An Omnidirectional Rasterizer for Differentiable Irradiance Rendering from 360° HDR RGB-D Images

Qian Zhang
Brown University

Min H. Kim
KAIST

James Tompkin
Brown University

Abstract

We present a differentiable equirectangular rasterizer for 360° RGB-D images. Inverse rendering from a single image is ill-posed: reflectance, geometry, and illumination are entangled, so many combinations of physical properties can describe the same observation. Wider-field observations can reduce this ill-posedness. 360° high dynamic range RGB-D images observe the environment radiance field and proxy geometry, making irradiance computable under specific assumptions rather than requiring it to be inferred. Our method constructs a closed mesh from omnidirectional depth, handles seam and pole discontinuities during equirectangular rasterization, and computes physically-based irradiance from the observed radiance field. Rasterizing in equirectangular space captures the full sphere in a single pass per surface point, rather than the six passes required by cubemap alternatives. We validate the rasterizer against path-traced synthetic scenes and characterize the behavior of joint normal and reflectance optimization under known illumination. These results establish differentiable irradiance rendering as a possible forward model for omnidirectional image inverse rendering.

1. Introduction

Inverse rendering attempts to recover physical scene properties from images. One subproblem is intrinsic image decomposition, which separates an image into reflectance and shading [2]. From a single image, this is ill-posed: many combinations of reflectance, surface geometry, and illumination can explain the same observation. To make progress, existing methods usually rely on strong priors, whether hand-designed [1, 12] or learned from data [3, 4, 10, 16, 30]. These priors may be embedded implicitly in network architectures, training data distributions, or optimization regularizers, making it difficult to separate what is constrained by the forward image formation model from what is imposed by prior assumptions.

Wider-field observations can reduce this ill-posedness. 360° high-dynamic-range (HDR) images capture the environment radiance field as physically meaningful values [5], while multi-camera 360° systems can provide scene depth [21, 27]. In this setting, the camera observes the outgoing radiance from all visible surfaces. With depth providing a proxy geometry, this observed radiance field can be used to approximate irradiance at each surface point, shifting the remaining ill-posedness toward geometry and reflectance. This opens the door to methods that compute irradiance from observed radiance rather than relying on learned shading models. The accuracy of the inverse result in such settings depends directly on how faithfully the differentiable forward model computes irradiance from the radiance field and proxy geometry. For example, Li et al. [13] use point-cloud reprojection with nearest-neighbor interpolation. But, without explicit surface connectivity, this misses occlusions and introduces interpolation artifacts, and their irradiance computation is not differentiable.

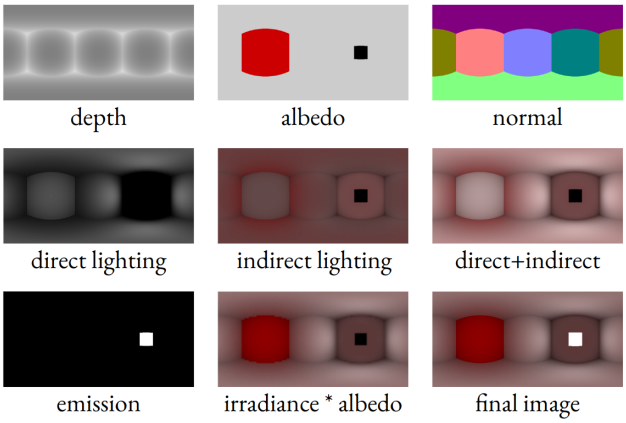


Figure 1. **Didactic toy scene.** Blender Cycles path traced depth, diffuse albedo, normal, direct and indirect diffuse lighting, and emission from render layers.

We address both limitations with a differentiable equirectangular rasterizer for 360° RGB-D images. The method constructs a closed mesh from the omnidirectional depth map, handles seam and pole discontinuities in equirectangular rasterization, and renders irradiance maps that are differentiable with respect to surface normals and reflectance. To compute irradiance at each surface point, we place a virtual camera there and render the surrounding

scene. As the camera is inside the enclosing mesh, nearby triangles subtend large solid angles and cover many pixels, making rasterization an efficient approach. We validate the rasterizer in a controlled setting with path-traced synthetic scenes, where ground truth illumination, depth, normals, and reflectance are all available. Then, we use the rasterizer in analysis-by-synthesis experiments that characterize the behavior of joint normal and reflectance optimization under known illumination.

In summary, we contribute:

- A differentiable equirectangular rasterizer for 360° RGB-D images with seam and pole handling;
- A physically-based irradiance computation method for near-field omnidirectional lighting; and
- Experiments that characterize gradient-based optimization of normals and reflectance under known illumination.

2. Related Work

Inverse rendering and intrinsic decomposition. Intrinsic image decomposition [2] separates an image into reflectance (albedo) and shading layers. The problem is inherently ill-posed from a single image: infinitely many reflectance–shading pairs can explain the same observation. Classic approaches impose hand-designed priors: Retinex [12] assumes piecewise-constant reflectance and smooth shading; Barron and Malik [1] combine shape, reflectance, and illumination priors within a unified optimization. More recently, deep learning methods learn priors from synthetic [16–18, 31] or human-labeled [3] datasets, and modern approaches use diffusion models to capture the distribution of plausible decompositions [4, 10, 30]. Self-supervised methods exploit multi-view consistency [29] or known outdoor illumination [24]. Our work differs in focus: rather than proposing a decomposition system, we build a differentiable forward model for 360° RGB-D images and study its use in gradient-based optimization.

Shape from shading. Recovering surface geometry from shading cues is a classical problem [8]. Photometric stereo [28] resolves the per-pixel normal ambiguity by using multiple images under different illumination, providing additional constraints. In the single-image setting, shape-from-shading requires priors on either illumination or reflectance. Our setting is intermediate: radiance is captured but we have only a single 360° image with depth.

360° scene understanding and image-based lighting. 360° HDR images provide direct access to the environment radiance field, enabling physically-based image formation [5, 19]. For indoor scenes, Li et al. [13] combine a data-driven normal and albedo estimator with a non-differentiable nearest-neighbor irradiance computation for screen-space refinement. This cannot optimize normals through shading because the irradiance computation is not differentiable. Other works explore 360° images for depth estimation [23], novel view synthesis [26], and scene relighting [6, 7]. The 360° setting is attractive because the environment radiance field is observed from the camera position, and our rasterizer exploits this advantage.

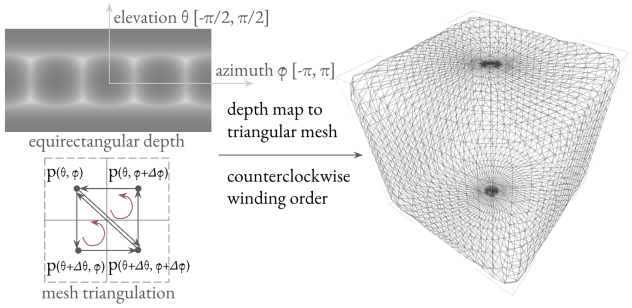


Figure 2. **Constructing a mesh from equirectangular depth.** We create a triangle mesh from the depth map with one vertex per pixel and local pixel connectivity. The winding order sets surface normals to point toward the camera at the scene center. Polar and longitudinal connectivity are omitted for clarity.

Differentiable rendering. Gradient-based optimization through the rendering process enables recovery of scene parameters from images. Approaches range from mesh rasterization [11, 15, 22] to physically-based path tracing [14, 20]. In inverse rendering, differentiable renderers are used to optimize scene parameters (geometry, materials, lighting) from images [9, 25]. Our work complements prior differentiable rendering efforts by adapting the rasterizer to equirectangular omnidirectional geometry and near-field irradiance computation.

3. Irradiance Rasterization from 360° Images

We construct a physically-based, differentiable forward model that maps normals and reflectance to an image under known illumination. We build this from an equirectangular mesh rasterizer atop nvdiffrast [11] and PyTorch. Its use in controlled experiments follows in Section 4. Throughout, we use a synthetic cube room scene as a didactic example: a cube with a square white light source on one face, red reflectance on the opposing face, and neutral reflectance elsewhere, making interreflections visible (Fig. 1).

3.1. Mesh construction

Let us assume we have a depth map computed from a camera system, e.g., from a multi-camera stereo 360° rig. We begin by creating a triangle mesh from the input depth map. We un-project the scene depth map to a point cloud, associate each point to one vertex of the mesh, and join neighboring pixels by two triangles (Fig. 2). For the 360° camera setting, the triangles are constructed such that the winding

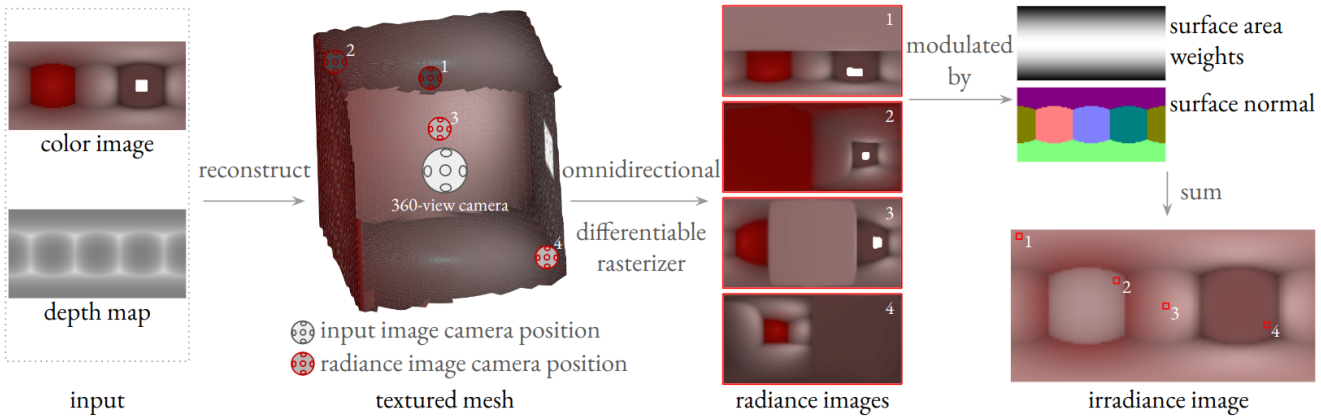


Figure 3. **Rendering irradiance from RGB-D images.** Given a vertex-colored triangle mesh, we move an omnidirectional camera to each vertex to render a radiance image, representing the illumination at that vertex from each direction. To obtain the irradiance at that vertex, we multiply the radiance map with the pixel surface area weight and the cosine between surface normal and inverse of incoming light direction, then sum all pixels. Iterating the process for each vertex produces the irradiance map. We show four example radiance images, where their corresponding pixel is indicated by red boxes in the irradiance image.

order determines the face pointing inwards to the camera. At the longitude wrap, vertices must be connected to their neighbors across the azimuth angle $\phi = \pm\pi$. For the zenith, vertices in the top row must be connected to a single new vertex at the pole; similarly for the nadir. We have now formed a closed geometry. We assume that all properties to be optimized over the mesh are stored at vertices; e.g., that texture is not a separate map but is stored as vertex color.

3.2. Equirectangular mesh rasterizer

Next, we define how to rasterize this geometry into an equirectangular map for an arbitrary camera point inside the scene. We will later compute irradiance by rendering hemispheres at each surface point. Rasterizing the geometry into an equirectangular map is convenient for this purpose. We specify a virtual camera center of projection within the interior of the geometry and equirectangularly project the geometry onto a plane for z-sorting and rasterization.

3.2.1 Longitudinal discontinuity

The vertices of each triangle must not be allowed to span $\phi = \pm\pi$, otherwise we will incorrectly draw large back-facing triangles over the plane and leave missing regions at the frame's edge. Instead, each of these triangles must be copied and drawn twice with individual vertex locations shifted $\pm\pi$ such that the first copy crosses the $\phi = -\pi$ line and the second copy crosses the $\phi = \pi$ line. First, we identify which of the mesh triangles would contain the line $\phi = \pm\pi$, then store their vertex indices. Then, just before projection, we dynamically create a new set of triangles from the same vertex indices: one with vertex positions adjusted such that the triangle spans $\phi = \pi$, and one such that it spans $\phi = -\pi$. For differentiation, the gradient through rasterization attributed to any triangle vertices must be summed back to the true triangles in the underlying mesh. As the dynamically-created triangles share the same vertices in memory, each vertex property receives the correct gradient contribution from both $\phi = \pm\pi$ copies.

3.2.2 Polar discontinuity

We use a similar approach to handle the poles, but these have additional complexity. As triangles approach the pole under projection, their edges should become less linear, but accounting for this through subdivision is expensive as performance is bound by the number of triangles. Given that our initial mesh is dense and the triangles are small, we assume that the linear approximation is sufficient.

Next, we consider the pole itself where the new zenith of the virtual camera position maps to the line $\theta = \pi/2$, and the new nadir to $\theta = -\pi/2$. We consider the case when a pole is contained within a triangle. A triangle that surrounds a pole will cause significant artifacts as its vertices will lie at arbitrary points along the x-dimension of the image, will potentially be back facing, and will leave empty space. First, just before projection, we identify the triangle that contains the pole, and do not draw it unless it is back facing. Next, we dynamically create two screen-space triangles each to span the potentially-empty top region of the frame, drawn behind any other content.

3.3. Alternative approaches

Cubemap rasterization. A cubemap rasterizer would render six perspective views per surface point to capture the surrounding radiance field. Standard perspective rasterization handles frustum boundaries natively, so each face requires no special discontinuity handling, and the formulation is differentiable when built from differentiable perspective rasterizers. However, it replaces a single omnidi-

Algorithm 1: Irradiance computation

Data: Near-field lighting mesh: vertex v, normal n
Result: Equirectangular irradiance image: L
Precompute camera ray directions $r(i, j)$;
Precompute area weights $A(i, j)$;
Initialize irradiance image pixel value: $L(\theta, \phi) = 0$;
for *each vertex* $v(\theta, \phi)$ *on the mesh* **do**
  Move virtual camera slightly inward to avoid self-intersection: $pos(\theta, \phi) = 0.95v(\theta, \phi)$;
  Render a 360-degree image: pixel value $P(i, j)$;
  Sum radiance received over the surface visible hemisphere at $v(\theta, \phi)$ and assign to the irradiance image pixel: $L(\theta, \phi) = \sum_{i,j} P(i, j) \cdot A(i, j) \cdot (n(\theta, \phi) \cdot r(i, j))$;
end

rectional render with six per surface point, multiplying the batch size and GPU memory required for simultaneous gradient computation. Instead, we adopt the omnidirectional rasterizer and handle the equirectangular seam and pole discontinuities described in Section 3.2.

Path tracing. A differentiable path tracer would support richer light transport, but our setting assumes only single-bounce diffuse irradiance under fixed geometry. Rasterization provides this at lower cost, as discussed in Section 1. We use path tracing only as a validation reference.

3.4. Irradiance computation

Most intrinsic decomposition methods estimate shading, which is a ratio at a 2D image pixel representing the change in color of a surface due to light interactions. 360° HDR images with depth let us compute irradiance: the physically-based amount of light arriving at a 3D surface point, integrated over the hemisphere of incoming directions. Given high-dynamic range inputs, we compute this up to scale, knowing the relative total radiant power at the surface.

3.4.1 Scene assumptions and limitations

We make the following simplifying assumptions about the scene, which are most naturally satisfied in enclosed indoor environments such as rooms in houses or offices, and explicitly scope our experiments:

- The RGB input is HDR and approximately linear, with negligible saturation in bright regions.
- The scene is mostly enclosed, and the observed depth map covers the dominant visible surfaces around the camera.
- Depth is bounded to a finite range and is treated as fixed during optimization.
- Surface reflectance is diffuse (Lambertian), so view-dependent specular effects are not modeled.
- Illumination is largely explained by radiance emitted from visible scene surfaces.
- Hidden emitters, strong ambient transport not represented by the visible geometry, and significant external directional lighting are not modeled.

Under these assumptions, a single 360° RGB-D image provides enough information to reconstruct a closed proxy geometry and compute near-field irradiance from the observed radiance field. When these assumptions are violated, for example, in scenes with missing geometry, glossy materials, hidden light sources, or strong outdoor illumination entering the room, the rendered irradiance becomes an approximation, and the residual must be absorbed by the optimization objective. In practice, the approximation also depends on the quality and any discretization of the proxy geometry. Noisy or incomplete depth, poorly tessellated thin structures, residual seam and polar rasterization error, and finite hemispherical sampling can introduce additional error.

3.4.2 Irradiance rendering

We show the irradiance computation process in Fig. 3, with details in Algorithm 1. For a scene with the presumed surface material and emitters, at a location x the total incident irradiance $L_{irr}(x)$, reflected radiance $L_r(x)$, and total outgoing radiance $L_o(x)$ are:

$$\begin{aligned} L_{irr}(x) &= \int_{\Omega} L_i(x, \omega_i) \cdot (n \cdot \omega_i) d\omega_i \\ L_r(x) &= L_{irr}(x) \cdot \rho(x) \\ L_o(x) &= L_e(x) + L_r(x) \end{aligned} \tag{1}$$

where n is the surface normal at x, ω_i the direction from x toward the light source, $L_i(x, \omega_i)$ the incoming radiance at x along direction ω_i, Ω the unit hemisphere centered around n, ρ the albedo at x, and $L_e(x)$ the emission at x.

For an input equirectangular image, we approximate the per-pixel irradiance $L(\theta, \phi)$ at the un-projected 3D point $v(\theta, \phi)$ by discretizing the hemisphere integral. We rasterize a 360°view of the scene at location $v(\theta, \phi)$; each pixel $p(i, j)$ in this rendered image represents radiance from the corresponding direction:

$$\begin{aligned} L(\theta, \phi) &= \frac{1}{\pi} \sum_{i,j} P(i, j) \cdot A(i, j) \cdot \max(0,\, n(\theta, \phi) \cdot r(i, j)) \\ A(i, j) &= (\phi_1 - \phi_0) \cdot (\sin(\theta_1) - \sin(\theta_0)) \end{aligned} \tag{2}$$

where $P(i, j)$ is the radiance intensity, $A(i, j)$ the subtended pixel area weight, and $r(i, j)$ camera ray direction.

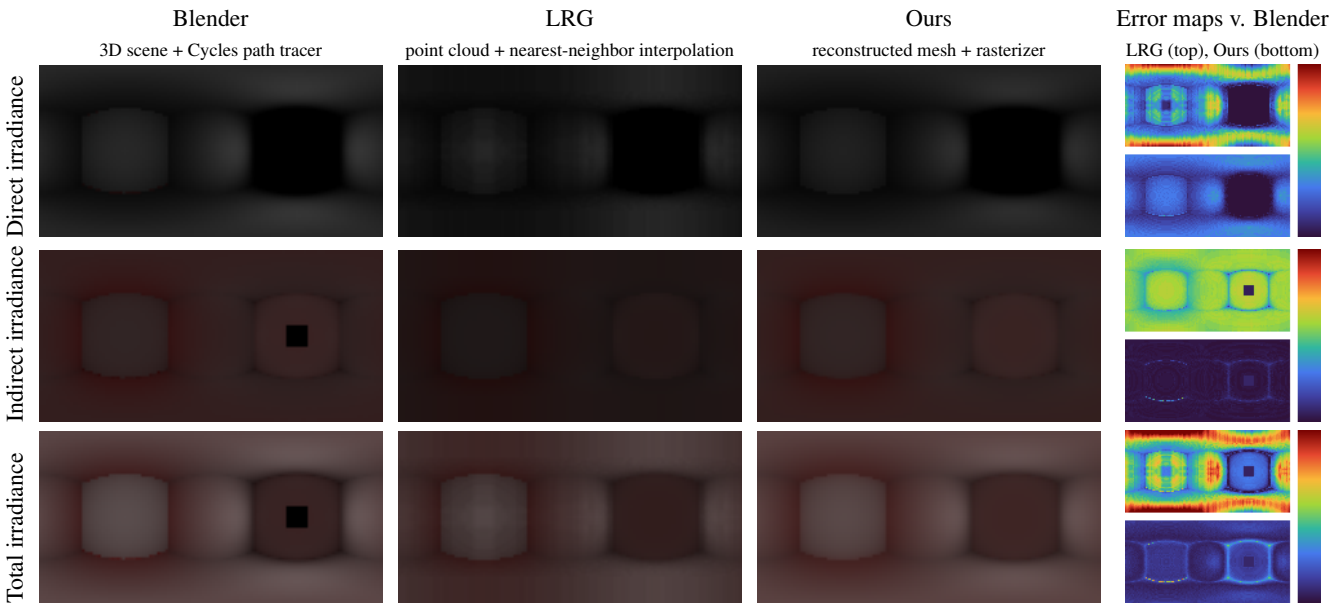


Figure 4. **Validation against a path-traced reference and LRG.** Our method renders indirect lighting more faithfully when comparing reflected-light components. Our rasterizer closely matches the path-traced reference, with remaining error concentrated near depth discontinuities. Blender reports zero irradiance on light-source pixels because the albedo is set to zero, so the square area light appears black. Although our rasterizer uses only the total observed radiance field to compute irradiance, this synthetic component-wise visualization helps explain where the methods differ. In particular, mesh rasterization better preserves visibility and interreflection structure, yielding a total reflected image closer to the Blender reference than nearest-neighbor interpolation. Images are rescaled HDR visualizations in a common linear intensity space, so their appearance may differ from other figures. All error map legends use the same global range: 0.0 to 0.095.

Precomputed radiance maps. Computing the hemispherical rasterization for every surface pixel at every optimization iteration is prohibitively expensive. Instead, we precompute the radiance field: for each of the 64×128 surface points, we rasterize the full environment and store the result as a 5D tensor $L(\theta, \phi, i, j, c)$ of shape $(64, 128, 64, 128, 3)$, i.e., the incoming radiance from all directions at each surface point. During optimization, we compute irradiance as a weighted sum over this precomputed tensor, which reduces the forward pass to a single matrix operation. This precomputation is valid as long as the depth (surface positions) remains fixed; e.g., in experiments where the target of optimization is material or surface normal parameters.

3.5. Image formation model

For our assumed scenes, light sources are captured in the 360°image. We use the image formation model:

$$I = I_{irr}(\mathbf{n}(\theta, \phi)) \cdot \rho(\theta, \phi) + E \tag{3}$$

where I is the input HDR image, I_{irr} the irradiance computed from the surface normal $\mathbf{n}$ and the known radiance field, ρ the albedo, and E the emitted energy from light sources (with light source albedo set to 0). The differentiable rasterizer also enables gradient-based optimization of surface properties, which we investigate in Section 4.

Table 1. **Quantitative evaluation of irradiance computation.** Results are shown on three synthetic scenes with ground-truth normals. For near-field reprojection, LRG [13] uses nearest-neighbor ray interpolation, whereas our method rasterizes a mesh.

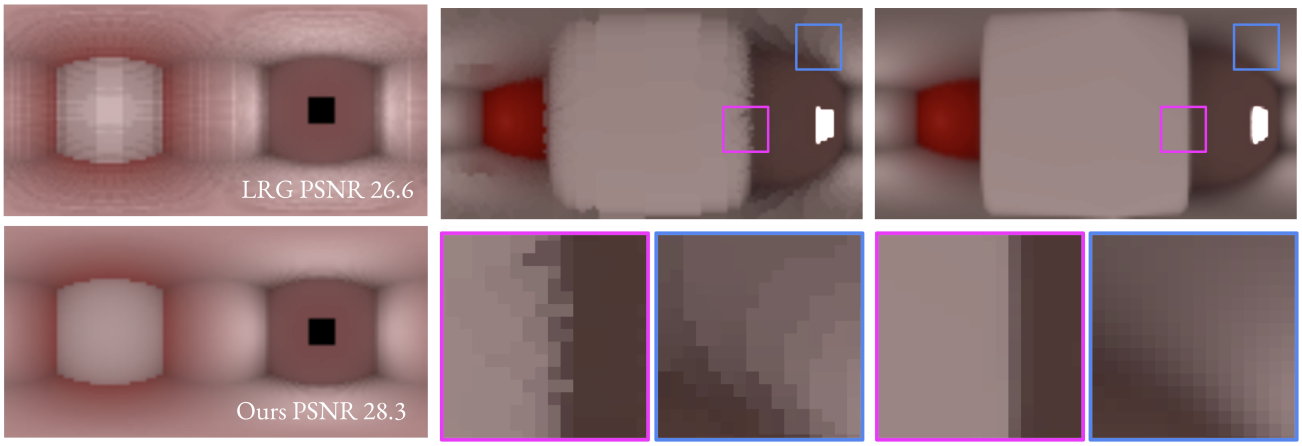
Scenes Irradiance	Method	L2 ▼ ×100	PSNR ▲	SSIM ▲	LPIPS ▼
Cube	LRG	0.0430	26.62	0.850	0.357
	Ours	**0.0241**	**28.32**	**0.964**	**0.047**
Classroom	LRG	0.5042	25.03	0.823	0.209
w/o chairs	**Ours**	**0.3401**	**25.12**	**0.871**	**0.126**
Classroom	LRG	1.6233	19.82	0.706	0.222
w/ chairs	**Ours**	**0.9115**	**20.91**	**0.756**	**0.189**

Table 2. **Runtime of irradiance computation.** Wall-clock runtime at two equirectangular resolutions.

Method	1024×512	128×64
LRG [13]	4h 50min	26s
Ours	45min	10s

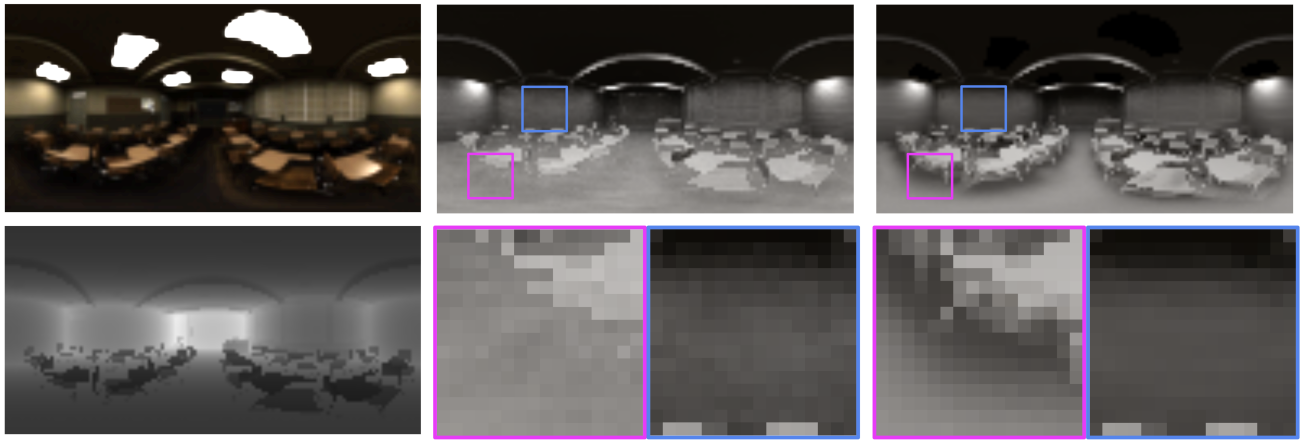
4. Experiments

We first validate the rasterizer against path-traced references and compare against the nearest-neighbor approach of Li et al. [13]. We then design controlled experiments to characterize the behavior of joint normal and reflectance optimiza-



(a) irradiance - LRG vs. ours (b) radiance - LRG (c) radiance - ours

Figure 5. **Comparison with LRG on rendered radiance and irradiance.** LRG [13] reprojects a point cloud and fills missing values by nearest-neighbor interpolation, whereas our method rasterizes a connected mesh. Using the same Gaussian-filtered input 360° image and depth map, our rendered radiance images are smoother near depth discontinuities. These radiance differences carry over to the irradiance maps, where LRG shows aliasing and ringing artifacts that are reduced by mesh rasterization.



(a) input RGB-D (b) irradiance - LRG (c) irradiance - ours

Figure 6. **Occlusion handling in irradiance of the classroom scene.** Mesh rasterization better preserves visibility at occlusion boundaries and restores shadows cast by desks and chairs, whereas nearest-neighbor reprojection [13] misses these effects because the scene is represented only as disconnected points rather than joined surfaces.

tion under known illumination.

4.1. Rasterization vs. Blender ray-traced output.

As a validation of our rasterizer, we compare our rasterized irradiance image against Blender Cycles path-traced ground truth (Fig. 4). The difference is small. Remaining error comes from: incomplete hemisphere sampling when integrating incoming light; rasterization artifacts, such as straight triangle edges near the poles rather than curved ones; or geometric differences due to limited tessellation, especially at depth edges. This error is sufficiently low to support optimization using a reconstruction loss.

4.2. Shading Approximation Comparison

We compare against the nearest-neighbor irradiance approximation used by Li et al. [13], which uses precomputed irradiance and therefore cannot backpropagate shading gradients to normals. It is a different numerical approximation of the same image-based lighting model: our approach uses mesh rasterization with per-triangle visibility, while LRG uses point-cloud reprojection with nearest-ray interpolation.

Irradiance quality. Our rasterization produces lower irradiance error across three synthetic scenes (Table 1). We use three complementary visual comparisons to explain this result. Fig. 5 compares our rasterized radiance and irradiance maps against LRG, showing reduced aliasing and more

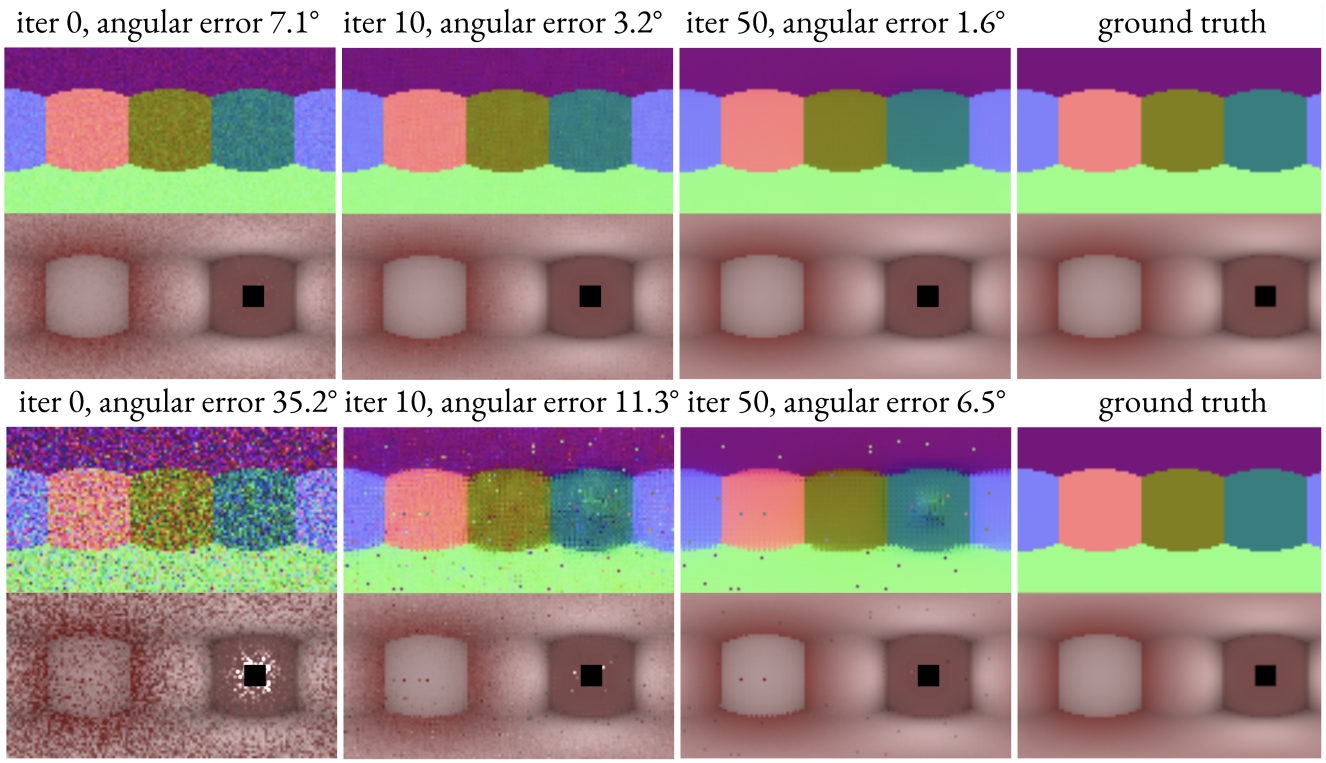


Figure 7. **Noisy normal estimates can in principle be optimized.** We add Gaussian noise with mean angular error of 7.1°(*top*) and 35.2°(*bottom*) to the ground-truth normal map. We then penalize the difference between irradiance rendered from the noisy normals and irradiance rendered from the ground-truth normals. We show iterations 0, 10, and 50, together with the ground truth for comparison.

coherent visibility near depth discontinuities. Fig. 4 provides a diagnostic component-wise comparison, illustrating how visibility and interreflection differences propagate into the total reflected light. Fig. 6 isolates an occlusion case in the classroom scene, where mesh rasterization recovers shadows that nearest-neighbor reprojection misses.

Runtime. Our rasterization-based approach is more efficient at computing irradiance at both tested resolutions than LRG (Table 2 wall-clock runtime).

4.3. Experimental Protocol

Synthetic scene. We use a Blender-rendered cube room scene with textured walls and one area light, rendered via path tracing to provide ground truth for all scene quantities: HDR illumination, depth, surface normals, and reflectance. We work at 64×128 equirectangular resolution for the optimization grid (surface pixels), with radiance maps precomputed at matching resolution. We also created a classroom scene, one without chairs and one with chairs to investigate the shading computation when occlusion is present.

Initialization. Normals and albedo are initialized from the LRG pipeline [13], which uses an RN-Net, trained on a synthetic dataset in a supervised manner, to predict initial estimates from ground truth depth.

Fixed quantities. Illumination is given by the 360° HDR image (used directly as the environment illumination). Depth is fixed to ground truth. The radiance field $L(\mathbf{x}, \omega)$ is precomputed for all 64×128 surface points using our equirectangular rasterizer.

Metrics. We report scale-invariant L2, PSNR, SSIM, and LPIPS for irradiance computation against ground-truth and angular error in degrees for normals (mean absolute error between predicted and ground truth normal directions). All metrics are computed on non-emission pixels only.

4.4. Normal optimization through irradiance.

As a proof of concept, we verify that the differentiable irradiance computation propagates gradients to surface normals. We add synthetic Gaussian noise to ground truth normals and minimize the irradiance difference between the noisy and ground truth irradiance. Fig. 7 shows that normals can be recovered from the irradiance signal alone: in two experiments, the optimization reduces noise from initial random errors of 7.1° and 35.2° toward the ground truth over 50 iterations.

4.5. Optimizing Normals and Reflectance

Loss. We further optimize normals $\mathbf{n}$ and albedo ρ jointly using Adam (lr $= 0.001$, StepLR with step 50 and $\gamma = 0.5$) for 50 iterations. The total loss is:

$$\mathcal{L} = \lambda_{\text{recon}}\mathcal{L}_{\text{recon}} + \lambda_{\text{am}}\mathcal{L}_{\text{am}} + \lambda_{\text{nm}}\mathcal{L}_{\text{nm}} \tag{4}$$

where $\mathcal{L}_{\text{recon}}$ is the photometric reconstruction loss; $\mathcal{L}_{\text{am}}$ and $\mathcal{L}_{\text{nm}}$ are regularization terms penalizing deviation from the initial estimates of albedo and normals.

$$\mathcal{L}_{\text{recon}} = \|I - (\rho \cdot S(\mathbf{n}) + E)\|_1 \tag{5}$$
$$\mathcal{L}_{\text{am}} = \|\rho - \rho_0\|_1 \tag{6}$$
$$\mathcal{L}_{\text{nm}} = \|\mathbf{n} - \mathbf{n}_0\|_1 \tag{7}$$

The baseline experiments use $\lambda_{\text{recon}} = \lambda_{\text{am}} = \lambda_{\text{nm}} = 1$.

Optimization behavior. We observe that normals do not improve through optimization. The optimizer successfully reduces reconstruction error, but primarily through changes in albedo rather than normals. Despite our improved irradiance accuracy over LRG, the normal optimization behavior is qualitatively similar: normals remain difficult to optimize under both approaches. Shading error decreases substantially with our approach, but normal angular error barely changes. This suggests that the observed optimization difficulty is not specific to our rasterization-based shading implementation. Optimizing normals together with albedo under a photometric objective remains challenging.

4.6. Discussion

These experiments show that the proposed differentiable rasterizer is accurate enough to support analysis-by-synthesis optimization from 360° RGB-D images. In our setting, reconstruction quality improves reliably, while recovery of surface normals remains more challenging than recovery of reflectance. We view these results primarily as evidence that the rasterizer is useful for studying inverse rendering in omnidirectional settings. A deeper analysis of the gradient behavior and the geometry-reflectance coupling is left for future work.

5. Conclusion

We presented a differentiable irradiance rasterizer for 360° RGB-D images. Our method constructs a closed mesh from omnidirectional depth, handles seam and pole discontinuities in equirectangular rasterization, and computes physically-based irradiance from near-field omnidirectional lighting. On controlled synthetic scenes, the rasterizer produces irradiance maps that closely match path-traced references and is efficient enough to support reconstruction-loss optimization. Further, we used the rasterizer in analysis-by-synthesis experiments under known illumination, showing that it serves as a possible forward model for optimization from omnidirectional observations. In future work, we will use this rasterizer to study optimization behavior and scene-property ill-posedness under known illumination.

References

[1] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2013. 1, 2

[2] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 2 (3-26):2, 1978. 1, 2

[3] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (TOG)*, 33(4): 1–12, 2014. 1, 2

[4] Chris Careaga and Yağız Aksoy. Colorful diffuse intrinsic image decomposition in the wild. *ACM Transactions on Graphics (TOG)*, 43(6):1–12, 2024. 1, 2

[5] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH'97*, pages 369–378, 1997. 1, 2

[6] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *arXiv preprint arXiv:1704.00090*, 2017. 2

[7] Marc-Andre Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagne, and Jean-Francois Lalonde. Deep parametric indoor lighting estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[8] Berthold K P Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. *MIT AI Laboratory Technical Report*, 1970. 2

[9] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 2

[10] Peter Kocsis, Vincent Sitzmann, and Matthias Nießner. Intrinsic image diffusion for indoor single-view material estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5198–5208, 2024. 1, 2

[11] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 2

[12] Edwin H Land and John J McCann. Lightness and retinex theory. *Josa*, 61(1):1–11, 1971. 1, 2

[13] Junxuan Li, Hongdong Li, and Yasuyuki Matsushita. Lighting, reflectance and geometry estimation from 360 panoramic stereo. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10586–10595. IEEE, 2021. 1, 2, 5, 6, 7

[14] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 2

[15] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 2

[16] Zhengqi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *Proceedings of the European conference on computer vision (ECCV)*, pages 371–387, 2018. 1, 2

[17] Zhengqin Li, Ting-Wei Yu, Shen Sang, Sarah Wang, Meng Song, Yuhan Liu, Yu-Ying Yeh, Rui Zhu, Nitesh Gundavarapu, Jia Shi, et al. Openrooms: An open framework for photorealistic indoor scene datasets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7190–7199, 2021.

[18] Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A Efros, and Noah Snavely. Learning to factorize and relight a city. In *European Conference on Computer Vision*, pages 544–561. Springer, 2020. 2

[19] Leonard McMillan and Gary Bishop. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, page 39–46, New York, NY, USA, 1995. Association for Computing Machinery. 2

[20] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019. 2

[21] Albert Parra Pozo, Michael Toksvig, Terry Filiba Schrager, Joyce Hsu, Uday Mathur, Alexander Sorkine-Hornung, Rick Szeliski, and Brian Cabral. An integrated 6dof video camera and system design. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019. 1

[22] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 2

[23] Manuel Rey-Area, Mingze Yuan, and Christian Richardt. 360monodepth: High-resolution 360deg monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3762–3772, 2022. 2

[24] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. Nerf for outdoor scene relighting. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[25] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Computer Graphics Forum*, pages 701–727. Wiley Online Library, 2020. 2

[26] Guangcong Wang, Peng Wang, Zhaoxi Chen, Wenping Wang, Chen Change Loy, and Ziwei Liu. Perf: Panoramic neural radiance field from a single panorama. *arXiv preprint arXiv:2310.16831*, 2023. 2

[27] Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360sd-net: 360 stereo depth estimation with learnable cost volume. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 582–588. IEEE, 2020. 1

[28] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980. 2

[29] Ye Yu, Abhimetra Meka, Mohamed Elgharib, Hans-Peter Seidel, Theobalt Christian, and Will Smith. Self-supervised outdoor scene relighting. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[30] Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. Rgb↔x: Image decomposition and synthesis using material-and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 2

[31] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *European Conference on Computer Vision*, pages 519–535. Springer, 2020. 2