

Neural Dynamic GI: Random-Access Neural Compression for Temporal Lightmaps in Dynamic Lighting Environments

Supplementary Material

1. Lightmap

At a high level, real-time rendering first identifies visible surface points (*e.g.* via rasterization) and then shades each fragment by combining emitted radiance with reflected radiance integrated over the incident hemisphere. Formally, the rendering equation [3] relates outgoing radiance to emission and reflection:

$$L_o(\mathbf{p}, \mathbf{v}) = L_e(\mathbf{p}, \mathbf{v}) + \int_{\Omega} f(\mathbf{l}, \mathbf{v}) L_i(\mathbf{p}, \mathbf{l}) (\mathbf{n} \cdot \mathbf{l})^+ d\mathbf{l}. \quad (1)$$

Here, $L_o(\mathbf{p}, \mathbf{v})$ is the outgoing radiance at point \mathbf{p} toward view \mathbf{v} . $L_e(\mathbf{p}, \mathbf{v})$ denotes self emission from \mathbf{p} toward \mathbf{v} . Ω is the visible hemisphere. $f(\mathbf{l}, \mathbf{v})$ is the bidirectional reflectance distribution function (BRDF), and $L_i(\mathbf{p}, \mathbf{l})$ is the incident radiance from direction \mathbf{l} . The reflected term is often decomposed as $f = f_{\text{diff}} + f_{\text{spec}}$, where the diffuse term is direction independent. This yields:

$$\begin{aligned} L_o^{\text{diff}}(\mathbf{p}) &= f_{\text{diff}} \int_{\Omega} L_i(\mathbf{p}, \mathbf{l}) (\mathbf{n} \cdot \mathbf{l})^+ d\mathbf{l}, \\ L_o^{\text{spec}}(\mathbf{p}, \mathbf{v}) &= \int_{\Omega} f_{\text{spec}}(\mathbf{l}, \mathbf{v}) L_i(\mathbf{p}, \mathbf{l}) (\mathbf{n} \cdot \mathbf{l})^+ d\mathbf{l}, \end{aligned} \quad (2)$$

and thus:

$$L_o(\mathbf{p}, \mathbf{v}) \approx L_e(\mathbf{p}, \mathbf{v}) + L_o^{\text{diff}}(\mathbf{p}) + L_o^{\text{spec}}(\mathbf{p}, \mathbf{v}). \quad (3)$$

Lightmap baking targets the view independent diffuse component. Concretely, we store the lighting information in lightmaps:

$$L_{\text{LM}}(\mathbf{p}) = \int_{\Omega} L_i(\mathbf{p}, \mathbf{l}) (\mathbf{n} \cdot \mathbf{l})^+ d\mathbf{l}. \quad (4)$$

A high-quality lightmap typically includes direct illumination as well as multi-bounce indirect illumination. At run-time, we fetch the baked diffuse term using fragment UVs and apply it during shading.

Generating lightmaps requires UVs. Only texels that map to visible surfaces are valid, so we use a binary mask to flag valid regions. Resolution sets the trade-off between detail and memory.

2. Block Compression

Block compression (BC) is a family of fixed-rate, lossy texture compression formats designed for real-time GPU decoding. The core idea originates from Block Truncation Coding (BTC) [1]: the image is partitioned into small

blocks (*e.g.* 4×4 texels), and the color values within each block are approximated by a compact set of representative colors together with per texel selection indices.

In the simplest and most representative case [4], each 4×4 block stores two color *endpoints* e_1, e_2 and a set of per texel interpolation weights $\{w_p\}$. At decode time, the color of each texel p is reconstructed by linearly interpolating between the endpoints:

$$c_p = (1 - w_p) e_1 + w_p e_2, \quad p = 1, \dots, 16. \quad (5)$$

More advanced variants (*e.g.* BC7) extend this scheme by supporting multiple partitions within a single block, each with its own pair of endpoints, thereby improving quality at the cost of a more complex encoding. This design enables constant-time random access to any texel, which is critical for GPU texture sampling. Different BC variants target different use cases.

Adaptive Scalable Texture Compression (ASTC) [5] generalizes this framework with flexible block sizes, which provides a continuous trade-off between quality and bitrate. ASTC supports LDR, HDR, and 3D textures and is widely adopted on modern mobile platforms.

A key advantage of block compression is that decoding is performed entirely in hardware during texture sampling, imposing small computational overhead on the shader pipeline. In NDGI, we leverage this property by applying BC7 to our learned feature maps \mathbf{F}_{uv}^{2D} and \mathbf{F}_{uv}^{3D} .

3. Virtual Texturing

Virtual texturing (VT) [2], also known as megatexture or sparse virtual texturing, is a streaming technique that decouples the logical texture space from the physical GPU memory. It enables applications to reference texture data far exceeding the available video memory, loading only the portions that are actually visible.

The key data structures of a VT system are:

- **Virtual texture.** The entire logical texture, which may be many gigabytes and is divided into fixed-size *tiles* (*e.g.* 128×128 or 256×256 texels). Tiles are the atomic unit of streaming.
- **Physical texture.** A GPU-resident texture atlas that holds only the currently needed tiles. Its capacity is bounded by available video memory.
- **Page table.** An indirection texture that maps each virtual tile to its location in the physical texture. During shading, the shader first samples the page table with the fragment's

UV coordinates to obtain a physical tile address, and then samples the physical texture to retrieve the actual texel data.

At runtime, the VT system operates in a feedback-driven loop. First, the GPU renders a low-resolution *feedback buffer* that records which virtual tiles are required for the current view. The CPU then reads back the feedback, identifies missing tiles, and transfers them from disk into the physical texture. Finally, the page table is updated to reflect the new mapping. Tiles that have not been referenced within a interval may be evicted to reclaim capacity for newly requested tiles.

This architecture benefits NDGI in two ways. First, only the tiles visible in the current frame need to be decoded, reducing runtime decompression cost. Second, once decoded, a tile remains cached in the physical texture across frames until the lighting state changes, avoiding repeated inference. In our pipeline, each lightmap is partitioned into fixed-size tiles with a dedicated NDGI model per tile. The VT system identifies required tiles each frame and invokes neural decompression via a compute shader. Decoded results are written into the physical texture and invalidated only when re-decoding is needed. To support hardware texture filtering, each tile is stored with a small border (*e.g.* 4 pixels per edge). We apply the same mirrored padding during training, ensuring seamless tile boundaries.

4. Dataset

4.1. Dataset Description

Our dataset comprises baked lightmap data across multiple scenes. For each scene, we provide two types of files: lightmap data files and mask files. The lightmap file stores 3-channel (RGB) lighting textures. We bake 24 sets per day at hourly intervals aligned to the top of the hour. For scenes that exhibit light-switching behavior, we additionally bake two extra sets around the on/off transition times to better capture fast lighting changes.

The mask is a single-channel image with the same spatial resolution as the lightmaps. Each texel indicates whether the corresponding lightmap texel is valid. In real-time rendering, only valid regions are sampled, and compression methods can leverage this information to further improve compression ratio.

We also provide a per scene configuration file that records the spatial resolution of each lightmap set and the correspondence among lightmaps, masks, and time indices. Unless otherwise noted, all lightmap values are treated as linear HDR intensities for evaluation and visualization.

4.2. Dataset Evaluation

We evaluate lightmap reconstruction quality using PSNR, SSIM [8], and LPIPS [9]. Metrics are computed in a tiled

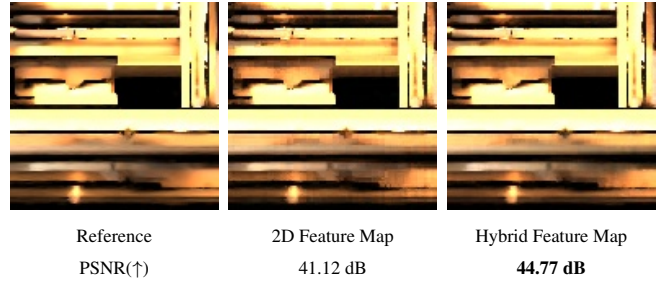


Figure 1. At comparable bitrates and with the same decoder, our hybrid feature maps outperform 2D-only feature maps, producing less noise.

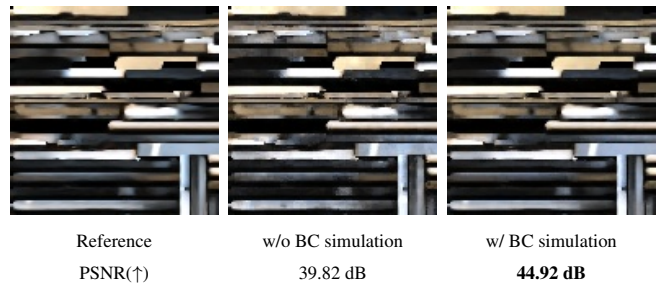


Figure 2. We compare variants with and without the BC simulation strategy and report PSNR. Parameterizing the feature map with BC endpoints and weights yields less noisy final reconstructions than directly optimizing feature parameters.

manner by partitioning each lightmap into non-overlapping 128×128 tiles. For each tile, we compute the metric within the valid region defined by the mask and report the result by averaging over all tiles across all lightmaps of that scene.

Because the data are HDR, we determine the dynamic range per tile: the minimum and maximum values within the tile’s valid region set the range (*e.g.* the peak value for PSNR), rather than adopting fixed global constants. Unless specified otherwise, metrics are computed in linear RGB. This tiled, mask-aware protocol yields stable and comparable scores across scenes with different UV packings and sparsity patterns.

Table 1. PSNR comparison for feature maps with different structures at comparable bitrates.

Feature Map Structure	BPP(↓)	PSNR(↑)
2D Feature Maps	0.73	42.1 dB
3D Feature Maps	0.69	36.6 dB
Hybrid Feature Maps	0.67	44.9 dB

5. Ablation Study

We validate the effectiveness of our hybrid feature representation. Figure 1 compares a baseline that uses only 2D feature maps with our hybrid features at comparable bitrates under the same decoder configuration. The hybrid representation captures lightmap content more faithfully, yielding smoother shading transitions and less aliasing, and it consistently improves reconstruction quality across view-points. Table 1 reports PSNR on a test scene across several feature map architectures at matched bitrates, showing that the hybrid representation attains higher accuracy with similar storage.

We also evaluate a BC simulation strategy and observe clear gains across test scenes and metrics. The results are shown in Figure 2 and Table 2. To ensure a fair comparison, both variants apply BC compression after training, while they differ in how the feature map is parameterized during optimization. In the first variant, we model the feature map with BC endpoints and weights during training, which better matches the target representation, encourages piecewise linear structure, and improves rate distortion behavior. In the second variant, we directly optimize the full feature map and compress it with BC at the end, which creates a mismatch with the BC quantization grid and introduces high frequency noise and block inconsistency. Empirically, the endpoints and weights modeling yields consistently higher fidelity and fewer artifacts, whereas direct optimization introduces substantial noise that degrades the final quality.

We further compare our tri-plane hybrid feature representation against a single 8-channel \mathbf{F}_{uv}^{2D} baseline on NDGI.M. As shown in Table 3, the tri-plane design achieves comparable or higher PSNR at a noticeably lower bitrate and smaller storage footprint. We attribute this to the additional \mathbf{F}_{ut}^{2D} and \mathbf{F}_{vt}^{2D} planes, which more effectively capture higher-frequency temporal variations that a single spatial feature map cannot represent as compactly.

6. Rendered Results

We provide additional qualitative comparisons of rendered results in the supplementary. Under matched bitrates, Figure 3 compares PRT [6], NTC L. [7] and our NDGI M. (Ours) against the reference. Our method delivers cleaner global illumination, fewer color shifts and less noise, and better preservation of fine details, which leads to

Table 2. PSNR comparison for modeling feature maps with endpoints and weights or not.

Feature Map Strategy	PSNR(↑)	SSIM(↑)	LPIPS(↓)
w/o BC Simulation	37.96 dB	0.975	0.013
w/ BC Simulation	44.20 dB	0.992	0.004

Table 3. Tri-plane vs. 8-channel \mathbf{F}_{uv}^{2D} on NDGI.M. Tri-plane yields smaller storage at higher quality.

Method	PSNR ↑	BPP ↓	Size (MB) ↓
8-channel \mathbf{F}_{uv}^{2D}	41.32	0.86	358
Tri-Plane	41.50	0.67	279

higher visual fidelity relative to the reference. Row annotations report BPP, PSNR, and SSIM, and our method consistently achieves a better trade-off between quality and bitrate across scenes.

References

- [1] E. Delp and O. Mitchell. Image compression using block truncation coding. *IEEE Transactions on Communications*, 27(9): 1335–1342, 1979. 1
- [2] Epic Games. Virtual texturing. <https://dev.epicgames.com/documentation/en-us/unreal-engine/virtual-texturing-in-unreal-engine>, 2025. Accessed: October 25, 2025. 1
- [3] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986. 1
- [4] Microsoft. Texture block compression in direct3d 11. <https://learn.microsoft.com/en-us/windows/win32/direct3d11/texture-block-compression-in-direct3d-11>, 2025. Accessed: October 25, 2025. 1
- [5] J. Nystad, A. Lassen, A. Pomianowski, S. Ellis, and T. Olson. Adaptive scalable texture compression. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, page 105–114, Goslar, DEU, 2012. Eurographics Association. 1
- [6] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 339–348. 2023. 3, 4
- [7] Karthik Vaidyanathan, Marco Salvi, Bartłomiej Wronski, Tomas Akenine-Moller, Pontus Ebelin, and Aaron Lefohn. Random-access neural compression of material textures. *ACM Transactions on Graphics*, 42(4):1–25, 2023. 3, 4
- [8] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 2
- [9] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 2












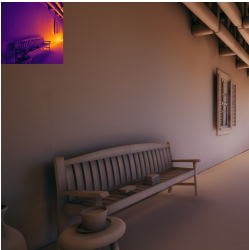





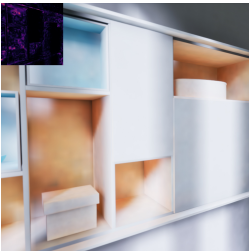
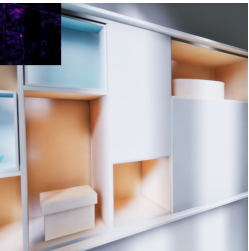






Lit Scene	PRT	NTC L.	NDGI M. (Ours)	Reference
				
BPP(↓), PSNR(↑), SSIM(↑)	0.92, 26.09 dB, 0.921	0.78, 45.56 dB, 0.998	0.67, 47.53 dB, 0.999	FarmLand
				
BPP(↓), PSNR(↑), SSIM(↑)	0.92, 24.39 dB, 0.971	0.78, 46.28 dB, 0.998	0.67, 47.55 dB, 0.999	FarmLand
				
BPP(↓), PSNR(↑), SSIM(↑)	1.00, 17.45 dB, 0.719	0.78, 45.41 dB, 0.997	0.71, 47.42 dB, 0.998	Yard
				
BPP(↓), PSNR(↑), SSIM(↑)	1.00, 15.18 dB, 0.667	0.78, 38.50 dB, 0.998	0.71, 41.07 dB, 0.999	Room
				
BPP(↓), PSNR(↑), SSIM(↑)	1.00, 10.37 dB, 0.688	0.78, 39.97 dB, 0.999	0.71, 42.25 dB, 0.999	Room

Figure 3. Additional comparisons of rendering quality. Compared to PRT [6], our method better captures global illumination tone variations, such as multi-bounce interreflections. Compared to NTC [7], it reconstructs lightmaps with higher fidelity and noticeably less noise.