Exploiting Temporal Coherence for Pre-computation Based Rendering

Ryan S. Overbeck

Submitted in partial fulfillment of the
requirements for the degree
of Master of Science
in the Graduate School of Engineering and Applied Sciences

COLUMBIA UNIVERSITY

2006

ABSTRACT

Exploiting Temporal Coherence for Pre-computation Based Rendering

Ryan S. Overbeck

Precomputed radiance transfer (PRT) generates impressive images with complex illumination, materials and shadows with real-time interactivity. These methods separate the scene's static and dynamic components allowing the static portion to be computed as a preprocess. In this work, we hold geometry static and allow either the lighting or BRDF to be dynamic. To achieve real-time performance, both static and dynamic components are compressed by exploiting spatial and angular coherence. *Temporal coherence* of the dynamic component from frame to frame is an important, but unexplored additional form of coherence. In this thesis, we explore temporal coherence of two forms of all-frequency PRT: BRDF material editing and lighting design. We develop *incremental* methods for approximating the *differences* in the dynamic component between consecutive frames. For BRDF editing, we find that a pure *incremental* approach allows quick convergence to an *exact* solution with smooth real-time response.

For relighting, we observe vastly differing degrees of temporal coherence accross levels of the lighting's wavelet hierarchy. To address this, we develop an algorithm that treats each level separately, adapting to available coherence. The proposed methods are othogonal to other forms of coherence, and can be added to almost any PRT algorithm with minimal implementation, computation, or memory overhead. We demonstrate our technique within existing codes for nonlinear wavelet approximation, changing view with BRDF factorization, and clustered PCA. Exploiting temporal coherence of dynamic lighting yields a 3×–4× performance improvement, e.g., all-frequency effects are achieved with 30 wavelet coefficients, about the same as low-frequency spherical harmonic methods. Distinctly, our algorithm *smoothly converges* to the *exact* result within a few frames of the lighting becoming static.

Contents

1 Introduction **2**

2 Previous Work **6**

3 Background **8**

3.1 PRT for Relighting 8

3.2 PRT for BRDF editing 9

4 Basic Incremental for BRDF editing **13**

4.1 BRDF Editing vs. Relighting 15

5 Analysis of Temporal Coherence for Relighting **17**

5.1 Comparison of Incremental and Non-Incremental 17

5.2 Detailed Analysis of Temporal Coherence 20

6 Per-Band Incremental Wavelet Algorithm **23**

6.1 Basic Per-Band Algorithm 23

6.2 Selecting When to Update Incrementally 25

6.2.1 Exhaustive Search 25

6.2.2 Simple and Fast Per-Band Test 26

6.2.3 Dealer Algorithm 27

6.3 Results and Discussion 27

7 Integration with PRT methods **30**

7.1 Basic Image Relighting 30

7.2 Changing View with BRDF Factorization 31
7.3 Clustered PCA 33

8 PBI for Spherical Harmonics **37**

9 Conclusions and Future Work **42**

Bibliography **43**

List of Figures

1.1 Comparison of our algorithm (per-band incremental or PBI) with standard (non-incremental) PRT for relighting. 5

3.1 Editing the BRDF of pearl earrings. 12

4.1 BI compared to standard NWA for BRDF editing. 16

5.1 Comparison of lighting approximations with 30 wavelet terms, for rotating the Grace Cathedral cubemap. 18

5.2 Rendered images for the lighting sequence in Figure 5.1 19

5.3 Coverage maps and histograms of BI and NWA for some frames from Figure 5.1. 21

5.4 A study of temporal coherence, independent of any algorithm. 22

6.1 Pseudocode for Per-Band Incremental Wavelet Algorithm (PBI). 24

6.2 Comparison of lighting accuracy over time for different algorithms 26

6.3 Comparison of images from PBI and basic incremental BI. 28

6.4 Incremental vs non-incremental updates for different bands using PBI. . . . 29

7.1 Comparison of different numbers of wavelet terms for PBI and standard NWA 31

7.2 Comparison of images with standard PRT and PBI for 30 wavelets, as well as standard PRT with 100 wavelets 33

7.3 CPCA, using our temporal coherence algorithm. 34

8.1 SetupBands() procedure for Per-Band Incremental Spherical Harmonics. . . 39

8.2 PBI for Spherical Harmonics using rotations from Figure 5.1. 40
8.3 Comparing 25 to 100 term approximation using linear spherical harmonics and NWA. 41

ACKNOWLEDGEMENTS

This thesis would not exist without the overwhelming amount of support and guidance I received from a few specific individuals. First and foremost, I must thank Assistant Professor Ravi Ramamoorthi, my thesis advisor. He, along with Assistant Professor Eitan Grinspun, introduced me to the fundamental ideas that led to this work and, more generally, to the computer graphics research process as a whole. Their enthusiasm was my fuel throughout this past year.

I began this project as an offshoot of Aner Ben-Artzi's work on real-time BRDF editing. Aner and I work very closely, and many of the ideas in this thesis are equally his. I also must thank him for introducing me to shower crayons.

Chapter 1

Introduction

Precomputed radiance transfer (PRT) addresses an important goal in computer graphics: real-time rendering with dynamic natural lighting, realistic materials and complex cast shadows [SKS02]. We focus on all-frequency PRT methods, which use wavelet representations for intricate lighting and shadowing effects. In their simplest form, these methods compute [NRH03]

$$B = MV, \tag{1.1}$$

where B is a vector of outgoing light intensities (or image pixels), M is a matrix representation of the static light transport, and V is a dynamic vector. To be specific, for relighting,

$$B = TL, \tag{1.2}$$

where L is the dynamic lighting environment and T is commonly known as the transport matrix where each column T_i represents the appearance of the scene under basis light L_i. For BRDF editing,

$$B = T'c \tag{1.3}$$

where T' is the transport integrated with a 4D BRDF quotient and c is a 1D BRDF curve (see [BOR06] for details). In both cases the matrix $M(T \text{ or } T')$ is multiplied against the vector $V(L \text{ or } c)$ at real-time rates.

PRT can be viewed as a *compressed, accelerated* matrix-vector multiplication for equation 1.1. Ng et al. [NRH03] compressed L in equation 1.2 using a nonlinear wavelet approximation (NWA), with only 100-200 terms. Liu et al. [LSSS04] and Wang et al. [WTL04] extended NWA to glossy materials with changing view via BRDF factorization. While these works exploited angular coherence in L, Liu et al. [LSSS04] also exploited spatial coherence in the scene to compress the transport matrix T using clustered principal component analysis (CPCA).

We identify a fundamentally new form of coherence: in real-time rendering, the dynamic vector V in equation 1.1 is *temporally coherent*. We design more efficient algorithms by *incrementally* compressing the *difference* in V between consecutive frames. Besides further accelerating PRT, our approach naturally yields a solution which *quickly and smoothly converges* to an *exact* representation of V when held static. This qualitatively enhances the user's experience. Our specific contributions are:

Analysis of Temporal Coherence: This thesis identifies temporal coherence as a key avenue for further research and compression in PRT methods. We find that a basic incremental (BI) approach (see Chapter 4) suffices for 1D BRDF curve editing since edits tend to be local. However for relighting, a series of experiments (see Chapter 5) on a rotating lighting environment exposes (i) the approximations and artifacts of alternative algorithms, and (ii) the inherent spatio-temporal coupling of the coherence in complex illumination (see Figure 5.4).

Per-Band Incremental (PBI) Wavelet Algorithm: We develop an algorithm (see Chapter 6) that adapts to the temporal coherence of each wavelet level, dynamically choosing an incremental update over standard NWA when profitable. The results, compared to standard PRT methods, are often dramatic (see Figure 1.1), and free of the flickering and ghosting artifacts of a straightforward basic incremental (BI) method (Chapter 4). When the evolution of the lighting is slow, static or changes only over a sparse set of directions, PBI is able to incrementally update all the wavelet bands, preserving or approaching a nearly exact solution. Even when the lighting changes rapidly, PBI preserves temporal coherence of the coarser wavelets.

Integration with PRT Methods: PBI integrates easily into existing all-frequency methods: it leaves open the alternatives for precomputing and representing T. We demonstrate PBI in the context of the original image relighting method [NRH03], the extension to changing view using BRDF factorization [WTL04], and clustered PCA [LSSS04] (see Chapter 7). In all cases, only about 100 lines of additional code is required, and the time and memory overheads are negligible.

While the lighting is changing dynamically, our method can usually lead to improvements by a factor of three or four. We obtain high-quality all-frequency effects with only 30 wavelet lighting terms (see Figure 1.1), comparable to the coefficient budget of *low*-frequency spherical harmonic methods. Within a few frames of the lighting remaining static (the user being idle), we converge to the *exact* result. The exact solution is maintained even under changing viewpoint in methods such as [LSSS04, WTL04].

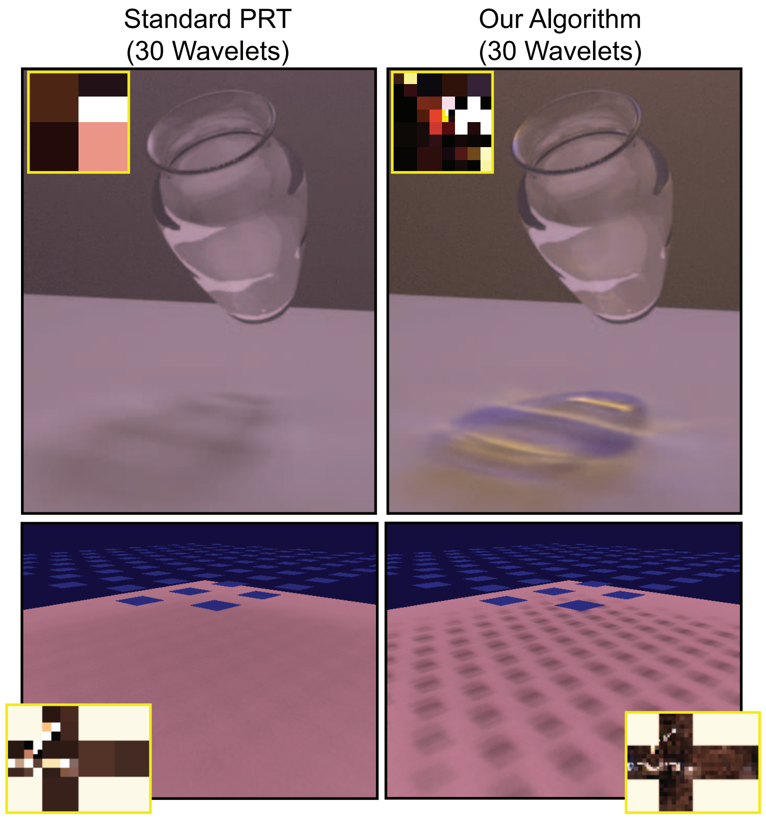

Figure 1.1: Comparison of our algorithm (per-band incremental or PBI) with standard (non-incremental) PRT *for relighting. PBI integrates easily into existing frameworks, such as image relighting (top) and clustered PCA (bottom). PBI (right) captures all-frequency effects including caustics (top) and sharp shadows (bottom) which at these framerates are blurred by non-incremental methods (left). Insets compare the quality of the lighting approximation.*

Chapter 2

Previous Work

We discuss previous precomputation-based rendering methods, and techniques for exploiting temporal coherence in other domains. We focus on PRT relighting since Ben-artzi et al. only recently applied PRT to BRDF editing in [BOR06]. We briefly summarize PRT for BRDF editing in Section 3.2.

Precomputation-based relighting or radiance transfer (PRT) was introduced by Sloan et al. [SKS02, SHHS03, SLS05] , building on prior work on design of time-dependent lighting by Dorsey et al. [DAG95] and others. Much of this work focuses on low-frequency effects, using spherical harmonics [RH01]. We will briefly discuss these methods in Chapter 8, but we focus primarily on all-frequency relighting [NRH03], which reproduces a richer class of visual effects and stands to benefit more from leveraging temporal coherence.

We work with the fundamental algorithms [NRH03, WTL04, LSSS04], that form the basic building blocks for all-frequency PRT. Our focus is on real-time rendering—thus, we do not consider all-frequency triple product algorithms [NRH04, ZHL$^+$05] that are not real-time. Recent advances (e.g., translucent materials [WTL05]) fit into our approach as they are variants of equation 1.2, differing only in the transport matrix T. Since we change the representation of L only, our method can be easily integrated into most existing or future PRT algorithms.

In general, the literature in rendering, and even beyond graphics, is rich in its coverage of temporal coherence. Since most of these previous approaches are not suitable for PRT algorithms, we give only a brief survey.

One may imagine applying video compression [SS00] to a pre-defined lighting sequence. However, the size of the lighting is small compared to the size of the transport matrix T. Moreover, the lighting sequence in an interactive system is not predetermined. Finally, our goal is really to accelerate the matrix-vector multiplication in equation 1.1, which is not sped up by compression techniques such as optical flow or sparse bitrate coding.

For offline rendering of dynamically-lit animations, Wan et al. [WWL05] exploit temporal coherence in importance sampling environment maps to reduce flickering. They build adaptive spherical quad-trees for creating point-samples in a raytracing framework, whereas PRT necessitates an orthonormal (wavelet) basis in equation 1.1. While reduced flicker is a side benefit of our approach, our main focus is on improved efficiency for real-time rendering.

In frameless rendering [BFMZ94, DWWL05], pixels update asynchronously, while in our approach, wavelet lighting coefficients update asynchronously; combining these orthogonal approaches remains future work. Similarly, in ray tracing, there are numerous ways to exploit temporal coherence. For example, [WDP99] reproject points that have already been shaded. Finally, our approach can loosely be interpreted as a form of multiplexing, because we update a small number of wavelet coefficients at each time frame, but ensure that a very much larger number are updated over a longer contiguous sequence of frames.

Chapter 3

Background

Here we present some of the fundamentals of PRT applied to both relighting and BRDF editing. PRT methods begin with the reflection equation

$$B\left(x, w_o\right) = \int_{\Omega_{4\pi}} L\left(w_i\right) V\left(x, w_i\right) \rho\left(w_i, w_o\right) \cos\theta_i dw_i, \tag{3.1}$$

where L is the distant lighting environment, V is the binary visibility, ρ is the BRDF, and w_i and w_o are incoming and outgoing directions at the point x. Depending on the application, x is either position of the vertices or position as seen through each pixel in the rendered image. Hence B will be the final color of either pixels or vertices. To make this integral tractable for real-time rendering, we seek an equation of the form of 1.1. Note that while the integral in equation 3.1 only represents direct illumination, it can be extended to global illumination as in [NRH03], and as shown in Figure 1.1. For now we use equation 3.1 for simplicity in the following derivations.

3.1 PRT for Relighting

If we fix the scene geometry and remove dependence on w_o by either restricting ourselves to Lambertian BRDFs or fixing the viewpoint, we can lump together the visibility, BRDF, and cosine term into the static light transport matrix T:

$$T\left(x, w_i\right) = V\left(x, w_i\right) \rho\left(w_i\right) \cos\theta_i. \tag{3.2}$$

$T(x, w_i)$ and $L(w_i)$ can be viewed as discrete vectors in w_i (often represented as cubemaps) allowing us to compute integral in 3.1 as a dot-product:

$$B(x) = T(x) \cdot L \tag{3.3}$$

When taken over all pixels in the image (or all vertices in the scene) x, we can view L as a vector and T as a matrix whose rows are the visibility at each pixel (or vertex) over all w_i, and columns are images of the scene as lit by basis light L_i over all x. The dot product in 3.3 grows to a matrix-vector product,

$$B = TL. \tag{3.4}$$

This is the standard PRT per-pixel image relighting equation. It is also used for per-vertex relighting with changing view and strictly diffuse surfaces. We will discuss extensions of PRT relighting to changing view with glossy materials in Section 7.2. T now contains all of the static elements of equation 3.1 and is computed as a preprocess. Finally, we project T and L into some orthonormal and compressible basis (usually wavelets [NRH03] or spherical harmonics [SKS02]) allowing us to approximate this matrix-vector product in real-time while arbitrarily varying the dynamic lighting vector L.

Ng et al. [NRH03] demonstrate that while computing B at image pixels allows arbitrary light transport effects such as reflection and caustics, it requires fixing the viewpoint. Therefore more recent work, [SKS02], [SHHS03], [LSSS04], [SLS05], [WTL04], and [WTL05] to name a few, compute B at scene vertices to permit real-time changing view with somewhat limited transport effects.

3.2 PRT for BRDF editing

Equation 3.4 is the most common factorization of the reflection equation 3.1 and can render complex shadows while changing view and lighting in real-time. Ben-Artzi et al. [BOR06] propose a different factorization for curve based editing of 1D BRDF factors. They assert that any BRDF $\rho(w_i, w_o)$ can be expressed as:

$$\rho(w_i, w_o) = \rho_q(w_i, w_o)\, c(\theta_h(w_i, w_o)), \tag{3.5}$$

where c is the editable 1D BRDF curve and ρ_q is the 4D quotient BRDF (the BRDF with c divided out). In this example, the editable dimension is θ_h, the half-angle, a 1D function of w_i and w_o, but other BRDF parameterizations provide other variable dimensions. We then discretize c by projecting it into a linear combination of J basis functions ([BOR06] use Daubechies 4 wavelets with $J = 256$),

$$c\left(\theta_h\left(w_i, w_o\right)\right) = \sum_{j=1}^{J} c_j b_j\left(\theta_h\left(w_i, w_o\right)\right), \tag{3.6}$$

and plug it into equation 3.5 then equation 3.1, and pull c_j outside the integral:

$$B\left(x, w_o\right) = \sum_j c_j \int_{\Omega_{4\pi}} L\left(w_i\right) V\left(x, w_i\right) \rho_q\left(w_i, w_o\right) b_j\left(\theta_h\left(w_i, w_o\right)\right) \cos\theta_i dw_i. \tag{3.7}$$

If we fix the viewpoint ($w_o = w_o\left(x\right)$) and lighting, we can factor out lighting, visibility, BRDF basis functions, and the cosine term into a new static light transport matrix:

$$T'_j\left(x\right) = \int_{\Omega_{4\pi}} L\left(w_i\right) V\left(x, w_i\right) \rho_q\left(w_i, x\right) b_j\left(w_i, x\right) \cos\theta_i dw_i. \tag{3.8}$$

Integration of the T_j's with the b_j's projects them into the same 1D basis as the c_j's. We are left with the matrix-vector product

$$B = T'c, \tag{3.9}$$

with variable coefficients c_j.

A smart BRDF factorization in equation 3.5 leads to coefficients c_j that are either linked to variables of an analytic BRDF or are themselves physically meaningful parameters of a measured BRDF. Again, since both T' and c are represented in a compressible basis (Daubechies 4 wavelets), we can approximate the matrix-vector product in real-time.

[BOR06] show how to factor basis functions out of the Cook-Torrance, Ashikhmin-Shirley, and other models as well as useful parameterizations of measured BRDFs.

For the examples in this thesis, we use the Cook-Torrance model [CT82] and measured BRDFs. The Cook-Torrance model,

$$\rho = \frac{F_{n,e}\left(\theta_d\right) G\left(w_i, w_o\right) D_\sigma\left(\theta_h\right)}{4\pi\left(w_i \cdot N\right)\left(w_o \cdot N\right)}, \tag{3.10}$$

has three variables of interest: the Fresnel index of refraction n, the Fresnel extinction coefficient e, and the mean slope distribution σ. When changing σ, the editable curve is actually the slope distribution function,

$$c\left(\theta_h\right) = D_\sigma\left(\theta_h\right), \tag{3.11}$$

which can be computed in real-time in respone to changes in σ. In reference to equation 3.7, the c_j's are $D_\sigma\left(\theta_h\right)$ projected onto wavelets along θ_h, and ρ_q is the rest of **??**:

$$\rho_q = \frac{F_{n,e}\left(\theta_d\right) G\left(w_i, w_o\right)}{4\pi\left(w_i \cdot N\right)\left(w_o \cdot N\right)}. \tag{3.12}$$

Measured BRDFs can be represented in many physically meaningful 3D and 4D parameterizations, and there are several options for factoring these high-dimensional data. Ben-Artzi et al. [BOR06] use homomorphic factorization [MAA01] for 3D isotropic BRDFs and non-negative matrix factorization [LRR04] for full 4D anisotropy.

To evaluate equations 3.9 or 3.4 in real-time we have to drop the smallest wavelet coefficients in L or c, keeping only a small fraction. If we simply recompute the product each frame, as per NWA, we achieve only a rough estimate of B. In this thesis, we aim to reuse results from previous frames to leverage both accuracy and real-time user response.

Figure 3.1 shows an example of using curve based edits of c to alter the appearance, B, of a pair of pearl earrings and a velvet cloth. The earrings use Cook-Torrance for their specular component and the cloth uses homomorphic factorization to separate the BRDF.

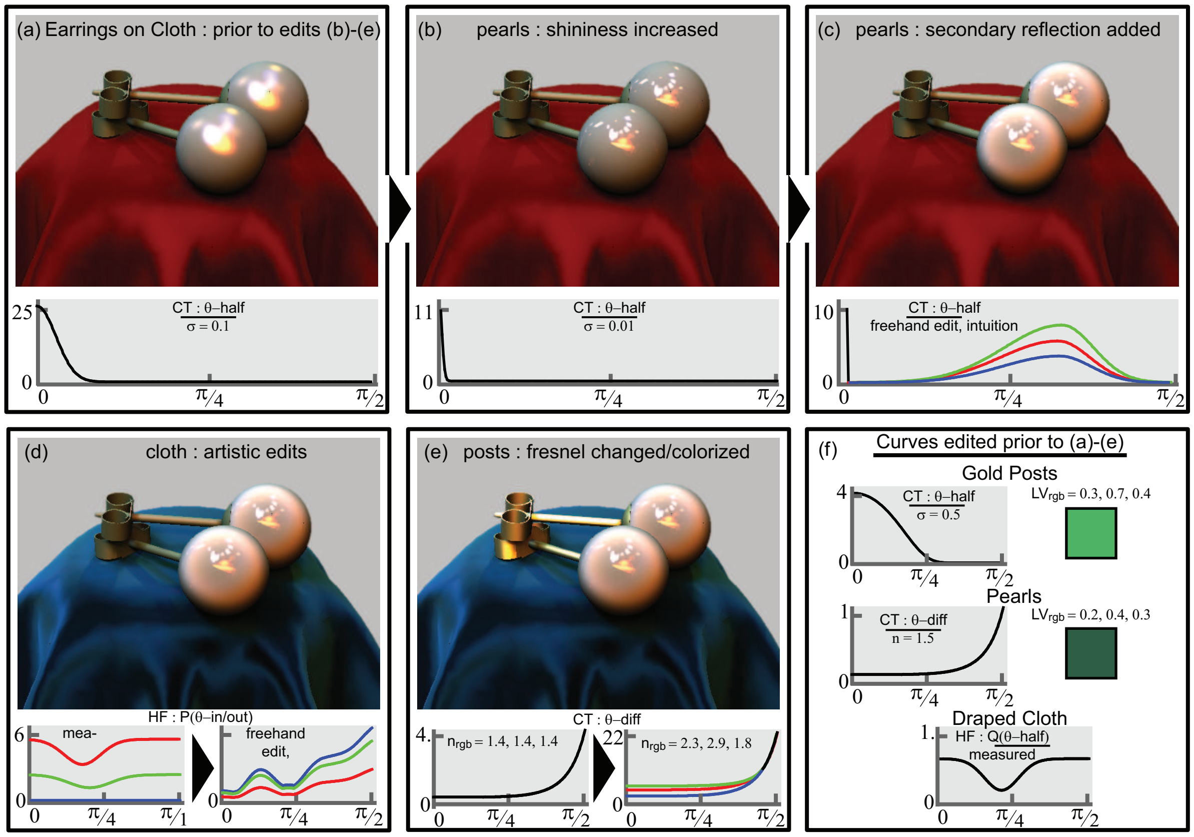

Figure 3.1: Editing the BRDF of pearl earrings. *A sample editing session shows before (a) and after (e) of a scene with pearl earrings on a cloth draped over a pedestal, as illuminated in Grace Cathedral. The pearls and posts use a Cook-Torrance specular term + LV diffuse term. The cloth uses homomorphic factorization as in [MAA01] to factor a measured BRDF. The session begins by setting some initial values for the editable BRDF curves (f), and loading data for red velvet. First (b), the pearls are given sharper reflections. The "hazy" secondary reflection, the signature of a real pearl, is added in (c). In (d) the user artistically edits the curve to produce blue velvet. Finally (e), the Fresnel term of the posts is adjusted to give them a metallic gold appearance.* This is Figure 2 from [BOR06].

Chapter 4

Basic Incremental for BRDF editing

Consider a basic incremental wavelet algorithm that leverages temporal coherence in V from equation 1.1(or c in 1.3 or L in 1.2). This algorithm, which motivates the remainder of the thesis, will need significant improvement for relighting, so we call it *basic incremental* (BI). To be concrete, consider equation 1.3[1] and [NRH03] (NWA) as the initial, non-incremental framework for our discussion. c is the BRDF vector in a full wavelet basis (typical size 256).

First, we rewrite equation 1.3 to make the approximation explicit,

$$B = T'\tilde{c} \ , \tag{4.1}$$

where $\tilde{c} = \text{Approx}(c)$ is the (compressed) dynamic vector in a truncated wavelet basis (typical dimension 10-30). Our basic idea is to consider the *change* in the vector from the previous frame, $\triangle c$, replacing equation 4.1 with the incremental update,

$$\begin{aligned} B^{new} &= B^{old} + \triangle B && (4.2)\\ \triangle B &= T' \triangle c \ . && (4.3) \end{aligned}$$

The computational and memory overhead is minimal. Storage of the previous frame B^{old} is negligible compared to the size of T', and the cost of computing equation 4.2 is negligible relative to the matrix-vector multiplication in equation 4.3 (or equation 4.1).

[1]Recall that this is equivalent to equation 1.2 for relighting.

Our insight is that $\triangle c$ *is much more compressible than* c. Therefore we write,

$$\triangle c = \text{Approx}\left(c^{\text{new}} - \tilde{c}\right) \ , \tag{4.4}$$

$$\tilde{c} = \tilde{c} + \triangle c \ . \tag{4.5}$$

We use a tilde for $\tilde{c}$ in equation 4.5, to signify that it is a wavelet *approximation* to the vector, which is updated at each frame.

Basic Incremental (BI) algorithm: Equations 4.2, 4.3, 4.4 and 4.5 make up the most basic approach to an incremental BRDF update. The method leverages the observation that $c^{\text{new}} - \tilde{c}$ can be more aggressively and sparsely approximated than c^{new}. To initialize[2], we usually assume $\tilde{c}^0 = c^0$ at the intial frame 0. In our BRDF editing implementation, we adopted the Daubechies 4 wavelet basis on a 256 element BRDF curve. However, there is nothing in the above discussion that restricts the basis representation used. For example, our relighting system uses a 2D Haar wavelet basis on a $6 \times 64 \times 64$ cubemap [NRH03].

Besides the high compressibility of $\triangle c$, a useful property of BI is that it *progressively converges* to the *exact* result when the user is idle (c^{new} is static) in a design session. Observe that a constant c^{new} acts as a fixed point under repeated iteration of BI. Indeed, if the vector is fixed,

$$\triangle c = \text{Approx}\left(c - \tilde{c}\right)$$

$$\tilde{c} = \tilde{c} + \triangle c,$$

where we drop the superscript since the vector is static.

This progressive convergence holds even in the extreme case of a wavelet basis truncated to a single term. Of course, when a more reasonable wavelet budget is used, convergence is very rapid. Since BRDF curve edits tend to be local, often a single incremental update achieves convergence. In contrast to an approximation to the static solution in equation 1.3 as per non-incremental NWA, the BRDF designer sees the *exact* result. In summary, BI

[2] To initialize, we compute the full matrix-vector multiply in equation 1.3. This takes a few seconds, which is generally small compared to the time required to load the (large) transport matrix T' into memory from disk, and initialize other auxiliary data structures that the relighting framework needs. An alternative is to initialize $\tilde{c}^0$ to the non-incremental NWA of c^0.

takes advantage of available CPU cycles to progressively improve a static (or slowly changing) image, whereas for NWA, error is always capped by wavelet budget.

Figure 4.1 shows a frame from a BRDF curve editing session. (a) Is the final pearl material from Figure 3.1. BI allows real-time BRDF curve edits under complex lighting with *exact* results in (a) and (c).

4.1 BRDF Editing vs. Relighting

BI may suffice for 1D BRDF curve editing. However, as we see in Chapter 5, BI applied to relighting causes significant artifacts when the entire wavelet hierarchy is subject to change.

BRDF edits tend to be localized, usually amplifying, suppressing, or translating individual peaks in the curve, allowing incremental updates to focus on the locus of change. Some relighting edits are likewise localized. Sparse lighting changes, such as positioning and changing the area of a single light source, are well approximated by BI. NWA must approximate both static and dynamic portions of the environment, while BI converges rapidly on the static environment, and thereafter allocates the full wavelet budget to represent the localized dynamic lighting component. Environment rotation, on the other hand, changes every pixel, and hence every wavelet, in the environment cubemap.

The size of the vectors in equations 1.2 and 1.3 also determine BI's utility. The 1D BRDF curve tends to be only 256 bins in length. If we restrict ourselves to updating only 30 wavelets per frame, we are still able to touch 12% of the BRDF curve each frame. The entire curve can change and we are still guaranteed to converge in at most 9 frames. The environment cubemap for relighting, on the other hand, tends to be $6 \times 64 \times 64 = 24576$. 30 represents about .1% of this vector, which is simply not enough to keep up with global changes to the wavelet hierarchy within a reasonable number of frames. Our experiments have shown that increasing the wavelet budget to 150-200 wavelets allows environment rotations without distracting artifacts, but at a high cost to interactivity. In chapter 6, we solve this problem by introducing a per-band incremental algorithm.

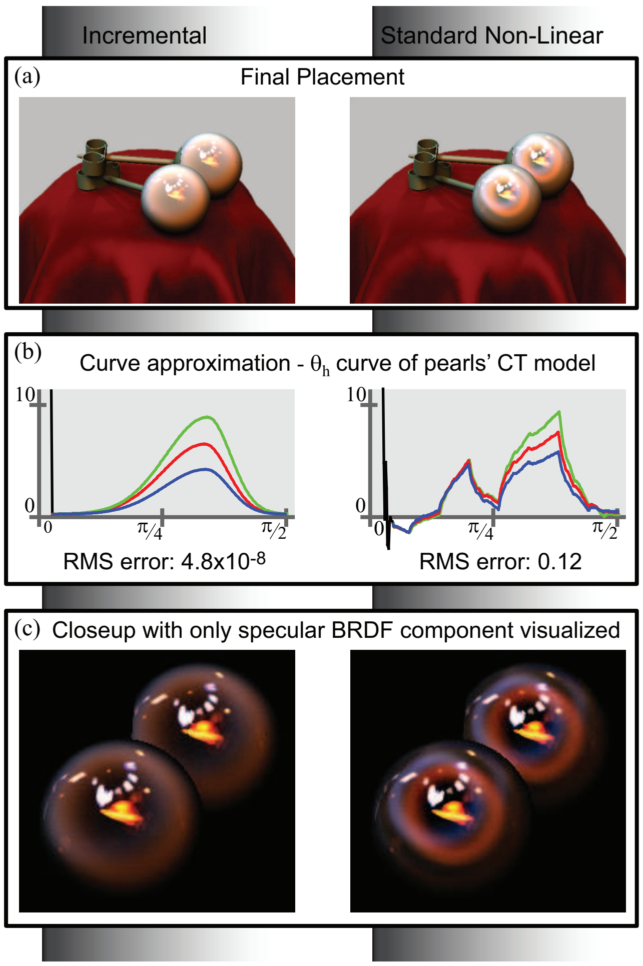
Incremental
Standard Non-Linear
(a) Final Placement
(b) Curve approximation - θh curve of pearls' CT model
10
0
0
π/4
π/2
RMS error: 4.8x10-8
RMS error: 0.12
(c) Closeup with only specular BRDF component visualized

Figure 4.1: BI compared to standard NWA in BRDF editing. *(a) uses the final pearl material from Figure 3.1. Changes to the θ_h BRDF curve (b) are incrementally approximated. (c) shows only the specular component being edited.* This is Figure 5 in [BOR06].

Chapter 5

Analysis of Temporal Coherence for Relighting

We now turn to the study of BI in the context of relighting to better understand BI and more generally temporal coherence of dynamic lighting. In this case, instead of equation 4.1, we have:

$$B = T\tilde{L}. \tag{5.1}$$

The observations in this section motivate the robust, efficient PBI method in Chapter 6.

5.1 Comparison of Incremental and Non-Incremental

Consider the rotation of the Grace Cathedral lighting environment. Figures 5.1 and 5.2 depict temporal evolution of the lighting cubemap and the rendered image, respectively (see also Figure 6.4–bottom). This example is representative of numerous experiments spanning a range of light manipulations, scenes, and shading complexities. Rotations are the most challenging test because the illumination is dynamic almost everywhere and they do not fit the special cases discussed above. We compare NWA, reference, BI, and (for completeness) PBI, always using 30 wavelet terms.

Initial Frames: Initially (frame 0) $\tilde{L}^0 = L^0$, and BI's lighting approximation exactly matches the reference. Indeed, early on, while rotation is relatively slow, BI's lighting

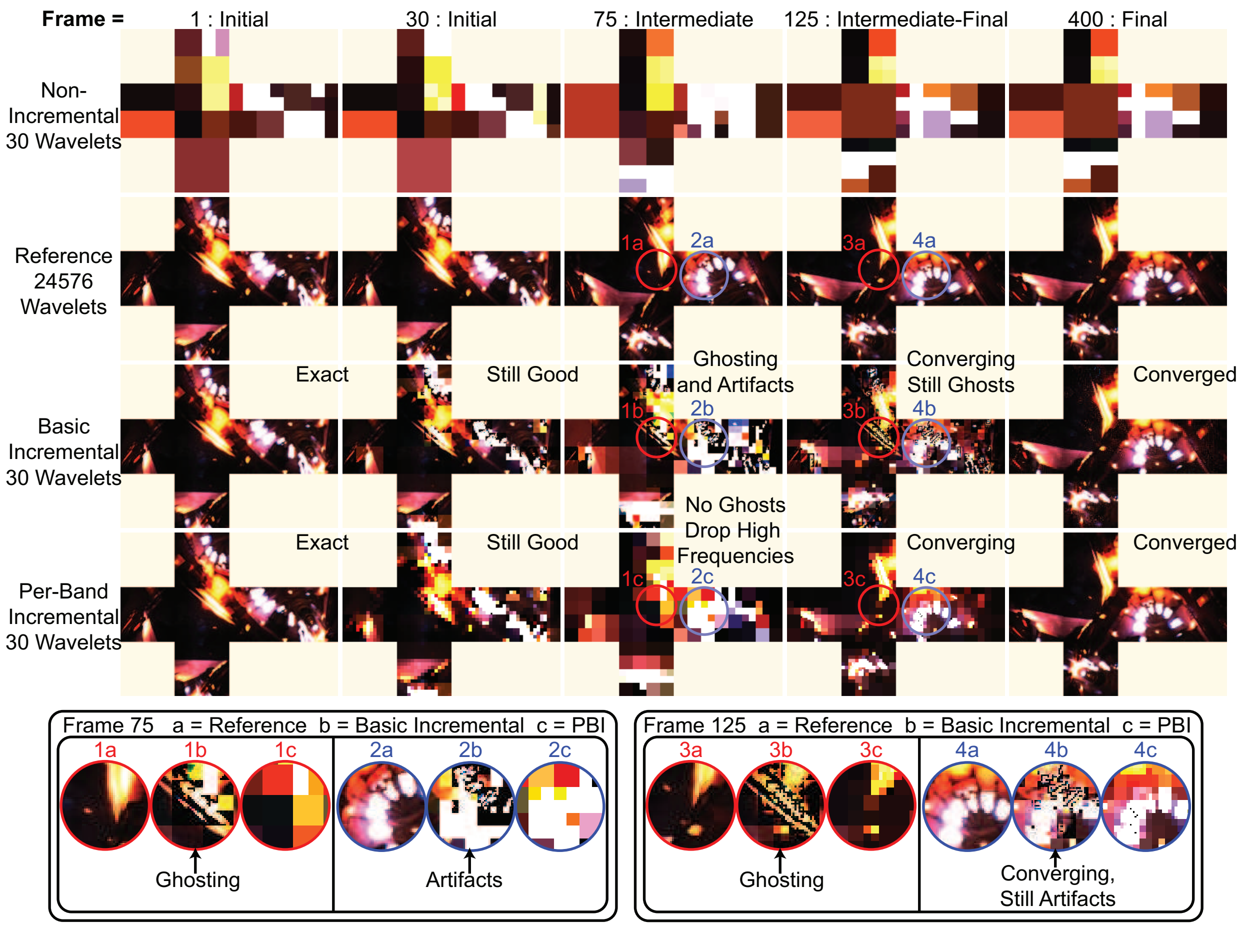

Figure 5.1: Comparison of lighting approximations with 30 wavelet terms, for rotating the Grace Cathedral cubemap. *The top row is NWA (non-incremental PRT), followed by the reference image, the basic incremental BI algorithm from Chapter 4, and the per-band incremental PBI method to be developed in Chapter 6. The bottom row shows details that reveal the performance and artifacts of the different algorithms.*

approximation is significantly sharper and more accurate than NWA's (see frame 30, Figure 5.1). The resulting images (see Figure 5.2) also display much sharper shadows, accurately matching the reference. This is because BI needs only to approximate the *change* in the lighting at each frame.

Intermediate Behavior and Artifacts: Next, consider intermediate times (see frame 75 in Figures 5.1 and 5.2). The lighting now differs significantly from its intial state, and rotation rate is relatively fast. BI's quantitative error is still smaller than NWA's. Even so,

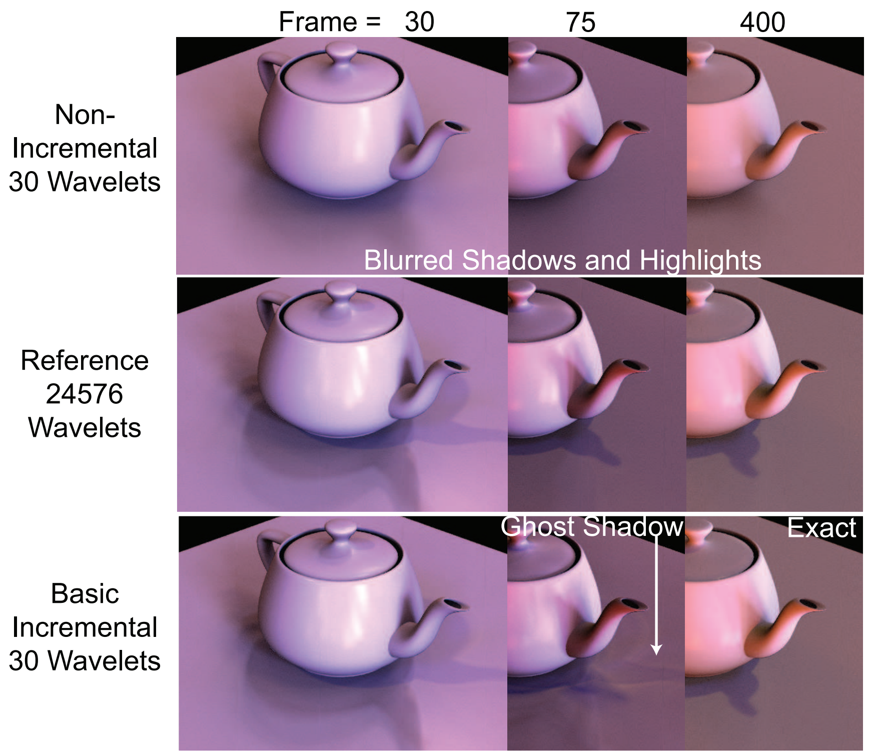
Frame = 30
75
400
Non-Incremental 30 Wavelets
Blurred Shadows and Highlights
Reference 24576 Wavelets
Ghost Shadow
Exact
Basic Incremental 30 Wavelets

***Figure 5.2: Rendered images for the lighting sequence in Figure 5.1**, comparing NWA (top), the reference (middle), and basic incremental BI (bottom). A comparison of BI with the PBI method is shown later, in Figure 6.3.*

while BI's shadows and lighting continue to be sharper than NWA's, they are inaccurate and spurious in many locations.

Figure 5.1-(1a/1b) highlights undesirable ghosting artifacts. For instance, consider the small bright light in the inset. As it rotates, a purely incremental technique such as BI must zero it in its old location *as well as* add it to the new location—an operation that can be more expensive than simply approximating it in the standard way with a coarser representation. With its limited wavelet budget, BI cannot keep up, with lights leaving trails or *ghosts* in the old locations. This can lead to spurious sharp shadows in the images (see frame 75, Figure 5.2). There are also significant high-frequency artifacts (see insets 2a–2b, Figire 5.1) where BI cannot approximate the lighting sharply enough. In Chapter 6, we introduce a per-band incremental algorithm (PBI) which avoids these artifacts by using an incremental update only for wavelet bands that have sufficient temporal coherence; compare Figure 5.1-(1b/1c) or 5.1-(2b/2c).

Final Frames and Convergence: We stop the rotation sequence at frame 99, and let the lighting be static. As discussed in Chapter 4, this allows the incremental algorithm to converge to the correct lighting. Since we are using 30 wavelets per timestep, frame 125 in Figure 5.1 is effectively using a 750-term wavelet approximation, and some regions have begun to converge (compare insets 4a and 4b). However, the previous ghosting is severe enough that some regions still show artifacts (compare insets 3a and 3b). Moreover, note from the insets that the PBI method in Chapter 6 is essentially converged at frame 125. Finally, at frame 400, the incremental algorithm has converged fully, and the image in Figure 5.2 accurately matches the reference. By contrast, the non-incremental algorithm does not improve with time, when the lighting is static.

5.2 Detailed Analysis of Temporal Coherence

We now show some more detailed results, characterizing the nature of temporal coherence. These observations will be taken into account in the Chapter 6, to design the improved PBI algorithm.

Coverage of Wavelets in Incremental and Non-Incremental: In Figure 5.3, we compare which wavelets are updated at each frame (what the coverage of the lighting is) for non-incremental NWA, versus incremental BI. Similar results also hold for the PBI method.

From the top of Figure 5.3, we see that BI by design updates different regions of the environment at adjacent frames (once a wavelet is updated, the change in the next frame will not usually warrant it being updated immediately again). By contrast, essentially the same wavelets are chosen at adjacent frames for non-incremental NWA. In these images, a pixel is shaded based on how many of the wavelet levels that overlap it are chosen at each frame. Coarser blocks indicate coarser wavelet coverage, and finer blocks indicate finer coverage in those regions. The bottom of Figure 5.3 considers the cumulative result over 5 frames of lighting motion. The non-incremental algorithm has a cumulative or average coverage that looks very similar to each individual frame. By contrast, BI updates a large number of wavelets with much finer frequencies over a 5 frame interval.

The bottom of Figure 5.3 also shows a histogram of how many wavelets are updated at

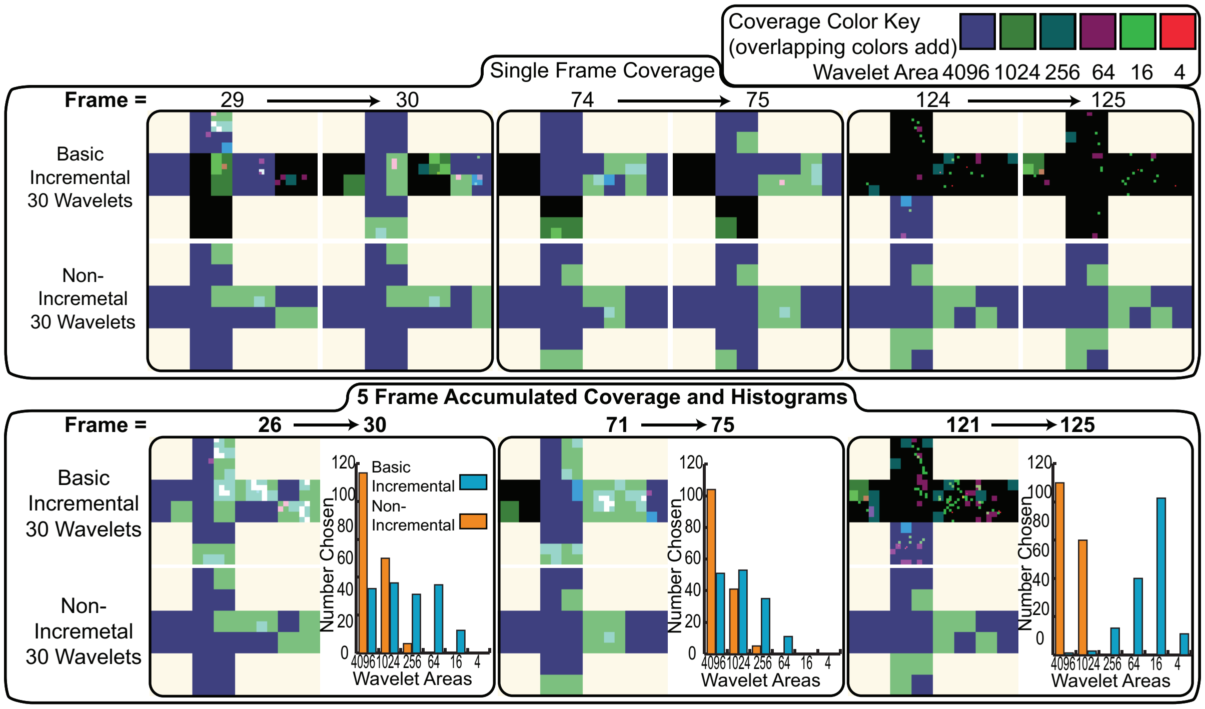

Figure 5.3: **Top:** *Coverage maps for incremental (BI) and non-incremental (NWA) algorithms for some frames from Figure 5.1.* **Bottom:** *Histogram and averages, over a 5 frame interval, of which wavelets and wavelet levels are chosen by incremental (BI) and non-incremental (NWA) algorithms.*

each level. NWA must always choose low-frequency coarse wavelets, that usually have the greatest energy. In fact, levels finer than 256 are not chosen at all, so the effective resolution of the environment map is only $6 \times 8 \times 8$. However, we will see that these coarse wavelets also exhibit the greatest temporal coherence, and BI can therefore update them only once every several frames, while still maintaining an accurate approximation. Hence, many more terms can be devoted to finer wavelets, producing a more uniform distribution into finer levels, and higher-quality images that use an effectively higher resolution environment map.

It is also instructive to compare the three frames (columns) in Figure 5.3. On the left (frame 30), BI can keep track of very high frequencies, as seen in the histogram. In the middle (frame 75), the lighting rotation is faster, and more updates must be given to lower frequencies, somewhat reducing the effective resolution. Towards the end (frame 125), the lighting is static and the approximation is converging, with work focused exclusively on

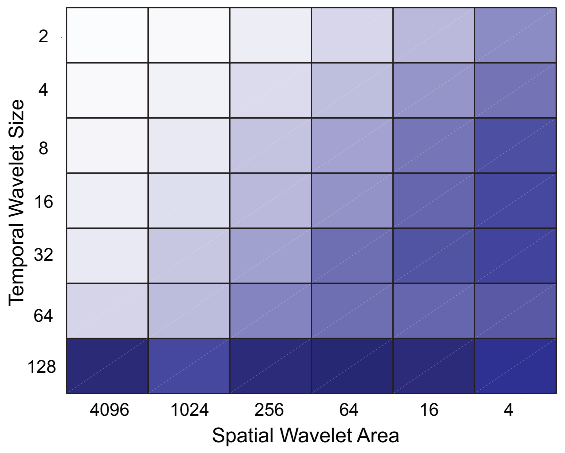

Figure 5.4: A study of temporal coherence, independent of any algorithm. *We show the norm of energy (darker is more) in each spatio-temporal wavelet band, as measured for the (uncompressed reference) rotation sequence of Figure 5.1. Columns correspond to spatial bands, rows to temporal bands, and the evident diagonal structure implies that progressively finer spatial bands exhibit progressively diminishing temporal coherence.*

the higher-frequency or smaller wavelet bands. By contrast, non-incremental NWA always updates essentially the same (coarse) wavelet levels.

Relation of Spatial Frequency and Temporal Coherence: Figure 5.4 visualizes temporal coherence, independent of any specific practical algorithm. We take the first 128 frames of the rotation sequence, wavelet transformed along the spatial (angular) dimensions in the normal way, and then apply a 1D Haar transform along the time dimension. In Figure 5.4, we plot the total energy for given spatial and temporal wavelet bands, with darker regions having more energy. The coarsest spatial wavelets with area $4096 = 64 \times 64$ have almost all of their temporal energy in the lowest frequency temporal band (size 128)—this follows from the observation that rotation does not significantly change the overall energy. As we go to finer spatial wavelets, there is more energy in finer temporal wavelets—the visible diagonal structure indicates that the extent of temporal coherence decreases with spatial wavelet frequency, with more coherence in low-frequency bands than in high-frequency bands. Unfortunately, the basic incremental algorithm treats each band similarly, which (due to the dark upper-right quarter of Figure 5.4) can lead to ghosting and artifacts at high spatial frequencies.

Chapter 6

Per-Band Incremental Wavelet Algorithm

Building on these observations, we propose a per-band incremental (PBI) lighting update algorithm that treats each wavelet band separately, choosing either an incremental or non-incremental approach, based on the available temporal coherence.

6.1 Basic Per-Band Algorithm

First, we group wavelets having area $4096 = 64 \times 64$ (the coarsest wavelet and scaling function) into one band, those with area $1024 = 32 \times 32$ (the next coarsest) into another band and so on. Since we consider cubemaps with resolution 64×64, there will in general be 6 wavelet levels or bands (1 to 6). For each band separately, we will decide whether to update it incrementally, as per Chapter 4, or in the standard non-incremental fashion, as per equation 4.1.

The details of our algorithm are summarized in Figure 6.1. First, we set up all the bands, determining whether they are updated incrementally or not (lines 2 and 5). How we do this optimally is a critical part of our algorithm, discussed in Section 6.2. Then, we must choose which wavelets to update (line 3). This is straightforward, since we simply need to sort them in the standard way based on their magnitudes. We use area weighting for choosing wavelets, as recommended in [NRH03], but transport (or any other) weighting

```
Per-Band Incremental Wavelets (PBI)
  Procedure SetupBands()              // Described in Sec. 6.2
1.  for all Bands $i$
2.    $\text{IsIncr}_i = \text{Incremental(i)}$;        // Should band i be incremental
3.    $W^i = \text{Wavelets}(i)$ ;          // Which wavelets in $i$ to update
4.  end ;
  Procedure PBI()                     // Per-Band Algorithm
5.  SetupBands() ;
6.  for all Bands $i$
7.    if $\text{IsIncr}_\text{i}$                       // Update incrementally
8.      for all chosen wavelets $j$ in $W^i$
9.        $\triangle L_j = L_j^{\text{new}} - \tilde{L}_j$ ;        // Equation (4.4)
10.       $\tilde{L}_j = L_j^{\text{new}}$ ;              // Equation (4.5)
11.       $B_i = B_i + T_j \triangle L_j$;      // Equations (4.2) and (4.3)
12.     end;
13.   else                            // Update non-incrementally
14.   $B_i = 0$ ; $\tilde{L}_{\text{Band i}} = 0$ ;         // Zero or reset lights and image
15.     for all chosen wavelets $j$ in $W^i$
16.       $\tilde{L}_j = L_j^{\text{new}}$ ;              // Equation (4.5)
17.       $B_i = B_i + T_j \tilde{L}_j$ ;         // Equation (4.1)
18.     end;
19. end ;
20. $B = \sum_{i=1}^{6} B_i$ ;                  // Sum over all bands
```

Figure 6.1: Pseudocode for Per-Band Incremental Wavelet Algorithm (PBI).

or unweighted selection could also be used. If a band is updated non-incrementally, as in standard PRT, we use the area-weighted magnitude of wavelet j, Area(j) $\mid L_j^{\text{new}} \mid$ for sorting; otherwise, we use the difference Area(j) $\mid \triangle L_j \mid =$ Area(j) $\mid L_j^{\text{new}} - \tilde{L}_j \mid$.

We treat each band separately (line 6), eventually summing their contributions (line

20). If the band is updated incrementally (lines 8-11), we use equations 4.2–4.5. For each wavelet j in that band's approximation, we compute the change $\triangle L_j$ relative to the current value $\tilde{L}_j$ (line 9), and also bring the current value up to date (line 10). In line 11, we add the contributions to the band image B_i. Since we are considering a single wavelet j, we will use a single column T_j of the transport matrix. Note that our method easily integrates with standard PRT, and can make use of any optimizations, such as division of the image into blocks for better caching [NRH03]. If the band does not have sufficient temporal coherence for an incremental update, it is simply updated as in standard PRT (line 17). We still update $\tilde{L}_j = L_j^{\mathrm{new}}$ in line 16, because future frames can (and usually will) still choose to update the band incrementally.

6.2 Selecting When to Update Incrementally

We need to know when there is enough temporal coherence to update a band incrementally. One possibility is to let the user specify a threshold, with coarser bands updating incrementally, and finer bands using standard PRT. However, a static threshold is difficult to specify or adapt to different speeds of motion. Ideally, we would like the algorithm to automatically pick coarser bands for incremental updates when the lighting changes rapidly, and finer bands for slower lighting changes where there is more temporal coherence. We have tested three automated approaches that range from exhaustive and expensive, to very simple and efficient.

6.2.1 Exhaustive Search

We consider every possibility for incremental vs non-incremental update over all bands, and pick the one that results in the least error for the lighting. For N wavelet bands, there are 2^N possibilities. While this method imposes too much computational overhead to be practical, it is exhaustive (optimal within the scope of one frame) by design and therefore serves as a useful baseline to compare alternatives. In an offline setting, another possible baseline might have involved a spacetime optimization over all frames, but in our interactive application the lighting is not known a priori.

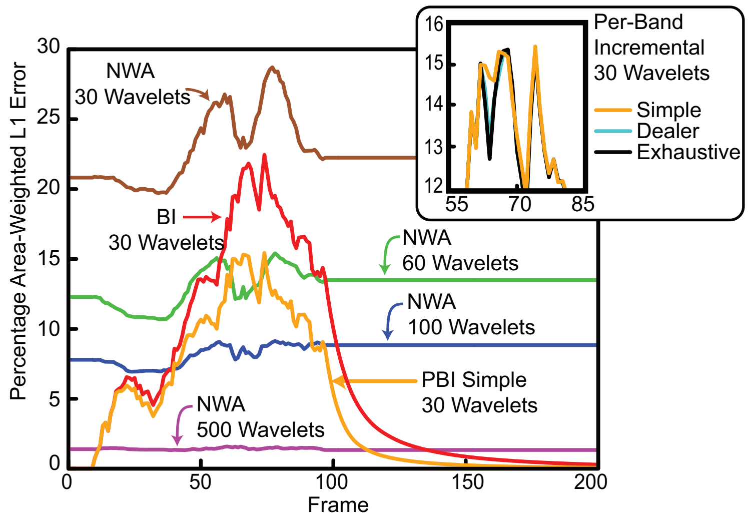

Figure 6.2: Comparison of lighting accuracy over time for different algorithms *(standard or non-incremental NWA, incremental BI, per-band incremental PBI). The inset compares the three selection methods for PBI.*

6.2.2 Simple and Fast Per-Band Test

At the other extreme, we simply test each band separately, determining whether it is better approximated incrementally or not. To do so, we compare the norm[1] for each band $\| L^{\text{new}} - \tilde{L} \|$ with $\| L^{\text{new}} \|$. If the former has a smaller error, we use an incremental update, using non-incremental otherwise. Note that non-incremental updating can be thought of as incremental with a previous value of 0, and our comparison is equivalent to seeing if the new lighting is closer to 0 or to the current approximation $\tilde{L}$. This makes it explicit that the lighting can sometimes drift so far from the current approximation, that it is better to reset or zero the band. The method is greedy because the error comparison is done once at the beginning, before knowing how many wavelet terms are actually allocated to the band. Because of its simplicity, this algorithm has little computational overhead, and is very easy to implement.

[1]In practice, we find the L_1 norm best for the perceptual quality of the final images. Similar quantitative results are also obtained with L_2.

6.2.3 Dealer Algorithm

The dealer algorithm "deals" out one wavelet coefficient at a time. It introduces some computational overhead, while providing a marginally better result than the simple per-band test. A wavelet coefficient is greedily "dealt" to the band where the net error will decrease the most. This decrease is measured as the maximum of net error decrease over two alternatives—non-incremental or incremental update of the band—holding all other bands fixed. Dealing then repeats until the wavelet budget is exhausted. In summary, this method simultaneously allocates wavelet coefficients to bands, while determining whether or not to update them incrementally.

6.3 Results and Discussion

We now discuss some properties of the PBI algorithm and compare it with basic incremental BI and non-incremental NWA methods.

Figure 6.2 plots the area-weighted L_1 error for the sequence in Figure 5.1. PBI clearly out-performs BI and non-incremental NWA. Moreover, it converges faster than BI. Note that BI always performs better quantitatively than NWA, but has relatively large errors in the middle of the rotation sequence because of the ghosting and artifacts. Its performance is close to PBI in the early part of the sequence, when both methods accurately approximate the lighting.

The inset in Figure 6.2 compares the three methods just discussed for selecting whether or not to update incrementally in PBI. In most cases, all three approaches perform nearly identically—we do not show all 3 curves in the main plot since one cannot distinguish them at that scale (the error axis in the inset is magnified). There are only marginal improvements for dealer and exhaustive over the simple per-band test. Hence, because of its implementation simplicity and low computational overhead, we will always use the simple test.

We can also attempt to see how many wavelets are needed in standard PRT to produce equal quality results as PBI. Because of the fundamentally different nature of the algorithms, we plot a number of curves in Figure 6.2. PBI with 30 wavelets is essentially always

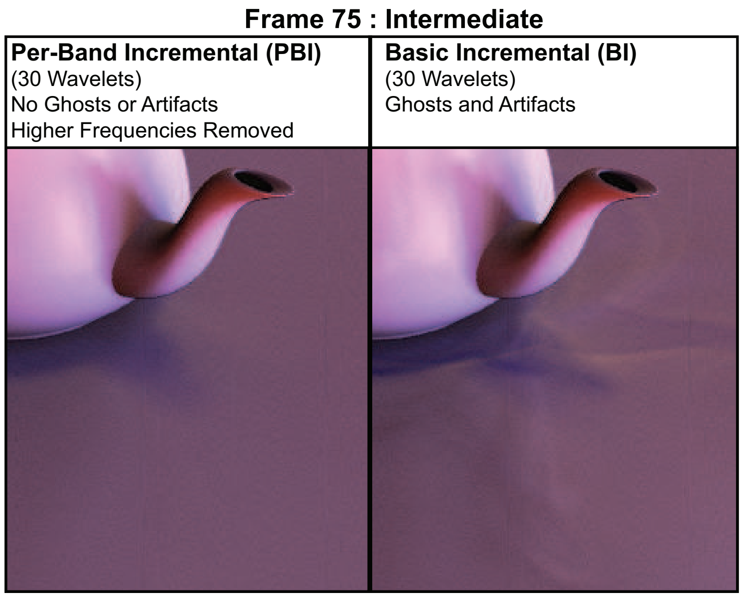

Figure 6.3: Comparison of images from PBI and basic incremental BI.

better than standard non-incremental NWA with 60 wavelets. Moreover, approximately 100 wavelets in non-incremental are needed to be comparable (sometimes better, sometimes worse) to PBI over the full sequence, while the lighting is rotating. However, if we include the static regions, where PBI converges, even a 500 wavelet non-incremental approximation cannot achieve equal quality as our method within 25 time steps of stopping rotation.

Figure 6.3 compares PBI to BI (for intermediate frame 75 from Figure 5.2). We can clearly see a sharp shadow without the ghosting and artifacts. Similarly, Figures 1.1 and 5.1, and the closeups, clearly show that PBI significantly outperforms BI and standard PRT.

Figure 6.4 shows the characteristic behavior of PBI for different wavelet sizes or bands. The vertical lines correspond to frames where that band was updated non-incrementally. As can be seen, the bands update incrementally most of the time, but are *occasionally reset or zeroed out*, updating non-incrementally for that frame. This follows our intuition—as the solution moves further away from the stored value, it is better to restart occasionally. The frequency of restarting (non-incremental frames) depends on the speed of motion (lighting change) and wavelet level. Coarser wavelets exhibit greater temporal coherence—in fact, the two coarsest levels (sizes 4096 and 1024) always update incrementally. As the wavelet

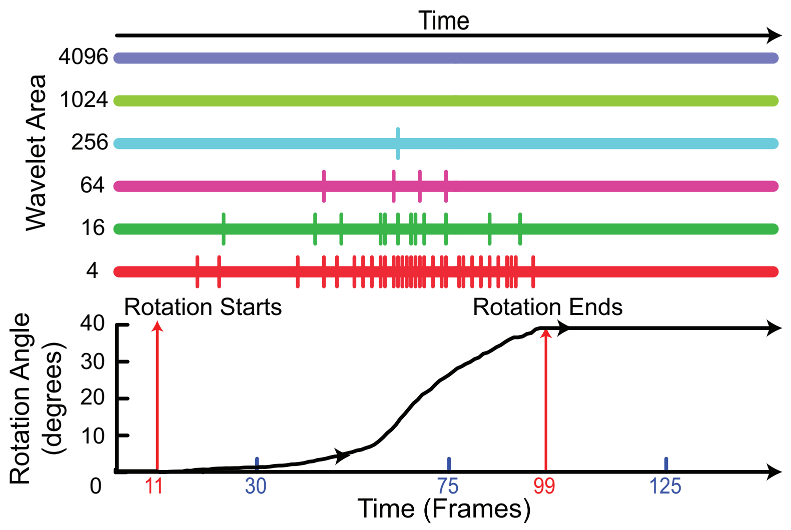

Figure 6.4: (top) Incremental (horizontal line) vs non-incremental (vertical line) updates for different bands using PBI. (bottom) Rotation angle for the lighting rotation sequence. *Bands are occasionally reset, or evaluated non-incrementally, when they drift too far from the stored value, with more frequent resets for higher frequency or finer wavelets.*

level gets finer, restarting becomes more frequent. PBI automatically adapts the frequency of restarts, or non-incremental updates, to the rate of illumination change and wavelet level.

Finally, we consider the computational and memory overhead for PBI. The memory overhead is primarily the stored value or previous frame's (floating point 512×512) image for each of the 6 wavelet bands. Together with (small) auxiliary data structures, the total extra storage is 19.2 MegaBytes. By comparison, the transport matrix and auxiliary structures for the scene in Figure 5.1 occupy 229 MB, and this can be larger for more complex scenes. Hence, the memory overhead is only 8% for this scene. The computational overhead comes primarily from adding the per-band images in line 20 of Figure 6.1. This is a fixed cost, and the relative time decreases as we increase the wavelet budget. Even if we only update 1 wavelet per frame, the overhead is only 20%. For realistic wavelet budgets, such as the dynamic lighting sequence in Figure 5.1 with 30 wavelets, the overhead is less than 5%—PBI averaged 14.2 frames per second, and standard NWA averaged 14.8 fps. Since the computational overhead for PBI is minimal, we refer to the number of wavelets used to quantify performance through out this thesis (rather than running times that are implementation and machine-specific).

Chapter 7

Integration with PRT methods

In this section, we integrate our per-band incremental (PBI) wavelet algorithm into three methods that form the basic building blocks for all all-frequency PRT algorithms, showing a variety of results.

7.1 Basic Image Relighting

We start with basic image relighting [NRH03], which we have already used to understand and develop the key ideas. For implementation, we simply modified the code framework in [NRH03] to incorporate PBI. The modifications affected only the lighting approximation and matrix multiplication phases, and required only about 100 lines of additional code. As seen in Figures 1.1 (top), 5.1 and 6.3, PBI is accurate, and produces significantly higher quality results than standard PRT without artifacts.

Figure 7.1 shows another example on a 512×512 image of the plant scene with intricate shadowing. We compare closeups as we increase the number of terms in both PBI and standard PRT. For equal time (30 or 100 wavelets), PBI has significantly sharper shadows in dynamic lighting. Three to four times as many terms are needed in standard PRT for equal quality across a fair range of wavelet approximations (about 100 in standard for 30 wavelets in PBI, and 300 in standard for 100 in PBI). Finally, within 5 frames of stopping lighting motion, PBI has essentially converged, and a 30 term approximation is comparable to 300 terms in standard PRT.

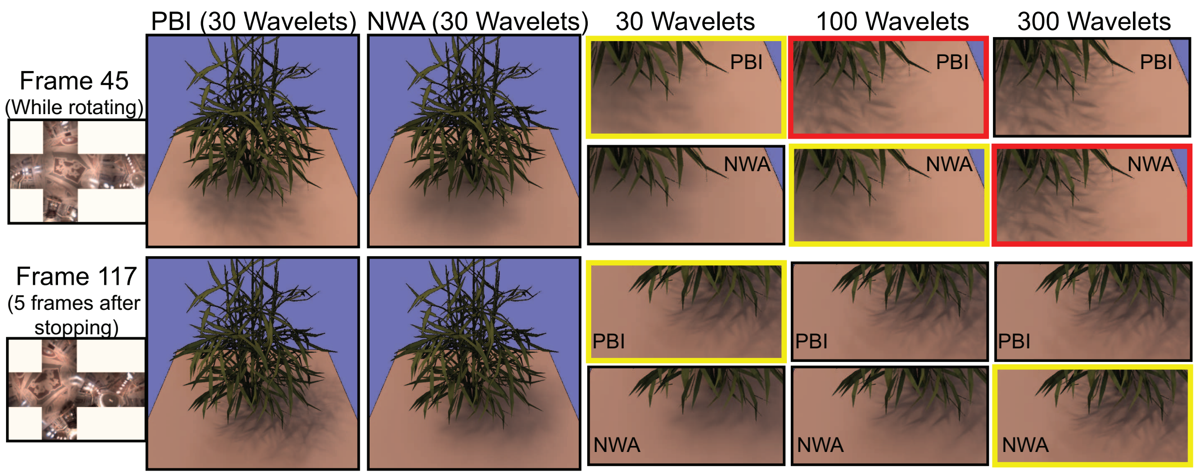

***Figure 7.1: Comparison of different numbers of wavelet terms for PBI and standard NWA**, while rotating (top) and within 5 frames of stopping (bottom). On top, we see that three to four times as many wavelet terms are needed for equal quality in standard PRT. Moreover, about 10 times as many terms is needed within a few frames of stopping (bottom). For equal time, with the same number of wavelets, PBI consistently has much sharper shadows than standard PRT.*

Since the quality of the image (such as the sharpness of shadows) in PBI depends on the speed of lighting variation, the shadows will get softer or sharper as the user speeds up or slows down the change in lighting. In many applications, such as lighting design, this is a very desirable behavior, with progressive refinement any time the user stops or even slows to make fine adjustments.

7.2 Changing View with BRDF Factorization

We now consider the extension to varying view as well as lighting, taking glossy materials into account. This section integrates PBI into a simple implementation of [LSSS04, WTL04].

Those methods use an in-out factorization of the BRDF,

$$\rho(\omega_i, \omega_o) = \sum_{k=1}^{K} g_k(\omega_i) h_k(\omega_o), \tag{7.1}$$

where ρ is the BRDF, factored into products g_k and h_k with a total of K terms. As reported in [LSSS04, WTL04], 3 to 10 terms suffice even for fairly shiny materials.

For PRT, one now folds the incident angle-dependent factor $g_k(\omega_i)$ into the transport matrix, with a separate T_k for each BRDF term. The outgoing dependence $h_k(\omega_o)$ is independent of the incident lighting, and simply modulates the final result,

$$B(x, \omega_o) = \sum_{k=1}^{K} h_k(\omega_o)(T_k L), \tag{7.2}$$

where the term $(T_k L)$ in parentheses is a matrix-vector multiplication, like equation 1.1, and the weighting by $h_k(\omega_o)$ allows for view-variation. There is minimal overhead for the addition above.

To take advantage of temporal coherence, we simply apply PBI to the lighting once, and then use this lighting approximation for all k, and each matrix-vector multiplication $T_k L$. The weighting by $h_k(\omega_o)$ to obtain the final result in equation 7.2 proceeds as before. In general, both the standard implementation and the method with PBI integrated operate much the same way as [NRH03], but are always per-vertex instead of per-pixel, and require K matrix-vector computations, as well as K times as much storage.

As before, the PBI method can be integrated in less than a hundred lines of code, and involves negligible computational or memory overhead. When the lighting is dynamic, we obtain a speedup (or increase in quality) by a factor of 3 or 4, and the solution rapidly converges to the exact result when the lighting is static.

Figure 7.2 shows a scene with 40,029 vertices (largely on the ground plane to capture intricate shadows), and complex BRDFs (note the fairly sharp Phong highlights on the street lamps, especially in the right column—we use $K = 4$ BRDF terms). We render this complex scene at real-time rates with PBI, with much sharper shadows than standard PRT, using only 30 wavelet terms. As seen in the bottom row, even 100 wavelet terms in standard PRT performs somewhat worse than PBI. To stress the generality of our method, the first two columns show two types of light manipulation—rotation as before, and interpolating two environments (Grace and StPeters), as one would for example when moving between two spatial locations in a video game. Note that the view can also simultaneously change in these examples. In the third column, we change viewpoint only. Since the lighting is static, the PBI algorithm very rapidly converges to the *exact* solution, which is accurately maintained while changing view, and is much sharper than the 100 term standard PRT comparison.

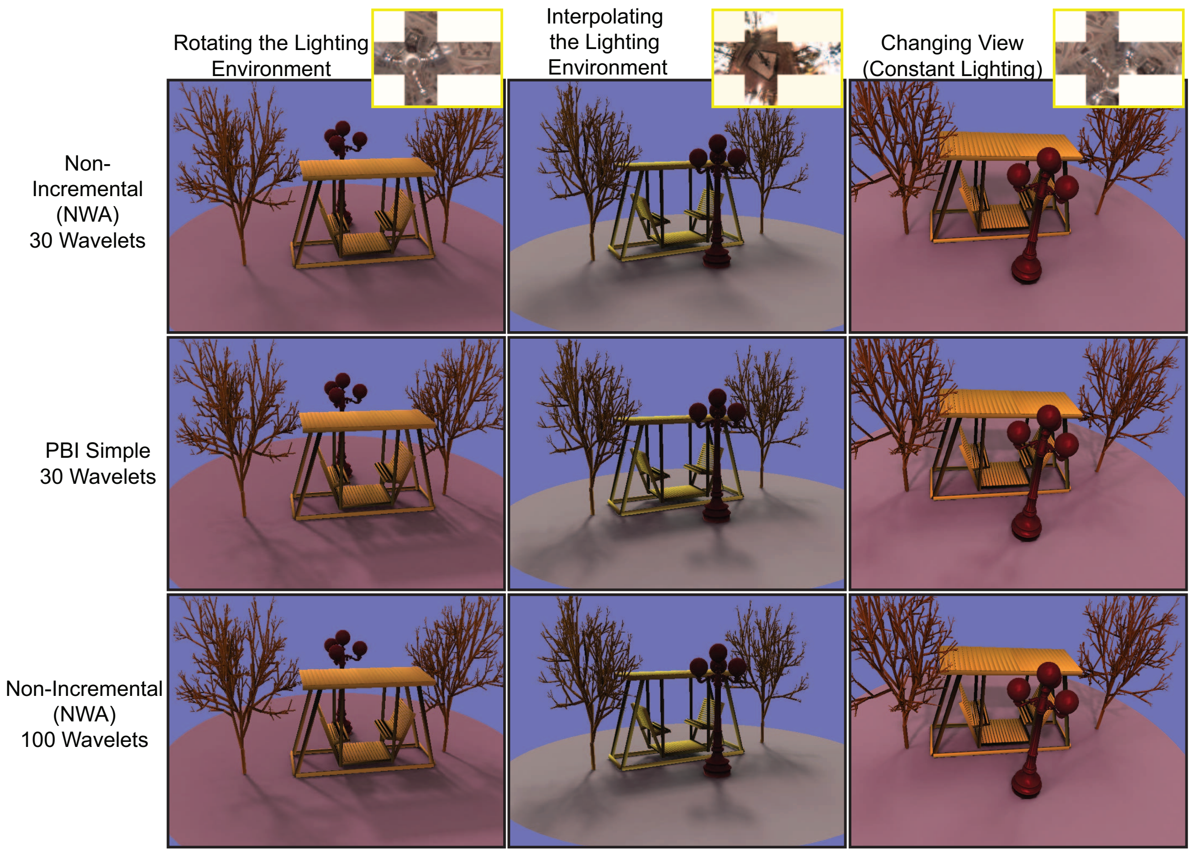
Rotating the Lighting Environment
Interpolating the Lighting Environment
Changing View (Constant Lighting)
Non-Incremental (NWA) 30 Wavelets
PBI Simple 30 Wavelets
Non-Incremental (NWA) 100 Wavelets

Figure 7.2: ***Comparison of images with standard PRT and PBI for 30 wavelets, as well as standard PRT with 100 wavelets*** *(which is marginally worse quality than the 30 term PBI approximation). For dynamic lighting (first two columns), PBI produces much sharper shadows than PRT with the same number of wavelet terms. We obtain exact results when only the view is changing in the right column—in this case, PBI is much sharper than the 100 term non-incremental result.*

7.3 Clustered PCA

Clustered PCA [LSSS04, SHHS03] or CPCA compresses the transport matrices T using spatial coherence, for greater compactness and efficiency. The vertices of the scene are broken into clusters, each of which is approximated with a low-dimensional PCA basis. We emphasize that our method can be applied "blindly" with any representation of T,

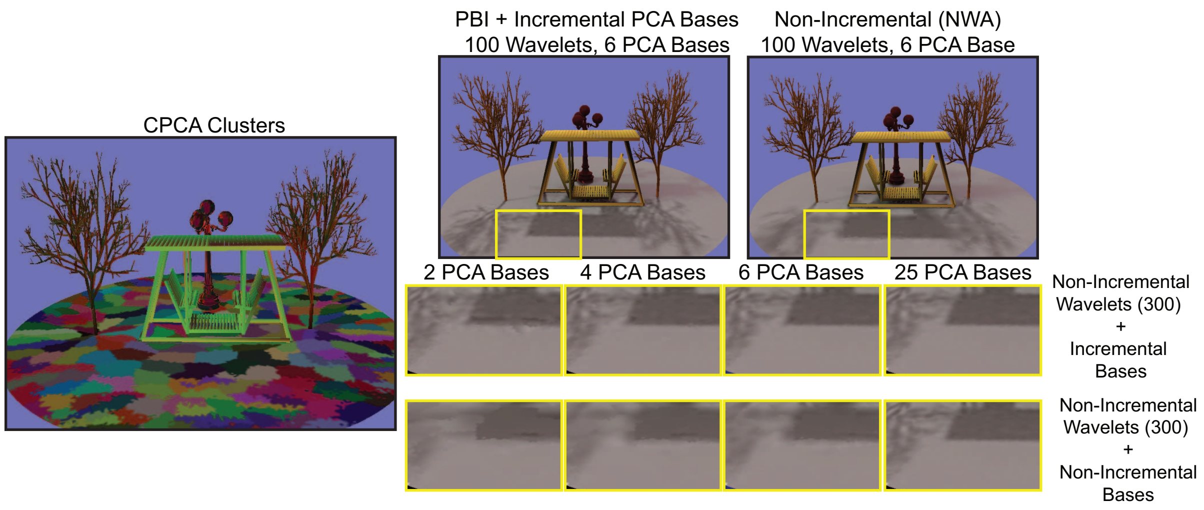
CPCA Clusters
PBI + Incremental PCA Bases
100 Wavelets, 6 PCA Bases
Non-Incremental (NWA)
100 Wavelets, 6 PCA Base
2 PCA Bases
4 PCA Bases
6 PCA Bases
25 PCA Bases
Non-Incremental Wavelets (300) + Incremental Bases
Non-Incremental Wavelets (300) + Non-Incremental Bases

Figure 7.3: CPCA, using our temporal coherence algorithm. *On left, we show the CPCA clusters color coded—we use several to accurately capture sharp shadows. On right, we compare our method with standard CPCA, clearly showing the higher quality in the images. The closeups below show the effect of changing the number of incremental terms in the second step of CPCA, and we see that $S/4 = 6$ is enough for very high quality (in the closeups, we always use high quality non-incremental updates for the lighting projection (first) step, so we can focus only on comparing incremental and non-incremental PCA bases).*

including CPCA, since we simply modify the lighting approximation L. However, even greater speedups can be obtained if we understand the CPCA method, and modify it to be fully incremental, as described below.

In the first rendering step, CPCA computes per-cluster coefficients,

$$P_i^c = M_i^c L, \tag{7.3}$$

where the superscript denotes the cluster number c, and the subscript denotes the PCA basis function i. M_i^c can be thought of as a $K \times N$ matrix, where N is the lighting resolution (in our case $6 \times 32 \times 32$). Each row of M_i^c corresponds to a specific term k in the BRDF factorization, and each element of the K element vector P_i^c is a dot product of this row in M_i^c and the lighting vector L.

In the second rendering step, the per-vertex weights are used to blend the coefficients P_i^c, with

$$U^v = \sum_{i=1}^{S} w_i^v P_i^{c(v)}, \tag{7.4}$$

where v is the vertex, $c(v)$ is its corresponding cluster, w_i^v is the weight for vertex v and basis function i, and we sum over all S PCA basis functions i. Note that U^v is a K element vector, with a separate value for each term of the BRDF. The final step weights by the BRDF factors h_k,

$$B^v(\omega_o) = \sum_{k=1}^{K} h_k(\omega_o) U_k^v. \tag{7.5}$$

Step 3, (equation 7.5) is usually very efficient, since K is small, and we compute it in the standard way. In [LSSS04], step 2, (equation 7.4) is expensive, since it is done for each vertex—but is usually much more efficient than standard PRT, since one needs to sum over only S basis functions. In their case, step 1 (equation 7.3) is relatively fast, especially with wavelet approximation of the lighting, since it needs to be done only once per cluster. However, in our experience, getting very sharp all-frequency shadows requires a large number of clusters, as well as more PCA basis functions than used by [LSSS04]. In this regime, steps 1 and 2 have comparable computational expense (as they do in the related technique of [NBB04]), and we would ideally like both steps to exploit temporal coherence.

Step 1 (equation 7.3) has essentially the same form as equations 1.1 and 4.1, and we can directly apply the PBI method to L, as for the previous algorithms. Step 2 (equation 7.4) is more interesting. For a given BRDF term k and cluster c, we can concatenate the weights w_i^v for all i into a large matrix W, whose rows correspond to vertices and columns to coefficients i. In that case, step 2 becomes

$$U = WP, \tag{7.6}$$

where P is an S-element vector of (dynamically-changing) coefficients for that cluster. We now have a very similar form to equation 1.1, with P as the dynamic vector V. Since there is no clear concept of bands, we apply the basic incremental algorithm of Chapter 4, which works well since S is usually small. We usually choose the number of incremental terms to be $S/4$, which gives us a four-fold improvement, while maintaining a high accuracy solution that avoids ghosting. In summary, we perform both steps of CPCA rendering incrementally, with PBI wavelets used for step 1 (lighting approximation and per-cluster coefficients), and basic incremental used for step 2 (per-vertex weights).

Figure 7.3 shows a complex scene, with 398 clusters and $S = 25$ PCA bases, which we render using 100 PBI wavelet lighting terms and 6 incremental basis functions per cluster. Another example is shown in the bottom of Figure 1.1. In both cases, our algorithm captures significantly sharper shadows than standard CPCA. The closeups in the bottom row of Figure 7.3 show how the quality improves as we increase the number of incremental terms in step 2 (equation 7.6). Clearly, $S/4 = 6$ terms suffices to give almost reference quality images in dynamic lighting. Hence, as with the earlier algorithms, we get a performance improvement by a factor of about four for both steps of CPCA in dynamic lighting, with rapid convergence in static lighting even if the view changes.

Chapter 8

PBI for Spherical Harmonics

There is nothing in equations 1.1, 4.1, and 5.1 restricting us to use wavelets or all-frequency methods. Indeed, we have conducted some preliminary experiments with spherical harmonic relighting that we discuss here briefly.

Low-frequency PRT involves a linear approximation of the lighting with a fixed basis, rather than a nonlinear approximation that picks the largest terms. As Ng et al. conclude in [NRH03], NWA converges exponentially faster than linear harmonic approximation as the number of terms increases making NWA a good choice for many applications. However, non-linear methods have some inherent flicker as they may choose disparate coefficients in consecutive frames. Although PBI can may sometimes reduce this flicker for NWA, any attempt to improve on spherical harmonics should preserve this primary benefit.

To resolve this, we modified the PBI method to first fill low-frequency bands to some tolerance before considering higher-frequency bands (guaranteeing we were never really worse in any band than standard low-frequency PRT). This creates a "stickiness" in each frequency band discouraging updates from jumping up and down between bands. A simple change to **SetupBands()** in Figure 6.1 is shown in Figure 8.1 that produces smoother appearance during rotations while somewhat hindering convergence. The tolerance variable (Tol in Figure 8.1) balances this trade-off between smoothness and more aggressive convergence.

We also have to be aware of how we compress the transport matrix. For wavelets, most transport coefficients are very close or equal to zero (usually more than 99%) permitting most of them to be simply dropped while still maintaining a near exact transport matrix. Spherical harmonics, on the other hand, distribute energy more evenly making it impossible to keep an exact representation of transport in memory. Instead, the maximum approximation order must be capped, usually at the 5^{th} or 6^{th} harmonic order making for transport matrices that are only 25 or 36 coefficients per row. Unfortunately, this means there is no hope of converging to an *exact* result. At best we can increment to a *better* approximation. Being restricted to a lower frequency approximation isn't all bad, however. Recall Figure 5.4, linking spatial frequency to temporal coherence. There is more exploitable temporal coherence at these lower frequencies, offseting the smoothness vs. convergence trade-off of the tolerance variable.

Figure 8.2 compares the modified PBI algorithm against linear 5^{th} and 10^{th} order spherical harmonic approximations. In this example, PBI increments only 25 coefficients per frame, and the transport matrix is large enough for a 20^{th} order approximation. We are able to achieve comparable, if not better, improvements as for wavelets, effectively a 25 term approximation (equivalent to 5^{th} order spherical harmonics) is comparable to 100 (10^{th} order) during rotation with minimal flickering. Unfortunately, the visual benefits are somewhat less dramatic than for wavelets. As depicted in Figure 8.3, the images produced by linear spherical harmonics change far less from 25 to 100 terms than for non-linear wavelets.

PBI for Spherical Harmonics

```
  Procedure SetupBands()            // Replaces SetupBands() in Figure 6.1.
1.  for all Bands $i$               // Initialize all bands to non-incremental.
2.    IsIncr$_i$ $= false$;
3.  end
4.  Tol $= 0.0005$;                 // Initialize Tolerance to a low value.
5.  $c = 1$;                        // Start at the first Spherical Harmonic coefficient,
6.  $fb = 1$;                       //     and the first frequency band.
// Dish out coefficients until budget is used up or total approximation error is negligible.
7.  while $c <$ Budget and $\sum_{i=1}^{NumBands}$ Err$_i > \epsilon$
8.    if Err$_{fb} >$ Tol           // Error in the band is above tolerance.
9.      Append($W^{fb}$, $c$);      // Append the next largest coefficient to $fb$'s update list.
10.     $c++$;
11.     IsIncr$_{fb}$ $= true$;     // Set this band to incremental.
12.   else                          // This band is acceptable,
13.     $fb++$;                     //     move on to next frequency band.
14.     if $fb >$ NumBands          // If all bands are acceptable,
15.       Tol $\times= 0.1$;        //     reduce the tolerance,
16.       $fb = 1$;                 //     and start back at the first band.
17.     end;
18.   end;
19. end;
```

Figure 8.1: SetupBands() procedure for Per-Band Incremental Spherical Harmonics.

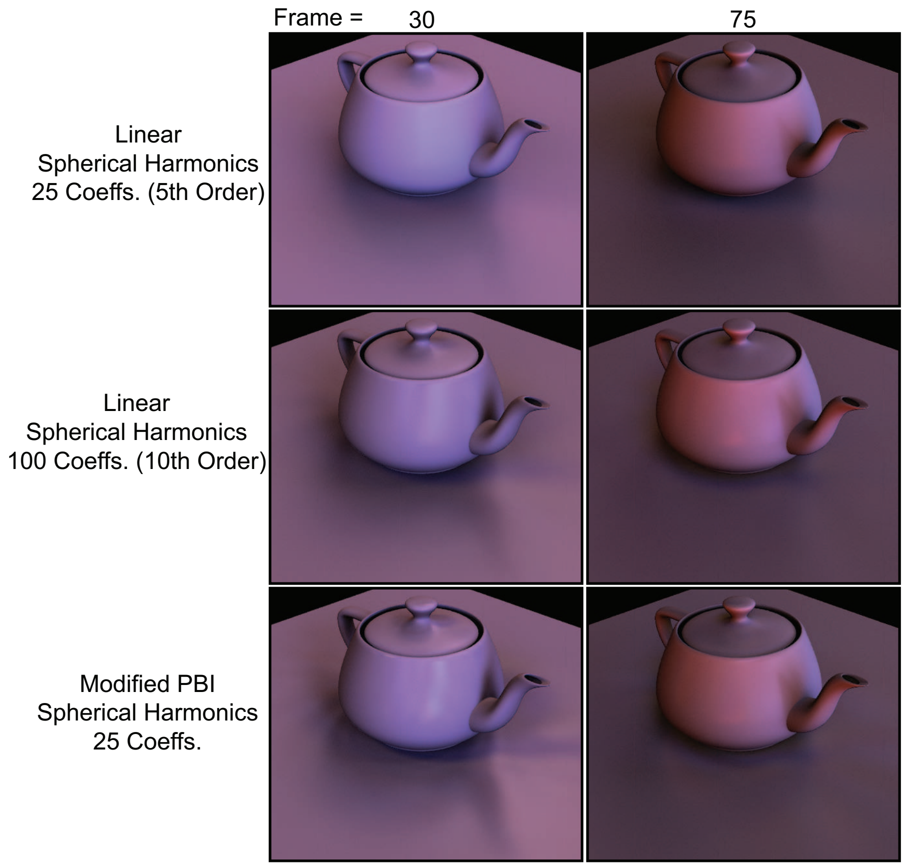

Figure 8.2: PBI for Spherical Harmonics using rotations from Figure 5.1, *comparing linear* 5^{th} *order (top), linear* 10^{th} *order (middle), and the modified PBI algorithm (bottom). In this example, we use a* 20^{th} *order spherical harmonic approximation of the transport matrix.*

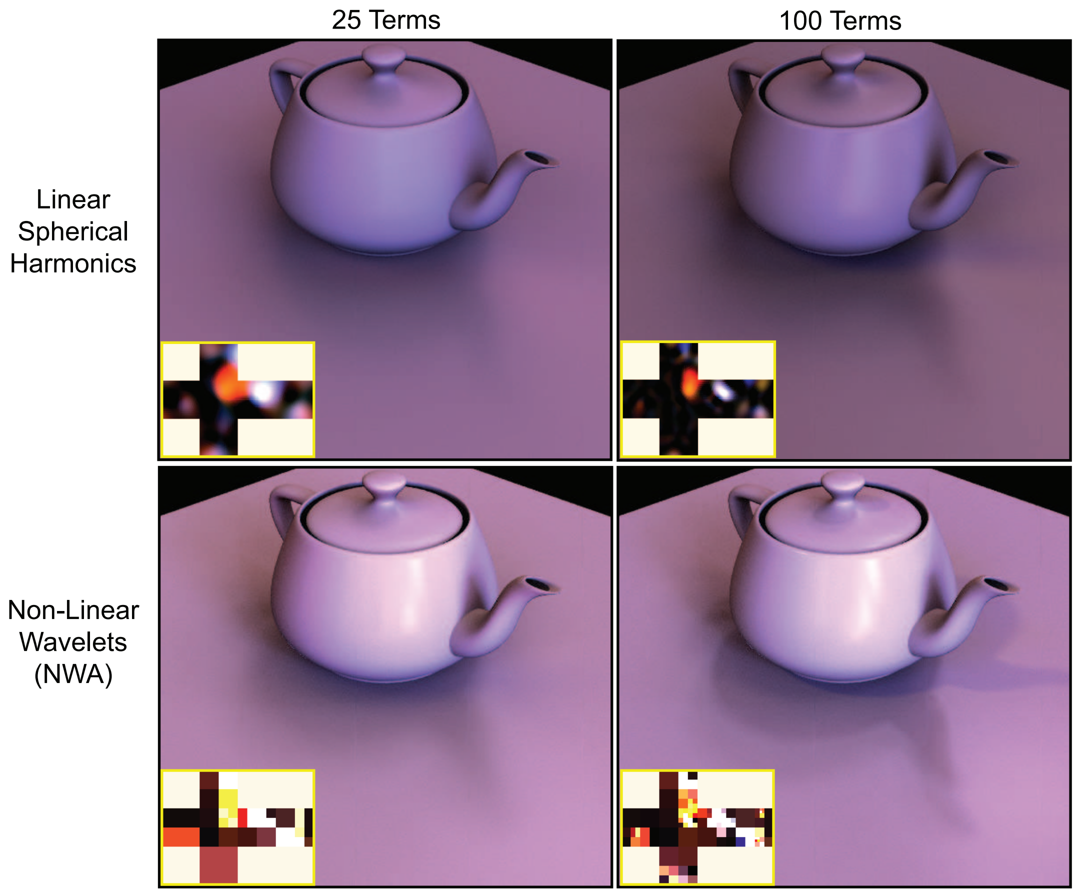

Figure 8.3: Comparing 25 to 100 term approximation using linear spherical harmonics and NWA. *NWA with only 25 terms gives a mostly low-frequency result with blurred shadows. With 100 terms, there are both soft and hard shadows. Meanwhile, linear spherical harmonics goes from very blurry shadows at 25 terms to slightly less blurry at 100 terms. Insets compare the environment approximation.*

Chapter 9

Conclusions and Future Work

We have identified a critical new source of coherence and compression in all-frequency PRT methods—temporal coherence in the lighting. We have analyzed the nature of temporal coherence, and developed an efficient per-band incremental wavelet algorithm. The method is very simple to implement and can be integrated with essentially all current real-time all-frequency PRT methods, while imposing minimal computational or memory overhead. For dynamically-varying lighting, we can obtain a performance improvement of $3\times$–$4\times$ without sacrificing quality. Equivalently, we can substantially increase the quality of all-frequency PRT methods, without sacrificing speed. Moreover, our algorithm converges to the exact result within a few frames of the lighting being static.

In the future, much further work can be done on understanding and analyzing the nature of temporal coherence. Algorithmically, one could imagine higher-order schemes for approximating the lighting coefficients, rather than the piecewise-constant approximation induced by selecting different wavelets at different times. However, this involves storing and updating image derivatives and Hessians, that can be quite complex and computationally expensive.

As discussed in Chapter 8, we have conducted only preliminary experiments with spherical harmonics. These experiments revealed compromises between transport size and approximation quality as well as between smoothness and convergence rate that don't exist in the non-linear wavelet domain. We are also hindered by our inability to converge to an exact result. More research is needed to determine how to integrate the incremental method into low-frequency approaches while benefitting more and costing less.

The results shown in figure 5.4 have deeper implications than those conveyed in this thesis. It seems intuitive that the link between temporal and angular frequencies and coherence should hold for any basis, not just wavelets. We also predict a similar relation between any forms of coherence, angular vs. spatial, spatial vs. temporal, etc.. This breakdown in coherence at higher frequencies may be a limiting factor to future applications. Understanding this limit may lead to enlightenment as to where and where *not* we can expext to benefit from coherence.

Modern GPUs are capable of speeding up many real-time rendering tasks by multiple orders of magnitude. As of yet we have only speculated on GPU implementations of the incremental methods discussed in this thesis. This is largely justified by the fact that there are no published all-frequency PRT implementations that we know of which claim faster results on the GPU than the CPU. We also predict GPU implementations of our methods to be complicated by the need for inter-frame and per-band knowledge, necessitating off-screen buffers and multiple passes. However, all current low-frequency implementations have corresponding GPU algorithms, often with blazingly fast results, and future GPU generations may open the door for all-frequency PRT. We would like to explore how temporal coherence might leverage GPU rendering algorithms.

Finally, we have exploited temporal coherence only in the lighting for static scenes. One could also exploit temporal coherence of the transport matrices for dynamic scenes, in applications like lighting design for pre-determined animated sequences. More generally, PRT is only one application, and temporal coherence should also be relevant to shadow mapping and other high-quality shading approaches. We predict that future PRT and other high-quality real-time rendering algorithms will be designed to take full advantage of temporal coherence in lighting, viewpoint and scene geometry.

Bibliography

[BFMZ94] G. Bishop, H. Fuchs, L. McMillan, and E. Zagier. Frameless rendering: Double buffering considered harmful. In *SIGGRAPH 94*, pages 175–176, 1994.

[BOR06] A. Ben-Artzi, R. Overbeck, and R. Ramamoorthi. Real-time BRDF editing in complex lighting. In *Accepted to SIGGRAPH 06*, 2006.

[CT82] R.L. Cook and K.E. Torrance. A reflectance model for computer graphics. *ACM Transaction on Graphics (TOG)*, 1(1):first article, 1982.

[DAG95] J. Dorsey, J. Arvo, and D. Greenberg. Interactive design of complex time dependent lighting. *IEEE Computer Graphics and Applications*, 15(2):26–36, March 1995.

[DWWL05] A. Dayal, C. Woolley, B. Watson, and D. Luebke. Adaptive frameless rendering. In *EuroGraphics Symposium on Rendering*, pages 265–275, 2005.

[LRR04] J. Lawrence, S. Rusinkiewicz, and R. Ramamoorthi. Efficient brdf importance sampling using a factored representation. *ACM Transaction On Graphics (TOG)*, 23(3):496–505, 2004.

[LSSS04] X. Liu, P. Sloan, H. Shum, and J. Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Eurographics Symposium on Rendering*, pages 337–344, 2004.

[MAA01] M. McCool, J. Ang, and A. Ahmad. Homomorphic factorization of BRDFs for high-performance rendering. In *SIGGRAPH 01*, pages 171–178, 2001.

[NBB04] S. Nayar, P. Belhumeur, and T. Boult. Lighting-sensitive displays. *ACM Transactions on Graphics*, 23(4):963–979, 2004.

[NRH03] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (SIGGRAPH 2003)*, 22(3):376–381, 2003.

[NRH04] R. Ng, R. Ramamoorthi, and P. Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 23(3):475–485, 2004.

[RH01] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH 01*, pages 497–500, 2001.

[SHHS03] P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics (SIGGRAPH 03 proceedings)*, 22(3):382–391, 2003.

[SKS02] P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM TOG (SIGGRAPH 02 proceedings)*, 21(3):527–536, 2002.

[SLS05] P. Sloan, B. Luna, and J. Snyder. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (SIGGRAPH 05 proceedings)*, 24(3):1216–1224, 2005.

[SS00] Y. Shi and H. Sun. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards.* CRC Press, 2000.

[WDP99] B. Walter, G. Drettakis, and S. Parker. Interactive rendering using the render cache. In *EGSR*, pages 19–30, 1999.

[WTL04] R. Wang, J. Tran, and D. Luebke. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Eurographics Symposium on Rendering*, pages 345–354, 2004.

[WTL05] R. Wang, J. Tran, and D. Luebke. All-frequency interactive relighting of translucent objects with single and multiple scattering. *ACM TOG (SIGGRAPH 05 proceedings)*, 24(3):1202–1207, 2005.

[WWL05] L. Wan, T. Wong, and C. Leung. Spherical Q2-tree for sampling dynamic environment sequences. In *EuroGraphics Symposium on Rendering*, pages 21–30, 2005.

[ZHL$^+$05] K. Zhou, Y. Hu, S. Lin, B. Guo, and H. Shum. Precomputed shadow fields for dynamic scenes. *ACM TOG (SIGGRAPH 2005)*, 25(3), 2005.