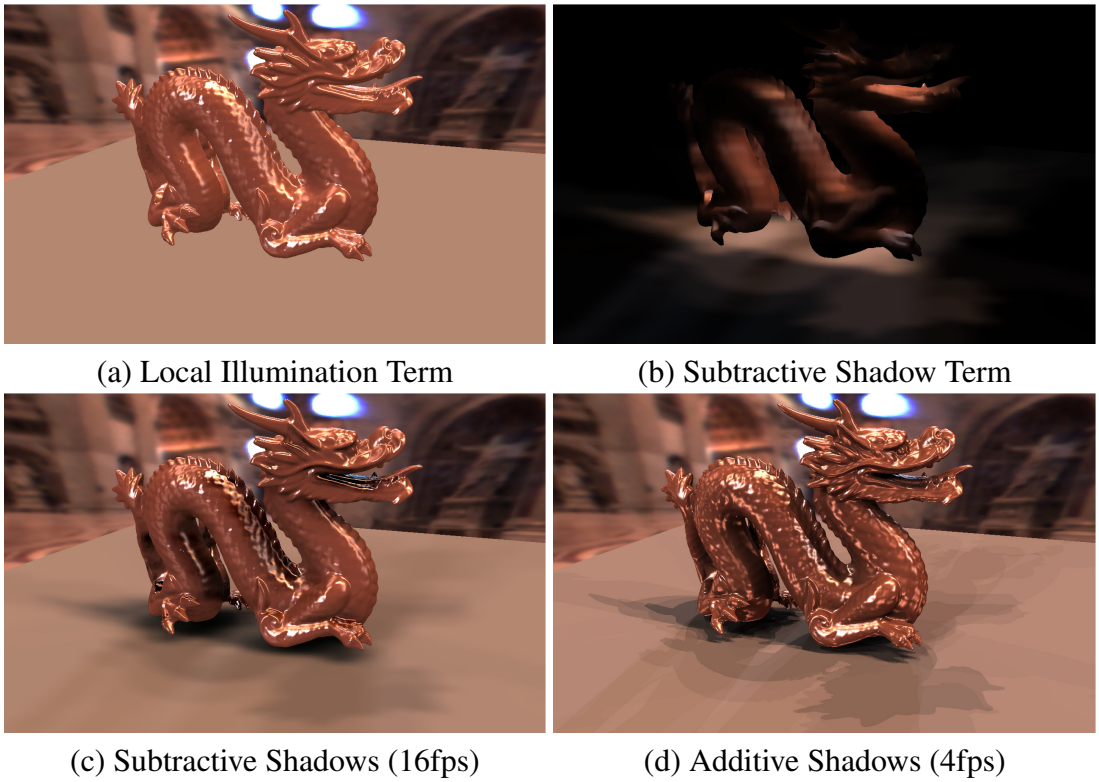
where $f_0 L_{e_1}$ is the local illumination (i.e. the unoccluded direct illumination, shown in Figure 4.1a) and the novel term $f_0 L_{e_1}(1 - V_0)$ is the extraneous light present in the local illumination (b) from which it is subtracted to produce the shadowed result. This formulation is equivalent to the original, but allows the local illumination term and the subtractive shadow term to be reconstructed and filtered independently in a manner that allows the most efficient manner of computation for each.

As discussed previously, extensive methods exist for prefiltering $f_0 L_{e_1}$, allowing a rendering of complete unoccluded direct illumination to be performed very quickly. This term, therefore, will be computed in our framework using such methods. Turning our attention to the subtractive shadow, we see that the term displays a dominance of low-frequency content; we will show that this may be computed by sampling at a lower rate (requiring fewer computations) and performing a resampling operation. We also offer a discussion of why the additive lighting representation of L_0 cannot be likewise subjected to efficient filtering methods.

This technique preserves that portion of the scene with the greatest visual importance – the high-frequency direct illumination – and allows shadows to be presented with lower fidelity in exchange for improvements in speed. This provides for flexible control over level-of-detail in shadow computation, with the capability of reducing both of lighting and camera samples while maintaining a reasonable approximation of the original. With the subtractive shadow filter, we are able to interactively manipulate and render arbitrary BRDFs and environment maps applied to complex, dynamic scenes with shadows, achieving in real time effects that previously required offline preprocessing.

4.1 Background

As we have discussed in Sections 2.3 and 2.4, while ray-traced rendering methods can compute the global illumination of a scene, and so inherently determine which regions are in shadow, most interactive rendering is performed using rasterization methods that instead utilize local illumination calculations. These methods do not inherently consider occlusion, the rendering term $V(\cdot)$ that varies over all points in the scene, so additional



(a) Local Illumination Term (b) Subtractive Shadow Term

(c) Subtractive Shadows (16fps) (d) Additive Shadows (4fps)

Figure 4.1: **Color and Shadow Buffers.** Even with low lighting detail (25 lights), subtractive shadows preserve high accuracy in the direct illumination along with soft shadowing. By contrast, additive shadowing produces an inaccurate speckled effect on the dragon due to undersampling, while the shadows are unpleasantly hard. In addition, by decreasing the shadow sampling detail to $1/16$, subtractive shadowing preserves real-time frame rates for this high resolution (1280x800) rendering.

techniques have been developed to simulate shadowing. The standard approach renders the scene illuminated by each light in turn, while limiting the effect to pixels visible from that light, which are identified as such by a *visibility determination* method (such as the well-known shadow volume [21] and shadow mapping [101] algorithms). The results from each light are accumulated to produce the final image, and so we refer to this as "additive" shadowing. We explore, however, the effect of doing the opposite: computing the complete, unoccluded local illumination of the scene, then, for each light, subtracting from the scene the energy occluded by cast shadows.

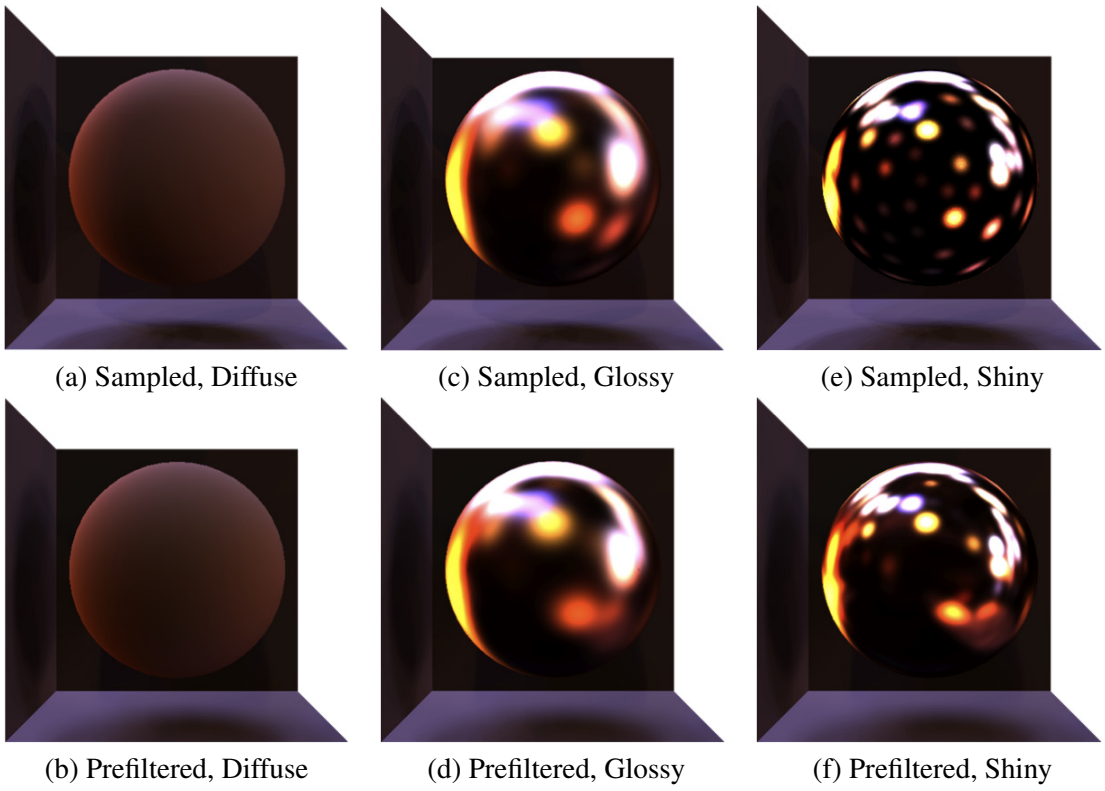
Our technique is intended to address complex illumination from natural lighting environments and realistic reflectance models, applied in an interactive setting that allows animation and editing. While illumination from small numbers of discrete point lights can be computed individually, real-world illumination and reflectance are defined by

continuous functions over the entire visible hemisphere, for which per-light methods are ineffective. To accurately render such lighting environments (commonly represented as tabulated spherical functions, or *environment maps*) researchers have developed several sets of techniques, of which we have provided an overview in Section 2.4. *Environment sampling* reduces the continuous function to a set of important directional lights; while the result is simple to render, it cannot account for the complex continuous distribution of incoming light – especially noticeable for non-diffuse materials. *Prefiltered illumination* methods allow the complete local lighting to be computed in constant time, yet they are ignorant of non-local geometry, and so are unable to represent cast shadows. *Precomputed radiance transfer* methods do consider geometry and visibility, but they do so only at the cost of significant offline precomputation, and so are unable to support dynamic geometry or materials.

Instead, we present a technique designed to support lighting and shadowing from realistic environment maps without significant preprocessing. It does so by leveraging the strengths of prefiltered illumination methods for direct lighting, with environment sampling for "subtractive" shadowing (Figure 4.2). It is based on the observation that the high-frequency direct illumination of the scene is of primary visual importance, and that the low-frequency shadowing, while providing essential visual cues, is secondary. By separating the two, our technique preserves direct illumination in full detail, yet allows the rendering system to perform a trade-off between shadow quality and speed – if a higher frame rate is needed, an interactive system can lower the level of detail present in the shadows until the target is reached. This is inspired by existing work on image compression and geometric simplification (see for example Figures 1.9 and 1.14), in which a particular level-of-detail (LOD) can be selected that approximates the original, yet is more efficient to store or faster to render – incurring rendering error for improvement in speed or size. Analogously, in many cases sacrificing accuracy of shadow computation in exchange for improved rendering speed is an acceptable trade-off, so long as the visual fidelity of the direct illumination is maintained. The technique of subtractive shadows does just that, as we will demonstrate.

4.2 Algorithm Description

We assume the availability of algorithms for prefiltered illumination, shadow determination, and environment map sampling (we will discuss this further when presenting



(a) Sampled, Diffuse (c) Sampled, Glossy (e) Sampled, Shiny

(b) Prefiltered, Diffuse (d) Prefiltered, Glossy (f) Prefiltered, Shiny

Figure 4.2: **Comparisons of Sampled and Prefiltered Illumination.** When using a sampled approximation, rendering speed is dependent on the number of lights, therefore fewer is preferable. With 100 lights, additive diffuse illumination (a) compares well to its prefiltered analogue (b); as specularity is increased (c) the illumination shows slight loss of continuous detail compared to prefiltering (d). For very shiny surfaces (e), illumination is very poorly represented, while the prefiltered illumination is highly plausible (f). Note as well that the shadows are more plausible after filtering, as well.

examples in Section 4.3). Our technique is as follows, and is illustrated graphically in Figures 4.1 and 4.3.

1. Sample environment map to create an approximation E
2. For each (shadow-casting) light $L \in E$:
   (a) Render radiance cache, if used for prefiltered illumination
   (b) Determine shadows of scene with respect to L
   (c) For each *shadowed* pixel, compute contribution of L
   (d) Composite into shadow buffer S, possibly lower resolution than color buffer
3. Render prefiltered, unshadowed direct illumination into color buffer C
4. Filter S to produce resampled S', equal in size to C
5. Subtract S' from C, producing final, shadowed, output

The benefit of this procedure is that it allows for two time-quality tradeoffs, parameters that we refer to as the *illumination detail* and the *sampling detail*. A renderer can dynamically reduce both of the levels of detail as needed to reach a target frame rate.

Illumination Detail. The bottleneck in this algorithm is the loop over shadow-casting lights L in Step 2. However, regardless of the size of L, the direct illumination C computed in Step 3 is maintained correctly due to the use of fast local illumination algorithms, such as prefiltered environment maps ([78], [41]). These display higher quality than that achievable by sampling (for example, for highly peaked BRDFs) as they represent continuous illumination from the environment, which sampling cannot. Therefore, we decouple shadow-casting lights from the illumination used to compute reflectance, allowing us to use fewer shadow-casting lights as necessary to improve frame rate. The resulting error is limited to shadow "hardening," which is then ameliorated by resampling via a rendering filter (see next section). Using fewer lights decreases both vertex and fill overhead for either class of shadow algorithms (shadow volumes or shadow maps). We refer to the number of lights in L as the *illumination detail*.

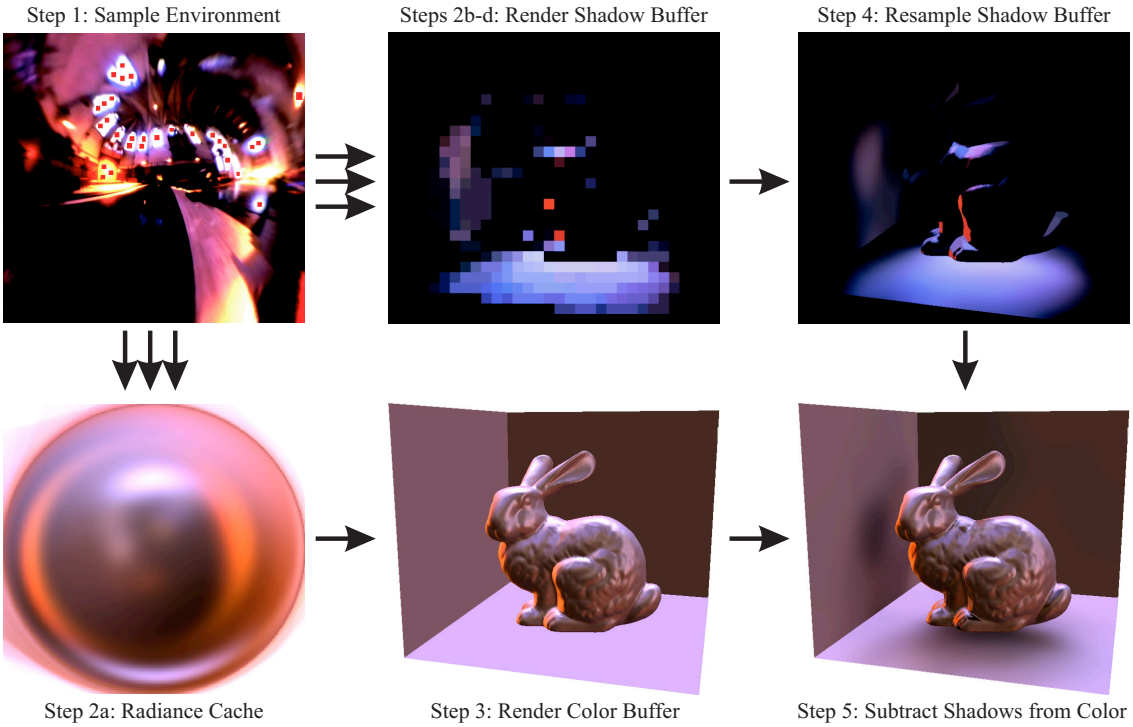


Figure 4.3: **Subtractive Shadows Overview.** An illustration of the various buffers used in the subtractive shadows technique. Multiple arrows indicate that multiple passes are required. The creation of the radiance cache is optional, and can be replaced with another fast local illumination algorithm.

Sampling Detail. Another flexible level-of-detail can be achieved in Step 4, which is key to providing plausibility for reduced-illumination detail shadows. Observe that shadows (especially the soft shadows that result from continuous lighting environments) are low-frequency phenomena, compared to the potentially very high-frequency direct illumination (consider perfect specularity, for example). We therefore can reduce fill overhead with minor loss of quality by reducing the shadows' *sampling detail*: reducing the resolution of the shadow buffer S computed in Step 2. The fill cost required by S can be the dominant component of the total fill requirements of the application (especially for shadow volumes, which may rasterize large portions of the screen for each light); we allow this overhead to be decreased in exchange for minor quality degradation. S is then filtered and resampled in Step 4 to match the size of C. As a rendering filter, this is a low-pass filter acting on $L(l \rightarrow x)V(x \leftrightarrow l)$, for all visible points x and all lights l

In addition to improving fill rate, this filtering also provides a better quality shadow by performing an effective (although not physically correct) approximation of soft shadows through blurring S, allowing fewer shadow-casting lights to be used for comparable quality. Also, as the low-frequency shadowing term is distinct from the high-frequency direct illumination, we are able to filter S only once, after all lights have been composited, and to do so without the loss of high-frequency detail in the direct illumination that would result from blurring the final rendered result of additive shadows. We may view this as an *a priori* decomposition of lighting into low- and high-frequency components, followed by appropriate filtering of the separate parts. The analogous technique for additive shadowing, which does not perform this decomposition, would instead require the per-light visibility to be blurred separately for *each* light, which is then modulated with the unoccluded light to produce the shadowed result.

The resampling must respect normal and depth variation. Our resampling filter uses the full-resolution positions and normals; for a given output pixel in S' we identify corresponding neighboring pixels in the downsampled shadow buffer S and penalize differences in position and normal, using the formula $\mathrm{G}(||S'_p - S_p||)(S'_n \cdot S_n)$; where p and n represent position and normal in their corresponding buffers. This simple approach is similar in concept to the bilateral filter, instead using position and normal as a range filter, and produces smooth results from low-resolution samples (see Figure 4.3). We have shown the range of effects achievable by filtered shadows, both physically plausible and stylized, in a previous work [24], to which we refer for additional analysis.

4.3 Examples

Implementation details. Our technique is not specific to any particular algorithms for prefiltered illumination, visibility determination, or environment sampling. However, for the examples shown here, we chose to use stenciled shadow volumes [40] for visibility, spherical harmonic irradiance maps [78] for fast local diffuse lighting, prefiltered environment maps [41] for fast local specular lighting, and structured importance sampling [2] of the environment map. More generally, for a fixed viewpoint, an arbitrary BRDF with distant lighting can be tabulated per-frame and cached as a texture, known as a *radiance cache*, which can be used as a form of prefiltered illumination [58]. Finally, we also use the technique of deferred shading [61, 39], which renders geometry once per frame, and uses image-space rendering passes to composite additional lights. We refer the reader to those papers for additional detail. All examples are rendered at 1280x800 using 32-bit floating point buffers, on a 3GHz Intel Core2 CPU with GeForce 8800 GPU. The bunny model (35K vertices), triceratops (5K vertices) and horse (50K vertices) in Figure 4.4 are shown lit with the Grace Cathedral, St. Peter's Basilica, and Eucalyptus grove datasets, respectively.

Variable shadow-casting lights. As we see in Figure 4.4 (left and middle columns) and Figure 4.6, decreasing the number of lights (the illumination detail) has a smaller visual impact for subtractive shadows (the shadows become "harder") than for additive shadows (direct illumination is also affected). As a result, the subtractive algorithm has a higher quality at a given frame rate. Further, the decrease in quality is limited to the shadows. The figures also demonstrate an important conclusion about our method, which is that it shows the largest improvements in quality for BRDFs with large peaks, such as high specularity. This is logical, as sampled representations are derived from the environment, not the BRDF, so we would expect these to perform best on diffuse surfaces, in which environment variation is most significant.

Resampled shadow buffer. By rendering the shadow buffer S at a lower resolution than C and resampling, we can achieve a increase in frame rate, as in Figure 4.4 (right column). Additionally, through filtering we can achieve a better approximation of the shadows in the reference than additive shadowing, even if not physically correct (see also Figures 4.1 and 4.5). The 4x downsampled ($1/16$ sampling detail) image is qualitatively comparable to the reference image, and while no longer accurate, the 32x downsampled ($1/1024$ sampling detail) image provides plausible soft shadows, with only 1000 pixel samples

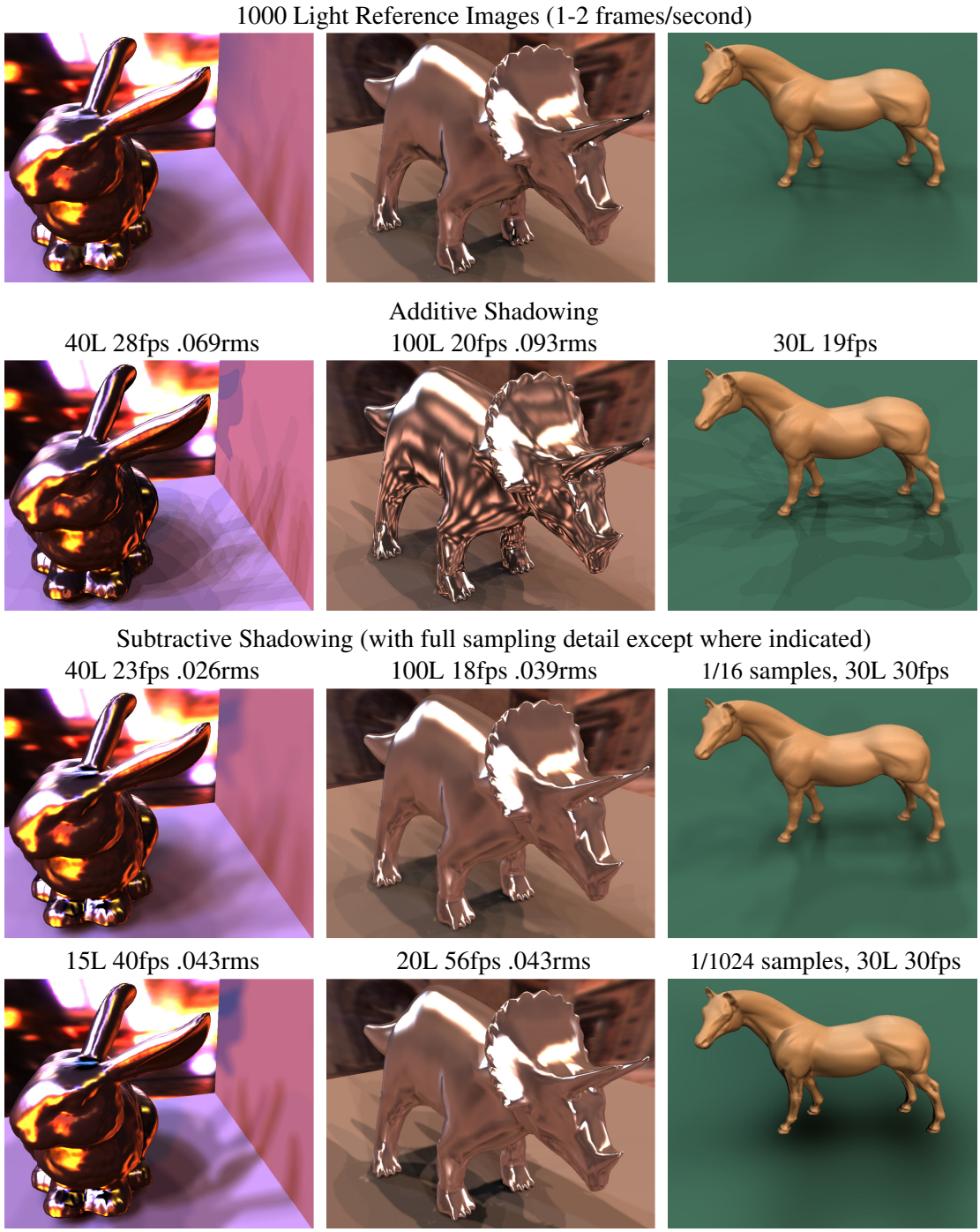


Figure 4.4: **Varying Parameters.** Additive rendering of the bunny (shininess 40) and triceratops (shininess 500) show degradation in quality with decreasing lights (note below the bunny's eye, and the root-mean-squared errors) relative to the reference. Subtractive shadows have higher quality and roughly equal speed at equal lighting detail, and maintain quality even at low level-of-detail; the only artifact is the hardening of the shadows. Further, decreasing sampling detail (right) reduces the fill cost typically associated with shadowing, while continuing to render plausible soft shadows.

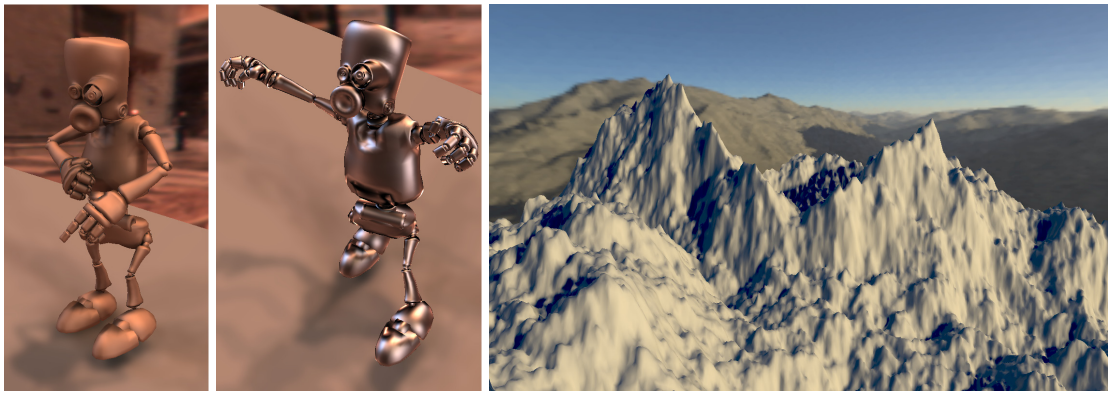


Figure 4.5: **Dynamic Scene and BRDF; Complex Geometry:** We show (left, middle) stills from a scene with dynamic geometry, camera and reflectance, captured at 30-40 FPS, with 50 lights and $1/16$ shadow samples for a 8500 vertex model. We enable editing of BRDF parameters (shown for glossy Phong and Torrance-Sparrow BRDFs) in real time with shadowing, while maintaining high quality through the use of subtractive shadowing with radiance caching for local illumination. The landscape model (66K vertices, 30 lights) demonstrates preservation of complex lighting by using a prefiltered diffuse environment (note the blue tint from atmospheric scattering; which would not be maintained with sampled lighting alone), while retaining important shadowing cues applied to a highly non-planar shadow caster and receiver.

of S for a 1280x800 rendering. This process reduces or eliminates the fill bottleneck typically associated with shadowing algorithms, in particular with shadow volumes (note that frame rate does not change from 4x to 32x, indicating that above $1/16$ detail in this example, fill rate is not a limiting factor). This is especially notable considering that fill overhead tends to be the bottleneck in many real-time applications (which generally use smaller models than those presented here).

Dynamic and Complex Scenes. We show in Figure 4.5 (left, middle) several frames from a scene with a moving camera, non-rigid deformations and complex dynamic reflectance functions. Because our method requires a minimal amount of precomputation per frame (shadow volume determination and radiance caching), no more than is commonly performed in many interactive applications, we are able to render such scenes at real-time rates. This example uses a 128^2 radiance cache, which we have found acceptable for most situations, and renders at 30-40 FPS. The radiance cache is a negligible overhead; by itself the cache renders at over 500 FPS. In Figure 4.5 (right) we demonstrate our method on complex shadow-casters and (self-)shadowing receivers; in which a prefiltered environment is necessary to preserve the subtle blue tint from atmospheric scattering (sampling instead concentrates around the sun).

4.4 Discussion

Our technique for rendering shadows under complex lighting readily invites comparison to the class of precomputed radiance transfer algorithms [85] (PRT), which provide a similar functionality. These methods precompute a transfer function that maps incoming to outgoing radiance for known geometry and reflectance. While these generalize directly to a much wider range of indirect illumination effects, such as interreflection and subsurface scattering, their inherent precomputation limits their use in many applications. For example, our technique was developed in the context of an interactive material editing system; changes in reflectance and their effect on the (dynamic) scene are necessarily required to be visualized immediately. We support the dynamic materials and animated geometry shown in the video stills in Figure 4.5; effects not possible with PRT.

Certain limitations of the implementation that we have chosen to demonstrate the subtractive shadows concept may restrict its use in a production context. A key example is the lack of physically-correct soft shadowing. Additionally, our system as implemented focuses specifically on environment map illumination, though local lighting can be directly integrated into our renderer by additively compositing the local lights along with the direct environment map illumination in Step 4. However, our goal was to focus the comparisons between additive and subtractive systems of equivalent implementation complexity, and so this additional functionality, while clearly important for many applications, was not implemented in this demonstration system. We anticipate that the gains of the system we demonstrate would also apply to more real-world systems. Also, our system shows less significant improvement with diffuse materials; although notable gains can be achieved in highly non-localized environments, such as the sky in Figure 4.5 (right). While there exist methods to suitably render diffuse scenes and discrete local lights, a main goal of ours was to demonstrate how to incorporate more complex materials and natural environment lighting in a shadowing-aware real-time system, without high precomputation complexity – and therefore limits on the dynamic nature of the scene.

As the results show, the subtractive shadows technique allows for simple yet flexible variable level of detail for shadow rendering. The technique generalizes to surfaces of arbitrary reflectance, and allows the developer to achieve high-quality natural illumination, while preserving the ability to render shadows at arbitrary speed with easily adjustable parameters. Through this technique, we give the lighting designer the same flexibility that geometric LOD algorithms have provided to modelers for years.

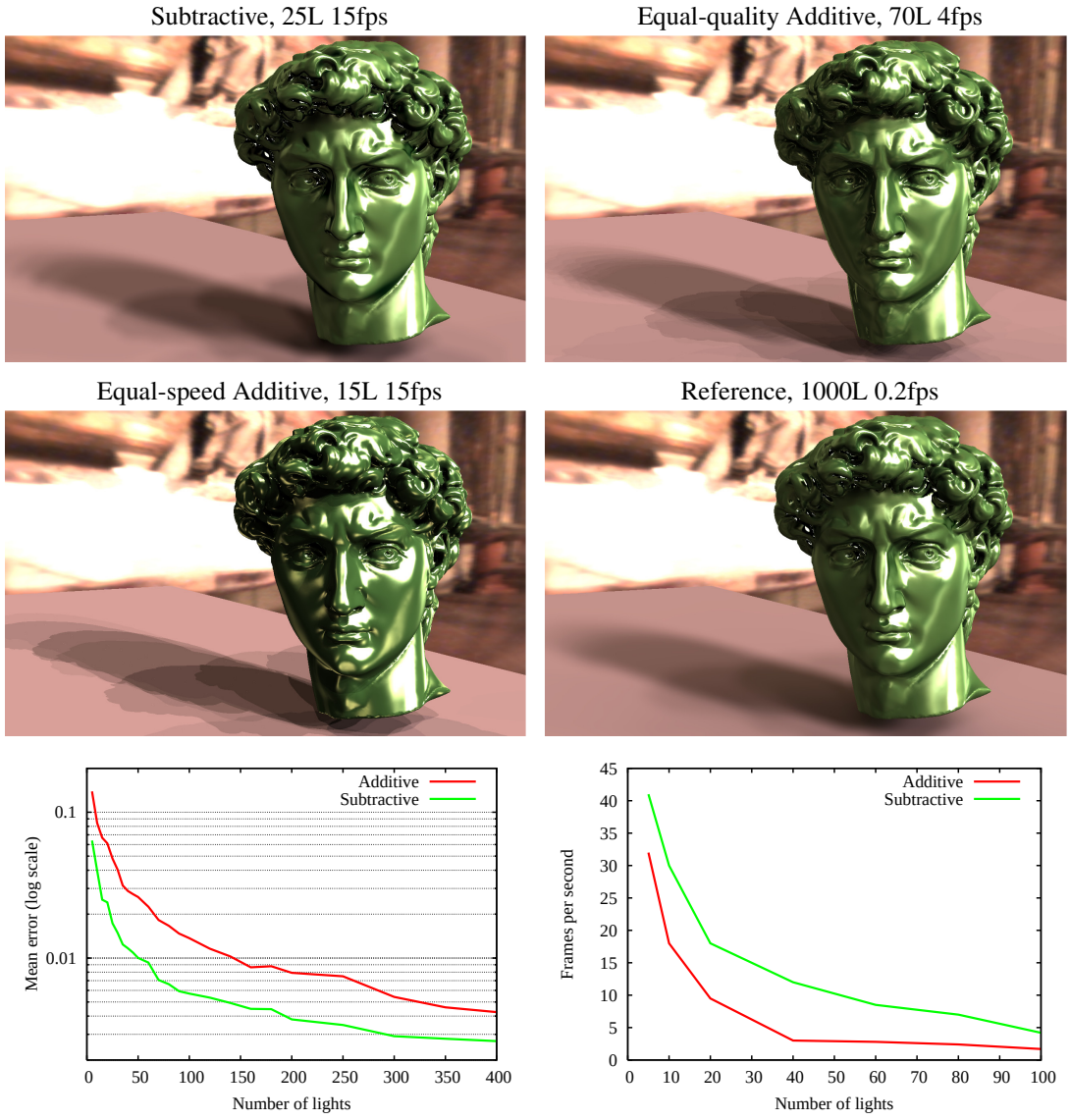


Figure 4.6: We compare the result of our subtractive shadows method with 25 lights and 1/4 sampling detail (top-left) to a high-quality reference rendering computed using 1000 lights (bottom-right). For roughly equivalent quality additive shadows (top-right), our method is significantly faster; while maintaining significantly higher quality compared to an equivalent speed additive rendering (lower-left). The major artifacts of our method, compared to the reference, are in the shadowed regions. We consider shadows cast by all objects in the scene, note for example the shadowed regions of the hair. However, the more noticeable direct illumination is well-preserved, which is not the case for the 15-light additive example. As shown in the graphs, subtractive shadows exhibits consistently superior quality and frame rate across varying levels of lighting detail. The model consists of approximately 125K vertices, rendered at 1280x800 with the Galileo's Tomb lightprobe.

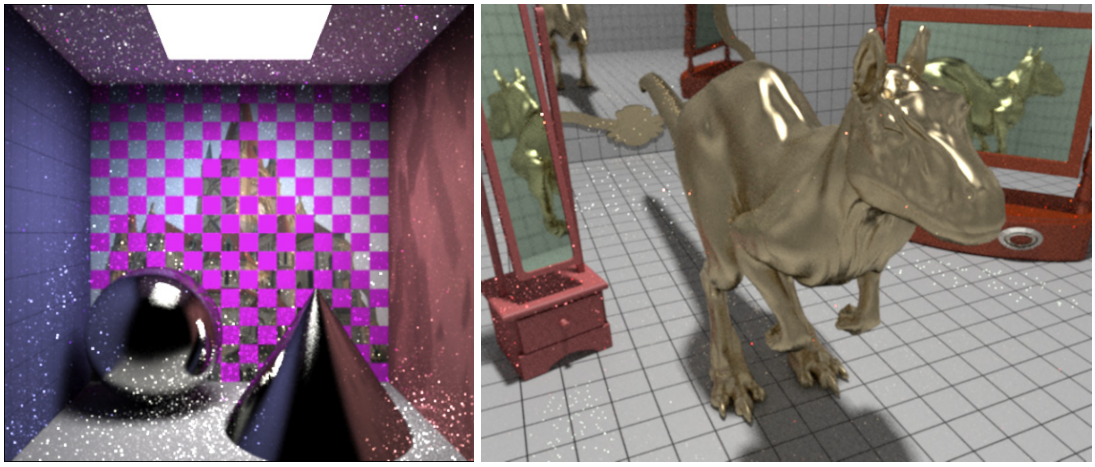
Chapter 5

Path-density Filters for Plausible Outlier Rejection

The problem of noise in Monte-Carlo rendering arising from estimator variance is well-known and well-studied. However, we suggest that certain conditions, which we refer to as *outliers*, have a particularly egregious effect on rendering quality. These leads to significant “spikes" of noise that are perceptually significant due to their high contrast with their surroundings. Most noise-reduction methods, such as importance sampling and stratification, attempt to generate samples that are expected *a priori* to have lower variance, but do not take into account actual sample values. While these methods are essential to decrease overall noise, we show that filtering samples *a posteriori* allows for greater reduction of spiked noise. In particular, given evaluated sample values, outliers can be identified and removed. Note that the use of the term “outlier" should not be taken as synonymous with “incorrect," rather, both here and in statistics in general, these refer to samples that distort the empirically-observed distribution of energy relative to the true underlying distribution. We will show that these are thereby characterized by their density across the set of all nearby paths, as well as their values, and that significant improves in quality can be achieved by filtering according to density.

5.1 Introduction

When simulating light transport using Monte Carlo methods such as path tracing [48], finite sampling rates produce the familiar noise artifacts as seen in Figures 5.1a and 5.1b.



(a & b) Original Renderings

(c & d) Renderings after Outlier Rejection

Figure 5.1: The standard linear reconstruction of these two rendering leads to significant peaks of noise, a result of outlying samples. We propose a method to identify and remove these outliers, leading to a significant reduction in perceptual noise. Importantly, just the noise samples are targeted by the filtering, while other salient features are left unmodified.

We draw attention to two distinct phenomena of noise: the subtle but globally-distributed high-frequency noise resulting from variance between correlated estimators, as well has the highly-localized, distinctive "speckling" resulting from statistical outliers. By removing these outliers in a principled manner, we significantly decrease the perceptual error, and subsequently make the task of smoothing inter-pixel variance amenable to existing filtering methods. Subsequent to outlier rejection and filtering, the same sampling rate results in significantly more plausible renderings, as shown in the two images on the right. The notion of plausibility is key: the filtered renderings are biased, but acceptable to the eye and consistent with the known data, in contrast with the unbiased renderings which contain unacceptable perceptual defects.

This type of peaked noise is highly scene-dependent. In Figure 5.2 we show a rendering of a scene in which the specular surfaces in Figure 5.1a are made Lambertian. Both have been rendered with the same number of samples. The specular surfaces, however, induce a significant increase in noise. In particular, the small percentage of paths that now undergo specular reflection towards a light have significantly higher energy than diffuse reflection paths, resulting in high variance.

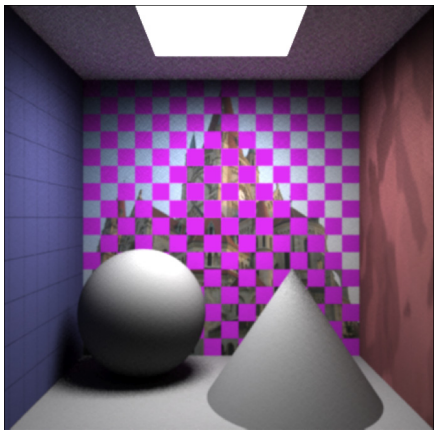


Figure 5.2: **All-diffuse version of Figure 5.1a.** Note absence of peaked noise.

This is not dissimilar to the situation frequently encountered in direct lighting: the energetic paths occupy only a small percentage of the total. However, since the distribution of light sources is known *a priori*, this information can be used to reduce variance through the method of importance sampling. Samples are disproportionately drawn in directions with known energy; resulting in most samples having similar energy, and thus low variance. Similar high-variance integrands result from peaks in the BRDF, in addition to those in incident illumination, as with highly specular reflectance. Multiple importance sampling [97] can combine multiple such sources of *a priori* information to reduce noise.

So long as low-likelihood events can be expected (such as the specular peak of a shiny BRDF) this can be accounted for using multiple importance sampling. The problem arises when the empirical distribution of energy differs widely from the expected distribution (i.e. the one from which samples are drawn). In particular, as indirect illumination is not

known in advance – it is the quantity which we are using path tracing to compute – it is accordingly more difficult to compensate for variance. In fact, importance sampling can compound the problem of outliers. When a region with low expected density is sampled, importance sampling weights such values higher to produce an unbiased estimator. However, when such samples in fact have high energy, this produces a significantly larger value. Techniques including defensive importance sampling [43] and others [67] have been developed in response to this phenomena.

We term such samples as *outliers*. This effect of speckling resulting from outliers is a common effect in rendering that has frequently been commented on in the related literature. See for example the discussion of exactly this point in [47] (Figure 9.1, Pg. 139), and note the similar artifacts in the closeups in [74] (Pg. 669, for example). We propose a definition of such outliers that allows them to be identified and removed, significantly decreasing noise in the rendered image.

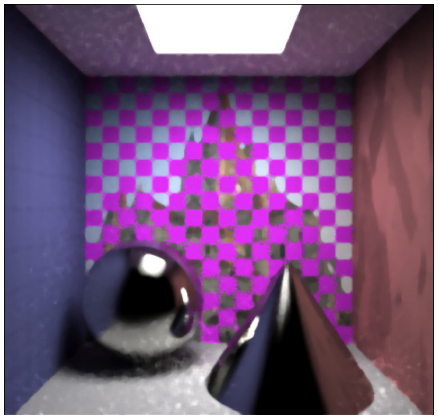


Figure 5.3: **Median Filter in image-space**, which fails due to information loss.

We have seen previously in Chapter 1 examples of image-space filters designed to remove noise, such as the median filter (shown used for noise removal in Figure 1.11) and the bilateral filter (Figure 1.10). The non-linear median filter in particular has often been recommended for removing the sort of speckled noise as often seen in Monte Carlo renderings. However, as image-space filters act solely on reconstructed pixels, significant data have already been lost by combining samples, and the existing data have been corrupted by outliers. For example, attempting to apply an image-space median filter to the example in Figure 5.1 results in Figure 5.3. In many regions of the image, enough pixels have been corrupted by outliers with the result that an accurate approximation of the original cannot be determined. Bilateral filtering performs worse; the range term *preserves* the noise, rather than discarding it. Instead, we will apply a rendering filter approach directly on samples of the rendering equation, which we show performs significantly better than the image-space approach.

5.2 Related Work

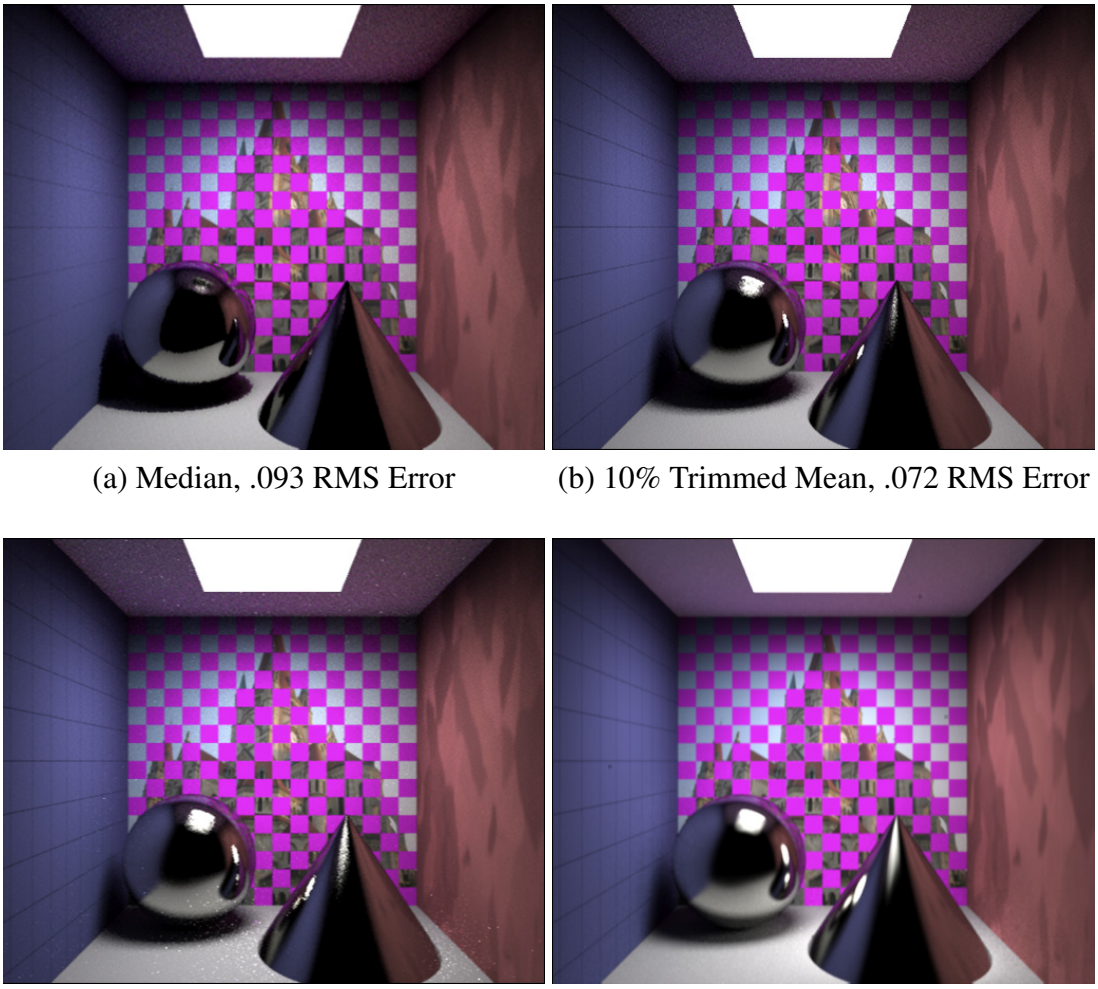
In contrast to methods such as importance sampling, other filtering-based methods use the values of the evaluated samples to reduce noise by using an alternate reconstruction method for the final image pixels. The common approach to reconstruction of an image $I(x)$ applies a linear filter W (representing the response function of an image sensor) to the continuous light transport equation

$$I(x) = W * L = \lim_{n\to\infty} \frac{1}{C} \sum_{i}^{n} W(||x - x_i||)L(x_i), \tag{5.1}$$

In words, the pixel $I(x)$ is a weighted average of points $L(x_i)$ on the image plane, where the weight W is determined entirely by the distance between x and x_i, and is independent of the radiance L. Being linear in L, any sample can have an unbounded influence on I by increasing its value; a single unrepresentative sample, when using a finite sampling rate, can significantly affect the reconstructed pixel value.

The percentage of values that, if modified, can force an arbitrary change in the estimate is known as the *breakdown point* [38]. As an estimator, mean has a breakdown point of 0, but is only one example of an estimator for a *location parameter*; a value that gives the translation of a statistical distribution (see [95] for an overview). Alternate estimators have higher breakdown points. The median is an example of a commonly used robust estimator; it has a breakdown point of 50%, that is, the estimate is resistant to arbitrary error until half of the data samples are corrupted. However, while the median has a high breakdown point, it has poor *efficiency*, measured as the variance of the estimator relative to the mean estimator for a normal distribution. The practical effect of this is shown in Figure 5.4a. While the outlier noise is removed, the transitions in smooth regions are made sharp, and the total level of energy in certain regions, such as the ceiling, is significantly reduced.

A generalization of both the median and the mean is the α-trimmed mean, which has been previously proposed as a robust filtering method for reconstruction in rendering [53]. For a value of $\alpha \in [0, .5)$, the upper and lower α order statistics of samples are discarded, and the location parameter estimated as the mean of the remaining samples. Note that the median is equivalent to the 0.5-trimmed mean. While the breakdown point is reduced from 50% to α, efficiency is also increased; as shown in Figures 5.4b and 5.4c, the 10%-trimmed mean significantly reduces variance in transition regions as compared to the



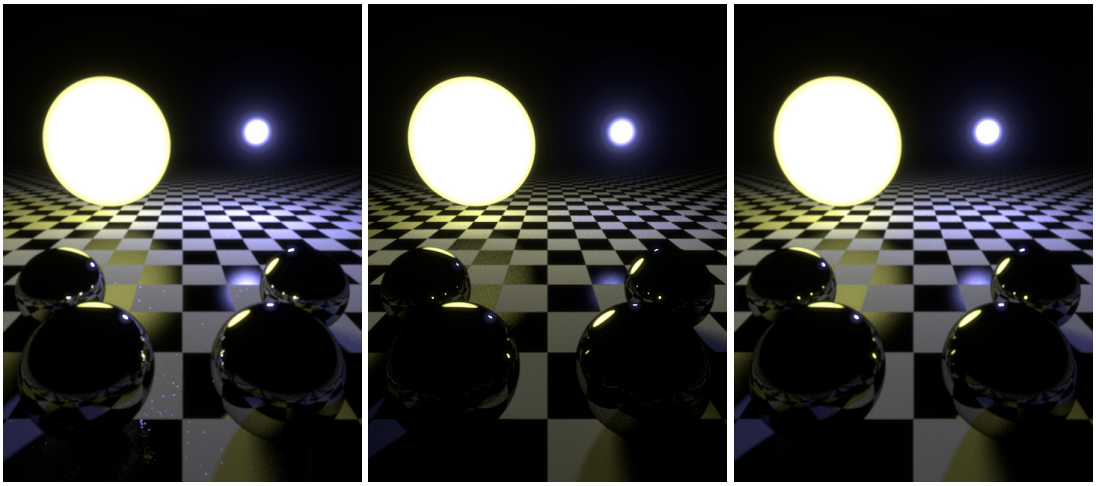
(a) Median, .093 RMS Error (b) 10% Trimmed Mean, .072 RMS Error

(c) 1% Trimmed Mean .051 RMS Error (d) Reference

Figure 5.4: **Alpha-trimmed Means:** Previous methods for outlier rejection using alpha-trimmed means remove noise at the cost of significant decrease in total energy (note especially the roof and specular highlights compared to the reference) and also in adding sharpness into previously smooth transitions (see the soft shadows). As a result, these reconstructions have significant root-mean-squared (RMS) error compared to the reference (computed with 1024 samples/pixel) that does not improve on the .0519 error of the noisy original. Our method (Figure 5.1c) removes noise while maintaining .041 RMS error.

mean, while continuing to remove noise. However, the energy levels of the ceiling and specular highlights are reduced, which continues to differ as compared to the reference even for a smaller α of 1%, at which point noise from outliers becomes noticeable.

This problem can be characterized as the effect of *heteroskedastic* data, in which the variance is non-constant across the image plane. As a result, a single choice of α is



(a) Original (b) 5% Trimmed Mean (c) Our Method

Figure 5.5: **Multi-modal Distributions:** In this scene with two lights (yellow and blue, seen in reflection), the presence of specular objects causes spiked noise due to outliers. While a trimmed mean will reject these outliers, samples on the blue light are rejected as well. Our method rejects only the clear outliers, while viewing the blue light samples as consistent with the data as a whole

insufficient to deal with the entire image. A potential solution is to first estimate α for each pixel, but this adds an additional level of complication. Other methods, such as the anisotropic diffusion filter [57] and the non-linear energy-preserving filter [82] attempt to adapt locally to the amount of noise in each region, and spread the excess energy out to neighboring pixels.

What these methods do not address, however, is the larger problem is of multimodal data. Estimators of location parameters are, by their nature, finding a single mode; the empirical distributions encountered in computer graphics have many modes of energy, induced by multiple lights (or partial occlusion of a single light), multiple modes of a BRDF, indirect illumination, and so on. Take for example the scene shown in Figure 5.5. While a trimmed mean removes the speckled noise, it also causes a significant shift in color, as rays from the smaller (but more powerful) blue light are rejected as outliers to the larger yellow light. Our method, to be presented in the next section, can remove the noise while preserving coloration.

5.3 Algorithm Description

We reject or reduce the influence of individual samples based on the observed *joint density* of each sample in path space. Therefore, this leads us to term this as a *path-density filter*. By the joint density, we refer to the density of each sample in the multidimensional space consisting of both image space coordinates (independent variables) and color coordinates (dependent variables). Samples with low density are rejected as outliers.

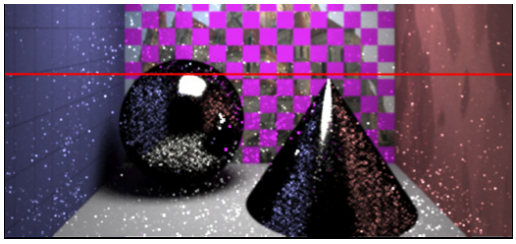
To motivate this method, Figure 5.6a shows a close up of a noisy image previously shown, where a single line of samples (corresponding to those along the red line) has been plotted independently in Figure 5.6b. The samples are plotted as image x-coordinate vs. radiance (with independent red, green, and blue channels overlaid). One can see clear outliers in the plot, which correspond to the sharp noise peaks seen in the reconstruction. Note however, that the density of samples, in addition to the magnitude, are important for identification of outliers. Compare to samples in the specular highlight, which have significantly large magnitudes, but also are densely sampled and therefore not outliers. Our algorithm is as follows:

1. Render samples
2. Store samples in space-partitioning tree
3. For each sample:
   (a) Find k-nearest neighbors using tree in joint image/color space
   (b) Use to estimate local scale and density
   (c) If density less than threshold, reject or down-weight

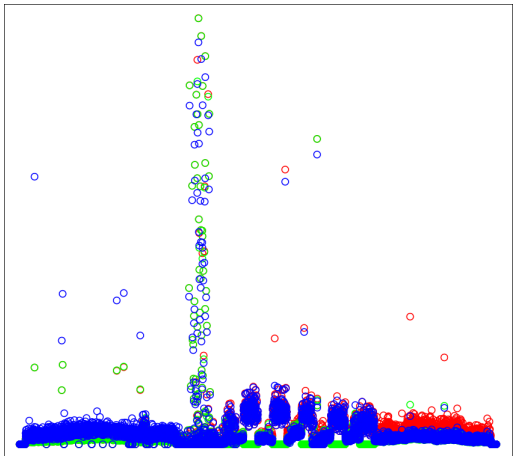
Density is a quantity defined as mass per unit of volume; in order to define a continuous density function $\rho(x)$ over a volume from finite point sample, we compute density from the volume of the hypersphere centered at x containing the k nearest neighbors. This is analogous to the density estimation used in methods such as photon mapping [46].

Figure 5.6c displays a visualization of the density function of the samples previously shown. Distinct colors represent isolines of equal density. Note that the outliers are located in the low-density region; rejection of all points with low densities removes the outliers and their associated noise (while preserving the higher-density features such as the specular peak). This produces the rendering originally shown in Figure 5.1a.

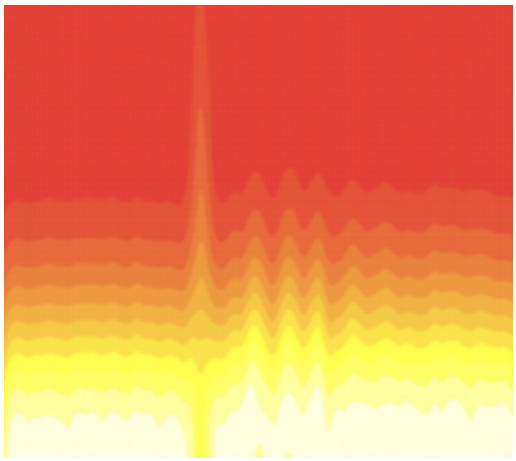
For our method we trace paths through the scene using existing methods, and rather than composite them immediately into an output image, we store them for later use in



(a) Original Noisy Image



(b) Plot of sample x-location vs. intensity



(c) Joint Density

Figure 5.6: **Visualization of Joint Density.** We show a slice of the noisy input image (a, indicated by red line) where samples have been plotted on a graph of their x-component versus intensity (b, note that red, green and blue color components are plotted separately). The existence of outliers can be seen clearly, and are characterized by their low density in the joint space relative to other samples, rather than by their values. By computing density explicitly (c, colors indicate lines of equal density) we can identify and remove these samples from the reconstructed image (Figure 5.1c).

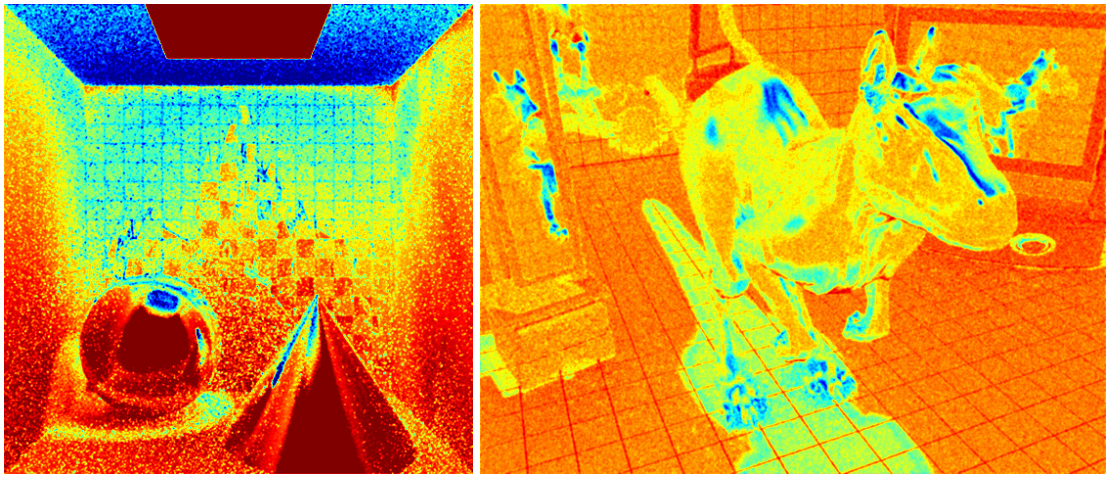
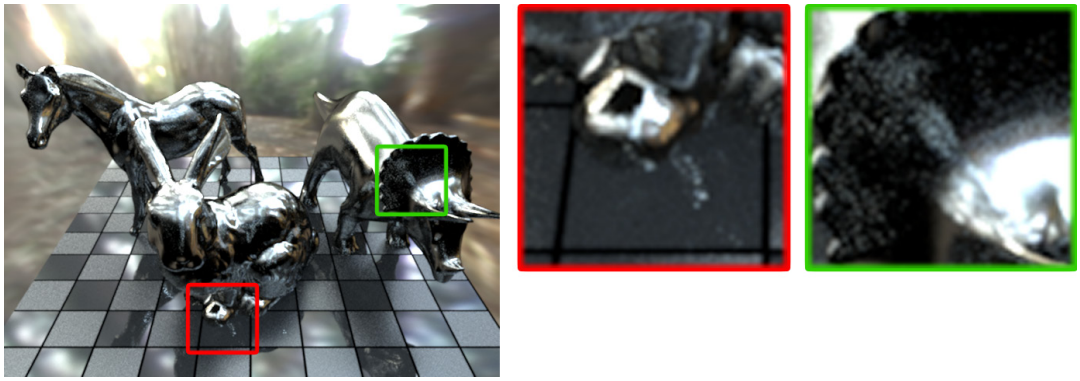


Figure 5.7: **Mean Density Per Pixel** The two images visualize the density for our test scenes by displaying the mean density of samples at each pixel, color-mapped such that blue is lowest density and red is highest. Note that areas such as the roof in the left image (which is lit entirely by indirect illumination) and shadow penumbrae in both images have low density compared to their surroundings.

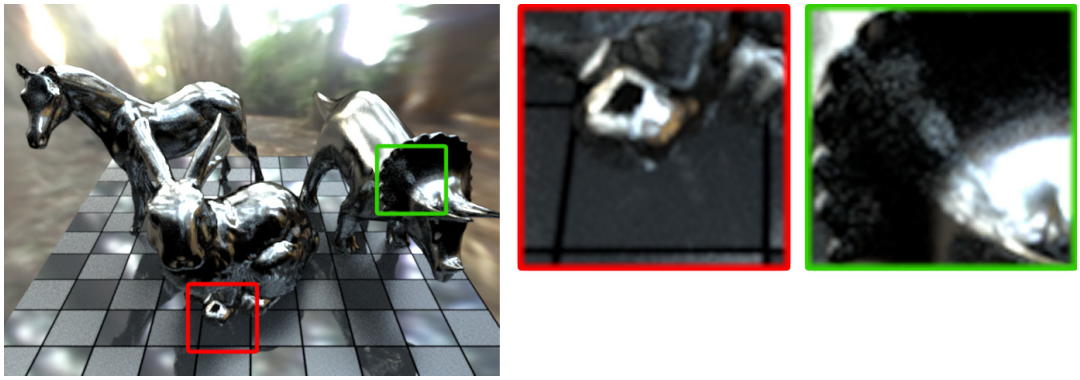
density estimation. Samples are collected into a space-partitioning tree to allow for fast neighbor computation. Then, for each sample x we use the tree to find the k-nearest neighbors, where k is a user-specified parameter. The neighbors are used to take an estimate of local scale, or dispersion, σ. In particular, σ is equal to the mean absolute distance from x to the k-nearest neighbors. Density is estimated as $G(\sigma)$, the standard normal Gaussian of the scale parameter.

The density $G(\sigma)$ maps σ, which may be of arbitrary magnitude, to the range $(0, 1/\sqrt{2\pi}]$. As the sampling rate increases, the neighbors of x increasingly converge to x itself, and the density approaches $1/\sqrt{2\pi}$. This acts as a measure of the individual sample, as opposed to measuring the variance of an individual pixel, and adapts to both heteroskedastic and multi-modal distributions. We use this measure to reject and reduce the influence of outliers; specifically, we choose to reduce the influence of all samples more than 2 standard deviations away, and reject samples 3 standard deviations away. The soft threshold reduces high-frequency artifacts that would otherwise be introduced when a particular density is culled. After reweighting, samples are composited into the output image as usual. We visualize the resulting densities for our test scenes in Figure 5.7.

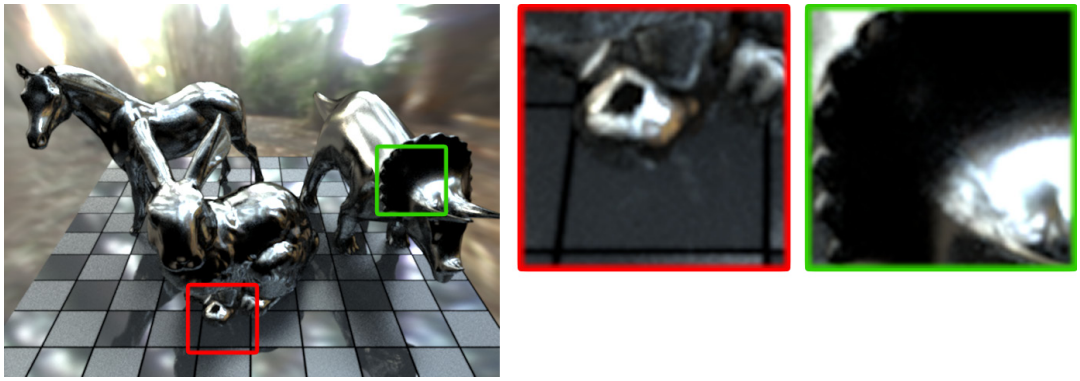
Choosing Neighborhood Size. The choice of the parameter k acts as a smoothing parameter. Larger values of k require samples to have stronger corroboration by its



(a) Low Confidence Threshold, $k = 10$

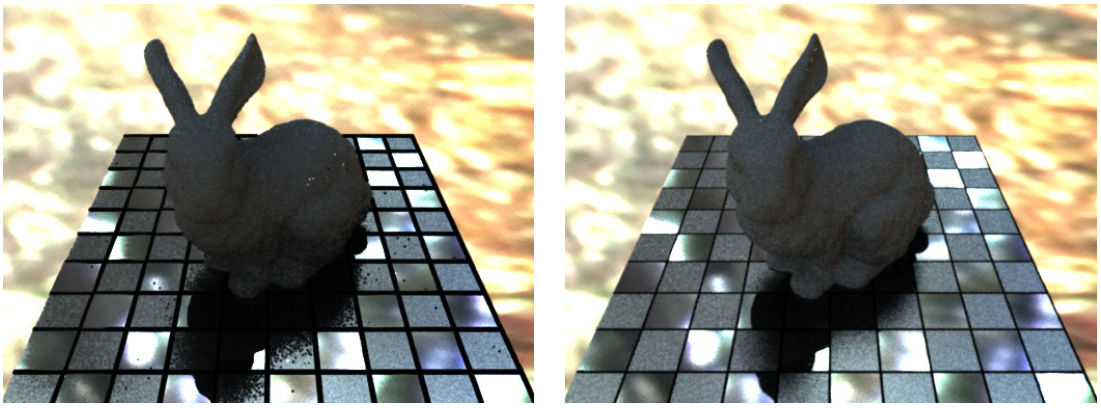


(b) Moderate Confidence Threshold, $k = 50$



(c) High Confidence Threshold, $k = 200$

Figure 5.8: **Choosing neighborhood size.** We show the Eucalyptus Grove scene using varying neighborhood sizes. As k is increased, the confidence required to retain a sample also increases. With fewer low-confidence samples in the reconstruction, noise is removed from the image. Note the how in the area under the feet of the bunny, high-frequency noise is removed between (a) and (b) while preserving the similarly-high frequencies given by the floor tile. Similarly, noise on the left of the triceratops head is removed when between (b) and (c).



(a) Using RGB distance (b) Using CIELAB distance

Figure 5.9: **Comparison of Distance Metrics.** When using a Euclidean distance metric in RGB color space, the colors of the grid cells are insufficiently distinguished from black. These border cells are viewed as outliers and rejected, causing the black grid lines to expand. Using CIELAB distance, the colors on the other side of the line are viewed as the nearest neighbors, despite being farther away, and the artifacts are removed.

neighbors before being accepted, leading to a removal of samples with low confidence. We demonstrate this in Figure 5.8.

Distance Metric. Before searching for neighbors, we must establish a notion of distance. For our purposes, distances between two points is defined as the sum of Euclidean distance between their image space coordinates, and their color coordinates in CIELAB space. The image-space distance is weighted to correspond to the width of the sensor response W. Color distance weight is a user parameter; we have fixed it to 20. We find that the use of CIELAB space, with its greater discriminative characteristics between dissimilar colors than that of RGB, is important for successful results. In Figure 5.9, an example is shown using both CIELAB distance and RGB distance. In the image using RGB, significant artifacts occur at edges, as the metric poorly distinguishes colors across edges. No such artifacts occur in the CIELAB image.

Approximate Nearest Neighbors. Because the k-nearest neighbors are used only as an approximation of the local neighborhood, it is not required to find the neighbors exactly. Instead, we can cut computation time almost in half through the use of an approximation algorithm [5]. The algorithm maintains the bound that for any error tolerance $\varepsilon > 0$, the k neighbor returned will have a distance no greater than $1+\varepsilon$ times the true k-th neighbor. This is implemented when traversing the space-partitioning tree; branches can be culled when it is determined that no element in the branch exceeds this error tolerance. Resultingly, an approximate kNN will always slightly underestimate density, a result of

larger scale in each neighborhood caused by the increased distances. With 10 M samples and $k = 200$, for example, approximate kNN causes a 0.026 RMS error relative to filtering with accurate kNN, which is reasonably low. Furthermore, most of this error is in noise pixels and is visually imperceptible, however, it leads to a speedup of about 85%.

5.4 Discussion

We have presented here a rendering filter that allows for plausible renderings across a wide range of sampling rates. Importantly, in many ways this method decouples the designer of the scene for responsibility over consideration of scene variance. Objects with complex, glossy reflectance can be used in a scene without penalty of introduced noise in other objects. Instead, the resulting noise is withheld from the scene until such a sampling rate has been achieved to allow the rendering of such phenomena without noise. Importantly, basic lighting features such as direct illumination can be computed relatively quickly, while the rendering filter isolates the low-noise direct illumination from corruption from high-noise indirect illumination. It is intended that this will allow for the use of progressive level-of-detail in realistic renderings, which itself will hopefully lead to greater adaptation of physically-based light transport methods and ray tracing to supplant the heuristic methods most commonly in place today.

Chapter 6

Conclusion and Future Work

As indicated in the title of this thesis, the main goals of the rendering filters that have been presented are twofold: control over detail, with the intent of enabling time/quality tradeoffs; and creation of effects, mostly from an artistic or creative purpose. While we believe that the presented filters in Part II are powerful tools that advance these goals, it is hoped more generally that the overall framework of the rendering filter in Part I will be employed towards the greater furtherance of such ends. In this Chapter, we discuss a number of directions that this future work could take.

6.1 Future Work for Controlling Detail

High-Dimensional Smoothing. The key contributions of the subtractive shadowing filters in Chapter 4 and the path-density filters in Chapter 5 is the tradeoff between time and quality, in which a plausible approximation can be generated in a fixed amount of time. These methods are useful in such cases where meeting the limited time budget is more critical than the highest possible quality. This is most critical in the real-time rendering context of Chapter 4, in which time is a *hard constraint*, one for which the system is considered unacceptable if the constraint is not met. However, meeting the *soft constraint* of a particular time-frame can also be a matter of significant cost savings or increased time-to-market for a large-scale photorealistic rendering, such as an animated film.

We would like to achieve a more flexible time-quality tradeoff than currently supported for plausible photorealistic rendering. One of the more promising directions for this

future work is to generalize our work on path-density filtering for outlier rejection into a more unified approach to filtering and reconstruction in high-dimensional spaces. As we mentioned in Chapter 5, the expression of Monte Carlo variance can broadly be divided into two phenomena: the localized, peaked noise due to significantly outlying samples, and the globally-distributed high-frequency noise due to pixel-to-pixel variation. While our method primarily addresses the former, it is hopeful that future extensions could make strides towards addressing the latter. One promising class of methods that bears a strong similarity to our work is that of the *mean-shift filter* [20], originally presented for image filtering. In the mean-shift framework, pixels are treated as high-dimensional points in a feature-space, just as in our method. Each point is adjusted towards the nearest mode of density, by employing an iterative gradient ascent. The authors show how this method can perform significantly better than bilateral filtering in many cases; indeed, it can be considered a generalization of the one-pass bilateral filter to an arbitrary number of update passes. As an image-space filter, however, the mean-shift encounters the same problems that we discussed as being inherent to such filters in Chapter 5; the most prominent limitation being that of information loss due to reconstruction, and the resulting inability to support multi-modal distributions. We have made some preliminary explorations of using mean-shift filters directly on rendering samples; these results indicate that the mean-shift procedure may allow for better reduction of noise compared to other methods, however the major limitation is the significant time and space complexity involved with storing and processing hundreds of millions of samples. We next provide a discussion of how this may be addressed.

Efficient Density Representations. The current outlier-rejection method represents density by storing all previous samples in a spatial-partitioning tree, and locating the k-nearest neighbors for each evaluation of density. While the time complexity is made manageable (logarithmic relative to stored samples) by the use of spatial partitioning, the space complexity seems unnecessarily high, and likely will be a limiting factor for use in rendering animations, which could require on the order of billions of samples. The use of very-high dimensional sample vectors further contributes to the space complexity. It is likely that more sophisticated methods of representation could more efficiently model the joint density. For example, the use of *expectation-maximization* (EM) for fitting a *Gaussian Mixture Model* (GMM) could potentially be applicable here. In such a model, the random variables are assumed to be generated according to a linear combination of multivariate Gaussian distributions; as the number of samples increases, we expect

that the empirical distribution will progressively approach the Gaussian. Given a GMM formulation of density, we are required only to store the mean and variance of each distribution; the EM algorithm is a particular iterative method for finding a particular choice of GMM for the input data. For further reference on this topic, and other topics in machine learning, we refer the reader to the textbook by Bishop [12].

6.2 Future Work for Creating Effects

Area lighting and indirect illumination. Our work on stylization of shadows is a first step towards more general control over artistic rendering parameters. Importantly, we may see the binary visibility buffers as a subset of the range of possible inputs to such a stylization algorithm. The most natural generalization, as we have mentioned previously, would be to support a continuous visibility buffer such as results from an area light source. In this case, rather than assuming a binary value, each pixel assumes a value between zero and one, reflecting the partial visibility of that pixel to the light. The first complication this causes to the existing algorithm is the requirement of an alternate distance metric; averaged distance to the shadow contour is no longer well-defined in such a context (consider the case where the entire shadow is penumbra). Potentially, this could require an integration over the entire visibility buffer, which would lead to increased time complexity. Even more general, however, would be the expansion of stylization to global illumination. A good example of this is recent work by Obert et al. [66], which allows the user to interactively "paint" indirect illumination effects.

Additional Control Parameters. A near-unlimited source of future work is the investigation and definition of additional stylistic control parameters to complement the four parameters of inflation, softness, brightness and abstraction previously given. We give the following example. In the description of the abstraction parameter, we defined the "most abstract" shadow as being equivalent to that of an ellipse. It follows, therefore, that the abstraction parameter can be viewed as an factor controlling the interpolation between the endpoints of the shadow of the original object and the shadow of a smoothed spheroid at the same location. To generalize this notion, we could add additional control parameters that configure this target shape to be one other than a spheroid: an *angularity* parameter could blend between the spheroid and a cube, allowing for sharper, yet still abstract shadows (for example, the outer penumbra in the top-left of Figure 3.2). For

greater user-guided control, a target shadow could be explicitly given, either in total, or by constraining user-specified regions of the shadow as projected onto the surface.

Alternate Distance Metrics. The Euclidean-like metrics of L_p-averaged distance are consider only straight-line distances through space, and as a result some behavior – particularly control over topology – can become unintuitive and difficult to control. In the Examples in Figure 3.5 with the Octopus model, a negative inflation value tends to "pinch off" the long, thin shadows cast by the tentacles. An example of this can be seen in Figure 6.1, in the lower-left quadrant, in which a portion of shadow forms a separate connected component to the main shadow. One proposal, therefore, would be to measure the distance of a shadowed point by a *averaged graph distance*. The shadowed pixels are the vertices of the graph, and two adjacent shadowed pixels are said to share an edge. By running an all-pairs shortest path algorithm, we determine the average distance of each vertex to all others. This could possibly lead to more intuitive behavior in this and other cases.

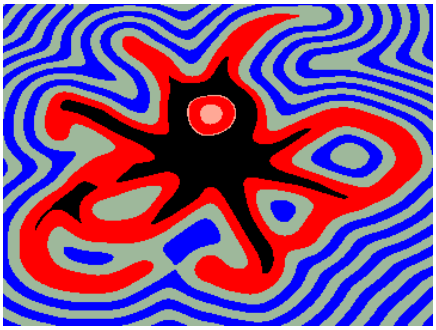


Figure 6.1: **A Shortcoming of L_p-averaged Distance.** Note the disconnected shadow region in the lower-left quadrant.

Additional and More Accurate Rendering Terms. We have discussed and shown examples of how our system can benefit from using additional rendering terms, such as approximate distance to shadow-caster/occluders, in recreating artistic effects. An important direction of future work is the investigation of which additional terms provide a benefit, and how those terms would be best used. For example, the use of surface normals could allow alternately both greater recreation of realism, for shadowing that naturally varies according to the surface's orientation towards the light, as well as abstraction, such as is used for *toon-shading*, which quantizes the surface normal for compositional purposes. Perhaps more important, however, would be the use of more accurate depth and visibility information as achieved through ray-tracing, as opposed to the rasterization-based shadows currently employed. This could allow more natural behavior as shadows are modified and animated. For example, in Figure 6.2, the region of the image near the red sphere appears to be distant from the shadow, as seen in the left image. However, the right image indicates otherwise; our algorithm would incorrectly compute an inflated shadow given a visibility buffer from the left image. This limitation is due to the limited

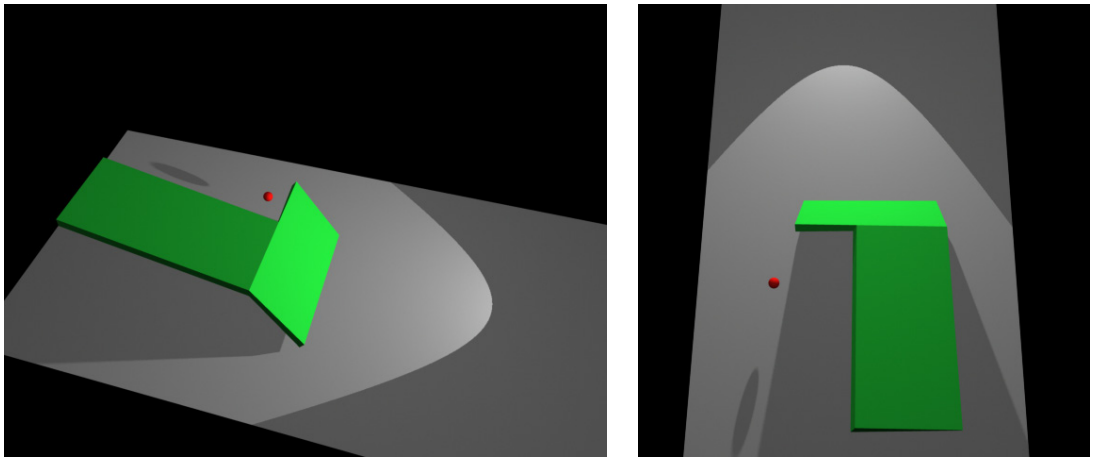


Figure 6.2: **Limitations of Rasterization-based Stylized Shadows.** The figures show an example of a scene in which shadow stylization could be better assisted using ray tracing, or alternative sources of light transport information. In the left image, the red sphere is apparently distant to any shadow boundaries. In the right image, however, we see that the sphere is indeed quite near, and that a small inflation would render the area beneath the sphere in shadow. This underscores that our current approach is limited by the information currently available to it.

information inherent in a rasterization system, where rendering terms are computed only for those pixels visible to the eye. Ray tracing could potentially be used in the situation to compute more accurate visibility, independent of the current camera position.

6.3 Final Thoughts

In this thesis, we have presented both a general framework and a specific selection of methods that we hope will prove useful to the challenge of rendering in computer graphics. The key feature behind these is the recognition of rendering as a process fundamentally similar to existing filtering processes, and that a transfer of existing expertise to this novel problem will allow for better solutions to novel problems. It is hoped that these examples of general principles point the way towards the future use of the rendering filter principle in controlling the efficiency and stylization of rendering.

Bibliography

[1] ADELSON, E., AND BERGEN, J. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing* (Jan 1991).

[2] AGARWAL, S., RAMAMOORTHI, R., BELONGIE, S., AND JENSEN, H. Structured importance sampling of environment maps. *ACM Trans. Graphics 22*, 3 (July 2003).

[3] ALTON, J. *Painting With Light*. University of California Press (republished in 1995), Berkeley, CA, 1949.

[4] ANNEN, T., KAUTZ, J., DURAND, F., AND SEIDEL, H.-P. Spherical harmonic gradients for mid-range illumination. In *Proc. Eurographics Symp. Rendering* (June 2004), pp. 331–336.

[5] ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. Y. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. ACM-SIAM Symp. Discrete Algorithms* (1994).

[6] ASSARSSON, U., AND AKENINE-MÖLLER, T. A geometry-based soft shadow volume algorithm using graphics hardware. *ACM Trans. Graphics 22*, 3 (2003), 511–520.

[7] ASSARSSON, U., DOUGHERTY, M., MOUNIER, M., AND AKENINE-MÖLLER, T. An optimized soft shadow volume algorithm with real-time performance. In *Proc. Graphics Hardware* (2003).

[8] ATTY, L., HOLZSCHUCH, N., LAPIERRE, M., HASENFRATZ, J., HANSEN, C., AND SILLION, F. X. Soft shadow maps: Efficient sampling of light source visibility. *Computer Graphics Forum 4* (2006).

[9] BARASH, D. Bilateral filtering and anisotropic diffusion: Towards a unified viewpoint. Tech. rep., Hewlett-Packard Laboratories Israel, 2001.

[10] BARZEL, R. Lighting controls for computer cinematography. *Journal of Graphics Tools 2*, 1 (1997), 1–20.

[11] BEN-ARTZI, A., OVERBECK, R., AND RAMAMOORTHI, R. Real-time BRDF editing in complex lighting. *ACM Trans. Graphics 25*, 3 (July 2006), 945–954.

[12] BISHOP, C. M. *Pattern Recognition and Machine Learning*. Springer Science + Business Media, 2006.

[13] BOER, W. H. D. Smooth penumbra transitions with shadow maps. *Journal of Graphics Tools 11*, 2 (2006), 59–71.

[14] CABRAL, B., OLANO, M., AND NEMEC, P. Reflection space image based rendering. In *Proc. ACM SIGGRAPH* (1999), pp. 165–170.

[15] CATMULL, E. E. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, The University of Utah, 1974.

[16] CHAN, E., AND DURAND, F. Rendering fake soft shadows with smoothies. In *Proc. Eurographics Workshop on Rendering* (2003), pp. 208–218.

[17] CHAN, E., AND DURAND, F. An efficient hybrid shadow rendering algorithm. In *Proc. Eurographics Symp. Rendering* (2004), pp. 185–195.

[18] CHEN, J., PARIS, S., AND DURAND, F. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graphics 26*, 3 (July 2007).

[19] CLARBERG, P., JAROSZ, W., AKENINE-MÖLLER, T., AND JENSEN, H. W. Wavelet Importance Sampling: Efficiently evaluating products of complex functions. *ACM Trans. Graphics 24*, 3 (July 2005), 1166–1175.

[20] COMANICIU, D., AND MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Analysis and Machine Learning* (January 2002).

[21] CROW, F. C. Shadow algorithms for computer graphics. In *Computer Graphics (Proc. ACM SIGGRAPH)* (July 1977).

[22] CROW, F. C. Summed-area tables for texture mapping. In *Computer Graphics (Proc. ACM SIGGRAPH)* (July 1984).

[23] DAUBECHIES, I., AND BATES, B. J. Ten lectures on wavelets. *Journ. Acoustical Society of America 93*, 3 (1993), 1671–1671.

[24] DECORO, C., COLE, F., FINKELSTEIN, A., AND RUSINKIEWICZ, S. Stylized shadows. In *Proc. Symp. Non-Photorealistic Animation and Rendering* (August 2007).

[25] DECORO, C., AND RUSINKIEWICZ, S. Subtractive Shadows: A flexible method for shadow level-of-detail. Tech. Rep. TR-781-07, Princeton University, 2007.

[26] DECORO, C., AND TATARCHUK, N. Real-time mesh simplification using the gpu. In *Proc. Symp. Interactive 3D Graphics and Games* (2007).

[27] DONNELLY, W., AND LAURITZEN, A. Variance shadow maps. In *Proc. Symp. Interactive 3D Graphics and Games* (2006), pp. 161–165.

[28] EASTMAN KODAK CO. KODAK Motion Picture Accessories & Services Catalog. http://motion.kodak.com/motion/uploadedFiles/Optical+Gelatin Filters.pdf, November 2008.

[29] ENCYCLOPÆDIA BRITANNICA. *"daguerreotype"*. Accessed from Encyclpædia Britannica Online, April 2009.

[30] FERNANDO, R. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches* (2005).

[31] GERSHUN, A. The Light Field. *Journ. Math and Physics 18* (1939).

[32] GONZALEZ, R., AND WOODS, R. Digital image fundamentals. In *Digital Image Processing*. Addison-Wesley, 1992, ch. 2.

[33] GONZALEZ, R., AND WOODS, R. Image enhancement. In *Digital Image Processing*. Addison-Wesley, 1992, ch. 4.

[34] GOULEKAS, K. Vaseline filter. In *Visual Effects in a Digital World*. Morgan Kaufman, 2001, p. 538.

[35] GREEN, P., KAUTZ, J., MATUSIK, W., AND DURAND, F. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proc. Symp. Interactive 3D Graphics and Games* (2006), pp. 7–14.

[36] GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. Real-time soft shadow mapping by backprojection. In *Proc. Eurographics Symp. Rendering* (2006), pp. 227–234.

[37] GUMHOLD, S. Maximum entropy light source placement. In *Proc. IEEE Visualization* (2002), pp. 275–282.

[38] HAMPEL, F. R., RONCHETTI, E. M., ROUSSEEUW, P. J., AND STAHEL, W. A. *Robust Statistics - The Approach Based on Influence Functions*. Wiley-Interscience, 1986.

[39] HARGREAVES, S. Deferred shading. *Game Developers' Conference* (2004).

[40] HEIDMANN, T. Real shadows, Real time. *Iris Universe, No. 18* (November 1991).

[41] HEIDRICH, W., AND SEIDEL, H.-P. Realistic, hardware-accelerated shading and lighting. In *Proc. ACM SIGGRAPH* (1999).

[42] HENSLEY, J., SCHEUERMANN, T., COOMBE, G., SINGH, M., AND LASTRA, A. Fast summed-area table generation and its applications. *Computer Graphics Forum 24*, 3 (2005), 547–555.

[43] HESTERBERG, T. Weighted average importance sampling and defensive mixture distributions. *Technometrics 37*, 2 (1995).

[44] JENSEN, H. W. Importance driven path tracing using the photon map. In *Proc. Eurographics Rendering Workshop* (1995), pp. 326–335.

[45] JENSEN, H. W. Global illumination using photon maps. In *Proc. Eurographics Rendering Workshop* (1996), pp. 21–30.

[46] JENSEN, H. W. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[47] JENSEN, H. W. *State of the Art in Monte Carlo Ray Tracing for Realistic Image Synthesis*. SIGGRAPH Course Notes. 2001.

[48] KAJIYA, J. T. The rendering equation. *Computer Graphics (Proc. ACM SIGGRAPH) 20*, 4 (1986), 143–150.

[49] KAUTZ, K., VAZQUEZ, P., HEIDRICH, W., AND SEIDEL, H. A unified approach to prefiltered environment maps. In *Proc. Eurographics Rendering Workshop* (2000).

[50] KAWAI, J. K., PAINTER, J. S., AND COHEN, M. F. Radioptimization - goal based rendering. In *Proc. ACM SIGGRAPH* (August 1993), pp. 147–154.

[51] KEMP, M. *Leonardo da Vinci: Experience, Experiment and Design*. Princeton University Press, 2006.

[52] LEE, C. H., AND HAO, X. Geometry-dependent lighting. *IEEE Trans. Visualization and Computer Graphics 12*, 2 (2006), 197–207.

[53] LEE, M. E., AND REDNER, R. A. A note on the use of nonlinear filtering in computer graphics. *IEEE Computer Graphics and Applications 10*, 3 (1990), 23–29.

[54] LINDSTROM, P. Out-of-core simplification of large polygonal models. In *Proc. SIGGRAPH* (2000), pp. 259–262.

[55] LOWELL, R. *Matters of Light & Depth*. Lowel-Light Manufacturing Inc., New York, NY, 1992.

[56] LUEBKE, D., REDDY, M., COHEN, J., VARSHNEY, A., B., W., AND HUEBNER, R. *Level of Detail for 3D Graphics*. Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers, San Francisco, CA, 2002.

[57] MCCOOL, M. D. Anisotropic diffusion for monte carlo noise reduction. *ACM Trans. Graphics 18*, 2 (1999), 171–194.

[58] MILLER, G. S., AND HOFFMAN, C. R. Illumination and reflection maps: Simulated objects in simulated and real environments. In *Course Notes for Advanced Computer Graphics Animation, SIGGRAPH*. ACM Press, 1984.

[59] MITRA, S. K. Digital filter design. In *Digital Signal Processing Laboratory Using MATLAB*. WCB/McGraw-Hill, 1999, ch. 7, pp. 107–126.

[60] MITRA, S. K. Digital processing of continuous-time signals. In *Digital Signal Processing Laboratory Using MATLAB*. WCB/McGraw-Hill, 1999, ch. 5, pp. 44–49.

[61] MOLNAR, S., EYLES, J., AND POULTON, J. PixelFlow: High-speed rendering using image composition. *Computer Graphics 26, 1992*, 2 (1992).

[62] NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graphics 22*, 3 (July 2003).

[63] NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graphics 23*, 3 (July 2004).

[64] NICODEMUS, F., RICHMON, J., HSIA, J., GINSBERG, I., AND LIMPERIS, T. Geometric considerations and nomenclature for reflectance. Tech. Rep. NBS Monograph 160, National Bureau of Standards, 1977.

[65] NIEDERREITER, H. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[66] OBERT, J., KŘIVÁNEK, J., PELLACINI, F., SÝKORA, D., AND PATTANAIK, S. iCheat: A Representation for Artistic Control of Indirect Cinematic Lighting. *Computer Graphics Forum 27*, 4 (2008), 1217–1223.

[67] OWEN, A., AND ZHOU, Y. Safe and effective importance sampling. *Journal American Statistical Assosciation* (2000).

[68] OXFORD ENGLISH DICTIONARY. *"Filter, n"*, 2nd ed. Oxford University Press, 1989. From *OED Online*, accessed Feb. 14th 2009.

[69] PATMORE, C. *Movie-Making Course: Principles, Practice and Techniques*. Barron's Educational Series, 2005.

[70] PELLACINI, F., TOLE, P., AND GREENBERG, D. P. A user interface for interactive cinematic shadow design. *ACM Trans. Graphics 21*, 3 (July 2002), 563–566.

[71] PELLACINI, F., VIDIMCE, K., LEFOHN, A., MOHR, A., LEONE, M., AND WARREN, J. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Trans. Graphics 24*, 3 (Aug. 2005), 464–470.

[72] PENG, J., KRISTJANSSON, D., AND ZORIN, D. Interactive modeling of topologically complex geometric detail. *ACM Trans. Graphics 23*, 3 (July 2004), 635–643.

[73] PETROVIC, L., FUJITO, B., WILLIAMS, L., AND FINKELSTEIN, A. Shadows for cel animation. In *Proc. SIGGRAPH* (July 2000), pp. 511–516.

[74] PHARR, M., AND HUMPHREYS, G. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004.

[75] PHONG, B. T. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317.

[76] POULIN, P., AND FOURNIER, A. Lights from highlights and shadows. In *Proc. Symp. Interactive 3D Graphics* (Mar. 1992), vol. 25, pp. 31–38.

[77] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes, The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[78] RAMAMOORTHI, R., AND HANRAHAN, P. An efficient representation for irradiance environment maps. In *Proc. ACM SIGGRAPH* (2001).

[79] RAMAMOORTHI, R., AND HANRAHAN, P. Frequency space environment map rendering. *ACM Trans. Graphics 21* (July 2002), 517–526.

[80] REDD, A. Chroma-cinema: The use of color in color and black-and-white films. 1999.

[81] ROSS, S. M. *Introduction to Probability Models*, 8th ed. Elsevier Academic Press, 2003.

[82] RUSHMEIER, H., AND WARD, G. Energy-preserving non-linear filters. *Computer Graphics (Proc. SIGGRAPH)* (1994).

[83] SHACKED, R., AND LISCHINSKI, D. Automatic lighting design using a perceptual quality metric. *Computer Graphics Forum 20*, 3 (2001), 215–226.

[84] SHANNON, C. Communication in the presence of noise. *Proc. Institute of Radio Engineers 37*, 1 (1949, Reprinted *Proc. IEEE*, Vol. 86, No.2, Feb. 1999).

[85] SLOAN, P., KAUTZ, J., AND SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic low-frequency lighting environments. *ACM Trans. Graphics 21*, 3 (July 2002).

[86] SLOAN, P.-P., LUNA, B., AND SNYDER, J. Local, deformable precomputed radiance transfer. *ACM Trans. Graphics 24*, 3 (July 2005), 1216–1224.

[87] STAMMINGER, M., AND DRETTAKIS, G. Perspective shadow maps. In *Proc. SIGGRAPH* (2002), pp. 557–562.

[88] STOCKMAN, A., MACLEOD, D., AND JOHNSON, N. Spectral sensitivities of the human cones. *Journal Optical Society of America, Optics and Image Science 10* (1993). Rendering originally from Wikipedia article 'Cone Cell'.

[89] STOLLNITZ, E. J., DEROSE, T. D., AND SALESIN, D. H. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufman, 1996.

[90] SUN, W., AND MUKHERJEE, A. Generalized wavelet product integral for rendering dynamic glossy objects. *ACM Trans. Graphics 25*, 3 (2006), 955–966.

[91] SUYKENS, F., AND WILLEMS, Y. D. Adaptive filtering for progressive monte carlo image rendering. In *Proc. Intl. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media* (2000).

[92] TAO, T. Distributions. In *Princeton Companion to Mathematics*. Princeton University Press, 2009, ch. 3.18, pp. 184–187.

[93] TAO, T. The Fourier transform. In *The Princeton Companion to Mathematics*. Princeton University Press, 2009, ch. 3.27, pp. 204–207.

[94] TOMASI, C., AND MANDUCHI, R. Bilateral filtering for gray and color images. In *Proc. Intl. Conf. Computer Vision* (1998), pp. 839–845.

[95] TUKEY, J. W. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[96] VEACH, E. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[97] VEACH, E., AND GUIBAS, L. J. Optimally combining sampling techniques for monte carlo rendering. In *Proc. SIGGRAPH* (1995), pp. 419–428.

[98] WALL, E. J. *A Dictionary of Photography*, 9th ed. Hazell, Watson and Viney, Ltd., 1889, p. 335.

[99] WARD, G. J., AND HECKBERT, P. S. Irradiance gradients. In *Proc. Eurographics Workshop on Rendering* (1992).

[100] WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. A ray tracing solution for diffuse interreflection. In *Computer Graphics (Proc. SIGGRAPH)* (1988), pp. 85–92.

[101] WILLIAMS, L. Casting curved shadows on curved surfaces. In *Computer Graphics (Proc. SIGGRAPH)* (1978).

[102] WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. Light space perspective shadow maps. In *Proc. Eurographics Symp. Rendering* (2004).

[103] XU, R., AND PATTANIK, S. Non-iterative, robust monte carlo noise reduction. *Computer Graphics and Applications* (February 2004).