PSDR-Room: Single Photo to Scene using Differentiable Rendering

Kai Yan
kyan8@uci.edu
University of California, Irvine
USA

Fujun Luan
fluan@adobe.com
Adobe Research
USA

Miloš Hašan
mihasan@adobe.com
Adobe Research
USA

Thibault Groueix
groueix@adobe.com
Adobe Research
USA

Valentin Deschaintre
deschain@adobe.com
Adobe Research
United Kingdom

Shuang Zhao
shz@ics.uci.edu
University of California, Irvine
USA

(a) **Input Photo**　　(b) **Our Result**　　(c) **Semi-auto Result**

Figure 1: Given an input photo of an indoor scene (a), PSDR-Room automatically retrieves, initializes and optimizes geometry, procedural materials and illumination to closely match the scene appearance. The reconstructed result (b) is a valid 3D scene, with editable camera, lighting, geometry, and materials. We provide an option for the user to pick a list of corresponding crop rectangles for appearance matching, resulting in slightly better color/material matching, as shown in our semi-auto result (c). Our method leverages optimization using path-space differentiable rendering, combined with initialization based on depth and segmentation neural models, and object and material search based on CLIP-space proximity.

ABSTRACT

A 3D digital scene contains many components: lights, materials and geometries, interacting to reach the desired appearance. Staging such a scene is time-consuming and requires both artistic and technical skills. In this work, we propose PSDR-Room, a system allowing to optimize lighting as well as the pose and materials of individual objects to match a target image of a room scene, with minimal user input. To this end, we leverage a recent path-space differentiable rendering approach that provides unbiased gradients of the rendering with respect to geometry, lighting, and procedural materials, allowing us to optimize all of these components using gradient descent to visually match the input photo appearance. We use recent single-image scene understanding methods to initialize the optimization and search for appropriate 3D models and materials. We evaluate our method on real photographs of indoor scenes and demonstrate the editability of the resulting scene components.

1 INTRODUCTION

Progress in computer graphics over recent decades has led to algorithms that turn detailed scene descriptions into highly photorealistic renderings. Such a scene description is composed of many components: lights, materials and geometries (themselves separated into sub-geometries with different material assignments). These pieces all interact to reach the desired appearance. However, the problem of composing such scenes (especially with complex materials and lighting) is a manual undertaking that requires significant technical and artistic skill, creating a need for methods that automatically handle parts of the arduous process of 3D content creation.

Composing a scene from a single photograph with a high level of automation is a long-standing challenge [Roberts 1963]. Several recent work made progress on parts of the problem, focusing either on geometry retrieval and placement [Izadinia et al. 2017; Nie et al. 2020] or materials and lighting [Yeh et al. 2022] independently. Recently, progress in physically-based differentiable rendering reached a point where full light transport computation can also provide additional *gradients* of the final pixel values with respect to all relevant scene parameters: geometric information, material parameters, and lighting parameters. For the first time, this theoretically opens the exciting possibility of jointly refining the estimates of all scene components through optimization.

In this work, we propose a system, PSDR-Room, to match the appearance of a single photograph of an indoor scene by initializing and optimizing all of these components: lighting, geometry and materials. To do so, we leverage recent path-space differentiable rendering (PSDR) approaches [Yan et al. 2022; Zhang et al. 2020a, 2021]. We also take advantage of the recent progress in scene understanding, using powerful models for image segmentation [Cheng et al. 2022], depth estimation [Ranftl et al. 2021, 2020], FOV estimation [Jin et al. 2023], and image latent-space encoding [Radford et al. 2021]. Furthermore, given their importance in industry, we retrieve and optimize materials represented as procedural node graphs, using a recent differentiable approach [Li et al. 2023; Shi et al. 2020]. With our system, given an input photograph, users can generate a visually matching 3D scene within a few minutes, using geometry and procedural material assets from their preferred library. Once optimized, the scene can easily be edited to change its components, or further modify the parameters of the selected procedural materials. Throughout our pipeline, differentiable rendering is crucial in adjusting initial guesses and reaching a high quality match to the target image.

Our method involves several stages as follows. First, the preprocessing step leverages state-of-the-art single image estimators for camera intrinsics, depth and object segmentation, which combine to estimate a point cloud and divide the scene into its component objects. Based on this initial information, the next stage finds an approximate room shape, which we further refine through physically-based differentiable rendering. The object stage then aims at retrieving, positioning and orienting each object detected by the segmentation. Again, our method first estimates a rough position based on the estimated point cloud and further refines this prediction using differentiable rendering. In the material and lighting stage, we assign a material, possibly based on a complex procedural node graph retrieved from a database, to each object and generate lighting, based on the layout of the scene. Once more, we leverage differentiable rendering to jointly refine all material properties as well as the scene lighting.

Our system also allows user guidance, by specifying pairs of crop between the target image and a rendering of our optimised scene to enforce material similarity in these areas. As our method is retrieval based, multiple objects of materials can be similar to the target. In our main setting (automatic), we select the top-1 result, but a user can easily adjust the select object/material among the closest match.

We evaluate our methods against real indoor scene photographs, obtaining visually close reconstructions. We further compare to recent work, showing that our approach is more accurate in jointly estimating lights and materials. In summary, we propose a method to go from a single photograph to a 3D scene with lighting and procedural materials in a few minutes with a minimal amount of user input. This is the first end-to-end system applying physically-based differentiable rendering to turn an input photo into a valid 3D scene with separate objects, capable of optimizing geometry poses, lighting and complex procedural materials.

2 RELATED WORKS

Scene-level inverse rendering. Creating entire 3D scenes from photographs alone is one of the big challenges at the interaction between Computer Graphics and Vision. Previous work leverages deep learning to directly reconstruct meshes [Nie et al. 2020] or infer per-pixel scene illumination, normal and material parameters [Li et al. 2020; Zhu et al. 2022]. Other approaches take advantage of the recent progress in differentiable rendering to optimise for texture and materials [Azinovic et al. 2019; Nimier-David et al. 2021]. Reconstructing both geometries and materials with a production-level quality is a particularly challenging task. Closer to our approach, different work proposed to instead retrieve geometries in a mesh database and optimise their positions given a target photograph, such as the IM2CAD system [Izadinia et al. 2017]. Other approaches [Huang et al. 2018; Nie et al. 2020] predict geometries from a single photo directly, not utilizing a database; it also predicts poses and camera parameters. As opposed to our approach, these works do not match lighting or materials and do not benefit from the recent progress in differentiable rendering to enable fine-grained adjustment to the generated scene.

On the material side, previous work proposed to transfer style or mood of a photograph into a 3D scene through uniform materials optimisation [Nguyen et al. 2012]. A recent method, PhotoScene [Yeh et al. 2022], retrieves and optimizes materials as well as lighting on a pre-existing geometry scene (manually created or predicted by a different method like Total3D [Nie et al. 2020]). This work is fairly close to our material and lighting stage, and also uses differentiable procedural materials as a texture prior. However, they use simpler approximations for differentiable rendering. Our material and lighting results improve upon theirs, due to having a more powerful full-scene differentiable rendering system, and CLIP-based search for materials. We also find that comparing texture on rectangular crops is more robust than masked VGG losses used by PhotoScene.

Differentiable rendering. Specialized differentiable renderers have long existed in computer graphics and vision [Azinovic et al. 2019; Che et al. 2020; Gkioulekas et al. 2016, 2013; Tsai et al. 2019]. Recently, several general-purpose ones like redner [Li et al. 2018], Mitsuba 2/3 [Nimier-David et al. 2019], and PSDR-CUDA [Zhang et al. 2020a] have been developed. Alternatively, Fischer et al. [Fischer and Ritschel 2022] proposed a stochastic parameter space sampling for gradient estimation. A key technical challenge in differentiable rendering is to estimate gradients with respect to object geometry (e.g., positions of mesh vertices). To this end, several approximated methods [Liu et al. 2019; Loubet et al. 2019] have been proposed. Unfortunately, inaccuracies introduced by these techniques can lead to degraded result quality, as demonstrated

by Luan et al. [2021]. On the contrary, recent techniques specifically sampling or reparameterizing visibility boundaries [Bangaru et al. 2020; Li et al. 2018; Zhang et al. 2020a, 2019, 2021] provides unbiased gradient estimates capable of producing higher-quality reconstructions.

Procedural material optimization. Procedural materials are an industry standard for material representation. The manipulation and generation of such materials has been an active field of research in recent years. The most recent work on procedural material is starting to enable procedural graph generation [Guerrero et al. 2022; Hu et al. 2022c], however the generated material quality do not yet match the existing databases. Most relevant to our goals are therefore procedural material parameters estimation [Hu et al. 2019] and optimization [Hu et al. 2022a; Li et al. 2023; Shi et al. 2020] methods. Combined with a new material retrieval approach, we leverage this progress in procedural materials optimization to better match the target scene appearance.

Single image scene understanding. Recent year saw significant progress in neural models for single image scene understanding. We utilize models for intrinsic camera parameter [Jin et al. 2023; Lopez et al. 2019; Zhang et al. 2020b] and depth estimation [Ranftl et al. 2021, 2020]. We benefit from this progress, to better understand the photograph FOV and scene geometry. To automate our method as much as possible, we benefit from recent improvements in instance level segmentation in a single image [Cheng et al. 2022; Kirillov et al. 2023], allowing us to separate the different geometries both spatially and semantically. Our approach relies on these scene understanding components, but is not directly tied to any specific method, meaning that it would benefit from any future progress in this area.

3 PRELIMINARIES

In this section, we introduce several technical tools, recently made available by the research community, which are key enablers of our approach. We give a brief introduction to path-space differentiable rendering [Zhang et al. 2020a, 2021] used by our PSDR-CUDA system. We also discuss the inverse procedural material approach MATch and its library of differentiable procedural nodes, DiffMat [Shi et al. 2020], as well as losses based on Gram matrices of VGG layers [Gatys et al. 2016].

3.1 Path-Space Differentiable Rendering

Physically based rendering is frequently formalized using the path-integral formulation by Veach [1997]. The intensity I of a final image pixel resulting is an integral over all light paths through that pixel:

$$I = \int_\Omega f(\bar{x})\,d\mu(\bar{x}), \tag{1}$$

where: Ω is the path space comprised of light paths $\bar{x} = (x_0, \ldots, x_N)$ of all lengths $N \geq 1$, connecting the camera to the light source through a number of scene interaction events; μ is an appropriate measure on the space of paths. Finally, f is the *contribution function* defined as the product of terms on each path vertex: *source emission* on the light source vertex; *bidirectional scattering distribution functions* (BSDFs) on vertices corresponding to scene geometry reflection and transmission events; *geometric terms* on the

path segments, and a *detector response* term on the camera vertex (typically based on a pixel reconstruction filter such as a box or a Gaussian).

A key challenge is to extend the above formulation to differentiable rendering, where the goal is to compute the derivative of the pixel intensity I with respect to some scene parameter $\theta \in \mathbb{R}$. We could think of θ as a time-like parameter, where the scene "evolves" with it, though this does not need to be the case; there could be many such parameters controlling lighting, materials and geometry.

In the latter case when scene geometry (i.e., union of object surfaces) $\mathcal{M}$ evolves with some parameters $\theta \in \mathbb{R}$, Zhang et al. [2020a; 2021] have demonstrated that the derivative $dI/d\theta$ of Eq. (1) can be expressed as the sum of an *interior* and a *boundary* path integrals where the latter is unique to differentiable rendering and capture light transport paths with a segment constrained on a visibility boundary.

In practice, our renderer PSDR-CUDA gives correct gradients with respect to parameters θ that cause geometric change, because it appropriately samples the boundary as well as interior terms. Note that this formulation can be easily adapted beyond solving full light transport, and we use it to compute anti-aliased depth and object mask images as well, by modifying the contribution function f accordingly.

3.2 Inverse rendering and losses

Optimization-based inverse rendering, or analysis by synthesis, infers a set of m_θ scene parameters $\theta \in \mathbb{R}^n$ by minimizing some predetermined rendering loss $\mathcal{L}$ between the rendered image I and reference image I_0. Solving this optimization using stochastic gradient descent methods such as Adam [Kingma and Ba 2014] requires differentiating the rendering loss $\mathcal{L}$ with respect to the parameters θ. According to the chain rule, the gradient $d\mathcal{L}/d\theta$ satisfies $\frac{d\mathcal{L}}{d\theta} = \frac{\partial\mathcal{L}}{\partial I}\frac{dI}{d\theta}$ where the $\partial\mathcal{L}/\partial I$ on the right-hand side can be obtained using differentiable evaluation of the loss $\mathcal{L}$, and $dI/d\theta$ is computed using differentiable rendering (§3.1).

Procedural materials. Substance materials [Adobe 2023] are an industry standard for defining realistic material textures through procedural node graphs, where nodes generates noises and patterns, and adjust them using image processing filters. The MATch approach and DiffMat library by Shi et al. [2020] implemented differentiable versions of many of the filter nodes in the Substance engine, which can be used to optimize their parameters to match a target material appearance. The DiffMat library was recently improved (DiffMatV2) for faster optimization and generator node optimization [Li et al. 2023]. Similar to PhotoScene [Yeh et al. 2022], we use this approach (we leverage the most recent version) to define a manifold of plausible textures, with a much smaller number of optimizable parameters than texels, providing important regularization such that even invisible parts of the scene objects receive valid texture.

In practice, the presence of procedural material parameters introduces another step into the chain rule above. The differentiable rendering will back-propagate the loss to a texture space gradient, after which the DiffMatV2 library (in Taichi) will take over and further back-propagate to a procedural parameters gradient. This

requires passing gradients between multiple systems written in different languages.

Texture descriptors. Gatys et al. [2016] leverage a pre-trained VGG neural network [Simonyan and Zisserman 2014] to guide style transfer, using the Gram matrices of extracted deep features from the VGG layers as their statistical representation. We use 5 layers (the ones after each pooling operation in the VGG19 variant), and concatenate the flattened Gram matrices into a single vector descriptor of the crop texture. We use this loss to compare the texture content between image crops, and found it to be the most reliable of the alternatives. Note that the descriptor size does not depend on the input crop size, and can be used to compare differently-sized crops.

Heitz et al. [2021] introduced an alternative sliced Wasserstein loss, which in theory compares the distributions of VGG activations more accurately, though this is at the cost of introducing more noise into the gradients. Some previous works have generalized these descriptors to arbitrarily-shaped image regions given by masks [Hu et al. 2022b; Yeh et al. 2022]; however, the masks often need to be processed by case-dependent amounts of erosion, and we find that rectangular crop regions remain more reliable.

4 METHOD

Overview. Our PSDR-Room system starts from a single input image of an indoor scene, with an object segmentation, depth and camera intrinsics (specifically field-of-view) provided by existing estimation methods. Our goal is to obtain a visually matching scene reconstruction, which includes picking the right objects, optimizing their poses, picking the right materials, and optimizing the lighting and materials in the scene. For most objects (at least the ones where texturing makes sense) our goal is to obtain tileable material textures, including diffuse color, roughness and normal maps.

Our method includes four high-level stages. First, in the *preprocess stage* we predict the camera field-of-view and estimate the depth map and segmentation per object; this stage is entirely based on previous methods. These inputs can be interpreted as an approximate segmented point cloud, which is sufficient for initialization decisions. Second, in the *room stage*, we initialize a coarsely aligned room box based on the subset of the point cloud belonging to the room's walls, floor and ceiling, and use differentiable rendering to refine the room box to satisfy an image-space mask loss in combination with a depth loss. Next, in the *object stage*, we search a database of objects to match the segmented objects in the input image, coarsely align them to their point clouds and use differentiable rendering to refine their poses to minimize their image-space mask loss and depth loss. Finally, in the *material and lighting stage*, we choose either a homogeneous material or a procedural node graph from a database for each material part, initialize lighting in the scene, and use differentiable rendering to jointly optimize the materials and lighting; for procedural node graph materials, this optimization backpropagates to their node parameters.

4.1 Preprocess stage

Camera intrinsics. We assume a simple pinhole camera model. Given an input image, we use PerspectiveFields [Jin et al. 2023] to estimate the camera vertical field-of-view and assume square pixels.

In our experiments, the model is robust enough and its predictions do not need manual adjustments.

Depth estimation. We use a recent monocular depth estimation proprietary model based on DPT [Ranftl et al. 2021] and Mi-DaS [Ranftl et al. 2020], to generate an approximate depth map. As the depth units returned are unknown, we simply used the normalised depth and do not assume anything about absolute scale. While not perfect, this depth information works reasonably well for room and object placement, which we will further refine using differentiable rendering, considering image-space losses. We can use depth information from any other source, e.g. a depth sensor.

Segmentation. We use Mask2Former [Cheng et al. 2022] to perform a panoptic segmentation, which yields a mask image for each object, including the room ceiling, floor and walls; these are labeled by the model, which we can use to establish correspondence with our rendered box and objects.

4.2 Room stage

For simplicity we assume the walls, ceiling and floor of the room can be approximated as a box; more complex floor plans could be supported in the future using an extension of our approach. We aim to initialize a coarsely aligned room box to the approximate point cloud estimated above, followed by a refinement stage, where we further optimize the box size and placement to match the image-space segmentation.

Room initialization. In the coarse stage, we take the depth and segmentation masks corresponding to room walls, ceiling and floor (with some erosion applied), to generate a segmented room point cloud, without objects. We estimate the floor plane using RANSAC on the floor segmented point cloud. We then estimate the room height through maximum height point value. If the room does not have a ceiling visible in the image, we simply set the room height sufficiently above the top visible point (1.2×maximum height point value). We rotate the room box horizontally around the predicted floor normal and find the rotation+scaling aligning best with the RANSAC estimated wall from the point cloud. This results in a coarsely aligned room box that approximately matches the predicted room point cloud, and is sufficient as initialization.

Room optimization. We next use differentiable rendering to further optimize the room box, improving visual alignment to the input image. We optimize the pose (object-to-world matrix) of the room, including rotation, translation and scaling. We base our differentiable renderer on PSDR-CUDA and optimise the walls, floor and ceiling positions based on depth maps and predicted maps differences. As PSDR-CUDA handles the edges discontinuities defined in Section 3.1, losses based on depth/mask can be differentiated and optimized smoothly. Note that the rendered masks have anti-aliased edges, which is crucial for gradient descent to work, as only the edge pixels will have non-zero gradients. We use the predicted depth and masks to compute our loss function, minimizing the mean L1 difference of depth predictions, and maximize the intersection over union (IoU) for room walls, ceiling and floor. The IoU metric is extended to correctly handle fractional anti-aliased pixels,
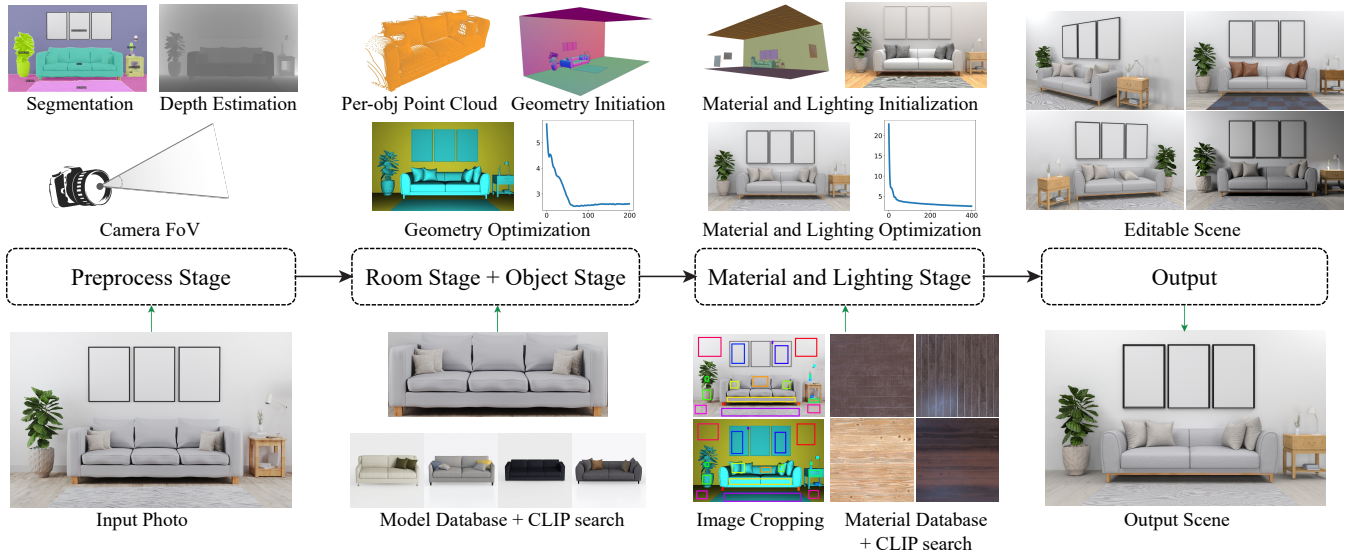
Figure 2: A high-level overview of our pipeline. In the *preprocess* stage (Sec. 4.1), we segment the input image and estimate camera field-of-view and per-pixel depth. A user can optionally edit these estimates. In the *room stage* (Sec. 4.2) and *object stage* (Sec. 4.3), we initialize and optimize the geometry of the box representing the room, as well as the objects in it. The objects are selected from the 3D-Future database by search using similarity in CLIP space. Finally, the *material and lighting stage* (Sec. 4.4) selects procedural node graph materials from a database using CLIP similarity, initializes material texture transforms and lighting, and refines the estimates using optimization, making use of corresponding crop pairs between input and rendered images, which can be automatically detected or user-provided. The final output is a reconstructed scene, which supports camera, geometry, material and lighting edits.
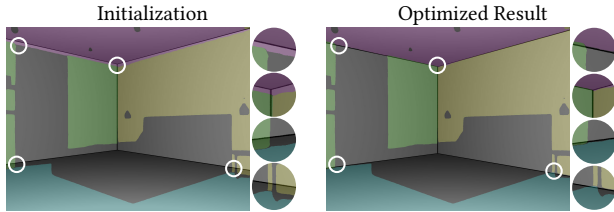

Figure 3: Room stage optimization: We show the estimated segmentation masks of the walls/floor/ceiling versus the rendered room box before and after the room optimization. The optimized room box is better aligned to the mask (see circular insets). Also note that the estimated masks are much smaller than the rendered box sides, as the room was not empty; our differentiable IoU metric is appropriate for this scenario.
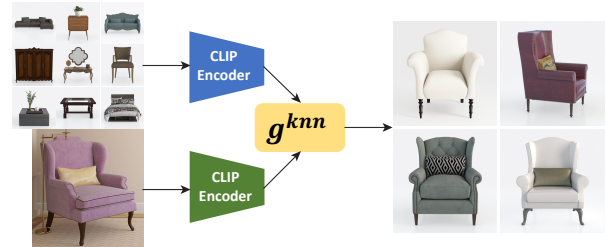

Figure 4: We use CLIP as a zero-shot ranked classifier of 3D geometries from the 3D-Future database. We encode the renderings provided with the database as well as a crop around each object from our input image, and find nearest neighbors using cosine similarity.

giving correct gradients from these terms.

$$\mathcal{L}_{box} = -\sum_i IoU(M_i^r(\theta_{box}), M_i^t) + \mathcal{L}_1(D_r(\theta_{box}), D_i), \quad (2)$$

where θ_{box} are the room "box" parameters, i varies over the set {wall, floor, ceiling}, M_i^r and M_i^t are the rendered and target masks respectively, and D^r and D^t are the rendered and target depths respectively. We normalize depths to a unit range.

4.3 Object stage

Model database. We use the 3D-FUTURE dataset [Fu et al. 2021] as the 3D model database. It contains a total of 10,000+ artist-made models (mostly furniture and similar assets typical in indoor scenes) with split material groups for each model, so that objects with multiple materials are possible (for example, a sofa with pillows of a different material). We discard the materials provided by the dataset, since we want to retrieve and optimize materials depending on the input photograph.

Model search. We use CLIP [Radford et al. 2021] to search 3D models from the database that match the input scene objects. We use the renderings available in the database and encode them all using CLIP into normalized 768-dimensional vectors. To search an appropriate model for each scene object in our segmentation, we crop a target image around each object's mask and encode the cropped image using CLIP. We then select the Nearest Neighbors –using a cosine distance- in CLIP space from the database, providing the closest assets in the database matching the input crop.

Coarse geometry prediction. Similarly to the room stage, we predict each object's pose by fitting to the approximate point cloud of the object and refining it using gradient-based optimization to match the image-space segmentation. For initial alignment, we first find the center as the median value of the point cloud over the three axes. We then compute the scaling factor by matching the mesh bounding box center to the point cloud median, and the mesh radius to half the point cloud radius, where radius is defined as median distance to center. While this process initializes the scaling to a smaller value than reality, it provides a good starting point for the later stage of geometry optimization; making the objects too small at initialization and letting them grow appears to make the optimization better behaved than trying to initialize the scale accurately.

We make the vertical axis for each object orthogonal to the room floor; this heuristic is appropriate for most objects common in indoor scene settings. We try a number of rotations around the vertical axis, which is kept orthogonal to the room floor. We pick as initial rotation the object rotation which has the minimum mean L1 difference between the rendered mask and object segmented mask. If the bottom face of an object's bounding box and the floor are close (distance < 0.1) and the mask edge is close enough to the floor mask (distance < 20px), we mark the object as "on the floor" and use a floor-distance loss when optimizing that object's position. If an object is floating, we search if the object mask edge is close enough (distance < 20px) of another object edge on the floor. If so, we enforce the floating object to stay on top of the parent object.

Refining geometry. Next, we use differentiable rendering to perform a joint pose optimization for all objects. We optimize the scaling, horizontal translation and rotation around the vertical axis –keeping alignment with the room floor/ceiling– for each object. This ensures that objects snapped to the floor/ceiling preserve this constraint. Similarly, for objects on the wall, the optimization transforms and scales them only along the wall. Our differentiable renderer generates the masks with anti-aliasing, which is critical for non-zero gradient in masks comparing losses. For the loss function, we use the mean L1 difference between the rendered and estimated segmentation mask multiplied by the depth of each object. We also use the L1 difference of heavily blurred versions of the masks, using a Gaussian blur.

$$\mathcal{L}_{obj} = \sum_i \mathcal{L}_1(M_i^r(\theta_{obj}) * D_r(\theta_{obj}), M_i^t * D_i) \\ + \sum_i \mathcal{L}_1(G(M_i^r(\theta_{obj}), G(M_i^t))), \quad (3)$$

where θ_{obj} are the object pose parameters, i varies over the set of objects, M_i^r and M_i^t are the rendered and target masks respectively,

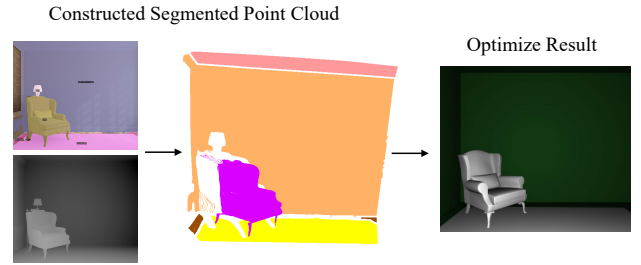Constructed Segmented Point Cloud

Optimize Result

Figure 5: Object Stage: We take the segmentation and depth map to construct segmented point clouds for coarse geometry prediction. We initialize object poses to the point cloud and use differentiable rendering in image space (with mask and depth losses) to further refine the poses.

and D^r and D^t are the rendered and target depths respectively, normalized to unit range. G is a 2D Gaussian convolution pyramid of a mask.

4.4 Material and lighting stage

In this stage, we predict and optimize the lighting and materials for the scene. For this, we establish a list of *crop pairs* between the input image and the rendering. Our optimization will match appearance between the crops of each pair, in addition to optimizing a low-resolution full-image loss. These crop pairs are normally automatically detected, but can optionally be provided by the user for best appearance match.

Crop pairs. To achieve the best robustness and closest appearance match for different material parts, we sample a number of crops inside the intersection region of a mesh material group rendered mask and the object segmentation mask. For each object i, let $M_i := M_i^r(\theta_{obj}) \cap M_i^t$ be the intersection between the rendered M_i^r and the target M_i^t masks for this object. Then, we compute a rectangular mask $W_i \subset M_i$ with maximal size to use as the crop window for this object in both the render and the target images. In this way, (I_i, R_i) forms a crop pair where $R_i = R[W_i]$ and $I_i = I[W_i]$ denote, respectively, crops of the rendered image R and the input image I using the window W_i.

To compare the textures between each pair of crops during optimization, we use a Gram matrix loss as we found it very reliable for optimization, outperforming alternative losses like masked VGG.

Light initialization. In this stage, we generate the initial lighting for the scene. We split the room ceiling into a grid of area lights and scale them down by a ratio of 0.8. We remove the lights that intersect the camera frustum, or are completely behind the camera. We also add one large area light behind the camera and one per invisible wall, simulating an invisible window. We also turn visible windows and lamps into emissive objects. For windows, we add a simple frame, though a database of window geometries could be used if available. For lamps, we make the largest visible rendered part into emissive.

We try a number of uniform light radiance values as initialization and pick the one with the minimum down-sampled L1 loss on
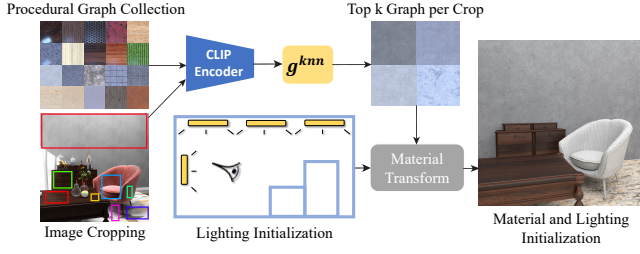
Figure 6: We use CLIP for zero-shot classification and ranking of materials, by comparing encoded thumbnails of our procedural materials with encoded input image crops. A user can optionally pick from top k matches. We further search for the best transform (scale and rotation) for the best texture match; this is done under the initial lighting. We keep some materials homogeneous (more detail in text).

luminance images as our initial light radiance. As light transport is linear in light intensity, a single rendering with unit radiance is needed for this step.

Material initialization and search. For material parts that do not have any available crop pair because of small mask area, or the cosine similarity for the best matched CLIP search result is worse than 0.25, we use homogeneous parameters (constant albedo, roughness, specular). We initialize homogeneous material albedo to the median color of the mesh-mask intersection area and a roughness of 0.5. If there is no intersection, we use the median color of the area of the entire object segment.

For each material part that needs a procedural material (i.e. that has at least one crop pair (I_i, R_i)), we use the crop I_i to run a CLIP search on rendered thumbnails of 118 procedural materials from the DiffMatV2 library provided by the MATch method [Li et al. 2023]. Our material search is based on cosine similarity of normalized CLIP encodings, same as the object search described above. If multiple crops are on a single material part, all pairs are used for CLIP search; for each crop we evaluate the top-10 similar materials. We use a voting scheme and select materials appearing for multiple crops. Out of these selected materials, we compare cosine similarities and select the material with highest mean cosine similarity. To determine the right texture transform, we sample the chosen material with different scales (0.5-8.0) and rotations (-45, 0, 45, 90) and find the one with minimum Gram matrix loss on that crop pair. If the Gram matrix loss of a homogeneous material is better than any of the texture transforms, we refrain from applying textures to this material and treat it as homogeneous. Note that we render and compare these crops under the initial lighting in the scene a the final lighting has not yet been optimized at this stage.

Full scene material and lighting optimization. In the final stage of our pipeline, we jointly optimize all materials and lights in the scene, under full global illumination. The loss function we use combines several terms: the L1 difference between the input image and rendering (downsized to 1/8), the Gram matrix difference for each crop pair, as well as the mean RGB color per object mask: The complete loss for joint material and lighting optimization is as
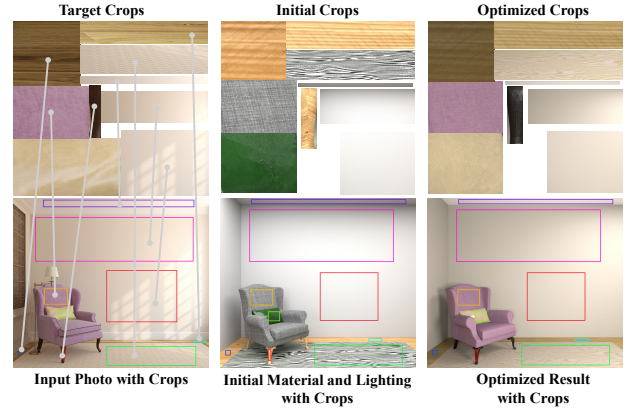

Figure 7: We visualize the crops from the input image, paired with rendered crops at initialization and during optimization. The optimized crops match the mean color and texture better comparing to the initialization. Losses based on Gram matrices of VGG layers are effective at matching texture in rectangular crops without need for pixel alignment.

follows:

$$\mathcal{L}_{final} = \mathcal{L}_1(R_{1/8}(\theta), I_{1/8}) \; + \\ \sum_i \mathcal{L}_1(\mu(M_i^r(\theta_{obj})), C_i) + \sum_i \mathcal{L}_1(T_G(R_i(\theta)), T_G(I_i)), \quad (4)$$

where θ is the vector of all material and lighting parameters, $R_{1/8}$ and $I_{1/8}$ are the rendered and input image, downsized to 1/8 of original size of height and width, i iterates over all crop pairs, (I_i, R_i) are the crop pairs, μ denotes the RGB mean over a mesh mask, C_i denotes the median color computed during initialization and T_G is the Gram matrix texture descriptor of a crop. Note that the optimization over procedural material parameters is bound to a plausible texture manifold allowed by the node graph, acting as a prior and ensuring the optimization generates only high-quality textures without baking in lighting cues, even in invisible parts of the scene.

5 RESULTS

Scene edits. In Fig. 8, we show examples of material editing. We can modify the parameters of the procedural node graphs (c) or switch to a completely different procedural graph material for the floor (d). Please refer to the Supplemental Material for more editing examples.

Ablations of design choices. We conduct ablation studies on various design choices, as demonstrated in Fig. 9, showing that a proper initialization before optimization using differentiable rendering is also important.

Ablation on database size. As described in §4, our pipeline retrieves object geometries from an input database. Fortunately, as demonstrated in Fig. 10, our pipeline is robust to the choice of this database. Even with only 5% of the data used (containing 800 3D models), our pipeline remains well-behaved.
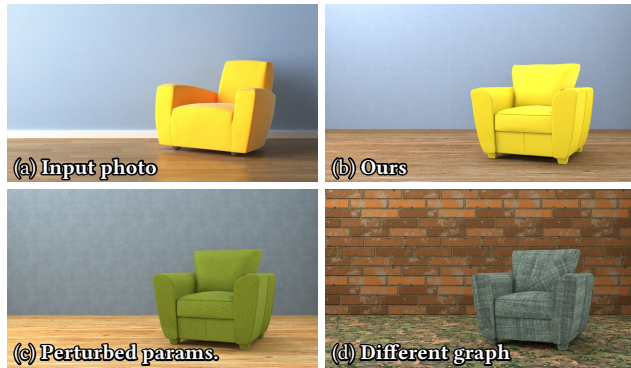
Figure 8: Material variations: We can modify the parameters of the procedural node graphs (c) or switch to a completely different procedural material for the floor (d).
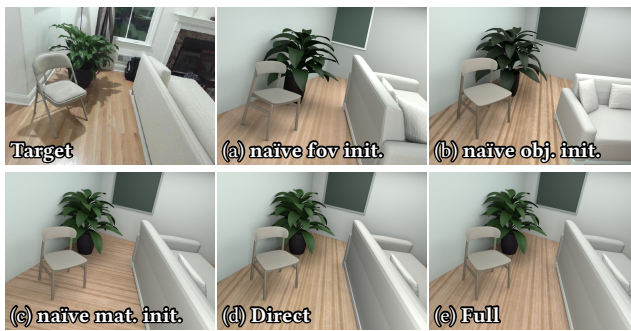

Figure 9: Ablations of design choices for the four ablation of our full result (e): (a) naïve initialization of fov = 40° **leads to distorted results; (b) naïve initialization of object poses; (c) without close initialization of texture transform; (d) direct illumination only.**

Comparison with IM2CAD. We compare to IM2CAD [Izadinia et al. 2017] in Fig. 11. Since the implementation of this technique has not been made publicly available, we apply our pipeline to two examples from their paper. To evaluate the results quantitatively, we use the LPIPS [Zhang et al. 2018] metric, which is appropriate for comparing images with some misalignment, and RMSE (computed using downsampled images) metrics. The quantitative errors are shown under the images, with the lowest error shown in bold, showing that our approach better matches the target photographs.

Comparison with PhotoScene. In Fig. 12, we compare results generated with our technique (b) and two pipelines (c, d) based on PhotoScene [Yeh et al. 2022]. Similar to the comparison with IM2CAD in Fig. 11, we compare LPIPS and RMSE metrics to show the effectiveness of our method. PhotoScene with our geometry (c) produces numerically good results, but worse than ours, due to several factors. They used simpler approximations for differentiable rendering compared to our path-space system. We also found that comparing texture on rectangular crops is more robust than the masked VGG losses used by PhotoScene. Further, our CLIP based search provides

Figure 10: Our pipeline is robust to the choice of shape databases. Rows (b) and (c) show our results for two example target images generated using, respectively, a full database (with 16563 models) and one containing 5% of randomly sampled data.

better results than VGG-based Gram matrix distance as demonstrated on flat surfaces in the supplemental materials. When using

LPIPS & RMSE: **0.481** / **0.038**   **0.598** / **0.039**

LPIPS & RMSE: 0.636 / 0.040   0.680 / 0.041

Figure 11: Comparison with IM2CAD: We compare results generated using our (automated) pipeline and IM2CAD using examples from their paper [Izadinia et al. 2017].

Example 1	Example 2	Example 3	Example 4	Example 5

(a) Target

(b) Ours

LPIPS & RMSE: **0.584 / 0.037** | **0.658 / 0.038** | **0.611 / 0.039** | **0.613 / 0.039** | **0.687 / 0.039**

(c) Ours + Photoscene

LPIPS & RMSE: 0.712 / 0.041 | 0.819 / 0.041 | 0.734 / 0.040 | 0.877 / 0.040 | 0.735 / **0.039**

(d) Total3D + Photoscene

LPIPS & RMSE: 0.800 / 0.040 | 0.768 / 0.040 | 0.781 / 0.040 | 0.822 / 0.040 | 0.809 / 0.040

Figure 12: Results and comparisons for five clean input photos of room scenes (a). We compare our (automated) pipeline (b), our geometric pipeline combined with PhotoScene's material and lighting pipeline (c), and Total3D's geometry prediction combined with PhotoScene's material and lighting (d). LPIPS and RMSE numbers are shown under the images, with the lowest error shown in bold. Our pipeline (b) offers the best visual and numeric quality for all examples. PhotoScene with our geometry (c) produces numerically good results, but visually worse than ours, due to factors we discuss in the main text. Total3D produces lowest-quality geometry (d), though it should be noted that it is solving a much harder problem of neurally predicting the meshes, rather than searching for them in a database. We also provide an additional 100 comparisons with full PhotoScene pipeline using sunRGBD dataset in the supplementary.

PhotoScene with geometries reconstructed by Total3D (d), the reconstruction quality becomes poor, although it should be noted that Total3D [Nie et al. 2020] is solving a much harder problem of directly predicting the meshes, rather than searching for them in a database.

Using user-specified crops. Results generated by our automated pipeline can be further improved by using minimal user inputs. We demonstrate this in Fig. 13 where user-specified crops are used to drive the material and lighting stage (§4.4).

Performance and supplementary materials. All experiments are conducted on a workstation with an AMD Ryzen 9 7950X and an Nvidia RTX 4090 graphics card. On average, the room optimization stage takes 100 iterations, with less than 0.1s per iteration. The geometry initialization and CLIP search take usually around 5s per object in the scene. The optimization time per iteration is around 1–3 seconds, with a total of 300 iterations. The fine stage usually takes 3–5 seconds per iteration, with about 120 iterations. A more detailed performance table is shown in the supplementary materials.

The supplementary also shows animations illustrating optimization progress and partial results.

6 LIMITATIONS AND FUTURE WORK

While PSDR-Room already gives useful results, there are some challenges left. First, while our fully automatic pipeline outperforms previous work, it is not perfect, and can sometimes benefit from user inputs. For example, user-provided crop pairs can focus attention on matching materials that matter to a human viewer, and establish correspondences between material parts that are semantically related but not aligned in image space. Further, a user picking from, say, top-3 materials and objects returned by CLIP search can sometimes make a better choice than simply using the highest-ranked result.

Our current box assumption is simple, and could be extended to more intricate floor plan layouts, as well as non-horizontal ceilings. Our work pushes even the most advanced differentiable rendering approaches to their limits. Intersection between objects causes erroneous zero gradient due to a missing boundary term. Differentiable
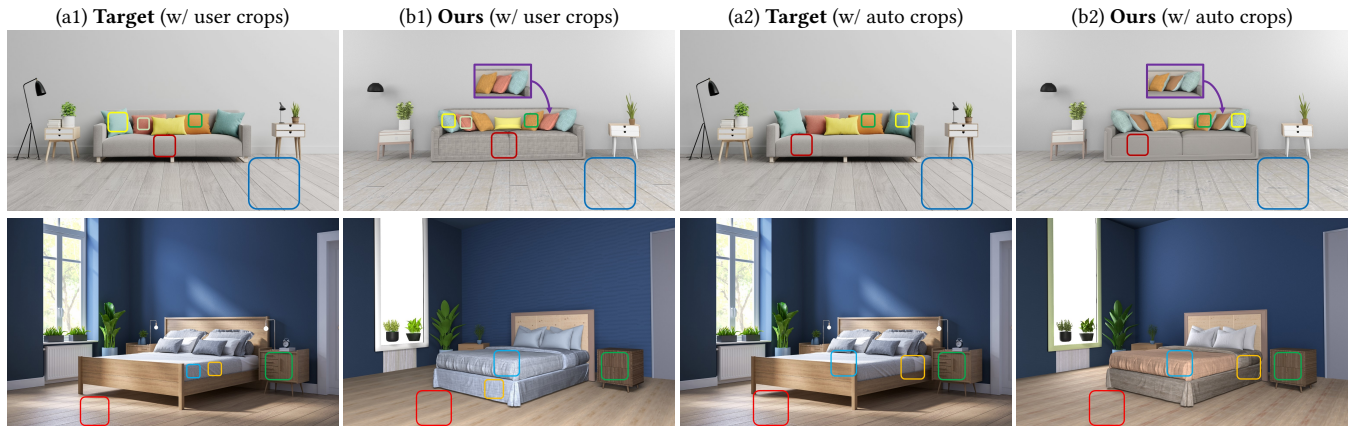
(a1) **Target** (w/ user crops)   (b1) **Ours** (w/ user crops)   (a2) **Target** (w/ auto crops)   (b2) **Ours** (w/ auto crops)

Figure 13: In the case where some correspondences are not trivial to establish (i.e between the different pillows (top row)) or the user desires to match a geometry to a different material in the scene (bottom row), user-specified crops can be used, potentially better matching their intent than our automatic crop selection.

collision detection during the optimization process could help avoid two objects intersecting, but remains out of the scope of this work.

Object databases with well-separated material parts are not easy to come by and our material database is fairly small; a commercial deployment of our method would likely invest in curating larger databases of objects and materials with the right properties.

7 CONCLUSION

In this paper, we introduced PSDR-Room, an end-to-end pipeline that generates 3D models of indoor scenes, complete with objects, lighting and textured materials, with minimal user input. Based on a single segmented image, our technique uses CLIP search to automatically select the shapes and materials for each object from databases. Further, our pipeline refines the object poses, lighting and materials of each object by leveraging physics-based differentiable rendering, significantly improving the reconstruction quality compared to previous work when evaluated on real indoor photographs. Our generated 3D models can be easily edited by the user as a post-process. We believe our pipeline provides a solid foundation for future improvement towards better learning-based initialization approaches, more complex room layouts, or other environments beyond room scenes.

REFERENCES

Adobe. 2023. Substance Designer. https://www.substance3d.com/.

Dejan Azinovic, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. 2019. Inverse path tracing for joint material and lighting estimation. In *Proc. IEEE/CVF CVPR*. 2447–2456.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020), 245:1–245:18.

Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Towards Learning-based Inverse Subsurface Scattering. In *2020 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 1–12.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1290–1299.

Michael Fischer and Tobias Ritschel. 2022. Plateau-free Differentiable Path Tracing. *arXiv preprint arXiv:2211.17263* (2022).

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 2021. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision* (2021), 1–25.

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2414–2423.

Ioannis Gkioulekas, Anat Levin, and Todd Zickler. 2016. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *ECCV*. Springer, 685–701.

Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. 2013. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.* 32, 6 (2013), 1–13.

Paul Guerrero, Milos Hasan, Kalyan Sunkavalli, Radomir Mech, Tamy Boubekeur, and Niloy Mitra. 2022. MatFormer: A Generative Model for Procedural Materials. *ACM Trans. Graph.* 41, 4, Article 46 (2022). https://doi.org/10.1145/3528223.3530173

Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. 2021. A sliced wasserstein loss for neural texture synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9412–9420.

Yiwei Hu, Julie Dorsey, and Holly Rushmeier. 2019. A Novel Framework for Inverse Procedural Texture Modeling. *ACM Trans. Graph.* 38, 6, Article 186 (Nov. 2019), 14 pages. https://doi.org/10.1145/3355089.3356516

Yiwei Hu, Paul Guerrero, Milos Hasan, Holly Rushmeier, and Valentin Deschaintre. 2022a. Node Graph Optimization Using Differentiable Proxies. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 5, 9 pages. https://doi.org/10.1145/3528233.3530733

Yiwei Hu, Miloš Hašan, Paul Guerrero, Holly Rushmeier, and Valentin Deschaintre. 2022b. Controlling Material Appearance by Examples. *Computer Graphics Forum* (2022). https://doi.org/10.1111/cgf.14591

Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. 2022c. An Inverse Procedural Modeling Pipeline for SVBRDF Maps. *ACM Transactions on Graphics (TOG)* 41, 2 (2022), 1–17.

Siyuan Huang, Siyuan Qi, Yinxue Xiao, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. 2018. Cooperative Holistic Scene Understanding: Unifying 3D Object, Layout, and Camera Pose Estimation. In *Advances in Neural Information Processing Systems*. 206–217.

Hamid Izadinia, Qi Shan, and Steven M Seitz. 2017. Im2cad. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5134–5143.

Linyi Jin, Jianming Zhang, Yannick Hold-Geoffroy, Oliver Wang, Kevin Matzen, Matthew Sticha, and David F. Fouhey. 2023. Perspective Fields for Single Image Camera Calibration. In *CVPR*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. arXiv:2304.02643 [cs.CV]

Beichen Li, Liang Shi, and Wojciech Matusik. 2023. End-to-End Procedural Material Capture with Proxy-Free Mixed-Integer Optimization. *ACM Transactions on Graphics (TOG)* 42, 4, Article 1 (2023), 15 pages.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6 (2018), 222:1–222:11.

Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. 2020. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2475–2484.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *ICCV*. 7708–7717.

Manuel Lopez, Roger Mari, Pau Gargallo, Yubin Kuang, Javier Gonzalez-Jimenez, and Gloria Haro. 2019. Deep single image camera calibration with radial distortion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11817–11825.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. 2021. Unified Shape and SVBRDF Recovery using Differentiable Monte Carlo Rendering. *Computer Graphics Forum* 40, 4 (2021), 101–113.

Chuong H. Nguyen, Tobias Ritschel, Karol Myszkowski, Elmar Eisemann, and Hans-Peter Seidel. 2012. 3D Material Style Transfer. *Computer Graphics Forum (Proc. EUROGRAPHICS 2012)* 2, 31 (2012).

Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. 2020. Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes From a Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. 2021. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association. https://doi.org/10.2312/sr.20211292

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.* 38, 6 (2019), 203:1–203:17.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al.

2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PMLR, 8748–8763.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12179–12188.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2020. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence* (2020).

Lawrence G Roberts. 1963. *Machine perception of three-dimensional solids*. Ph. D. Dissertation. Massachusetts Institute of Technology.

Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. 2020. MATch: Differentiable Material Graphs for Procedural Material Capture. *ACM Trans. Graph.* 39, 6, Article 196 (Dec. 2020), 15 pages.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

Chia-Yin Tsai, Aswin C Sankaranarayanan, and Ioannis Gkioulekas. 2019. Beyond Volumetric Albedo–A Surface Optimization Framework for Non-Line-Of-Sight Imaging. In *Proc. IEEE/CVF CVPR*. 1545–1555.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Vol. 1610. Stanford University PhD thesis.

Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph.* 41, 4 (2022), 123:1–123:13.

Yu-Ying Yeh, Zhengqin Li, Yannick Hold-Geoffroy, Rui Zhu, Zexiang Xu, Miloš Hašan, Kalyan Sunkavalli, and Manmohan Chandraker. 2022. PhotoScene: Photorealistic Material and Lighting Transfer for Indoor Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18562–18571.

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020a. Path-space differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 143:1–143:19.

Chaoning Zhang, Francois Rameau, Junsik Kim, Dawit Mureja Argaw, Jean-Charles Bazin, and In So Kweon. 2020b. Deepptz: Deep self-calibration for ptz cameras. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1041–1049.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM Trans. Graph.* 38, 6 (2019), 227:1–227:16.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021. Path-space differentiable rendering of participating media. *ACM Trans. Graph.* 40, 4 (2021), 76:1–76:15.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Rui Zhu, Zhengqin Li, Janarbek Matai, Fatih Porikli, and Manmohan Chandraker. 2022. IRISformer: Dense Vision Transformers for Single-Image Inverse Rendering in Indoor Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2822–2831.