A Data-Driven Paradigm for Precomputed Radiance Transfer

LAURENT BELCOUR, Unity Technologies, France
THOMAS DELIOT, Unity Technologies, France
WILHEM BARBIER, ENSIMAG, France
CYRIL SOLER, Inria, France

Fig. 1. **Real-Time Direct to Indirect Transfer**. We tackle the problem of rendering in real-time effects such as indirect illumination in hair, surfaces, or volumes on key assets such as this Lizard mage. We build a data-driven formulation of Precomputed Radiance Transfer and construct a practical baseline algorithm using this paradigm. We further demonstrate its use in a commercial game engine.

In this work, we explore a change of paradigm to build *Precomputed Radiance Transfer* (PRT) methods in a data-driven way. This paradigm shift allows us to alleviate the difficulties of building traditional PRT methods such as defining a reconstruction basis, coding a dedicated path tracer to compute a transfer function, etc. Our objective is to pave the way for Machine Learned methods by providing a simple baseline algorithm. More specifically, we demonstrate real-time rendering of indirect illumination in hair and surfaces from a few measurements of direct lighting. We build our baseline from pairs of direct and indirect illumination renderings using only standard tools such as Singular Value Decomposition (SVD) to extract both the reconstruction basis and transfer function.

CCS Concepts: • **Computing methodologies** → **Rendering**.

Additional Key Words and Phrases: Real-time Rendering, Precomputed Radiance Transfer

ACM Reference Format:
Laurent Belcour, Thomas Deliot, Wilhem Barbier, and Cyril Soler. 2022. A Data-Driven Paradigm for Precomputed Radiance Transfer. *Proc. ACM Comput. Graph. Interact. Tech.* 5, 3, Article 26 (July 2022), 15 pages.
https://doi.org/10.1145/3543864

Authors' addresses: Laurent Belcour, Unity Technologies, France, laurent.belcour@gmail.com; Thomas Deliot, Unity Technologies, France, thomasd@unity3d.com; Wilhem Barbier, ENSIMAG, France, wilhem@wbrbr.org; Cyril Soler, Inria, France, cyril.soler@inria.fr.

2577-6193/2022/7-ART26 $15.00
https://doi.org/10.1145/3543864

1 INTRODUCTION

Motivation. Despite the recent introduction of hardware ray-tracing, full real-time global illumination (GI) remains too expensive to solve for modern video-game engines. As a result, many local GI effects such as subsurface scattering or indirect illumination in hair are still handled using alternative techniques that are specific to a particular effect [Golubev 2018; Tafuri 2019]. In this work, we are interested in generic solutions for real-time low-frequency GI that do not require ray tracing capabilities at runtime.

Precomputed Radiance Transfer (PRT) is the ideal candidate to evaluate indirect illumination. It is both efficient (runs in real-time) and generic (handles different kinds of transport). PRT methods [Sloan et al. 2002] store a light transport matrix at each vertex of the geometry in order to reproduce, at runtime, indirect illumination given the incident illumination. During rendering, this matrix translates the incident illumination into outgoing indirect illumination. While those methods solve light transport effectively, they come at a cost in their design. They require: a user defined (basis) representation for radiance; a tedious and dedicated precomputation stage; an advanced clustering method to cope with memory cost, *etc.* Therefore, we investigated whether a solution free of those constraints was achievable.

Data-Driven Approaches. It would be temping to alleviate those constraints using Machine Learning (ML) approaches. Indeed, using a dataset, an optimization process could construct a transfer function and an internal representation for direct to indirect transport. However, the trade-off between complexity and output quality would be hard to assess. To pave the way for such approaches, we need a simple baseline to assess the efficiency of ML methods when benchmarking them. Our aim is to provide such baseline.

A New Paradigm for PRT. In this paper, we show that a different paradigm for PRT liberates us from computing the transfer matrix as well as the radiance representation. Inspired by ML's formalism, we express PRT in a data-driven paradigm. While we retain the advantages of ML approaches, it allows us to construct a lean baseline method. Such baseline defines a reference point that more complex ML-based approaches can benchmark against.

Outline. We structured our contributions as follows:

- In Section 3, we introduce a data-driven formulation of direct-to-indirect transfer and show that classical PRT can be inferred solely from data.
- In Section 4, we introduce a simple algorithm: our proposed baseline. Specifically, we describe a meshless radiance transfer algorithm that takes only a few measurements of direct illumination to infer indirect illumination in texture space.
- In Section 5, we describe practical details to avoid temporal aliasing, support high dynamic of light and animation.
- In Section 6, we validate that our method runs in real-time, and support multiple transport scenario (surfaces, volumes, hair).

2 PREVIOUS WORK

In this section, we review some of the most relevant works for solving GI effects in real time. For a more thorough survey of real-time GI, we refer the reader to the survey of Ritschel et al. [2012]. Modern game engines usually implement various methods for computing and/or approximating global illumination under the performance constraints of real-time rendering.

Static Global Illumination. Lightmaps store the result of diffuse global illumination computed offline [Abrash 2000; Seyb et al. 2020; Silvennoinen and Sloan 2019]. While very efficient, this

approach imposes static lighting at runtime and cannot reproduce sophisticated optical effects such as subsurface scattering.

Path-Tracing provides a means to solve global illumination using Monte Carlo integration [Kajiya 1986]. However, small effects such as indirect illumination in hair or highly diffusing participating media can take little space on screen, but require significant computing efforts to converge. While path tracing can be accelerated using techniques such as using virtual point lights [Keller 1997], importance sampling [Bitterli et al. 2020], radiance caching [Müller et al. 2021] or denoising [Işık et al. 2021], those methods often target global variance reduction and would struggle with localized indirect light transport. More dedicated methods exist to approximate light transport, but often requires a fair amount of runtime power.

Precomputed Radiance Transfer. PRT [Sloan et al. 2003, 2002, 2005] computes per-vertex transfer operators that map incident radiance projected in a basis to outgoing radiance. Such methods can simulate both direct (soft shadows) and indirect effects (indirect illumination) on static objects from any distant light illumination settings. However, such generality comes at a cost. For example, one must choose a basis representation for incident lighting. Many have been tested such as Spherical Harmonics [Ramamoorthi and Hanrahan 2001; Sloan et al. 2002], Spherical Gaussians [Green et al. 2006; Tsai and Shih 2006], and Wavelets [Ng et al. 2003].

Geometric Constraints. Typically, PRT works by linearly interpolating per vertex outgoing radiance. This naturally makes its storage and computational cost proportional to the amount of geometric details. A hierarchical basis on meshes [Lehtinen et al. 2008] or in screen-space [Hašan et al. 2006] can avoid this problem. However, those bases must resolve all possible illuminations. It makes them over-generic when only a subset of lighting conditions are experienced at runtime. Furthermore, they require the explicit computation of the transfer matrix. In our work, both the basis and the transfer matrix are extracted from the dataset. Furthermore, our meshless formulation is compatible with both volumetric, subsurface, or complex geometries without a theoretical change.

PRT Optimized for Lighting Scenarios. Closest to our work are Modular Radiance Transfer (MRT) [Loos et al. 2011] and Reduced Aggregate Scattering Operators (RASO) [Blumer et al. 2016]. Both used a custom lighting scenario to obtain better basis representation for direct illumination. However, in a second step they estimate the indirect illumination basis and the transfer function using the classical approach. We show that such a two-steps approach is not necessary and embrace a fully data-driven approach.

3 DATA-DRIVEN RADIANCE TRANSFER

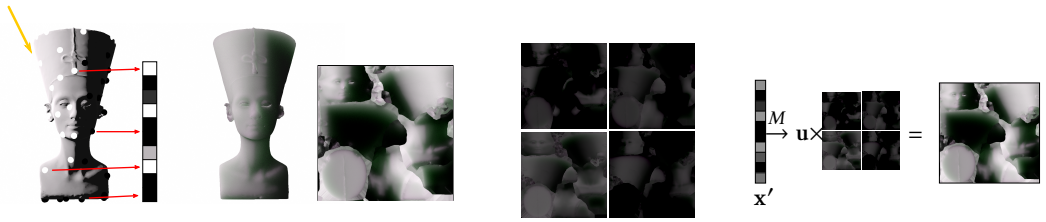
Given a dataset of N direct illuminations $\{\mathbf{D}_k\}_{0...N}$ (see Figure 2 (a)) along with the corresponding full indirect illuminations $\{\mathbf{I}_k\}_{0...N}$ (see Figure 2 (b)) we want to express the direct-to-indirect transfer:

$$\mathbf{I}_k = f(\mathbf{D}_k) \, \forall k \in [0, N]. \tag{1}$$

We do not yet explicit the nature of direct and indirect illumination data, but we treat them as vectors. These can be either volumetric, point-based measurements, textures, *etc.* In the following section, we describe how to build the necessary elements for Precomputed Radiance Transfer from this dataset.

3.1 Estimating the Transfer Matrix

We denote as D (resp. I) the matrix obtained by packing the direct (resp. indirect) illumination data $\mathbf{D}_k$ (resp. $\mathbf{I}_k$) as columns. Since light transport is a linear operator, there exists a linear mapping



x
a) direct illumination

y
b) indirect illumination

c) basis for indirect $\mathbf{U}_j$

d) runtime

Fig. 2. Outline of Data-Driven PRT. A lighting condition gives us a vector of direct illumination sampled around the object $\mathbf{x}$ (a), and a baked indirect illumination in texture space $\mathbf{y}$ (b). Randomizing the lighting condition, we extract a set of basis elements for the indirect illumination (c). From it, we compute a transfer matrix M that enables to compute coefficients $\mathbf{u} = M \times \mathbf{x}$ of indirect illumination (d).

between the direct and indirect illuminations in a scene. Thus, there is a light transport matrix M for which

$$I = MD. \tag{2}$$

Depending on how direct and indirect illuminations are encoded, matrices I and D can be very large matrices, and M is hence a potentially very large matrix. We assume that the dataset of direct lighting can compactly represent any lighting configuration. Hence, any given direct illumination vector $\mathbf{x}$ can be expressed as a linear combination of the columns of D with coefficients $\mathbf{c} \in \mathbb{R}^N$, we have

$$\mathbf{x} = D\mathbf{c}, \tag{3}$$

and by multiplying by the transfer matrix M, we obtain:

$$M\mathbf{x} = MD\mathbf{c} = I\mathbf{c}. \tag{4}$$

Hence, the indirect illumination caused by $\mathbf{x}$ can be computed as a linear combination of the columns of I. In other words, computing M is not necessary but it requires us to know the value of $\mathbf{c}$. We can avoid its direct calculation using Equation 3 and expressing $\mathbf{c}$ with respect to $\mathbf{x}$:

$$\mathbf{c} = (D^T D)^{-1} D^T \mathbf{x}. \tag{5}$$

Injecting this formula into Equation 4, we obtain the formula of the indirect illumination $\mathbf{y}$ w.r.t. direct illumination $\mathbf{x}$:

$$\mathbf{y} = M\mathbf{x} = I(D^T D)^{-1} D^T \mathbf{x}. \tag{6}$$

This expresses the light transport matrix as an ordinary least squares problem. It can also be seen as the transfer of the projection of $\mathbf{x}$ in the dataset of direct illumination: $\mathbf{x}^\perp = D^T \mathbf{x}$.

3.2 Indirect Illumination Basis Extraction

In Equation 6, the columns in I are used as a non-orthogonal basis to approximately represent the space spanned by M. This basis may have too many elements for practical use. However, similar to Loos et al. [2011] we found that the space of indirect illumination is effectively low dimensional (see Figure 3). We solve this problem using the singular value decomposition of I:

$$I = U\Sigma V^T. \tag{7}$$

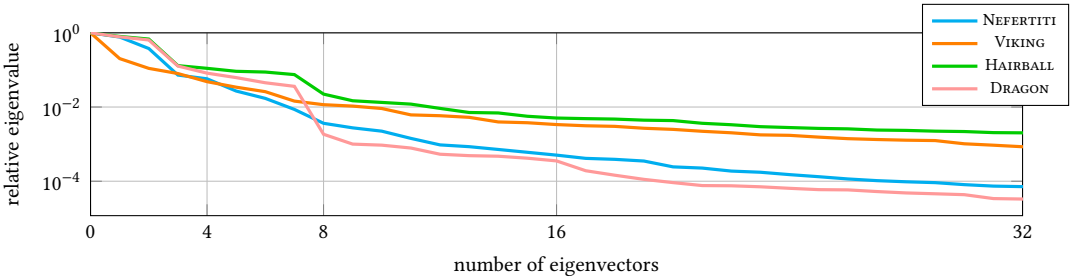


Fig. 3. Relative Eigenvalues of the indirect illumination. We display the relative eigenvalues of the aucorrelation matrix of the dataset of indirect illumination vectors for the different test scenes. We found out that they drop quickly below 1% and only a few dimensions are relevant for reconstruction.

In this decomposition, the columns of U form an orthogonal basis (see Figure 2 (c)), and the diagonal elements in Σ measure the importance of each of these vectors in representing this space. We approximate I by keeping only the n most important diagonal elements of Σ, and write our approximation of Equation 7 as

$$I \approx U_n \, \Sigma_n \, V_n^T = U_n \, C_n, \tag{8}$$

with C_n the matrix of coefficients. Plugging in Equation 6, we get

$$\mathbf{y} \simeq U_n C_n (D^T D)^{-1} D^T \mathbf{x} \tag{9}$$

$$\simeq U_n M_n \mathbf{x}, \tag{10}$$

with $M_n = C_n (D^T D)^{-1} D^T$.

The above equation computes the coefficients $\mathbf{u} = M_n \mathbf{x}$ of indirect illumination using a set of orthogonal basis vectors (columns of U_n) that are tailored to the lighting conditions seen in the matrix D and I. So far we have reduced the output of the transport matrix using the SVD. To make our method fully practical, we need to ensure that the dimension of the direct vectors is constrained.

3.3 Direct Vectors

Traditional PRT methods do not compute the transfer matrix between direct and indirect illumination. They rather store the transfer between incident illumination and indirect illumination. Incident illumination is expressed using a finite number of Spherical Harmonics coefficients $\mathbf{w}$. In such a case, we can write the transport operator that maps SH coefficients into direct illumination:

$$\mathbf{x} = T\mathbf{w}. \tag{11}$$

Thus, we can write

$$\mathbf{y} = MT\mathbf{w}. \tag{12}$$

As long as T defines a bijection between $\mathbf{x}$ and $\mathbf{w}$ (that is, it is invertible), the transfer matrices M and $M \times T$ are equivalent. Thus, we have described the same transfer function as Precomputed Radiance Transfer for indirect illumination. It follows that the placement of direct illumination is critical in both the conditioning of the transfer matrix, and the achievable equivalent incident illumination.

3.4 Discussion w.r.t. PRT

Meshless Transport. Our formalism for data-driven PRT is not tied to the geometry of the asset. While we could follow traditional PRT methods and compute transfer at vertices of the mesh, our formalism permits decoupling them. In practice, we advise for a meshless approach as reconstructed high frequencies are not linked to geometrical density.

Baking the Transport Matrix. PRT methods bake the transport matrix using implicit light sources defined by the illumination basis. Those light sources shade the asset with positive and negative radiance values. Hence, a dedicated light transport algorithm is used for them. In contrast, our formulation does not need to evaluate such lights and can take advantage of any advances in rendering algorithms (importance sampling, guiding, ...).

Lighting Agnostic Basis. While traditional PRT methods approximate the light transport matrix M in a basis made of data-agnostic elements (e.g. basis functions such as wavelets, polynomials, etc.), our method designs its own basis elements (the columns of U_n) to represent the output of M without explicitly encoding M. In essence, this is similar to random low-rank matrix approximation [Halko et al. 2011]: we build a space representing the output of M from *random* inputs to the light transport operator. This data-dependent approach also brings us a major advantage over light-agnostic PRT: our representation of the illumination is not limited in frequency in regions where the indirect illumination changes abruptly and does not waste discretization elements in regions where the illumination is always smooth.

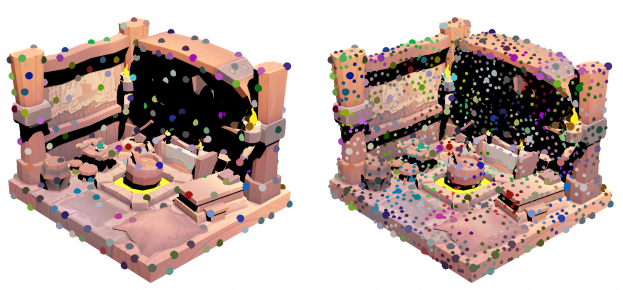
4 A LEAN BASELINE

In this section, we describe the baseline method we designed. First, we detail our baking process. It follows two steps: first the database creation (Section 4.1), then the basis and transfer matrix extraction (Section 4.2). Then, we detail the runtime component (Section 4.3).

4.1 Database Creation

In this first stage, we sample many random lighting configurations and render both direct and indirect illumination to build matrices D and I. This step setups lights positions, orientations and intensities around the asset for each configuration. Light interaction with the asset is evaluated using a path tracer for indirect transport and the runtime renderer for direct illumination. We evaluate indirect illumination in textures using the UV-space of the asset, thus each row of I is a lightmap. We evaluate direct illumination at discrete points we call *measurement points*. Each measurement point consists of a position, normal, and material information (for example, albedo). Depending on the type of geometry considered, we place them either on the surface of the object (for subsurface scattering) or on the geometry hull (for hair). Other patterns could be used as well, but we did not experiment with them. We store the measurement points for later use during runtime (see Section 4.3).

Measurement Points Distribution. We distribute measurement points evenly on the target surface (either the object or its convex hull). This distribution better conditions the Gram matrix DD^T. Measurement points too close to each other could produce the same lighting contribution and prevent us from inverting this matrix. To produce a blue-noise distribution of measurement points we use sample elimination [Yuksel 2015]. This method selects a subset of a denser set of uniformly distributed random points to produce an even distribution as shown in Figure 4 (a).



a) Measurement points b) 8× sampling

Fig. 4. Measurement points. We measure direct illumination at punctual positions in the scene. To better condition the incident irradiance vector, we distribute the measurement points as a blue noise on the target shape (a). To reduce flickering during runtime, we average the contribution of neighboring points (b). We show a super-sampling of 8× with averaging color coded.

4.2 Basis and Transfer Matrix Extraction

Once the database is computed, we first extract the basis U_n using Singular Value Decomposition (SVD) and then compute the direct to indirect transfer matrix M_n.

Indirect Illumination Basis. In our experiments, we always had more pixels than examples in matrix I. Thus, we extract U from the covariance $I^T \times I$, a $N \times N$ matrix [Turk and Pentland 1991]. To perform the SVD, we always subtract the mean of the indirect illumination data as this is common practice. Since the basis works with mean subtracted images we need to add it back afterwards, at runtime. So we store the mean along with the basis. Hence, when we report using 16 basis elements, we effectively use the mean and 15 basis elements. To work with any light intensity, we always normalize the indirect vector by the norm of the direct vector. At runtime, we scale the mean with the norm of the direct vector.

Transfer Matrix. Once the basis is computed, we use Equation 10 to compute M_n. Despite the measurement points distribution, the inverse covariance matrix of the direct illumination $\left[DD^T\right]^{-1}$ can be ill-posed. To overcome this, we always use the Moore-Penrose pseudo inverse [Penrose 1955] to perform it:

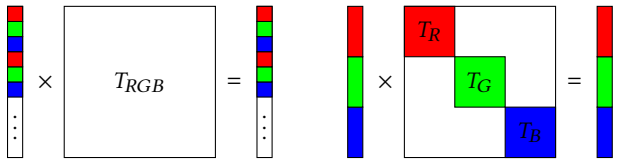
$$M_n \simeq C \left[DD^T\right]^+ D^T \tag{13}$$

4.3 Runtime Rendering

Our runtime algorithm works solely on GPU: first, we render the direct illumination vector; then, in a compute shader, we evaluate the coefficients $\mathbf{u}$ for the indirect illumination; finally, in the fragment shader, we blend the basis elements with these coefficients.

Fake G-Buffer Rendering. We use the set of measurement points as a G-Buffer that we shade using Deferred Shading [Saito and Takahashi 1990]. This works with any lighting type (point lights, area lights, environments lights, ...) and easily integrates into modern rendering engines.

Deringing. Because we reconstruct indirect illumination with a few basis elements, ringing artifacts (oscillation with negative values) may occur. Since our formulation expresses the transfer matrix as an ordinary least-squares problem, we add a Tikhonov regularization (or ridge regression)



a) multi-channel basis b) mono-channel basis

Fig. 5. **Multi vs Mono channel basis.** In our framework, we can either encode in the basis the color channels or separate them. In the latter case, this results in applying three different transfer matrices to the incident illumination by separating the color channels.

term to Equation 6:

$$M_n \simeq C \left[DD^T + \Lambda \right]^{-1} D^T \tag{14}$$

where Λ is a diagonal matrix. We usually use $\Lambda = \lambda I$, where I is the identity matrix and λ is a small scalar.

5 PRACTICAL DETAILS

5.1 Direct Illumination Downscaling

The length of the direct illumination vector defines a trade-off between performance and quality. Using only a few measurement points enables us to efficiently evaluate the direct illumination using Deferred Shading and perform the matrix-vector product $M_n \mathbf{x}$. However, too few measurement points will result in aliasing in both the direct illumination evaluation and the indirect reconstruction. We choose to counter aliasing by increasing the number of measurement points when doing shading (see Figure 4 (b)), but downscaling the vector of direct illumination afterwards. We do so by averaging neighboring direct samples when the measurement points are geometrically close (similar to mipmapping). We opted for a compute-shader pass that downsamples an arbitrary number of measurement points per pixel for this task.

5.2 Mono or Multi-Channel Basis

So far, we assumed that the direct and indirect illumination, $\mathbf{D}_k$ and $\mathbf{I}_k$, are vectors. They do include color information using an interleaved pattern: $\mathbf{I}_k = \left[\mathbf{I}^R_{k,0}, \mathbf{I}^G_{k,0}, \mathbf{I}^B_{k,0}, \dots \right]$. Hence, the basis decomposition of the indirect lighting data will incorporate color reconstruction and by construction the transfer matrix as well. This can be detrimental when trying to change the color of the light sources with colors not seen in the dataset. In practice, we treat color channels as separate vectors to build a unique reconstruction basis for all color channels. However, using such splitting, we have to perform a transfer matrix multiplication per color channel (see Figure 5).

GPU Storage and Evaluation. Since each basis element is used to reconstruct the three color channels, we pack them into RGBA textures to store four basis elements per texture. At runtime, we upload the reconstruction coefficient as 4×3 matrices on the GPU and perform the matrix/vector multiplication in the fragment shader to get the output color.

5.3 Working in Gamma Space

We found that for appearances with high contrast -such as subsurface scattering where details of diffusion are also to be found in low values- the reconstruction of indirect illumination was introducing high frequency contrast. To correct those artifacts, we perform the direct to indirect

transfer in Gamma space. During precomputation, we Gamma-correct the values in the dataset and evaluate the basis and transfer matrix. At runtime, we Gamma-correct the incident irradiance values before applying the transfer matrix. After reconstructing the indirect illumination, we invert the Gamma correction. We show in Figure 6 how working in Gamma space leads to visually more acceptable results.

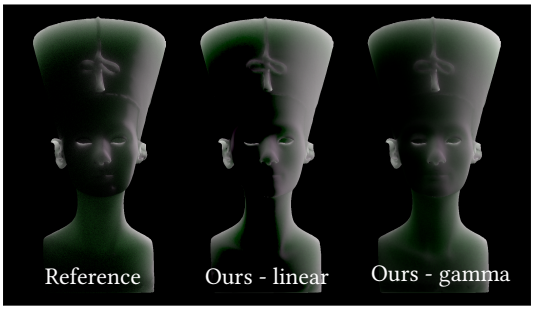


Fig. 6. **Working in Gamma space.** We perform transport and reconstruction in Gamma space to overcome the inability of the basis decomposition to cope with high contrast such as this side light. We found that this resolves hard discontinuity as shown in the middle rendering.

5.4 Working with Animated Assets

When working with animated assets such as rigged meshes, we construct a database of random animation keyframes with random lighting. With this database, we choose to generate a single basis that works across keyframes. Then, we can either construct a transfer matrix per keyframe or a single transfer matrix for the whole animation. We found that for subsurface scattering, this later approach worked well in practice (see our supplemental video).

6 RESULTS

We prototyped our method in the Unity game engine using the High Definition Render Pipeline [Lagarde et al. 2018]. There, we used the Deferred Shading for all our results. We validated our method for different light transport scenarios: indirect illumination on surfaces (Section 6.1), translucent materials (Section 6.2), and multiple scattering in hair (Section 6.3). We also benchmarked the performance and quality of our method with a varying number of basis components (Section 6.5).

6.1 Indirect Illumination on Surfaces

With the Cornell Box scene in Figure 7 we showcase the use of our method to render Global Illumination in a static scene. Because our basis provides a global support for the indirect illumination, we found that it is not a good fit for large scene GI as artifacts are likely to appear. Precomputed Radiance Transfer here would tessellate the scene to perform local reconstruction of indirect lighting resulting in fewer artifacts but requiring more computational power.

With the Viking scene in Figure 8, we show how our method is tailored to a specific lighting scenario. In this figure, we change the spot lighting used to build the dataset to point lights close to the surface of the mesh. While the resulting indirect illumination looks coherent when moving the point lights, the reconstructed result departs from the ground truth. This validates that our basis decomposition is optimized for specific lighting conditions.

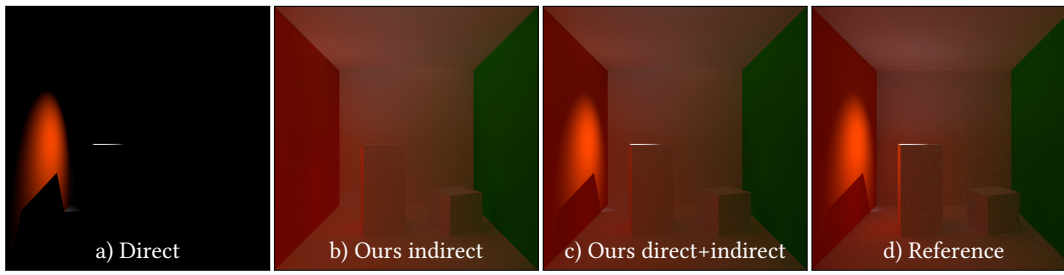
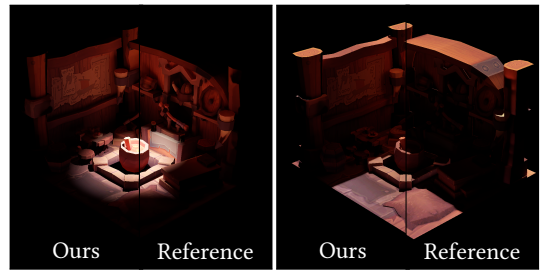


Fig. 7. **Reconstructing a frame of the Cornell Box scene.** We render the direct illumination (a) using traditional techniques (in our case a Deferred Renderer). Using a custom Deferred pass on the fake G-buffer, we estimate the vector of direct illumination and reconstruct the indirect lighting by weighting the bases together with the estimated weights (b). Summing both direct and indirect together provides the final output (c). We compare it to a reference for the same lighting condition (d). In this setup, we used 128 RGB values to estimate direct illumination, 16 basis elements and a regularization of $\lambda = 0.01$.



Same light scenario Different light scenario

Fig. 8. **Changing light scenario.** Our reconstruction is tailored to the lighting conditions in the dataset. When we relight the scene with the same type of lighting -a spotlight- the result closely approximate the reference (left). However, when a different light condition is applied -a directional light- (right), the result, while being coherent, departs from the reference.

6.2 Subsurface Scattering

Our method supports diffusion in thick media enclosed in a geometry. Lizard, Nefertiti, and Dragon scenes (see Figure 1, 6 and 9 respectively) demonstrate its use in such context. Our method captures the self shadowing and the color saturation due to multiple scattering within the object. As shown in Figure 6, the rendering of subsurface scattering creates high contrasts that do not reconstruct well in linear space. By applying a Gamma correction during the direct to indirect transfer, we get visually closer results.

6.3 Scattering in Hair

We also use our method to render indirect illumination in hair as demonstrated in scenes Lizard (Figure 1) and Hairball (Figure 10). Unlike rendering on a mesh we evaluate direct illumination on the hull defined by hair fibers. The extraction of the indirect illumination basis follows the same process, however the texture-space parameterization of hair is different.

Texture Space for Hair. When using hair, we parameterize hair strands using length and strand index as 2D coordinates. We further pack this texture coordinate by using 10 pixels per hair strand

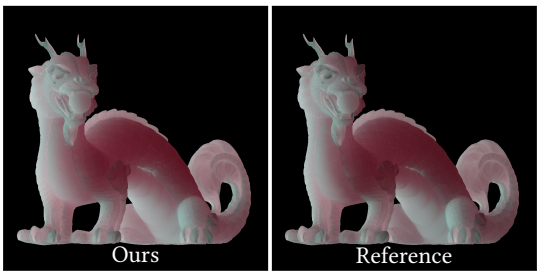


Fig. 9. **Rendering Subsurface scattering.** Our method enables to render translucency due to thick participating media in real-time. It handles light transmission well, something that screen-space methods struggle to cope with.

and concatenating multiple strands per row. Using this packing, we have 100k hair strands in a 1000×1000 texture (see Figure 11). The Lizard and the Hairball have respectively $10k$ and $140k$ hair strands. In our results, we reconstruct a radially constant illumination at each point on the fiber. While directional variation would increase the accuracy at the expense of additional storage, we found that it was not visually necessary on our assets.

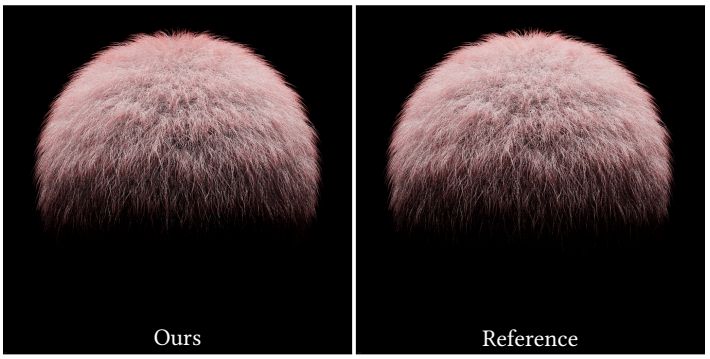


Fig. 10. **Hair rendering.** We render the interreflection of light in this Hairball asset containing 140k hair strands. The direct illumination is evaluated on the hull of the hair volume. We store the basis for the indirect lighting in textures using 10 texels per fiber (see Figure 11).

6.4 Comparison with Classical PRT

We compare our method with classical PRT [Sloan et al. 2002] to render indirect illumination from surfaces in the Viking scene lit by a directional light (in Figure 12). There, we display solely the indirect illumination to highlight some key differences with our work. First, because classical PRT restricts the frequency content of the incoming lighting, the resulting lighting condition will differ from the reference. Our method does not restrict the frequency content of incoming light but rather the space of possible indirect illumination. Hence, we can better reproduce such lighting scenario. Furthermore, classical PRT is performed on the vertices of the asset. This can cause interpolation and occlusion artifacts when the asset is not designed with PRT's contraints in mind. Furthermore, it links performance to the vertex count. Since we rely on a meshless approach, we are free of such issues.

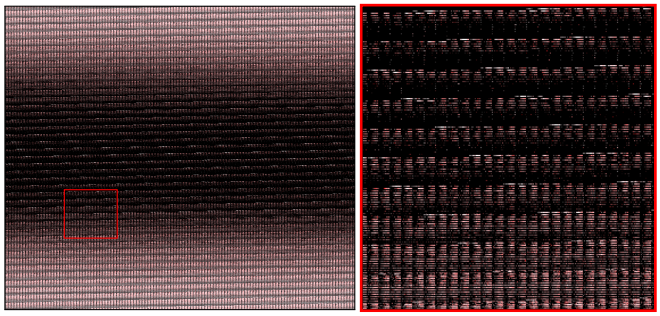


Fig. 11. **Hair parameterization.** We parameterize indirect illumination in hair using hair index and arc length that we pack using 10 texels per strands (see inset on the right). In this figure, we display one element of the dataset of the Hairball asset. For each texel, we average the outgoing radiance along the hair's azimuth as well as along the arc length inside the texel.

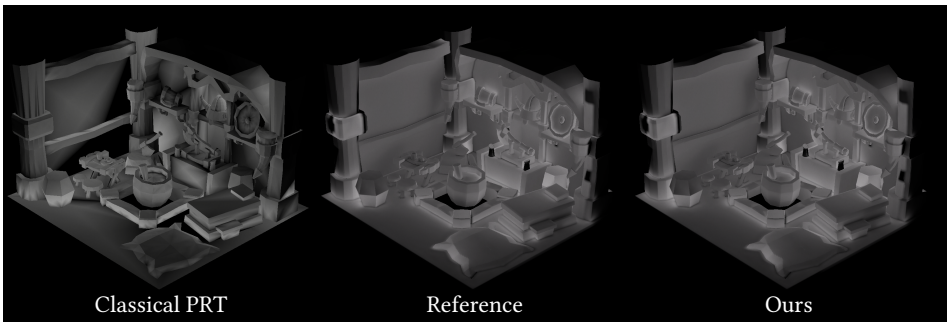


Fig. 12. **Comparing to Classical PRT.** We compare our method to a classical PRT method [Sloan et al. 2002]. While classical PRT both blurs the incident illumination and is linked to the tessellation of the asset, our method is free of those issues.

6.5 Performance, Storage, & Quality

We captured the performance of our method on an Nvidia RTX 2080 and compared it to Unity's standard shader with no indirect illumination as a baseline. We report the timings when using 4, 16, 64 basis elements (respectively 1, 4, and 16 RGBA textures) as well as baking times for a dataset of 128 elements in Table 1. The rendering time is decomposed into: shader evaluation; fake G-buffer computation; and matrix transfer to compute the indirect illumination coefficients. We found that this later part is independent of the number of basis elements in the 3 tested configurations.

The storage footprint for a 16 monochannel basis at 1024×1024 resolution with 32 bits per channel is 64MB on GPU (4 RGBA floating point textures) and the associated transfer matrix for 256 measurement points weighs 36KB.

Quality naturally increases as we grow the number of basis elements (see Figure 13). Furthermore, we found that using only a few direct illumination samples as input produces a reconstruction error close to projecting the complete indirect illumination in the basis. We report the RMSE of both approaches in Figure 14. For this particular test, we set the regularization to $\lambda = 10^{-3}$.

Table 1. **Performance.** We compare our method to a Standard shader with no indirect illumination. For our baseline, we report timings of the final fragment shader, as well as, fake G-buffer rendering (GPU), coefficients evaluation (GPU), and baking time (CPU).

	Cornell	Viking	Dragon	Hairball
No GI	221 μs	232 μs	418 μs	8729 μs
Ours (4 basis)	325 μs	348 μs	490 μs	8910 μs
Ours (16 basis)	428 μs	483 μs	658 μs	9547 μs
Ours (64 basis)	892 μs	1068 μs	1358 μs	11971 μs
Fake G-buffer	36 μs	34 μs	32 μs	34 μs
Matrix transfer	15 μs	15 μs	15 μs	15 μs
Bake time	6.48 min	6.89 min	13.37 min	49.07 min

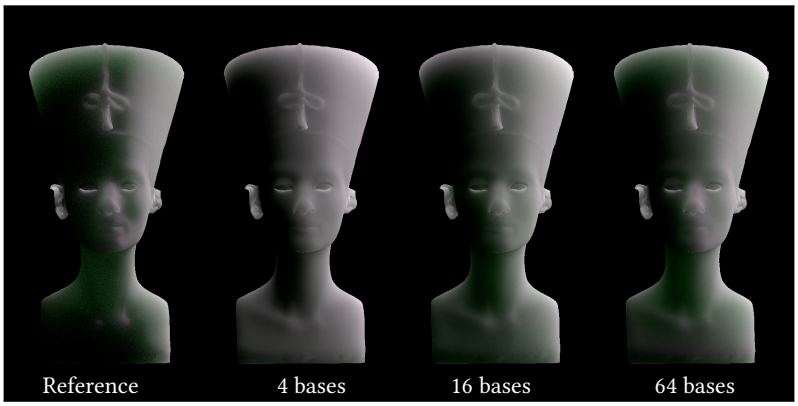


Fig. 13. **Increasing the number of basis.** The quality of the reconstruction increases as we add more basis elements. We found that for most of our assets, using 64 basis elements (hence, 16 textures) would provide accurate results. However, using 16 basis elements (4 texture) already gives plausible results.

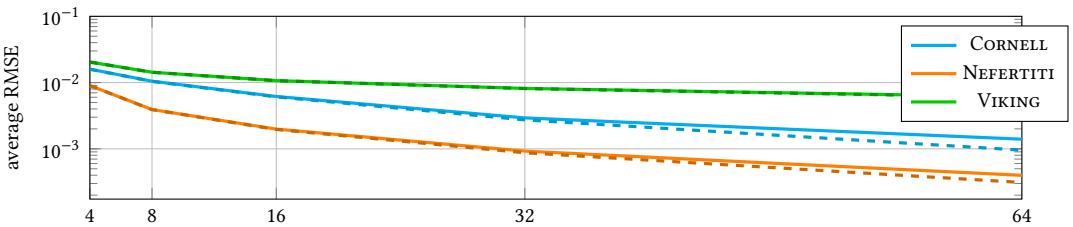


Fig. 14. **Quality.** Our method (plain) produces a decreasing RMSE with respect to the reference when increasing the number of basis (abscissa), close to the RMSE of projecting the reference in the basis (dashed).

7 LIMITATIONS & FUTURE WORK

Sparse Illumination Measurement. As shown in Section 3.3, the sampling of the measurement points is linked to the dimensionality of the space of reproducible lighting. Thus, it needs to be sufficiently dense to reproduce the space of observable lighting configurations in the dataset. It

follows that a lighting scenario mixing many light types should require a denser sampling than a single light scenario.

No Directionality. We reconstruct a diffuse appearance when reconstructing indirect illumination. However, since our method does not depend on the encoding of the measured indirect illumination, it can be extended to reconstruct glossy appearances e.g. directional distributions using directional sampling or any basis such as Spherical Harmonics. However, our method is likely to be restricted to low frequency gloss here and will not work to render specular reflections.

Large Assets. Our solution is not designed to handle assets such as levels in a game. Because we handle light transport globally and reduce it with a handful of basis functions, we cannot reconstruct the interconnected interiors or large environments in which the combinatorics of possible illumination is large. For such case, our method would require to be extended to handle modular transfer between disjoint transport solutions (Similar to Loos et al. [2011]).

8 CONCLUSION

We have presented a new data-driven paradigm for Precomputed Radiance Transfer methods. We have shown that it could lead to a lean and efficient method that uses images of direct and indirect lighting obtained from renderers such as path tracers to compute a transfer matrix and a reconstruction basis that are tailored to render the indirect lighting of a lighting scenario in real-time with a small footprint. This method applies to the rendering of indirect illumination from surfaces, subsurface scattering, or hair rendering and could serve as a baseline for future work in Machine Learning.

REFERENCES

Michael Abrash. 2000. Quake's lighting model: Surface caching. *Graphic programming black book* (2000).

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 148–1.

Adrian Blumer, Jan Novák, Ralf Habel, Derek Nowrouzezahrai, and Wojciech Jarosz. 2016. Reduced aggregate scattering operators for path tracing. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 461–473.

Evgenii Golubev. 2018. Efficient screen-space subsurface scattering using Burleys normalized diffusion in real-time. In *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games Course*.

Paul Green, Jan Kautz, Wojciech Matusik, and Frédo Durand. 2006. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. 7–14.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

Miloš Hašan, Fabio Pellacini, and Kavita Bala. 2006. Direct-to-indirect transfer for cinematic relighting. *ACM transactions on graphics (TOG)* 25, 3 (2006), 1089–1097.

Mustafa Işık, Krishna Mullia, Matthew Fisher, Jonathan Eisenmann, and Michaël Gharbi. 2021. Interactive Monte Carlo denoising using affinity of neural features. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–13.

James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.

Alexander Keller. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 49–56.

Sebastien Lagarde et al. 2018. Unity High Definition Render Pipeline. https://unity.com/srp/High-Definition-Render-Pipeline

Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François X Sillion, and Timo Aila. 2008. A meshless hierarchical representation for light transport. In *ACM SIGGRAPH 2008 papers*. 1–9.

Bradford J Loos, Lakulish Antani, Kenny Mitchell, Derek Nowrouzezahrai, Wojciech Jarosz, and Peter-Pike Sloan. 2011. Modular radiance transfer. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–10.

Thomas Müller, Fabrice Rousselle, Jan Novák, and Alexander Keller. 2021. Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021).

Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. 2003. All-frequency shadows using non-linear wavelet lighting approximation. In *ACM SIGGRAPH 2003*. 376–381.

Roger Penrose. 1955. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, Vol. 51. Cambridge University Press, 406–413.

Ravi Ramamoorthi and Pat Hanrahan. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 497–500.

Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The state of the art in interactive global illumination. In *Computer graphics forum*, Vol. 31. Wiley Online Library, 160–188.

Takafumi Saito and Tokiichiro Takahashi. 1990. Comprehensible rendering of 3-D shapes. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. 197–206.

Dario Seyb, Peter-Pike Sloan, Ari Silvennoinen, Michał Iwanicki, and Wojciech Jarosz. 2020. The design and evolution of the UberBake light baking system. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020). https://doi.org/10/gg8xc9

Ari Silvennoinen and Peter-Pike Sloan. 2019. Ray Guiding for Production Lightmap Baking. In *SIGGRAPH Asia 2019 Technical Briefs* (Brisbane, QLD, Australia) *(SA '19)*. Association for Computing Machinery, New York, NY, USA, 91–94. https://doi.org/10.1145/3355088.3365167

Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. 2003. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 382–391.

Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH 2002*. 527–536.

Peter-Pike Sloan, Ben Luna, and John Snyder. 2005. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics (TOG)* 24, 3 (2005), 1216–1224.

Sebastian Tafuri. 2019. Strand-based Hair Rendering in Frostbite. In *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games Course*.

Yu-Ting Tsai and Zen-Chung Shih. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on graphics (TOG)* 25, 3 (2006), 967–976.

Matthew Turk and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3, 1 (1991), 71–86.

Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2015)* 34, 2 (2015), 25–32.