DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Intrinsic Image Distillation for Room-Scale Material Reconstruction

Bendeguz Timar

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Intrinsic Image Distillation for Room-Scale Material Reconstruction

Destillation Intrinsischer Bilder zur Rekonstruktion von Materialien in Raumgröße

Author:	Bendeguz Timar
Supervisor:	Prof. Dr. Matthias Nießner
Advisor:	Peter Kocsis
Submission Date:	2025. 12. 10.

I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 2025. 12. 10. Bendeguz Timar

Acknowledgments

I would like to thank my mom, dad and siblings for constantly reminding me of the big picture while working on this project. I would furthermore like to express my sincere gratitude and formally acknowledge the contributions of my supervisor, Péter Kocsis, without whom this research never would have started or finished.

Abstract

Recent developments in Score Distillation Sampling led to pipelines generating high-quality, visually plausible 3D assets for virtual environments with image and text conditioning. The primary limitation of these approaches is that they do not separate the material from the light, resulting in textures with baked-in lighting effects. In contrast, we proposed a method that utilizes a pre-trained monocular generative intrinsic image estimator to distill Physically Based Rendering material textures for room scale scenes. Given the scene geometric mesh and a set of images spatially covering the environment, our model synthesizes albedo, roughness, metallic, normal and irradiance maps using score distillation. Our method outperforms state-of-the-art optimization-based methods and provides clean material maps, which can subsequently be used in rendering engines for the purpose of relighting or material editing.

Kurzfassung

Jüngste Entwicklungen im Bereich Score Distillation Sampling (SDS) haben zu Pipelines geführt, die qualitativ hochwertige, visuell plausible 3D-Assets für virtuelle Umgebungen unter Nutzung von Bild- und Textkonditionierung generieren. Die primäre Einschränkung dieser Ansätze ist, dass sie das Material nicht vom Licht trennen, was zu Texturen mit ëingebrannten"(baked-in) Beleuchtungseffekten führt.

Im Gegensatz dazu schlagen wir eine Methode vor, die einen vortrainierten monokularen generativen intrinsischen Bild-Estimator nutzt, um Materialien für das Physically Based Rendering (PBR) für Szenen in Raumgröße zu destillieren. Unter Verwendung des geometrischen Meshs der Szene und eines Bilddatensatzes, der die Umgebung vollständig abdeckt, synthetisiert unser Modell Albedo-, Roughness-, Metallic-, Normalen- und Irradiance-Maps mittels Score Distillation. Unsere Methode übertrifft die aktuellen optimierungsbasierten State-of-the-Art-Verfahren und liefert saubere Material-Maps, die anschließend in Rendering-Engines zum Zweck des Relightings oder der Materialbearbeitung verwendet werden können.

Contents

1. Introduction

The automatic generation of high-fidelity 3D assets is a crucial task in modern computer graphics, having prominance in a number of areas such as digital entertainment, virtual and augmented reality and large-scale digital twins. The capabilities of image generation to synthesize high-quality, visually coherent imagery has been integrated throughout 3D asset creation pipelines to optimize 3D representations using 2D supervision in a number of applications, such as multi-view image generation for reconstruction ([1], [2])), text-to-3D synthesis ([3], [4]) or texture generation ([5], [6], [7]). Texture generation is a key subdomain of 3D asset generation and aims to synthesize visually plausible surface detail and material properties of a 3D model. The introduction of Score Distillation Sampling (SDS) in generative AI provided a powerful new mechanism for this task, leveraging a pre-trained 2D diffusion model as high-quality image prior to generate semantically coherent texture maps onto virtual objects.

Methods utilizing SDS for texture generation face one crucial limitation, as they bake in lighting conditions into the synthesized texture, restricting the utility of the generated asset from subsequent editing such as material editing or environment relighting. Previous approaches to texture generation mitigate this by discarding SDS and instead utilizing physically-based inverse rendering methods such as inverse path tracing ([8], [9]), which can exploit optical rules to refine material estimations of the scene. However, physically-based inverse rendering methods are typically more computationally expensive than the optimization process driven by SDS ([8], [3]). Instead, the primary objective of this research is the development of a novel generation pipeline designed to disentangle material parameters leveraging SDS.

The proposed method specifically aims to output separate texture maps for the parameters of Physically Based Rendering, namely albedo, metallic, roughness, normal and irradiance maps, allowing for the successive editing of the leveraged virtual scene such as arbitrary relighting or material editing. The model necessitates solely the scene geometry and image captures of the scene as input and exploits the domain of already available, vast, high-quality diffusion models as score estimators, significantly increasing output fidelity and outperforming current state-of-the-art material distillation methods ([8], [9]).

Furthermore, our approach integrates of an image-space learning objective to mitigate errors such as oversmoothing commonly observed in pure latent score distillators; as well as a visibility-based loss weighting to overcome visual artifacts stemming from varying observation frequency.

In summary, our core contributions are:

1. The proposed pipeline leverages diffusion models for PBR material optimization, lifting

already available generative decomposition models to tackle a 3D optimization task

2. We shift problem domain to image-space by utilizing image-space loss and regularization
3. Combined, the proposed modifications improve albedo prediction PSNR by 1.58 dB compared to current state-of-the-art approaches

2. Related work

2.1. Intrinsic Image Decomposition

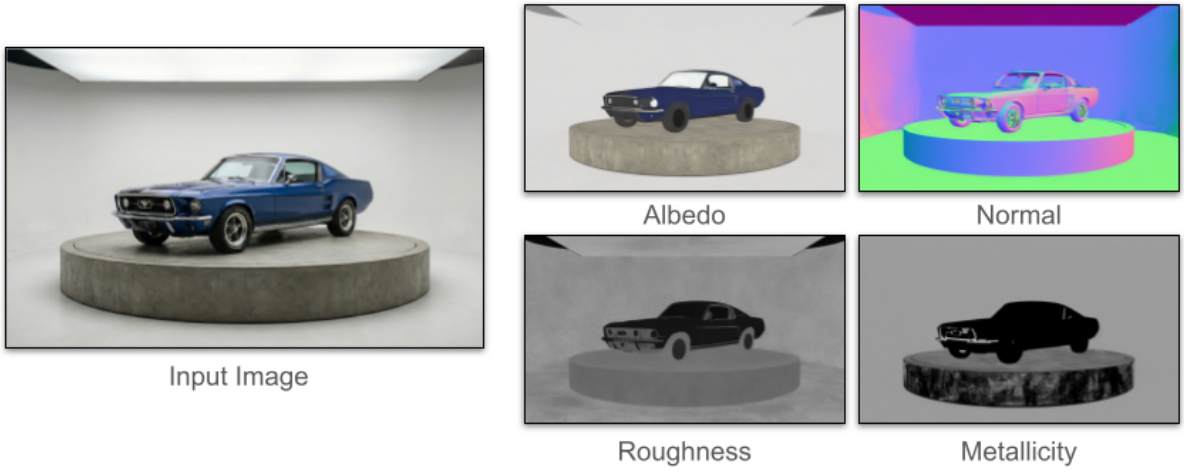


Figure 2.1.: **Input image and output material texture maps of an Intrinsic Image Decomposition pipeline**. The aim of Intrinsic Image Decomposition is to disentangle an input image into its respective intrinsic physical components. In the context of Physically Based Rendering, this means albedo color, normal, roughness and metallicity maps. These separated physical elements can subsequently be used in computer graphics tasks such as relighting, material editing and scene deconstruction.

The goal of Intrinsic Image Decomposition is to tackle the inverse problem of separating a single input image into its intrinsic physical components, namely the reflectance and the shading layer, such that the composition of these two layers result in the original image. Reflectance is usually represented by the GGX microfacet model ([10]) or the Disney BRDF ([11]). In the context of Physically Based Rendering, the latter typically encompasses albedo color, metallicity and roughness, as illustrated in figure 2.1. The shading layer includes geometry and illumination of the scene such as normals, light emitters and specular residual components ([12]). Computer graphics tasks such as material editing, object re-lighting and scene deconstruction crucially depend on this concept.

Decomposing an image into its intrinsics is an inherently underconstrained problem ([13]) as multiple unknown variables must be estimated from a single observation, and a multitude of material and lighting combinations can result in visually identical images ([14], [15]). To

mitigate this, traditional methods relied on highly limiting assumptions such as single-color illumination or Lambertian surfaces to find a unique solution ([16]). One novel approach to overcome these restrictions was to leverage human perceptual clues to constrain the solution space. More specifically, humans are more adept at judging ordinal relationships between pixel intensities and not as sensitive to absolute values, for instance the relativity between the intensity of the shading of two pixels ([17]). This idea has been further refined with color-aware decomposition, producing even higher fidelity ([18]).

Further fundamental challenge rises from the limited availability of ground-truth data used for training. Initial datasets only contained simple objects in controlled lighting environments. Recently, significant steps are taken in ground-truth data acquiring using crowdsourced approaches. These methods put importance on the aforementioned relativity of human perceptual clues on shading and reflectance as annotations ([19]). Still, the size and label density of these datasets is highly limited.

To solve the high ambiguity between material and lighting properties and the substantial lack of ground-truth data, contemporary methods employ probabilistic diffusion models such as Denoising Diffuse Probabilistic Model (DDPM) or more recently Flow Matching. Leveraging these pipelines, this ill-posed decomposition problem is transformed into a well-posed conditional generative sampling problem ([20] [21]). In these approaches the intrinsic factors of the image are optimized in a simultaneous and mutually dependent fashion, generating physically plausible material maps and reducing the ambiguity between their components. They allow for even more consistency with Physically Based Rendering standards by introducing further constraints on the material and lighting maps. More recent approaches treat image synthesis and decomposition as a unified invertible task, and result in diffusion models capable of composing an RGB image from its components as well as decomposing it. Enforcing this cycle consistency leads to more robust light estimation and reconstruction, thanks to the invertibility of the relationship between the image and its components ([22]). Inspired by this, the proposed pipeline leverages Score Distillation Sampling (SDS), which works with already available diffusion models pre-trained on the vast image data accessible from the public internet.

2.2. Texture Generation

The automatic generation of high-fidelity, semantically correct textures is a critical research area in computer graphics, having prominence in virtual reality, simulation, and digital entertainment. The fundamental challenge lies in generating textures possessing multi-view style and color consistency while still maintaining geometric coherence onto the complex 3D scene.

Traditional texture generation pipelines relied on procedural([23], [24]) or exemplar-based synthesis methods ([25], [26]). Procedural synthesis algorithms employ stochastic functions and noise primitives to simulate naturally occuring materials, but often lack photorealistic fidelity in their output, necessitating post-process parameter tuning. Conversely, exemplar-based synthesis leverage an initial source texture sample, which is subsequently reshuffled to

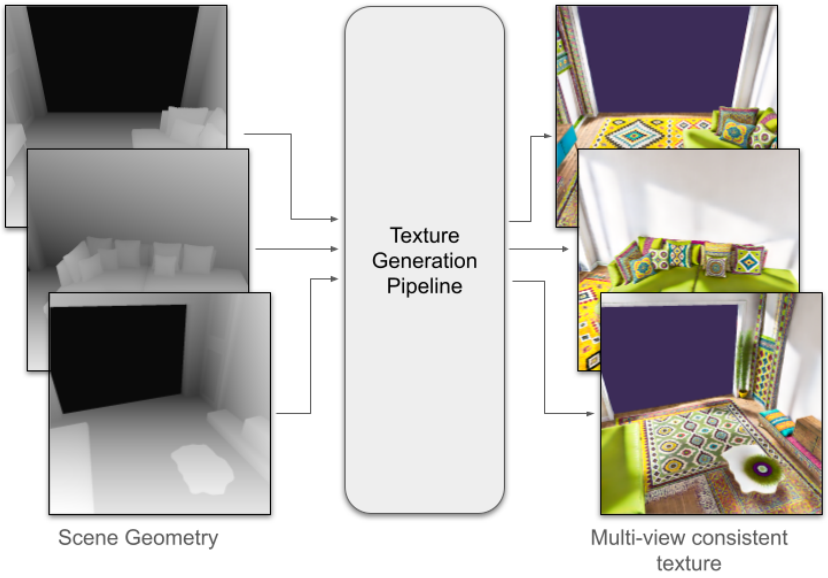


Figure 2.2.: **High-level overview of texture generation**. The goal of texture generation is to optimize texture maps that possess multi-view style and color consistency based on input geometric data.

generate perceptually similar outputs. It utilizes Markov Random Fields to generate pixels on a local basis, preserving local structural coherence. Unfortunately, it is a computationally intensive method and is prone to repetition artifacts.

Traditional methods were superseded by approaches leveraging Vision-Language Models (VLMs) such as Contrastive Language-Image Pre-Training (CLIP) ([27]). These systems exploit the ability of VLMs to embed both visual information and natural language into one shared latent space while keeping semantic correctness. The embeddings of this latent space are used as control signal to guide existing Generative Adversarial Network (GAN) backbones, such as StyleCLIP leveraging StyleGAN backbone ([28]) or VQGAN-CLIP ([29]). While the success of these approaches showed the capability of natural language as high-level control signal for visual synthesis tasks, they suffer from two crucial challenges inherited from the application of GANs. One fundamental problem is the highly unstable training due to the GAN backbone and the susceptibility to mode collapse ([30], [31]). Secondly, the reliance of these methods on localized convolutions raise the difficulty of capturing the full data distribution, resulting in in plausible local patches but inconsistent or incoherent global compositions. This renders VLMs suitable only for separate objects and not scalable to larger environments such as entire rooms ([32]).

To address these critical flaws, current state-of-the-art texture generation methods utilize 2D latent diffusion models as geometric priors. Research has branched to two separate primary strategies. The first approach uses the score of the frozen pretrained prior and backpropagate it into the underlying 3D representation through Score Distillations Sampling ([3]); either by inpainting directly into texture space ([7], [6]) or by supplying it to a score-distillation-based

objective function ([5], [33]). While these methods are capable of synthesizing both 3D geometry and texture, a well-known limitation of these methods is the reliance on lengthy multi-view rendering and optimization, where the 3D model has to be rendered from a given viewpoint, processed through the diffusion prior and updated using the score. The second strategy removes this issue by working directly on the 3D representation or 3D-aware latent space of the scene using structural guidance ([34], [35]). This enables synthesizing the texture in a single forward pass or in a limited number of refinement steps.

Despite these achievements, leveraging a pretrained 2D latent diffusion priors to optimize 3D representations leads to geometric inconsistencies such as the Janus problem, where gradients derived from different viewpoints can differ considerably on the same point of geometry. This introduces seaming artifacts across views where rendered views overlap or blend. Further topological challenges stem from the continuous surfaces appearing in indoor environments such as walls, which give rise to loop closure issues; meaning that the synthesized texture at the start of the loop does not seamlessly match itself at the end of it, resulting in visible discontinuity. To mitigate this, the proposed pipeline shifts the learning objective from latent-space to the image-space by working with an image-space loss function and image-space loss regularization detailed in section 4.2.1 and 4.3.

2.3. Inverse Rendering

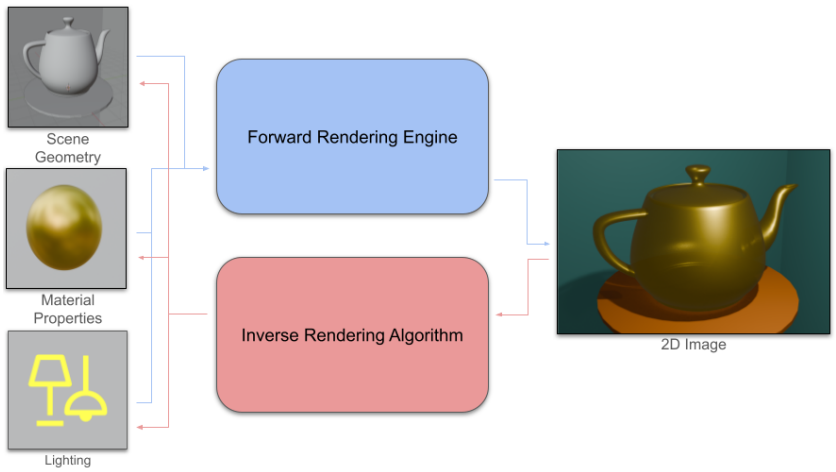


Figure 2.3.: **Comparison of forward versus inverse rendering**. The traditional forward rendering process synthesizes a novel image based on input scene geometry, material properties and lighting. Contrary to this, inverse rendering aims to estimate the underlying physical properties of a scene from rendered observations. Similarly to Intrinsic Image Decomposition, its goal is to disentangle the reflectance and shading layers of the image. However, it takes one step further in also separating the shading layer into respective geometric and illumination data.

Inverse rendering is the process of estimating underlying physical properties of a scene from one or more rendered 2D observations. It is the inverse of the forward rendering problem, where based on scene geometry, materials and lighting an image is synthesized, as illustrated in figure 2.3 ([36], [37]). It goes a step further than Intrinsic Image Decomposition by not only extracting the reflectance and shading layers from an image, but also disentangling the shading layer, breaking it down to geometric and lighting information respectively. Doing so allows not only relighting of the scene but geometrical transformations such as rotation. Along with novel view synthesis and photorealistic relighting, Inverse Rendering is a crucial part of scene understanding, enabling the creation of photo-realistic digital twins of real-world environments ([38]). The exact properties of the scene the inverse rendering process is estimates are the scene geometry, the Bidirectional Reflectance Distribution Functions (BRDFs) of the objects of the scene, the scene lighting and the intrinsic and extrinsic parameters of the capturing device ([39]).

Subsequently, Inverse Rendering is a highly ill-posed problem, since a single pixel intensity value serves as the basis for the estimation of a number of factors. Similarly to Intrinsic Image Decomposition detailed in section 2.1, Inverse Rendering suffers from the high ambiguity between physical properties of the scene, where a number of material and lighting combinations can result in identical pixel intensities. One way to lower this ambiguity is to employ constraints on the scene or the material models, for example prior knowledge on the specific illumination and light sources of the scene ([40]).

Neural inverse rendering aims to solve the problem of inverse rendering employing deep neural networks to approximate scene properties ([41]). This method often uses implicit scene representations, where instead of traditional representations like meshes and texture maps, the scene is represented by Multi-Layer Perceptrons (MLPs) implicitly encoding the properties of traditional representations, such as Neural Reflectance Decomposition utilizing Neural Radiance Fields to encompass the scene ([42]). To enable gradient-based optimization of the resulting 3D representation, Neural Inverse Rendering utilizes Differentiable Rendering, where the whole forward rendering process is fully differentiable. This allows for the rendering process to serve as computational graph between the MLP representation of the scene and a given rendered observation. Subsequently, an image-space reconstruction loss, often simple L2 norm, is backpropagated on this computational graph ([43]).

Another approach aiming to transform the inverse rendering problem well-posed is Multi-view Inverse Rendering, exploiting parallax and view-dependent effects. Multi-view Inverse Rendering utilizes captures from multiple different camera positions in the same scene, thereby decoupling geometry from appearance ([44]). Using multiple views significantly reduces the ambiguity between lighting and material, as surface reflections are view-dependent properties of the rendering, and different camera positions yield different reflections. It also enables separation of specular and diffuse components of the material by analyzing the change of intensity and color between views. While earlier implementations of Multi-view Inverse Rendering adopted explicit scene representations, newer approaches use combine multi-view constraints with implicit representations to achieve photorealistic material estimation ([45]).

Factorized Inverse Path Tracing (FIPT) aims to apply global illumination models to inverse

rendering, a significant challenge before. Instead of relying on simplified shading models or local shading assumptions, FIPT utilizes path tracing, which is a physically-based forward rendering method further detailed in section 3.2.1. To make the optimization tractable, FIPT factorizes the light transport used in path tracing to decompose it into terms dependent only on the specific scene parameters, enabling gradient flow. This separation also supports visual physical effects such as inter-reflections or soft shadows ([8]). Inverse Rendering of Indoor Scenes from Low Dynamic Range Images (IRIS) builds upon FIPT to apply inverse rendering specifically to real-world indoor environments captures using Low Dynamic Range (LDR) images. While LDR images are widely available, their limited information presents significant challenges for generation tasks such as saturation artifacts. To overcome this, IRIS combines geometric reconstruction, material and illumination estimation using domain-specific loss functions, designed to mitigate the effects of specular highlight clipping ([9]). The high physical plausibility however comes with increased computational costs ([8]). To avoid this, the proposed pipeline leverages a pre-trained diffusion model to estimate scene properties in an SDS framework, which is the core technique used by Multi-view Inverse Rendering methods for 3D optimization, yielding lower processing complexity ([46]).

2.4. Intrinsic Image Generation

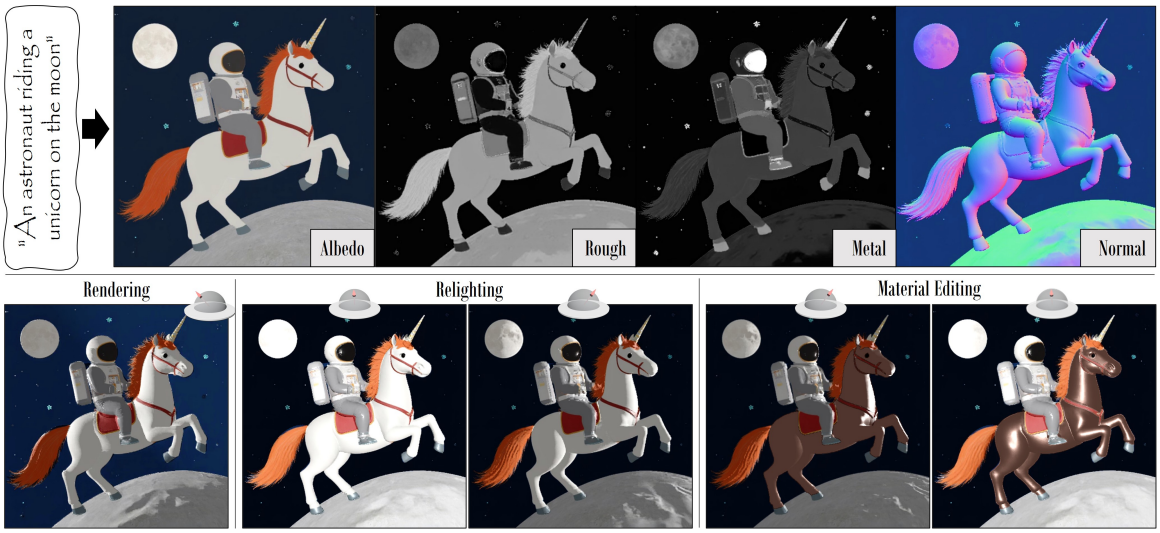


Figure 2.4.: **Results and subsequent graphics capabilities of an Intrinsic Image Generation pipeline ([47])**. The goal of Intrinsic Image Generation is the synthesis of novel PBR material maps from a singular textual input. In contrast to images possessing baked-in lighting generated by conventional text-to-image diffusion pipelines, an Intrinsic Image Generation framework produces disentangled texture maps for the individual PBR parameters. This allows for subsequent relighting and material editing of the scene in computer graphics programs.

Intrinsic Image Generation introduces a substantial paradigm shift in the synthesis of PBR materials compared to Intrinsic Image Decomposition. Inspired by conventional text-to-image diffusion models, this task utilizes a single text prompt. Contrary to traditional methods which optimize a singular output image with baked-in lighting, it leverages text conditioning to generate the individual PBR materials of a single image. These high-quality material maps are then ready for downstream 3D computer graphics applications to achieve relighting and editing the generated scene or further texture generation, as illustrated in figure 2.4.

The learning objective is to map a textual description to a set of spatially aligned feature maps, albedo, normals, roughness and metallic maps respectively, independent of a specific lighting condition. The pipeline achieves this by pre-training separate diffusion models for each PBR material component.

The primary challenge raised by this approach is the lack of geometric input data. In traditional image-conditioned tasks such as Intrinsic Image Decomposition, the geometric properties of the scene are explicitly provided by the image input. Relying solely on text-input disables this opportunity, and naive implementations can easily result in disjointed material maps describing semantically completely different scenarios still adhering to the same text prompt.

To mitigate this problem, recent approaches in Intrinsic Image Generation introduce innovative cross-attention formulations, such as Cross-Intrinsic Attention. Cross-Intrinsic Attention is a modification on the standard self-attention mechanism commonly seen in diffusion UNet architectures. This novel definition the key K and value V are concatenated and derived from the different map generation heads. Subsequently, during the generation of one modality, the generative prior attends to the structural features of another modality, enforcing a shared semantic layout and low-frequency structural alignment. To preserve high-frequency details, methods often leverage on a rendering-aware loss function. Latent-space optimization inherently decreases available quality thanks to the lower spatiality of its dimensions. To counter this, a rendering loss is introduced, utilizing the higher resolution image-space signals to constrain the optimization process which is crucial to achieve sharp details ([47]). A similar image-space learning objective is utilized in the proposed pipeline as well to retain high-frequency information, but generalized to 3D optimization.

3. Background

3.1. Image Formation Process using Rasterization

The aim of the image formation process in computer graphics is to produce an image by projecting 3D objects onto a 2D raster grid leveraging a series of coordinate transformations in a rendering framework. In each of these stages the geometric data is converted into a new coordinate space better suiting the subsequent rendering operation such as lighting, clipping or projection. The resulting 2D output of the image formation process is a raster image, suitable for display on a flat screen. This visualization step is leveraged by texture generation pipelines not only in the final rendering step to showcase the optimized texture maps, but during the optimization process to synthesize high-resolution input to a pre-trained diffusion model.

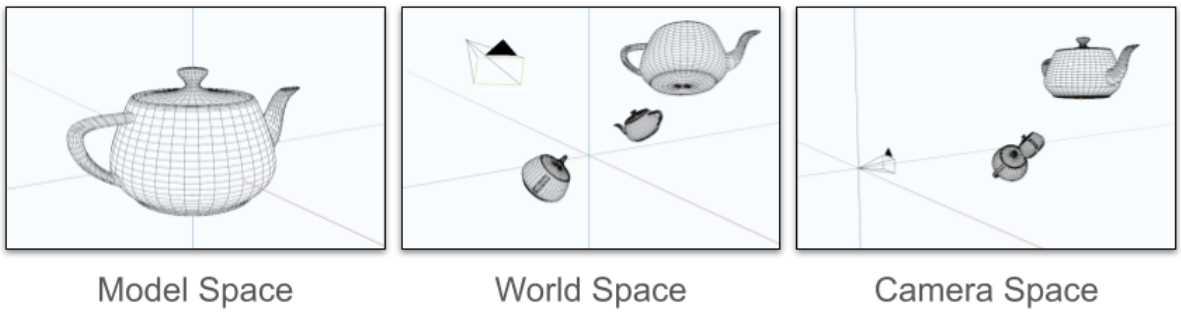


Figure 3.1.: **Coordinate systems of the image formation process**. The end result of the image formation process is a 2D image rasterized from an underlying 3D representation of the scene. First, the objects of the scene are each defined in their individual Model Space where the object is in the origin of the system, as illustrated on the left. The objects are subsequently transformed into the World Space using Model Transformation Matrix $\mathbf{M}_{model}$, which encompasses all objects of the scene set relative to a common origin as seen in the middle. Lastly, the scene is shifted to Camera Space leveraging $\mathbf{M}_{camera}$, where the rendering camera is placed into the origin of the coordinate system to render the scene from the perspective of a virtual observer.

The individual transformations are formalized as simple composition of homogeneous matrices $\mathbf{M} \in \mathbb{R}^{4\times4}$ called transformation matrix. Utilizing this representation, the process of the transformation is simplified to elementary matrix multiplications on the 3D object. The final 2D position P_{image} of a 3D point P_{local} in the scene can therefore be derived as follows

$$\mathbf{P}_{image} = \mathbf{M}_{viewport} \cdot \mathbf{M}_{clip} \cdot \mathbf{M}_{proj} \cdot \mathbf{M}_{camera} \cdot \mathbf{M}_{model} \cdot \mathbf{P}_{local} \tag{3.1}$$

On a conceptual level, a virtual object is designed as a series of 3D vertices represented as Euclidean vectors $\mathbf{v} \in \mathbb{R}^3$ defined as $(x, y, z)^T$. Similarly to the transformation matrices, the rendering framework operates on the homogeneous representations of these vectors. A Euclidean vector $\mathbf{v}$ is augmented into projective space $\mathbb{P}^3$ as $\mathbf{v}' = (x, y, z, w)^T$, where the homogeneous component w is initialized to 1 in standard geometry.

Initially, the virtual object is defined within a designated local coordinate frame called the Model Space, with the object commonly being the origin of the system, as illustrated in figure 3.1. The second stage of the process produces a global Cartesian coordinate frame called the World Space, encompassing all objects of the scene relative to a common origin. The transformation from Model Space to World Space utilizes the Modeling Transformation Matrix, denoted $\mathbf{M}_{model}$ in equation 3.1, typically combining affine transformations such as translation, rotation and non-uniform scaling.

Next, the scene is put into the perspective of the virtual observer by shifting the camera to the origin of the system, now called Camera Space. The alignment of the world axis with the local basis vectors of the camera is achieved by rigid body transformations rotation and translation composed in the Camera Transformation Matrix, indicated as $\mathbf{M}_{camera}$ in equation 3.1. The viewing direction is aligned with the standard axis, varying based on the used convention.

Perspective foreshortening is subsequently achieved using Projection Matrix $\mathbf{M}_{proj}$, which introduces non-linearity on the z-axis. The resulting space, called the Clip Space, encompasses vertices satisfying the condition $-w \leq x, y, z \leq w$, where w is now proportional to the original linear depth. This information is used to compute Normalized Device Coordinates commonly ranging in $[-1, 1]$, using Clipping Transformation Matrix $\mathbf{M}_{clip}$ to divide the coordinates x, y and z with homogeneous component w. Geometry outside the range of the convention is clipped.

In the final stage, the Normalized Device Coordinates are transformed into Screen Space using the Viewport Transformation Matrix $\mathbf{M}_{viewport}$. The Screen Space defines its point with respect to the actual viewport with discrete raster coordinates, taking the width and the height of the viewport into account. The geometric data of the Screen Space is subsequently used by the rasterizer to generate fragments and populate the framebuffer ([48]). The image rasterized from the setup illustrated in figure 3.1 is shown in figure 3.2.

3.1.1. Camera Extrinsic and Intrinsic Parameters

The extrinsic parameters of in the pinhole camera model define its pose within the 3D space, namely its position and orientation. They are mathematically denoted in a 4×4 homogeneous matrix $\mathbf{E}_{\text{cam}}$, which is composed of 3×3 rotation matrix $\mathbf{R}$ and 3×1 translation vector t, as follows

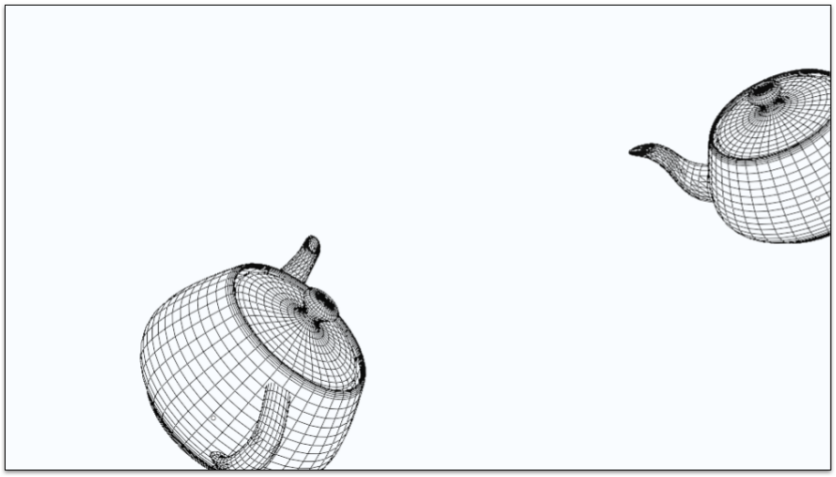


Figure 3.2.: **Final rasterized image in Screen Space**. A perspective projection using Clipping Transformation Matrix $\mathbf{M}_{clip}$ is employed on the scene in Camera Space, resulting in Normalized Device Coordinates commonly ranging between -1 and 1. All geometry outside this range is clipped. The final image coordinates in Screen Space are computed using the Viewport Transformation Matrix $\mathbf{M}_{viewport}$, which takes the height and the width of the viewport to retrieve raster coordinates.

$$\mathbf{E}_{\text{cam}} = \begin{bmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \mathbf{R}_{1,3} & t_1 \\ \mathbf{R}_{2,1} & \mathbf{R}_{2,2} & \mathbf{R}_{2,3} & t_2 \\ \mathbf{R}_{3,1} & \mathbf{R}_{3,2} & \mathbf{R}_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Subsequently, the intrinsic parameters of the pinhole camera model characterize its internal projective properties within the pinhole camera model. These parameters encode geometric attributes of the camera sensor, specifically focal lengths f_x and f_y defining scaling factors between physical units and pixels, principal point offsets c_x and c_y specifying the intersection of the optical axis with the image plane; and skew coefficient s, accounting for non-orthogonality between sensor axes. The intrinsic parameters are mathematically expressed using a homogeneous matrix $\mathbf{I}_{\text{cam}}$ as

$$\mathbf{I}_{\text{cam}} = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In the image formation process, the extrinsic parameter matrix $\mathbf{E}_{\text{cam}}$ essentially fills the role of Camera Transformation Matrix $\mathbf{M}_{camera}$, acting as a rigid body transformation aligning the global scene with the viewpoint of the camera, utilizing the camera pose for translation and orientation for rotation of the scene. Similarly, the intrinsic parameters of the camera

are leveraged during perspective projection onto the normalized image plane in Projection Matrix $\mathbf{M}_{proj}$.

Accurately determining both the extrinsic and intrinsic parameters of the camera thereby is a crucial task in quantitative computer vision tasks to ensure mathematical consistency of back-projected rays. Errors in the extrinsic data lead to significant misalignment of multi-view correspondences and drifts in trajectory tracking. Similarly, even minor deviations in focal length or principal point can introduce artifacts like systematic radial or tangential distortions and depth biases ([49], [50]).

3.2. Physically Based Rendering

The inherent problem of rasterization-based image formation is its local independency. The projection stage discards geometric relationships, meaning to determine a color of the pixel the fragment shader only possesses data regarding the currently processed geometric surface and fixed light sources, and no knowledge of other geometry in the scene. This makes direct computing the effects of Global Illumination such as soft shadows, inter-reflections and ambient occlusion impossible, leaving multi-pass approximation techniques as only solutions like shadow mapping or Screen Space Ambient Occlusion. Naturally, such approximations introduce unwanted artifacts and aliasing in the rendered result, breaking the illusion of photorealism. The aim of Phisically Based Rendering (PBR) is to tackle these issues and achieve photorealistic rendering results based on optical rules.

To compute plausible visual appearance, PBR solves the Rendering Equation. This equation leverages on Bidirectional Reflectance Functions (BRDFs) to define the total outgoing radiance from a given surface point $\mathbf{x}$ in direction ω_o. This effectively unifies direct and global illumination into a single statement, as both light directly hitting the surface and light bouncing between surfaces are covered. The equation can be mathematically formulated as follows

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)\, d\omega_i \tag{3.2}$$

It states that the total radiance L_o leaving the surface point $\mathbf{x}$ in outgoing direction ω_o is the sum of the self-emitted light $L_e(\mathbf{p}, \omega_o)$ at $\mathbf{x}$ and the light reflected from the surface, as illustrated in figure 3.3. The reflected light is computed as the integral of the incident radiance $L_i(\mathbf{x}, \omega_i)$ from incoming direction ω_i on the total surrounding hemisphere Ω. This incoming contribution is modulated by the BRDF f_r defining material reflectivity, and a geometric attenuation term $\mathbf{n} \cdot \omega_i$ inflecting light arriving at shallow angles. As the equation employs normalized vectors, $\mathbf{n} \cdot \omega_i$ is effectively the cosinus of the angle between the surface normal $\mathbf{n}$ and the incoming light direction ω_i ([51]).

In PBR, BRDFs are used characterize the material properties of a surface. The BRDF specifies the directional variations of opaque surfaces in reflectance and emissivitiy, namely the ratio of radiance exiting the surface element in a set outgoing direction (ω_o) to the incident radiance from a given incoming direction (ω_i). It encapsulates both the behavior

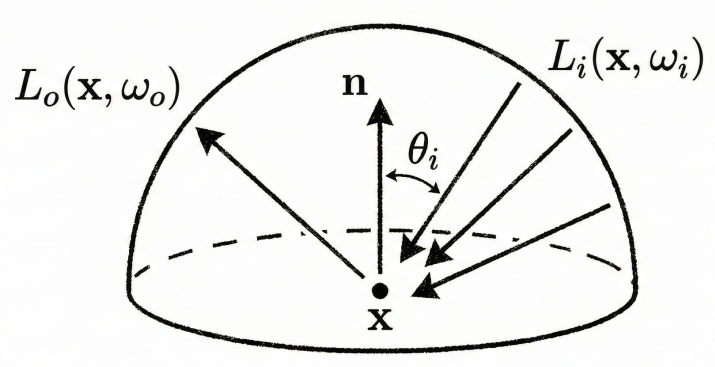


Figure 3.3.: **Physically Based Rendering Equation**. The rendering equation defines the outgoing radiance L_o leaving surface point $\mathbf{x}$ in direction ω_o as the sum of the light emitted by the surface and the reflected incident radiance $L_i(\mathbf{x}, \omega_i)$. The reflected light is characterized by the surface BRDF and the angle between surface normal $\mathbf{n}$ and incoming light direction ω_i.

of diffuse and specular reflectance([52]). Each BRDF follows two important properties, the Helmholtz reciprocity principle and the energy conservation principle. The former states that changing up the incoming and outgoing directions should not change the value of the BRDF ($f_r(\omega_i, \omega_o) = f_r(\omega_o, \omega_i)$); while the former expresses that the reflected energy must be less than or equal to the incident energy ([53]). The Microfacet Theory is one such BRDF, stating that any surface point is composed of statistically distributed microscopic mirrors, whose alignment determines whether the surface acts more diffuse or specular ([54])

In the rendering pipeline, separate texture maps are used to store the input variables of BRDF, allowing for spatially varying physical properties in surfaces. The four primary maps in industry-standard workflows are albedo, normal map, roughness map and metallic map.

Albedo defines the base color of the surface, representing the fraction of incoming light scattered diffusely. It does not contain soft shadows, directional lighting or ambient occlusion. Normal maps enrich the surface with high-frequency details avoiding the increase in geometric complexity for more detailed shading. It stores local surface orientation used to perturbate the initial surface normal vector $\mathbf{n}$. The roughness map encodes variance of microscopic surface slopes, characterizing the ratio of diffuse versus specular scattering from the surface. It governs the tightness and intensity of specular highlights on the surface. Similarly, the metallic map describes the degree conductivity versus dielectric behavior of the material. The primary difference between these two material types is that conductor materials have suppressed diffuse reflection and therefore the albedo is used to define specular color, while insulator materials have diffuse colors and achromatic white specular colors.

3.2.1. Path Tracing

Path Tracing solves the Rendering Equation with a Monte Carlo approximation of the rendering integral to compute realistic radiance values for the points of 3D scene. Contrary to

rasterization-based rendering where a 3D geometric points are projected into Screen Space to find out which pixel they accommodate, Path Tracing progresses in the opposite direction in that it traces rays shot from screen coordinates into the 3D scene to follow the path the light takes as it reaches into the camera. Effectively, from each pixel of the screen a ray is shot, and the Rendering Equation is computed at the first point of intersection with scene geometry. Subsequently, the path of the shot ray is traced along reflections from the surface point to achieve Global Illumination.

However, as illustrated in equation 3.2, solving the Rendering Equation involves an integral over an entire hemisphere to calculate the light incoming at that surface point, making it analytically unsolvable for general scenes. To address this, Path Tracing applies Monte Carlo integration, a statistical estimation of integration through repeated random sampling, thereby replacing the integral with a finite sum. Mathematically, the outgoing radiance $L_o(\mathbf{x}, \omega_o)$ is estimated by averaging N random samples as follows

$$L_o(\mathbf{x}, \omega_o) \approx L_e(\mathbf{x}, \omega_o) + \frac{1}{N} \sum_{k=1}^{N} \frac{f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\mathbf{n} \cdot \omega_i)}{p(\omega_i)}$$

where $p(\omega_i)$ represents the Probability Density Function (PDF). It describes the probability of a specific ray direction being chosen for sampling incoming radiance. This division ensure that the estimate remains unbiased, meaning that as number of samples N approaches infinity, the error of the approximation approaches zero ([55]).

With this approximation set, the path tracer execution follows a recursive logic. Firstly, a ray is projected from each pixel of the screen. At first intersection with a scene object, the path tracer explicitly checks for light sources visible from the hit surface point to compute direct lighting. Subsequently, a new ray direction is chosen based on the BRDF of the material, and the ray is followed in this direction until the next intersection, leveraging that the incoming radiance $L_i(\mathbf{x}, \omega_i)$ is just the outgoing radiance $L_o(\hat{\mathbf{x}}, \omega_o)$ of the next hit surface point $\hat{\mathbf{x}}$. The ray is followed until a termination probability q calculated at the point of intersection is missed, a technique called Russian Roulette. After termination, a new ray is shot from each of the pixels with slight stochastic perturbations in origins and directions and the algorithm recursively repeats.

However, employing Monte Carlo integration introduces a fundamental artifact in the results of Path Tracing, namely a high-frequency noise visible as grain in the image. As the estimator is probabilistic, some pixels lose incoming radiance which was simply not sampled. To counter this, modern Path Tracing solutions heavily utilize Importance Sampling, which samples ray directions with bias to important features like lights or BRDF peaks. Still, unfortunately this error makes Path Tracing an unreliable input source for texture generation pipelines, as the noisy rendering produces noisy gradients in the backpropagation process of the differentiable renderer. This leaves rasterization-based rendering the only viable solution for such tasks.

3.3. Diffusion Models

Diffusion Models represent a class of diverse generative methods inside Deep Generative Models inspired by non-equilibrium thermodynamics. Unlike Generative Adversarial Networks (GANs) they do not suffer from training instability and mode collapse, yielding results of higher quality and much more stable training objectives. Early diffusion models such as Denoising Diffusion Probabilistic Models (DDPMs) operated directly in the high-dimensional image-space. Contemporary approaches however, known as Latent Diffusion Models, leverage Variational Autoencoders (VAEs) to transform the domain of the problem to a compressed latent-space, significantly speeding up and further stabilizing the optimization routine by reducing computational complexity. Diffusion Model frameworks consist of two complementary processes, namely the fixed forward diffusion process that destroys information and the learned reverse diffusion process that reconstructs it ([56]).

The forward process, also called the diffusion phase, gradually degrades the input data distribution into complete noise by injecting a controlled-level Gaussian noise over a fixed sequence of time steps. This is formalized as a Markovian transformation process, where a state in step t depends exclusively on the state at step $t-1$. Injecting the noise can be denoted as sampling a simple Gaussian distribution as follows:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

where x_t is the noised sample in time step t, $\beta_t \in (0,1)$ represents the variance in time step t and $\mathbf{I}$ is the identity matrix. Consequently, $\beta_t\mathbf{I}$ is the added isotropic Gaussian noise. As t approaches infinity, the data distribution x_t converges to an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

A critical property of this Gaussian formulation is that at any given time step t, x_t can be expressed directly in terms of the original input data x_0. Using the reparameterization trick and defining $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, the marginal distribution at any timestep can be derived as

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

This allows for the direct expression of x_t as a linear combination of the signal and noise as follows

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ([57]).

In the reverse process, also called the denoising phase, the model learns to generate denoised samples x_0 from pure Gaussian noise. Ideally, the aim would be to learn to reverse the forward Markov chain step-by-step and recover the original data sample. This true reverse transition, denoted as $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$, is however intractable.

Applying Bayes's theorem on $p(\mathbf{x}_{t-1}|\mathbf{x}_t)$ yields

$$p(x_{t-1}|x_t) = \frac{q(x_t|x_{t-1})q(x_{t-1})}{q(x_t)}$$

where both the transition $q(x_t|x_{t-1})$ and marginal distribution $q(x_t)$ are known, but computing marginal distribution $q(x_{t-1})$ would require integration over the entire input space, making solving true posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ intractable for complex high-dimensional data.

To resolve this, DDPMs condition the reverse transition on the ground truth x_0, yielding conditional distribution $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$. Leveraging the property that the forward process is a fixed Gaussian Markov chain, this conditional distribution can be derived using Bayes' theorem, resulting in a closed-form mathematical formula for the mean $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ and covariance $\tilde{\boldsymbol{\Sigma}}_t$ of the distribution. Diffusion Models therefore learn to approximate the true reverse distribution by predicting the Gaussian noise $\boldsymbol{\epsilon}$ that was added to x_0 at step t, effectively learning the reverse transition as a Gaussian:

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

This simplifies the training objective to a Mean Squared Error (MSE) loss between the noise predicted by the model $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and the actual noise added $\boldsymbol{\epsilon}$ ([58]):

$$L_t = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2 \right]$$

In the context of diffusion-based texture generation, timestep t semantically defines the magnitude of detail the model is actively optimizing. Given a large t, the model assumes a significant amount of noise has been added to the input data, effectively hiding smaller details. Consequently, the model focuses on large-scale semantic structures, such as defining a larger region of the scene geometry as part of a furniture. On the other hand, low values of t mean the diffusion process has not applied too much noise to the input yet. In such a state the model refines high-frequency details, such as the grain texture on wooden surfaces.

3.3.1. Denoising Diffusion Implicit Models

The reliance on the Markovian formulation of the forward diffusion process is a recognized bottleneck as it necessitates thousands of sequential steps to be reversed accurately. Denoising Diffusion Implicit Models (DDIMs) in contrast reformulate the forward process as a non-Markovian inference scheme by utilizing the joint distribution of the entire trajectory. This means that DDIMs are able to use the same noise predictors as DDPMs but with a fundamentally different, deterministic sampling mechanism, leading to significantly fewer steps to produce results of matching quality.

The DDIM framework defines a family of inference distributions sharing the same marginal distributions $q(\mathbf{x}_t|\mathbf{x}_0)$ as DDPMs, but introducing a stochasticity parameter σ_t. Setting $\sigma > 0$ leaves the stochastic nature of DDPMs, making DDIMs a generalization of the DDPM denoising step. Setting $\sigma_t = 0$ transforms the reverse process into an Ordinary Differential Equation, the denoising process becomes fully deterministic, the sampling trajectory a fixed path uniquely determined by the initial latent noise. As the trajectory now has significantly lower curvature, the integration approximation is allowed to take larger step sizes, allowing for skipping steps and producing high-quality output in a fragment of the steps required in

DDPM. Furthermore, since the trajectory is uniquely defined by the initial noise, denoising the same noise always results in the exact same image. This property enables smooth interpolation between the outputs of the model, such as simulating a moving picture by smoothly interpolating between two noise vectors as model input. Similarly, the denoising process can be reversed to find the exact noise map a set output image was generated from ([59]).

3.3.2. Flow Matching

Traditional diffusion models aim to learn estimating a score function, the noise, derived from Stochastic Differential Equations or discrete Markov chains. Contrary to this, Flow Matching generalizes the diffusion process by shifting the learning objective to learn a Vector Field instead, the velocity, based on Continuous Normalizing Flows. This allows for the construction of straight, deterministic probability paths between the noise and the data in contrast to the Gaussian paths of curved trajectories in high-dimensional space enforced by standard diffusion models. The more simple probability paths lead to significantly more efficient sampling, making Flow Matching the state-of-the-art diffusion paradigm.

In Flow Matching, the generation process is a time-dependent flow of physically moving the probability mass from a source configuration, the noise, to a target configuration, the data. This movement is described by a probability density path $p_t(x)$ where $t \in [0,1]$. $t = 0$ means the distribution is pure noise, while $t = 1$ means the distribution is the data; and is defined using an Ordinary Differential Equation. Following fluid dynamics, the movement of a sample x is computed using the Vector Field $v_t(x)$ encompassing the velocity of the particle. Mathematically, the motion is described as

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \tag{3.3}$$

where $\phi_t(x)$ is the trajectory of sample x.

Furthermore, the Transport Equation enforces that the probability mass is conserved, as follows

$$\frac{\partial p_t}{\partial t} + \nabla \cdot (p_t v_t) = 0$$

The task is now to learn the Vector Field v_t, in order to generate data by simply sampling pure noise x_0 and integrating the differential equation 3.3 from $t = 0$ to $t = 1$ using a traditional solver. Unfortunately, to learn the true Vector Field v_t, ground-truth data is needed on the intermediate steps of the differential equation 3.3, which is unavailable. To counter this, Conditional Flow Matching shifts the focus from learning the transport of the entire input noise to data to learning the motion of a single noise sample to a single data sample.

This new probability path $p_t(x|z)$ is conditioned on a specific data point z. Common implementations make constructing the Vector Field trivial as they move the sampled point x_0 to x_1 on a straight line, making the Vector Field the simple derivative of the position with respect to time as follows

$$u_t(x|x_1) = x_1 - x_0$$

The new conditional objective function on a single noise data pair can be written as

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t\sim[0,1],x_1\sim p_{data},x_0\sim p_{prior}}\left[||v_\theta(t,x_t) - u_t(x_t|x_1)||^2\right]$$

where $t \sim [0,1]$ is the time uniformly sampled, $x_1 \sim p_{data}$ is a data point sampled from the empirical distribution of the training dataset and $x_0 \sim p_{prior}$ is noise sampled from a standard Gaussian distribution. Lipman et al. explicitly proved that learning to match these simple conditional vector fields and averaging them is equivalent to learning the complex global vector field, thereby regressing the model to learn the Marginal Vector Field which correctly pushes the entire noise distribution to the data distribution.

As previously mentioned, Flow Matching allows for a straight probability path between the sampled noise and the denoised data. This enables the differential equation solver to take bigger leaps in the process, leading to the possibility of generating high-quality images in a single Euler step ([60], [61]).

4. Methodology

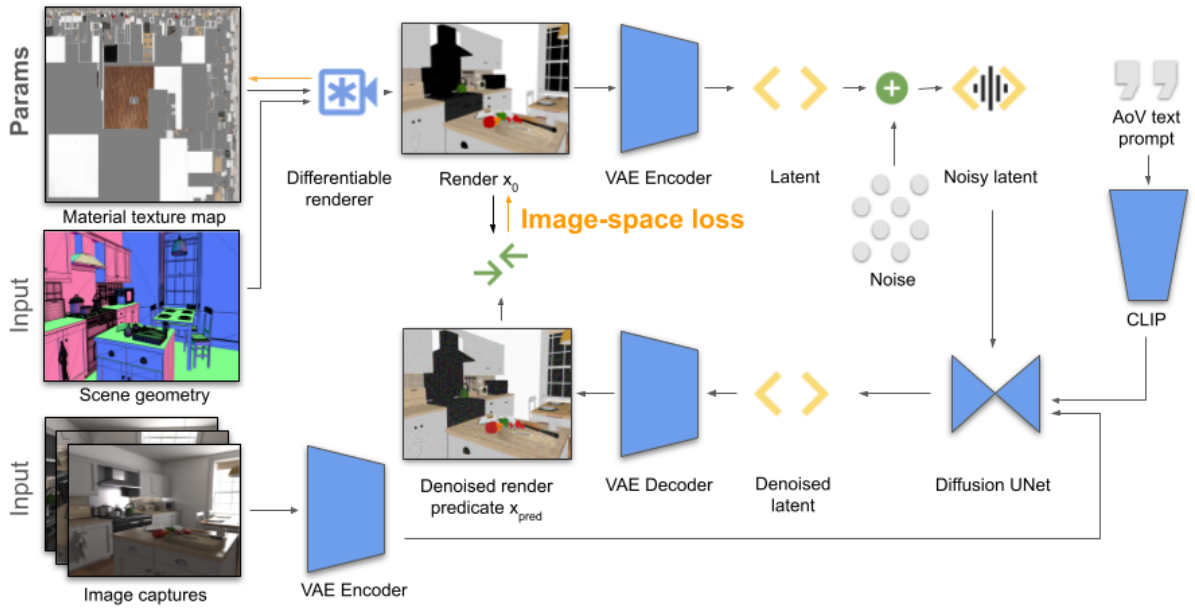


Figure 4.1.: **Complete training pipeline**. The pipeline takes two sets of inputs, the scanned geometry of the scene and the image captures of the scene used for conditioning. The material texture map are the parameters optimized during training. The pipeline initializes a texture map for the target rendering parameter and uses a differentiable renderer to render the scene from a set camera poses, yielding render x_0. The render is fed to a VAE encoder and Gaussian noise is applied to the resulting latent representation. The desired texture parameter is denoted by a text prompt internally and Contrastive Language-Image Pre-training (CLIP) is used to retrieve a corresponding text embedding from the prompt. Provided the noisy latent, the text embedding and the relevant image capture, a Diffusion UNet is used to predict a completely denoised sample. This prediction is forwarded through the decoder part of the VAE, resulting in denoised render predicate x_{pred}. An image-space loss is calculated between x_0 and x_{pred}, and backpropagated to update the target texture representation.

This chapter details the overall system architecture proposed for multi-view consistent synthesis of Physically Based Rendering (PBR) materials. Firstly a high-level overview of the pipeline is presented, establishing foundational structure and assumptions, followed by in-depth analysis of the individual processes and modules. The primary objective is the

generation of physically plausible, multi-view consistent 3D texture maps for PBR rendering parameters, namely albedo, normals, roughness, metallic and irradiance maps, leveraging scene geometry and the related image captures of the scene as conditioning inputs.

To follow current state-of-the-art texture generation techniques, the proposed approach adapts principles from Score Distillation Sampling (SDS). At a conceptual level, the iterative optimization process begins with the scene geometry initialized with a preliminary texture map. The pipeline utilizes a differentiable renderer to project the scene from a sampled camera pose, yielding the initial rendered image x_0. This rendering is subsequently encoded into latent-space representation and perturbed with a controlled level of noise, resulting in noisy latent x_{noisy}. Concurrently, a text embedding c_{text} is generated from the text prompt denoting the desired rendering parameter (e.g. albedo, roughness, ...) with Contrastive Language-Image Pre-training. Next, both noisy latent x_{noisy}, text embedding c_{text} and the corresponding conditioning image capture c_{image} of the scene are jointly passed to frozen pre-trained diffusion model. The diffusion model predicts the underlying denoised rendering, which is then decoded back to its image-space representation as the denoised estimate x_{pred}. Distinct to pure gradient-based SDS, the proposed pipeline employs an image-space reconstruction loss, calculated between the initial rendering x_0 and the predicted denoised estimate x_{pred}. This loss is backpropagated through the differentiable renderer to update the underlying 3D texture parameterizations. This process is repeated across a set of diverse camera poses using the updated texture maps, yielding an iterative and view-consistent optimization of the target material properties, as shown in figure 4.1.

4.1. Assumptions

The pipeline operates on specific geometric and photometric prerequisites. The scene geometry is ingested either as a collection of the individual mesh objects of the scene or as a monolithic, already conjoined scene mesh. In both cases supplied UV-mapping of the meshes is assumed. Furthermore, the former case mandates the inclusion of the scene configuration, detailing the spatial transformations of the scene objects.

A corresponding set of photometric captures of the scene is a further required input. Crucially, these images should provide comprehensive coverage of the entire scene and must be taken using a consistent sensor setup; namely the same camera or cameras of the same intrinsic parameters and operating under invariant lighting conditions. The pipeline does not do camera calibration, meaning it is presumed that the complete camera configuration is provided, detailing the intrinsic parameters of the camera and the extrinsic parameters corresponding to the respective image captures. Furthermore, it is assumed that this camera configuration follows one of the established coordinate system conventions, the specification of which must be supplied to the pipeline to ensure the generation of plausible results.

4.2. Score Distillation Sampling

Diffusion Models are powerful tools for 2D image synthesis, but not directly applicable to 3D generation task, due to the cubic increase in data dimensionality compared to 2D images leading to exponential growth in computational requirements as well as the sparsity of 3D training datasets ([62]). Score Distillation Sampling ([3] was introduced to resolve this problem by utilizing a pre-trained, frozen 2D Diffusion Model as a critique to optimize a separate trainable model.

The Diffusion Model functions as a score estimator for a data distribution, more specifically the perturbed distribution $q(\mathbf{x}_t|c)$ of noisy samples conditioned on c. In standard SDS, the gradient is derived not from a scalar image-space loss function, but from the Score Matching Principle, making it the expected square of the difference between the noise predicted by the model and the noise required to make the rendered image more likely under the Diffusion model's learned distribution. In effect, SDS optimizes the trainable set of 3D parameters θ such that the rendered image would look more like what the pre-trained Diffusion Model expects. The canonical SDS gradient update can be mathematically described as follows:

$$\nabla_{\boldsymbol{\theta}} L_{\text{SDS}} = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[w(t) \cdot (\boldsymbol{\epsilon}_{\phi}(\mathbf{x}_t, t, c) - \boldsymbol{\epsilon}) \cdot \nabla_{\boldsymbol{\theta}} \mathbf{x} \right]$$

where $w(t)$ is a weighting function for prioritizing set time steps, $\boldsymbol{\epsilon}_{\phi}(\mathbf{x}_t, t, c)$ is the noise predicted by the Diffusion Model, $\mathbf{x}_t$ is the rendered image after with injected noise vector $\boldsymbol{\epsilon}$, c is the condition guiding the generation, typically text embedding derived from the input text prompt and $\nabla_{\boldsymbol{\theta}} \mathbf{x}$ is the gradient of the rendered image with respect to trainable parameters θ. The pre-trained nature of the Diffusion Model is propagated by this last gradient component, giving direction to the derived SDS-loss ([3]).

4.2.1. Image-Space Loss

As previously detailed, standard SDS-based 3D generation approaches employ latent-space loss. Latent-space offers faster training due to its lower dimensionality compared to image-space, exponentially reducing the computational intensity of the Diffusion Model's noise prediction step. Inspired by other state-of-the-art texture generation methods ([63], [64]) however, the proposed pipeline exploits an explicit image-space MSE-loss to correct over-saturation artifacts and improve high-frequency local details. The loss is calculated between the pixels of the original rendering x_0 and the denoised estimate x_{pred}, as shown in figure 4.1. The denoised estimate x_{pred} is still predicted using a latent diffusion model, but subsequently decoded into its image-space representation. Utilizing this custom image-space objective has been observed to provide higher quality results, as it mitigates the variance often observed in pure score-based updates.

The proposed image-space objective is formulated as

$$L_{\text{MSE}} = \frac{1}{H \cdot W \cdot C} \sum_{h,w,c} \| \mathbf{x}_0[h,w,c] - \mathbf{x}_{\text{pred}}[h,w,c] \|_2^2$$

where h, w and c are image height, width and channel running until H, W and C respectively, $\mathbf{x}_0$ is the current differentiable rendering and $\mathbf{x}_{\text{pred}}[h, w, c]$ the denoised rendering estimation provided by the diffusion backbone.

4.3. Loss Regularization

To avoid overfitting and thereby ensuring good generalization on unseen problem domain, contemporary generative pipelines employ robust regularization strategies. The proposed pipeline utilizes regularization mechanisms inspired by standard methods but customized to fit domain-specific requirements.

4.3.1. Visibility Weighting

An inherent issue in optimization-based texturing is the biased observation frequency. The number of times a point in scene geometry is seen from different camera views can vary to a great scale as scene regions occlude each other. Consequently, geometry observed from multiple camera poses accumulates excessive gradient updates, while blocked areas remain under-optimized. To mitigate this, the proposed pipeline employs a Visibility Accumulation Map $\mathcal{V}$, similar to the target 3D texture maps, as illustrated in figure 4.2. The Visibility Accumulation Map $\mathcal{V}$ is subsequently used in a weighting mechanism, improving visual consistency across the entire scene representation.

During a pre-generation phase, the entire scene is rasterized from all available camera poses, populating $\mathcal{V}$ with the intersection count for every texel. As such, a texel value is simply the number of times the corresponding geometric point has been observed. During optimization, $\mathcal{V}$ is sampled to calculate a weighting applied to the loss as follows

$$W(\mathbf{p}) = \text{clip}\left(\frac{1}{2 \cdot \text{clip}(\mathcal{V}(\mathbf{p}), 1, 6)}, 0.1, 10\right)$$

where $\text{clip}(x, y, z)$ constrains x to the interval $[y, z]$ for numerical stability.

The final weighted loss gradient $\nabla_{\boldsymbol{\theta}} L^{weighted}$ is then calculated as follows

$$\nabla_{\boldsymbol{\theta}} L^{\text{weighted}} = \mathbb{E}\left[W \odot \left(w(t) \cdot (\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, t, c) - \boldsymbol{\epsilon}) \cdot \nabla_{\boldsymbol{\theta}} \mathbf{x}\right)\right]$$

where $\odot$ denotes element-wise multiplication, permitted by the introduced image-space loss detailed in section 4.2.1.

Utilizing this inverse weighting scheme balances the effect the individual geometric points have on the loss, as the higher weight stemming from lower observation count increases the magnitude of the received gradient. Prioritizing rarely seen geometric features therefore leads to more plausible samples and higher consistency across the entire 3D model.

4.3.2. Stochastic Regularization via Image-Space Dropout

The goal of this technique is to introduce stochasticity into the optimization objective to further enhance generalization capabilities. Inspired by path-tracing used in PBR rendering

Figure 4.2.: **Frequency texture and example renderings of view counts**. As pre-training phase of the pipeline, the scene is rendered from all camera poses. The frequency texture is a map of the observation frequencies of the individual geometric points. It is used to weight the loss in an inverse fashion,w granting higher effect to geometric points seen from less camera poses. On the left the texture map containing the view counts itself can be seen, next to it a number of sample views utilizing this texture map. Brighter color means higher view count

pipelines where rays are sampled randomly from the image plane, the Image-Space Dropout aims to mimic the behavior of a ray initialized in a way that results in no intersection with light sources. Since the pipeline utilizes a rasterization-based rendering framework, such sampling can only be emulated by a regularization term on the Image-Space objective function.

The core idea is to mask the photometric loss for a random subset of pixels during the backward pass, to emulate the described effect of path-tracing. A random binary mask $\mathbf{M} \in \{0,1\}^{H \times W}$ can be generated by sampling its elements from a Bernoulli distribution:

$$M_{i,j} \sim \text{Bernoulli}(p_{keep})$$

where p_{keep} is the retention probability. Leveraging mask $\mathbf{M}$, the regularized loss $\mathcal{L}_{regularized}$ is computed as

$$\mathcal{L}_{regularized} = \sum_{i,j} (M_{i,j} \cdot \hat{\mathcal{L}}_{pixel}^{(i,j)})$$

where $\hat{\mathcal{L}}_{pixel}^{(i,j)}$ is the initial per-pixel loss.

4.4. Multi-Modal Classifier-Free Guidance

Classifier-Free Guidance (CFG) enables conditional generation guided by diverse modalities such as text prompts or images, without the need of a separately trained Classifier in contrast to Classifier Guidance, which introduced instability in the diffusion process and created a trade-off between sample quality and diversity. Instead, Classifier-Free Guidance trains jointly

trains the noise-prediction model for both conditional and unconditional sampling. The core concept is that during reverse diffusion the model originates from the unconditional and extrapolates towards the conditional prediction, using the guidance scale $w \geq 1$ as step size, enhancing the influence of the condition on the output. The standard CFG output can be calculated as

$$\tilde{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \varnothing) + w \cdot (\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c) - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \varnothing)) \tag{4.1}$$

where $\tilde{\boldsymbol{\epsilon}}$ is the final noise estimate, $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \varnothing)$ is the unconditional prediction and $(\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, c)$ is the conditional prediction. ([65]).

As the proposed pipeline is dually conditioned on both image data and text embeddings, the implemented Classifier-Free Guidance is extended to follow multiple modalities. The final noise estimate linearly combines the unconditional score $\boldsymbol{\epsilon}_{\text{uncond}}$, text conditioned score $\boldsymbol{\epsilon}_{\text{text}}$ and the image conditioned score $\boldsymbol{\epsilon}_{\text{image}}$. The final noise prediction is formulated as follows

$$\boldsymbol{\epsilon}_{\text{pred}} = \boldsymbol{\epsilon}_{\text{uncond}} + w_t(\boldsymbol{\epsilon}_{\text{text}} - \boldsymbol{\epsilon}_{\text{image}}) + w_i(\boldsymbol{\epsilon}_{\text{image}} - \boldsymbol{\epsilon}_{\text{uncond}}) \tag{4.2}$$

where w_t and w_i denote the guidance scales for text and image modalities respectively.

4.5. Differentiable rendering

The goal of differentiable rendering is to allow the computation of gradients from the 2D rendering output back to the 3D input set, bridging the discrete domain of mesh rasterization and the continuous domain of neural optimization. Since traditional rendering is non-differentiable due to discrete operations such as occlusion, differential rendering utilizes continuous or soft approximations such as soft rasterization ([66]).This allows the direct backpropagation of image-space loss L_{MSE} into the UV texture space.

4.6. Variational Autoencoder

As mentioned in section 3.3, modern diffusion frameworks encorporate Variational Autoencoders (VAEs) to further facilitate efficiency, as diffusion models operating directly in image-space are computationally prohibitive. VAEs are inspired by standard Autoencoders (AEs), and their two main components are the encoder and the decoder, both implemented as neural networks as illustrated in figure 4.3.

However, VAEs extend on AEs by combining its concept with Bayesian inference and probabilistic graphical models, to learn instead a probabilistic mapping between data and its compressed representation. Their core idea is that input data x is generated from latent variable z governed by a prior distribution $p(z)$.

The encoder is designed to approximate the intractable true posterior $p(\mathbf{z}|\mathbf{x})$ by a tractable variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$. The output of the encoder is the mean $\boldsymbol{\mu}_\phi$ and variance $\boldsymbol{\sigma}^2_\phi$ of this multivariate Gaussian distribution:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\Sigma}_\phi(\mathbf{x}))$$

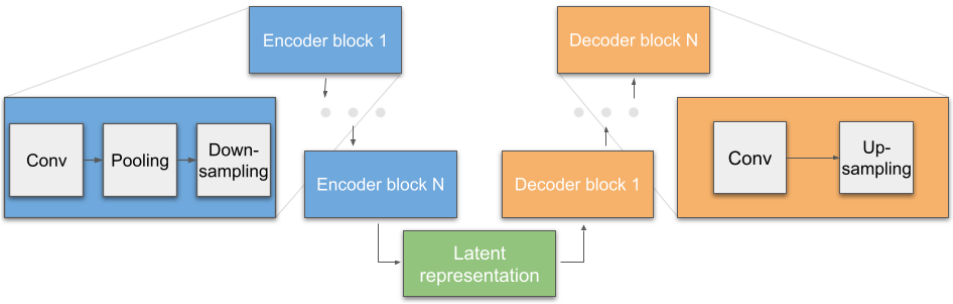


Figure 4.3.: **Variational Autoencoder structure**. The symmetric structure of a VAE consists of an encoder and a decoder part, with the latent representation laying in-between. Most often an encoder block is made up by a convolutional layer followed by max-pooling and downsampling layers. Consequently a decoder block includes a convolutional layer followed by an upsampling layer.

The encoder most commonly consists of repeating blocks of convolutional, pooling and downsampling layers.

The objective of the decoder is reconstruct the data-space representation from the latent embedding vector, parameterizing the likelihood function $p_\theta(\mathbf{x}|\mathbf{z})$. However, sampling directly from variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is a non-differentiable stochastic operation, which blocks the backpropagation of gradients and thereby optimization based on gradients. To circumvent this, VAEs utilize the reparameterization trick to compute the latent vector deterministically from the mean and the standard deviation:

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \odot \boldsymbol{\epsilon}, \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

In this formulation, $\boldsymbol{\epsilon}$ is an independent noise injection. This moves the stochasticity to an input node and allows gradient flow rom μ and σ to $\mathbf{z}$. The decoder typically encompasses transposed convolutions and nearest-neighbor interpolation upsamplers to restore the original spatial resolution ([67]).

In the proposed pipeline, a pre-trained VAE is leveraged to follow state-of-the-art texture generation approaches. The chosen VAE architecture possesses requisite generative capacity to process the non-RGB input modalities required by the task domain, namely albedo, roughness and metallic maps ([22]).

4.7. UNet Denoising Backbone

In contemporary diffusion frameworks the core generative engine is a UNet architecture. It is fundamentally a symmetric encoder-decoder structure augmented with skip connections linking the feature maps of the individual levels of the encoder and decoder components. Skip connections aim to solve the problem of vanishing and exploding gradients by directly transferring fine-grained spatial information often lost during the encoding-decoding process. On a given level of the UNet architecture, the skip connection can be characterized as follows

$$\mathbf{F}_{\text{skip},l} = \text{Concat}(\mathbf{F}_{\text{encoder},l}, \mathbf{F}_{\text{upsampled},l-1})$$

where $\mathbf{F}_{\text{encoder},l}$ is the output feature map of the encoder block at level l and $\mathbf{F}_{\text{upsampled},l-1}$ is the output feature map of the decoder block at level $l-1$. The resulting $\mathbf{F}_{\text{skip},l}$ feature map is used as input on the respective level of the decoder ([68]).

To inject conditioning signals such as text embeddings or image data, the UNet architecture is enhanced with cross-attention layers placed between the residual blocks, in the optional bottleneck layers and in the decoder path. This cross-attention is mathematically formulated as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

where the query $\mathbf{Q}$ is the internal feature maps of the UNet, the key $\mathbf{K}$ and the value $\mathbf{V}$ are derived from the conditioning vector. This mechanism forwards the semantic conditioning to spatially modulate the intermediate representations of the texture during the reverse diffusion process ([69]).

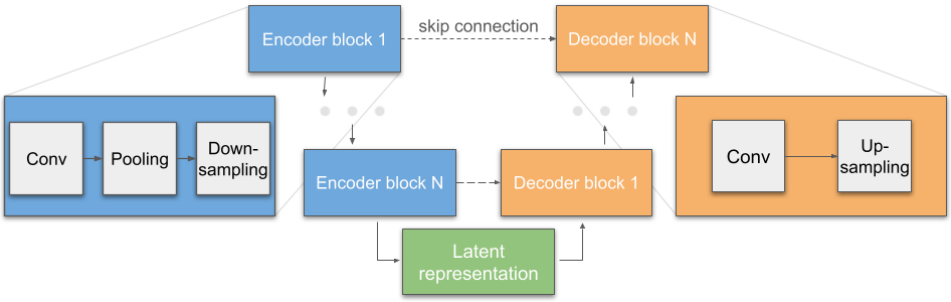


Figure 4.4.: **UNet structure**. The structure of a UNet architecture heavily resembles that of a VAE illustrated in figure 4.3, with sequential encoder blocks followed by a series of decoder blocks. The key difference is the skip connections running between the opposing layers of the encoder and decoder, transferring fine-grained spatial information between them. This property tackles the problem of vanishing or exploding gradients.

4.8. Text embedding with Contrastive Language-Image Pre-training

To align meaning of the generated textures and user intention, the proposed pipeline employs Contrastive Language-Image Pre-training (CLIP). The aim of CLIP is to learn the semantic relationship between raw text and raw image data by mapping them into a shared, high-dimensional embedding space using a joint image and text encoder. The shared nature of the embedding space allows for measuring of the similarity of any given text and image pair utilizing simple cosine similarity.

Contrary to traditional computer vision models utilizing human-annotated datasets (ImageNet), CLIP was trained on internet images and their captions. On a conceptual level, the learning objective of CLIP is to maximize the similarity of positive image-text pairs while minimizing the similarity of all negative pairs. The learning objective can therefore be formulated as follows:

$$L = \frac{1}{2N} \sum_{i=1}^{N} \left(L_{\text{image},i} + L_{\text{text},i} \right)$$

where $L_{\text{image},i}$ cross-entropy loss ensures that the i-th image embedding is assigned the caption with the highest similarity and $L_{\text{text},i}$ cross-entropy loss ensures that the i-th text embedding is assigned the image with the highest similarity ([27]).

In the proposed pipeline, CLIP is first leveraged to tokenize the text prompt denoting the desired texture parameter using Byte-Pair Encoding. The purpose of tokenization is to derive the variable-length text into a uniform set of numerical IDs found in the CLIP vocabulary. Subsequently the set of these IDs is used to look up the text embedding of the text prompt denoting the desired texture parameter. This text embedding is then utilized to condition the generation process using cross-attention mechanisms detailed in sections 4.4 and 4.7.

5. Experiments

5.1. Datasets

5.1.1. Benedikt Bitterli's Rendering Resources

Figure 5.1.: **Sample scenes of Benedikt Bitterli's Rendering Resources ([70])**. The Rendering Resources is a set 3D environment with manually configured PBR materials and lighting. Additionally, it includes image captures for with defined camera poses

The Rendering Resources is a collection of high quality 3D scenes compiled by Benedikt Bitterli, as seen in figure 5.1. In contrast to unstructured raw internet data, this set consists of curated scenes with cleaned up geometry and manually set up PBR materials, lighting and defined camera poses. The scenes range from simple, academic test setups like the Utah Teapot to more complex, photo-realistic interior environments including global indirect illumination ([70]). The proposed pipeline leverages the Rendering Resources twofold. First, the dataset serves as the input for training, providing a collection of multi-view image captures alongside their respective calibrated camera poses, as well as an already whole scene geometry mesh containing all the objects of the scene. Furthermore the dataset supplies its own ground-truth data for the individual rendering parameters, which can be used for retrieving quantitative results.

5.1.2. ScanNet++

The ScanNet++ dataset consists of 3D scene geometry reconstructed from large-scale real-world 3D scans, scene image captures and associated camera poses as well as PBR rendering and semantic labels. It is an extension of the ScanNet repository in that it increases the

Figure 5.2.: **Sample scenes of the ScanNet++ dataset ([71])**. The Scannet++ dataset is a collection of real-world 3D scans, accompanied by image captures and their respective camera poses. The dataset provides its own set of ground-truth PBR material observations and further semantic labels

number of scenes and their complexities. It aims to reduce noise and the number of artifacts introduced by traditional scanning. It currently includes 460 scenes and 280'000 captured DSLR images ([71]). The pipeline utilizes this dataset in a fashion similar to that of Benedikt Bitterli's Rendering Resources, detailed in subsection 5.1.1, as in it leverages the complex scene geometry, image captures and associated camera poses for training and employs the associated ground-truth values of rendering parameters for quantitative comparison.

Fundamentally, processing real-world data poses more challenges compared to synthetic data. Firstly, the scene geometry is more complex and reconstructed from different types of scannings of real environments, such as laser scanning or photogrammetric processes like Structure from Motion.These processes of geometric sampling inherently introduce geometric artifacts, linear edges undergo deformation and planar surfaces showcase perturbed, non-planar characteristics. As the Scannet++ database operated with laser scannings, the scene geometry is represented as an extensive set of vertices, magnitudes more than common in synthetic data, applying further pressure on the differentiable rendering pipeline.

Secondly, the material and appearance complexity of real-world environments far exceeds that of synthetic scenes. Subtle variations in the texture, color and microstructure of non-homogenic real-world objects make it impossible to perfectly model their respective BRDF functions. Subsurface scattering and translucency elevates this complexity, necessitating rather approximations utilizing simplified standard models.

5.2. Implementational details

The proposed pipeline is implemented utilizing the frozen pre-trained components of the RGB↔X pipeline, more specifically its diffusion timestep scheduler and UNet, VAE and text encoder architecture. RGB↔X is specifically designed for cross-domain synthesis of

separate PBR feature maps using multi-modal text and image conditioning, making it a more suitable architecture for distilling individual texture maps while disentangling appearance from geometry. The model takes a different approach than joint estimation methods that try to optimize all material maps simultaneously, and instead opts to generate them sequentially. This allows for parallelization and for distinct feature extraction for the different physical properties.

State-of-the-art asset generation approaches often utilize Variational Score Distillation (VSD), which theoretically offers higher diversity than Score Distillation Sampling (SDS) with a significant memory overhead to maintain its variational distribution. Unfortunately, empirical testing showed that such memory overhead exceeded the capacity of the target hardware, possessing 24GB VRAM. Subsequently, the pipeline runs traditional SDS to fit in memory constraints while maintaining high structural fidelity. The diffusion time-step of SDS is truncated to the range $t \in [0.2, 0.98]$, to avoid burn-in artifacts caused by very low noise levels. Furthermore, time-step annealing is explicitly disabled. Empirical testing indicated that, while producing lower average error during training, introducing timestep annealing significantly degraded the output visual fidelity. These observations led to the inactivation of the timestep annealing strategy, in favor of fully random timestep sampling.

The pipeline leverages an AdamW optimizer, selected for its decoupled weight decay implementation, which often yields better regularization in generative tasks than standard Adam, with a learning rate $\alpha = 0.01$. Other learning rates such down until $\alpha = 0.001$ were tested, and shown to yield the same local optima with the convergence speed significantly slower. The effective batch size $b = 1$, due to the architectural limitations of the RGB↔X pipeline.

As detailed in section 4.4, the pipeline conditions the generation on both text and image modalities, leveraging Multi-Modal Classifier-Free Guidance (CFG) for noise prediction. However, extensive ablation showed that best visual fidelity is yielded when $w_i = w_t = 1$. Based on equation 4.2, this means that the final noise prediction is simplified to only use the text-conditioned prediction:

$$\boldsymbol{\epsilon}_{\text{pred}} = \boldsymbol{\epsilon}_{\text{text}}$$

Contrary to traditional SDS methods, the proposed pipeline employs an image-space learning objective inversely weighted by a custom Visibility Accumulation Map $\mathcal{V}$, as discussed in sections 4.2.1 and 4.3.1. The inverse weighting scheme ensures that the optimizer does not neglect less visible regions, increasing global consistency across the entire 3D scene. Further loss regularization was implemented in the shape of image-space loss dropout as detailed in section 4.3.2, but was ultimately disabled as testing showed not only no improvement in the visual fidelity of model output, but significant degradation.

The target output of the model is a standard UV-mapped RGB texture, to ensure compatibility with standard rendering pipelines, in contrast to hash-grid representations. A texture resolution of 4096×4096 was found to yield the most detailed results, but a 2048×2048 size exhibits only marginal degradation in visual quality. Since the computational intensity scaling is manageable, the 4096×4096 size is preferred. The individual texture maps are initialized differently based on the domain. Albedo, normal and irradiance maps are initialized as

zero tensors. Metallic maps are initialized to full -1 in denormalized $[-1, 1]$ space. This maps to 0 in the normalized $[0, 1]$ space, loading every texel point as dielectric. Similarly, roughness maps are initialized to full 1, indicating every texel as matte in normalized space. This domain-specific initialization prevents stopping in highly specular local minimas in early training phases.

All experiments were launched on a workstation equipped with a single NVIDIA GeForce RTX 3090 GPU with 24GB VRAM available, running CUDA Version 12.9. A floating-point precision of 32-bits was used, as 16-bit precision results in gradient explosion and oversaturated model outputs. Average runtime of generating a single material map was found to be 4.5 hours with 17K training iterations. This heavily depends on the complexity of the scene geometry. Clean scenes, such as Country Kitchen in the Rendering Resources dataset, run for roughly 3 hours on average, while complex scenes, like Apartment in ScanNet++, train for up to 6 hours.

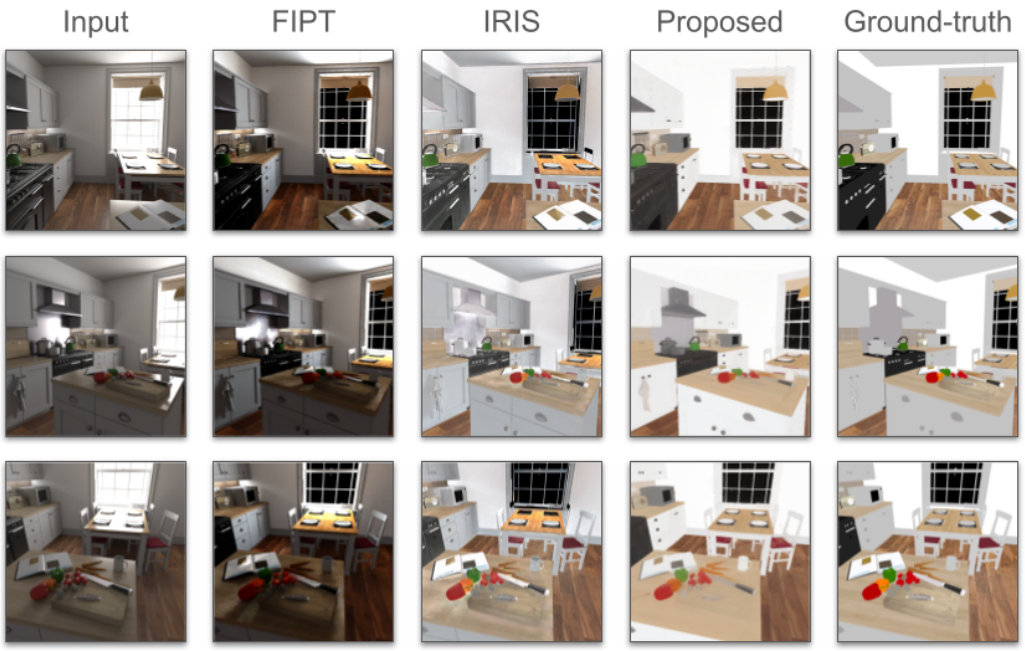
5.3. Comparisons

In the following, qualitative and quantitative comparisons are detailed between the proposed pipeline and two state-of-the-art approaches, FIPT ([8]) and IRIS ([9]). To compare the albedo, metallic and roughness predictions, four scenes of the Rendering Resources dataset are utilized, namely "Country Kitchen", "Contemporary Bathroom", "Bedroom" and "The White Room". On these four scenes both previous approaches and the developed framework is launched. Subsequently, the predicted PBR material parameters are separately used to render the scenes from pre-established camera poses. These renders, as well as the respective image-capture and ground-truth renders with the individual PBR material provided by the dataset are visually compared against one another for qualitative evaluation in figures 5.4, 5.6 and 5.8.

As visualized in figure 5.4, the developed pipeline qualitatively outperfoms current state-of-the art approaches in optimizing albedo parameters, yielding higher visual fidelity when compared to the ground-truth data. In contrast, both metallic and roughness material estimation fails to deliver the quality of results retrieved by the aforementioned methods. This is outcome is the consequence of both the used image prior and the loss formulation.

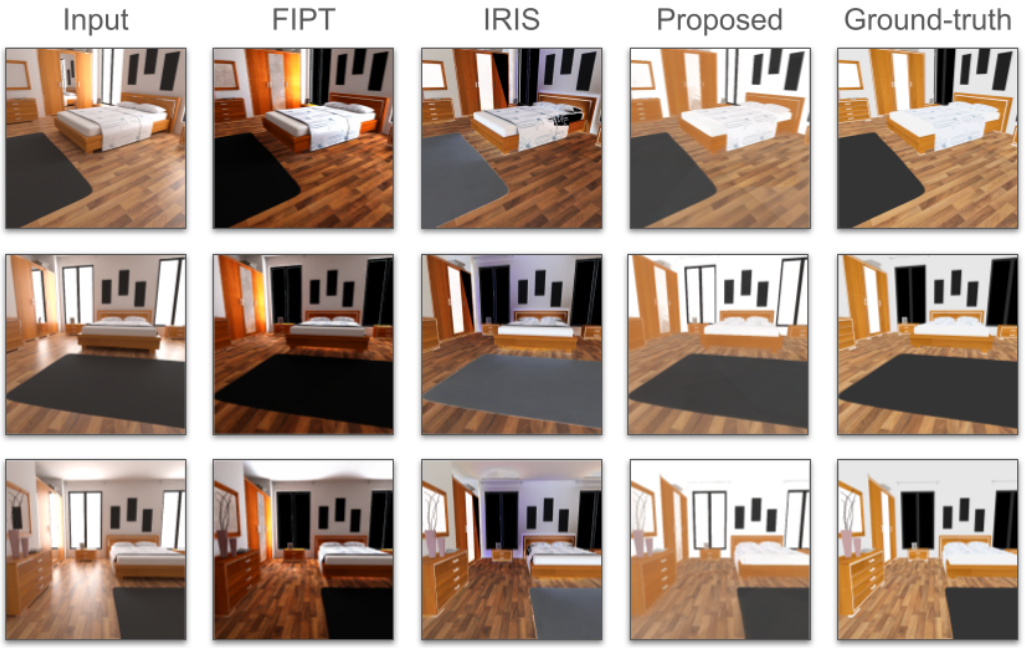
Firstly, the training of the estimator was done using everyday scenes of simple BRDFs. However, the target BRDF domain is much more complex with highly specular, non-Lambertian light transport. Consequently, the estimator struggles to generalize from the domain it was trained on to the domain it is applied to, leading to unplausible results. Secondly, the pipeline applies no rerendering loss. The rerendering loss aims to increase photometric consistency by rendering an image with the generated material maps and identical lighting as the original input image. The two renders are compared to calculate the rerendering loss. Unfortunately, the disjoint nature of material optimization does not allow this approach.

Quantitative assessment is run in a similar fashion, where the respective texture maps optimized by the aforementioned approaches are used to render the scene from all available camera poses. The resulting renders are accumulatively compared to ground-truth data



(a) **Albedo material estimations on the Rendering Resources ([70]) "Country Kitchen" scene**

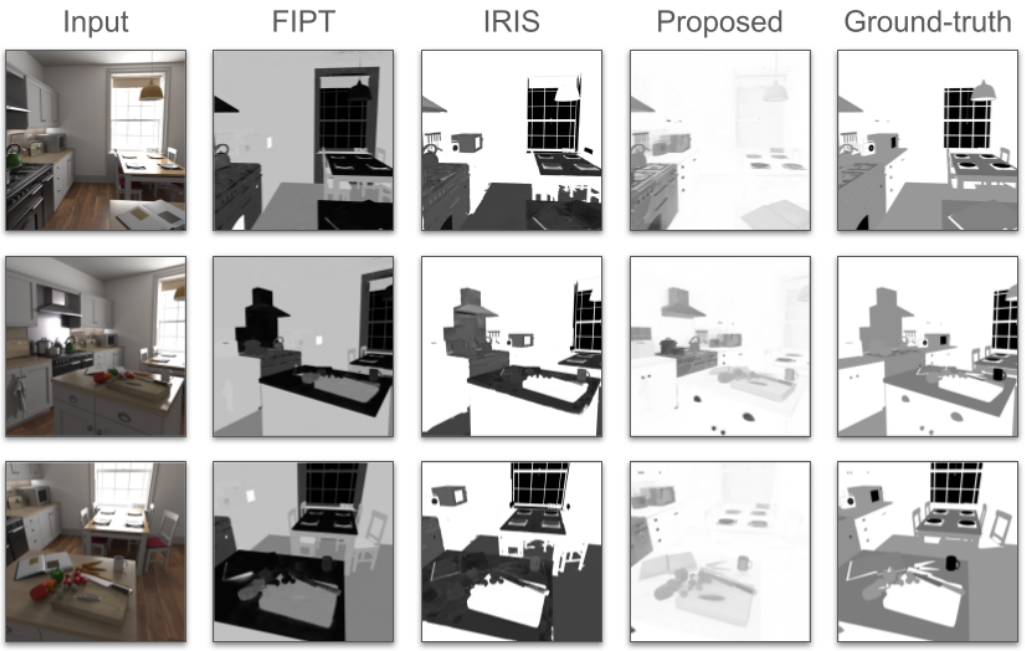
(b) **Albedo material estimations on the Rendering Resources ([70]) "Contemporary Bathroom" scene**



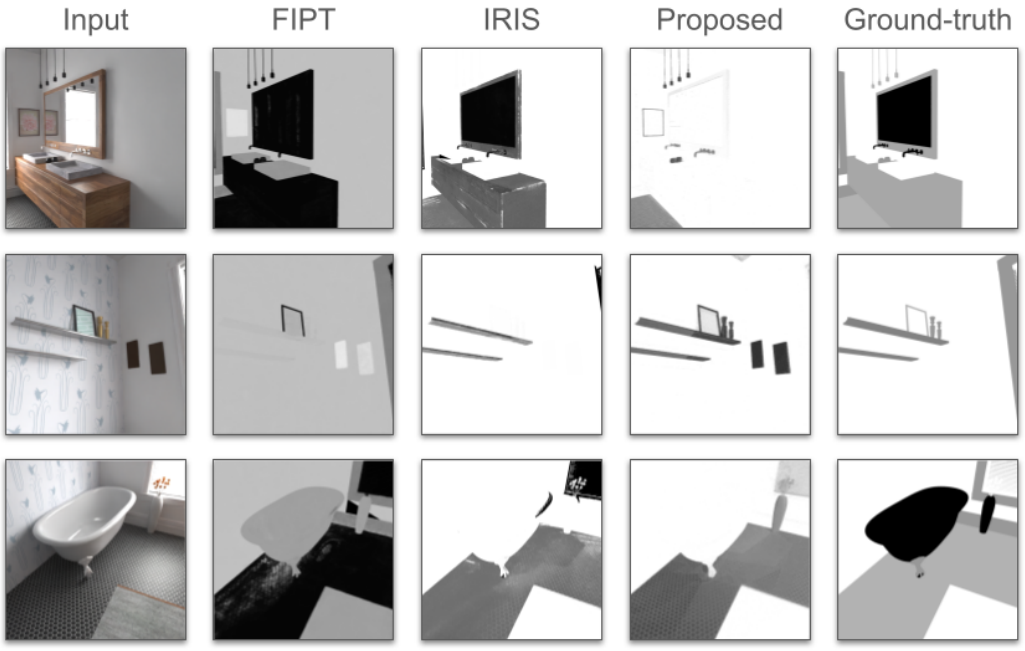
(a) **Albedo material estimations on the Rendering Resources ([70]) "Bedroom" scene**

(b) **Albedo material estimations on the Rendering Resources ([70]) "The White Room" scene**

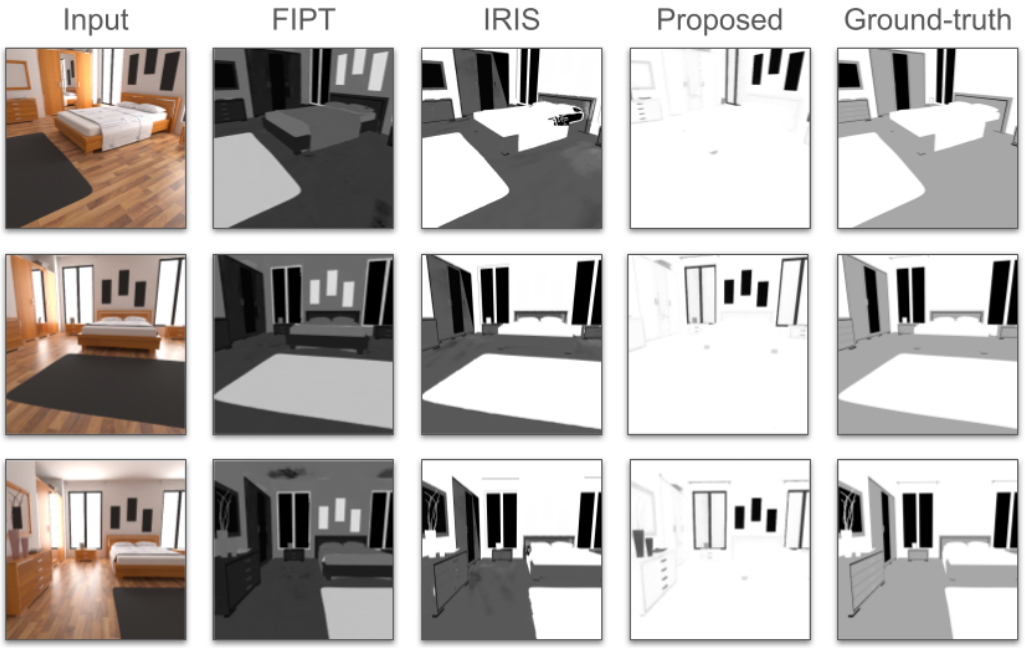
Figure 5.4.: **Albedo material predictions of the proposed pipeline compared to state-of-the-art approaches FIPT ([8]) and IRIS ([9]) and the ground-truth data on four Rendering Resources ([70]) scenes**. The developed pipeline visually outperforms other leading methods, more closely approximating the target albedo map. On the left the PBR scene rendering respective to the estimations is shown.



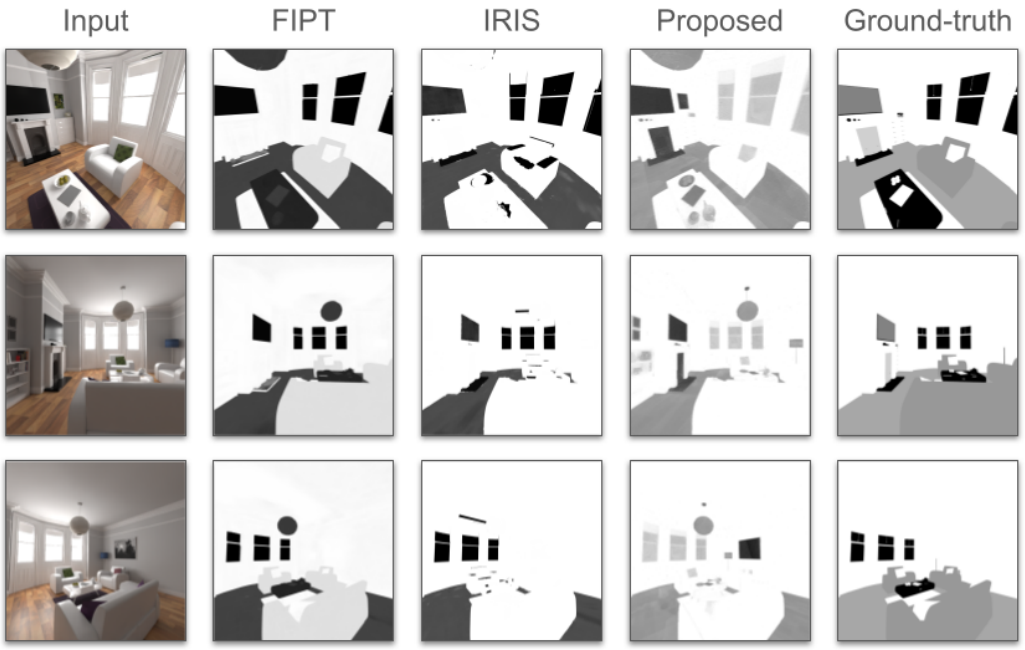
(a) **Roughness material estimations on the Rendering Resources ([70]) "Country Kitchen" scene**



(b) **Roughness material estimations on the Rendering Resources ([70]) "Contemporary Bathroom" scene**

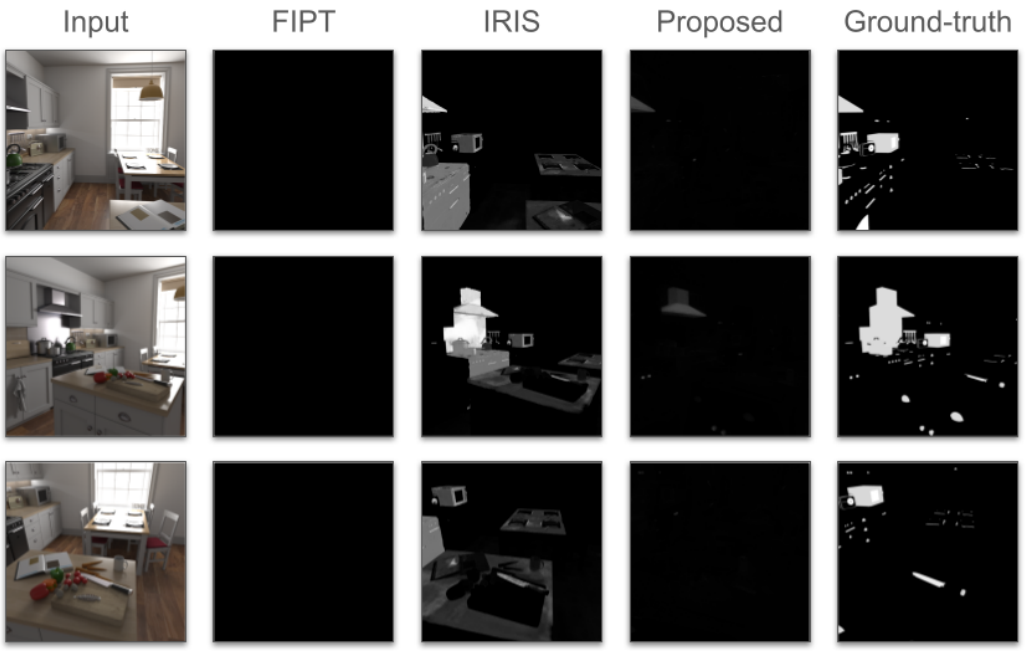


(a) **Roughness material estimations on the Rendering Resources ([70]) "Bedroom" scene**

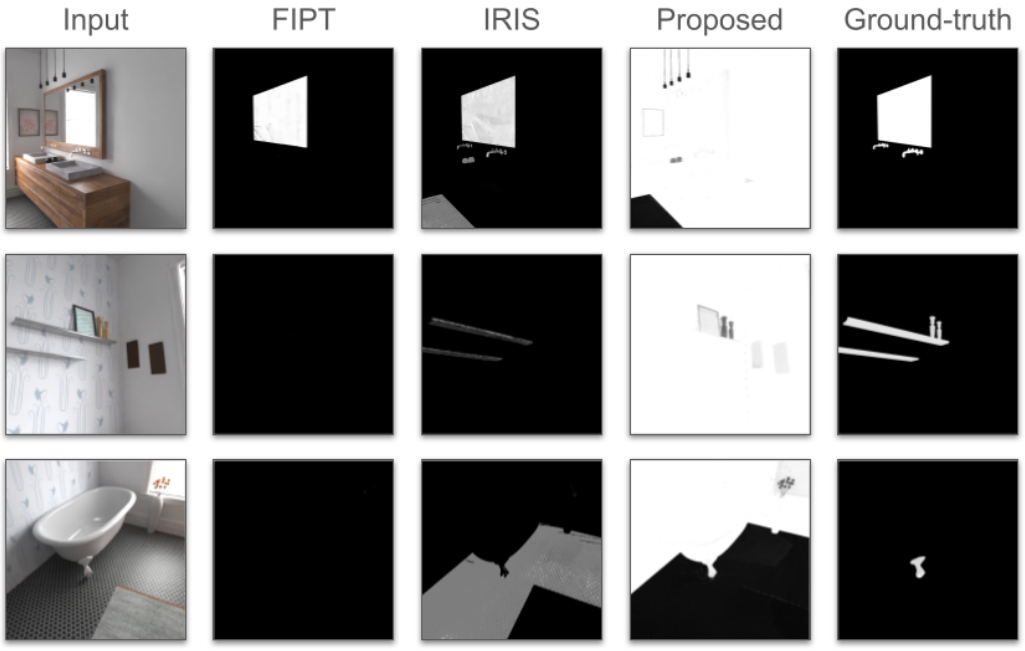


(b) **Roughness material estimations on the Rendering Resources ([70]) "The White Room" scene**

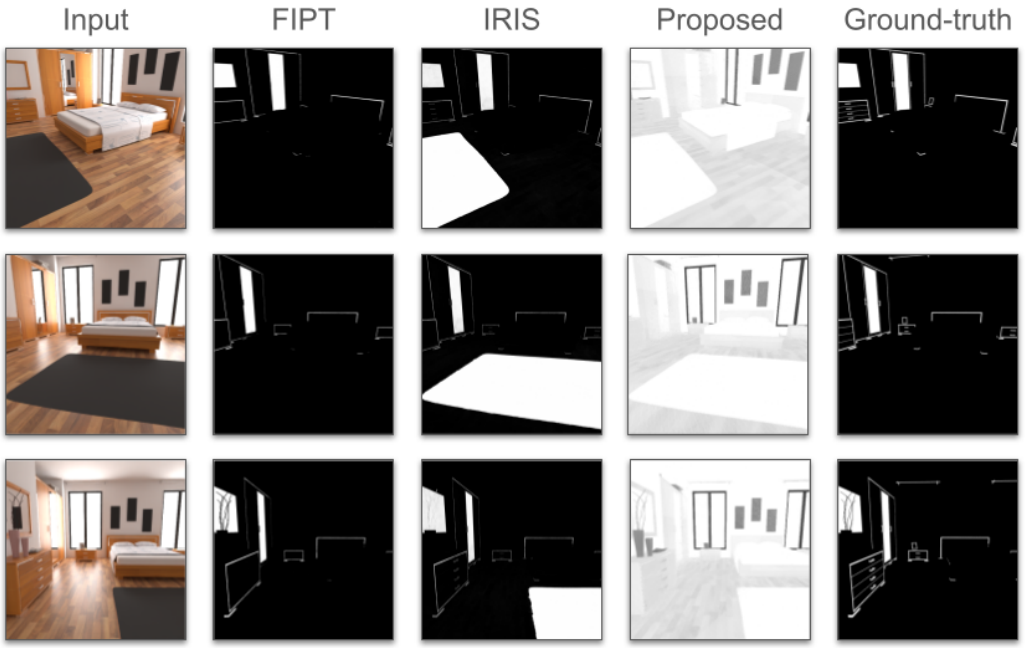
Figure 5.6.: **Roughness material predictions of the proposed pipeline compared to state-of-the-art approaches FIPT ([8]) and IRIS ([9]) and the ground-truth data on four Rendering Resources ([70]) scenes**. The developed pipeline fails to capture material consistency compared to the ground-truth data, due to the utilization of rasterization-based optimization, which discards optical information disambiguating metallic and roughness parameters



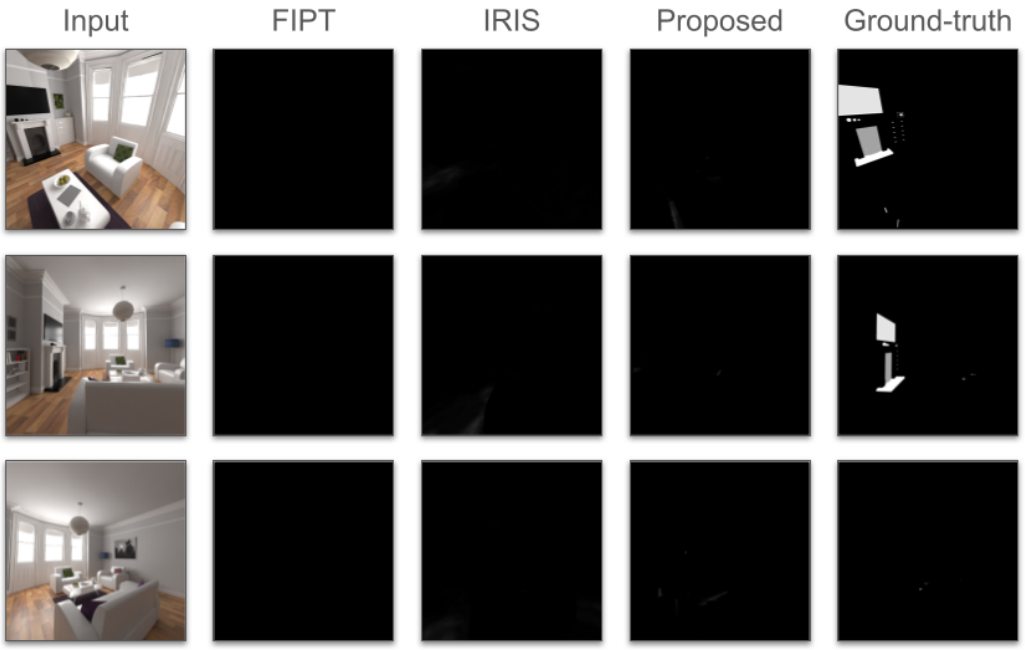
(a) **Metallic material estimations on the Rendering Resources ([70]) "Country Kitchen" scene**



(b) **Metallic material estimations on the Rendering Resources ([70]) "Contemporary Bathroom" scene**



(a) **Metallic material estimations on the Rendering Resources ([70]) "Bedroom" scene**



(b) **Metallic material estimations on the Rendering Resources ([70]) "The White Room" scene**

Figure 5.8.: **Metallic material predictions of the proposed pipeline compared to state-of-the-art approaches FIPT ([8]) and IRIS ([9]) and the ground-truth data on four Rendering Resources ([70]) scenes**. The developed pipeline fails to capture material consistency compared to the ground-truth data, due to the utilization of rasterization-based optimization, which discards optical information disambiguating metallic and roughness parameters

Method	Albedo PSNR ↑	Albedo SSIM ↑	Albedo LPIPS ↓	Rough L2 ↓	Metal L2 ↓
FIPT [8]	10.63	0.661	0.403	0.110	**0.006**
IRIS [9]	15.86	0.735	**0.307**	**0.056**	0.040
Proposed pipeline	**17.44**	**0.758**	0.397	0.143	0.403

Table 5.1.: **Baseline comparisons.** Quantitative results leveraging different metrics are averaged over all the views of four synthetic scenes from the Rendering Resources dataset ([70]). The developed model outperforms current baseline methods in albedo map synthetization, but needs further tuning in roughness and metallic material generation. In contrast to the proposed pipeline, The baseline methods do not generate normal or irradiance maps, which are therefore not evaluated here.

utilizing a number of conventional metrics. For simplified interpretability, the quantitative results aggregate the individual scene scores, yielding a single global metric. For albedo comparison Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) are used in a scale-invariant fashion, to follow current literature. As both roughness and metallic texture maps are grayscale, a simple L2 norm is leveraged there to assess performance.

PSNR is a popular metric for image and video processing to quantify reconstruction quality of a lossy compressed image compared to the uncompressed counterpart, based on Mean Squared Error. It is expressed on a logarithmic decibel scale, where higher values denote a better reconstruction quality. Contrary to this, SSIM aims to explicitly model human visual perception of structural information. It measures quality based on luminance, contrast and structure. Similarly to PSNR, higher score signifies better quality. Lastly, LPIPS is a deep-learning based metric. It works on a feature representation extracted by a Deep Convolutional Neural Network, which is pre-trained on a large image classification task utilizing human-labeled dataset of image pairs labeled by perceptual similarity. A lower LPIPS score indicates higher qualitative similarity.

As seen in table 5.1, the proposed pipeline achieves superior performance compared to existing state-of-the-art approaches in the field of albedo map generation. Roughness and metallic material synthetization, while comparable to baseline efficiency, needs further fine tuning. As reflected in the table, the baseline methods generate no normal or irradiance maps, consequently falling outside the scope of this investigation.

5.4. Applications

To showcase the relighting capabilities of the model, a number of qualitative applications were run, detailed in the following sections. The primary objective here is to evaluate visual coherence, sharpness and material consistency of the generated textures by relighting with different lighting environments. Therefore, the two unique 3D scenes of the ScanNet++ dataset, namely "Bedroom" and "Apartment", are relit from a number of poses with a number of unique lighting conditions.

The "Bedroom" scene offers a constrained topological space, ideal for small-scale testing with complex occlusion patterns originating from wrinkled curtains or personal electronics. Even though the scene is of smaller scale, it already introduces large material variety. Subsequently, the larger scope of the rooms of the "Apartment" scene is utilized to showcase global consistency.

The scenes are relit employing multiple distinct lighting environments and subsequently rendered using a consumer-grade computer graphics program from a number of camera poses to showcase the robustness of the generated textures. As illustrated in figure 5.9 and 5.10, the proposed pipeline has great capabilities in distilling visually plausible PBR materials. Furthermore, figure 5.9 showcase the efficiency of the model to capture small details such as the drawings on the wall or the cable running in the right corner of the room.

Figure 5.9.: **Relighting of ScanNet++ ([71]) "Bedroom" scene**. This scene offers a more constrained environment for relighting experimentation. The experiments were launched with a number of different lighting conditions from unique camera poses, showcasing visually plausible outcomes. Furthermore, the results show the capability of the model to capture small details of the scene, such as the drawings on the wall or the cable running in the right corner of the room. For reference, the respective image capture of the dataset is provided.

Raw results of the training method used in the aforementioned light editing use-cases are also provided in the form of the optimized material maps and respective sample renderings of the ScanNet++ "Bedroom" scene, illustrated in figure 5.11 and 5.12.

Figure 5.10.: **Relighting of ScanNet++ ([71]) "Apartment" scene**. The visual fidelity of the results on the "Apartment" scene, a geometrically more complex environment, prove the efficiency of the proposed pipeline. Similarly to figure 5.9, individual unique view points and lighting environments are utilized to demonstrate output quality. For reference, the respective image capture of the dataset is provided.

5.5. Ablations

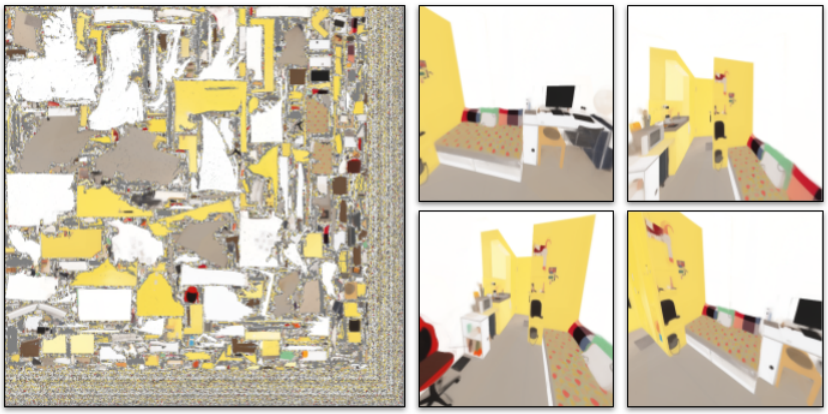
In the following the effects of individual design decisions are detailed, providing further justification on the selections made.

5.5.1. Classifier-Free Guidance

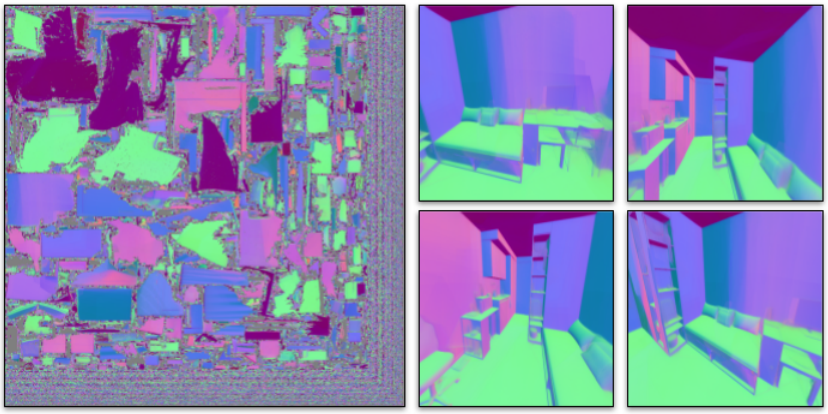
The goal of CFG is to implement guided generation without the utilization of a separately trained classifier, by instead interpolating from the unconditional prediction to the conditional prediction. The step size of the interpolation is defined by guidance scales, more specifically image guidance scale w_i and text guidance scale w_t in the context of the proposed pipeline.

To enhance output fidelity, a systematic hyperparameter search was initiated on the guidance scales. It was found that low w_t and w_i values resulted in a rapid degradation of feature separation, causing the output texture to shift uniformly toward a yellow color bias. On the contrary, increasing the w_t and w_i values enhanced the overall hue intensity but failed to introduce the necessary fine-grained structural detail and property separation.

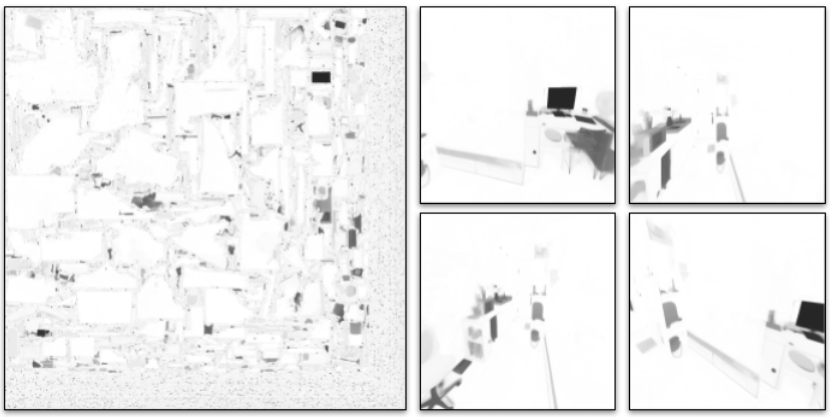
Subsequently, further experiments on hyperparameter tuning focused improving model output quality by finding individual values for text guidance scale w_t and image guidance scale w_i for achieving a more granular control over the influence of the conditioning view versus the semantic text prompt. Though initial results remained unsatisfactory, the effect of letting one conditioning dominate the noise prediction by setting it a larger scale was observed, leading to possible leads on the reason of low-quality outputs. It was noted that elevating the value of w_i relative to w_t resulted in hyper-saturation of the scene hue characterized by a predominant purple chromatic bias, as well as extreme contrast disparity



(a) **Albedo material map and sample renderings of the ScanNet++ ([71]) "Bedroom" scene**

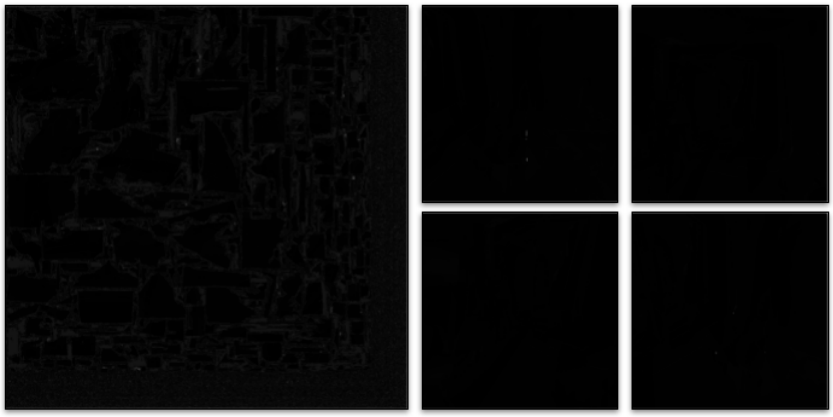


(b) **Normals material map and sample renderings of the ScanNet++ ([71]) "Bedroom" scene**



(c) **Roughness material map and sample renderings of the ScanNet++ ([71]) "Bedroom" scene**

Figure 5.11.: **Albedo, normal and roughness material maps optimized on the ScanNet++ ([71]) "Bedroom" scene**. Sample renderings from equivalent camera poses are demonstrating the resulting scene fidelity.



(a) **Metallic material map and sample renderings of the ScanNet++ ([71]) "Bedroom" scene**

(b) **Irradiance texture map and sample renderings of the ScanNet++ ([71]) "Bedroom" scene**

Figure 5.12.: **Metallic and irradiance material maps optimized on the ScanNet++ ([71]) "Bedroom" scene**. Sample renderings from equivalent camera poses are demonstrating the resulting scene fidelity.

between scene objects, as showcased in figure 5.13.

Figure 5.13.: **Results optimized with image guidance scale w_i dominating the model**. Setting image guidance scale w_i to be an order of magnitude higher than text guidance scale w_t destabilizes the synthesis process, resulting in a chromatic shift.

Ultimately, it was observed that setting both w_i and w_t to 1.0 critically increases the output plausibility. Based on equation 4.2, this setting results in only relying on the text-conditioned noise estimation, suggesting that the text prompt is the most effective and dominant prior for the current task. This observation likely stems from the high information density provided by image conditioning, which effectively reduces feasible solution manifold, promoting rapid convergence to a consistent solution; in contrast to text-only SDS, where a high guidance scale is commonly required ([46])

5.5.2. Image-Space Loss

The primary motivation to shift the objective calculation to image-space instead of the commonly used latent space is that it allows the error to be computed at the native resolution of the renderer. This approach can retain high-frequency details frequently discarded in encoded representations. Furthermore, applying an image-space loss enables the loss regularization methods detailed in section 4.3. However, experimental evaluation demonstrated that utilizing latent-space loss in the current pipeline environment exhibits prohibitively slow convergence, making it unfeasible, as illustrated in figure 5.14. Extensive hyperparameter search would be necessary to incorporate latent-space objective. Even if a feasible configuration was achieved, latent-space introduces oversmoothing property inherent to the compression mechanism, manifesting in perceptual blurriness. Consequently, the image-space objective is leveraged.

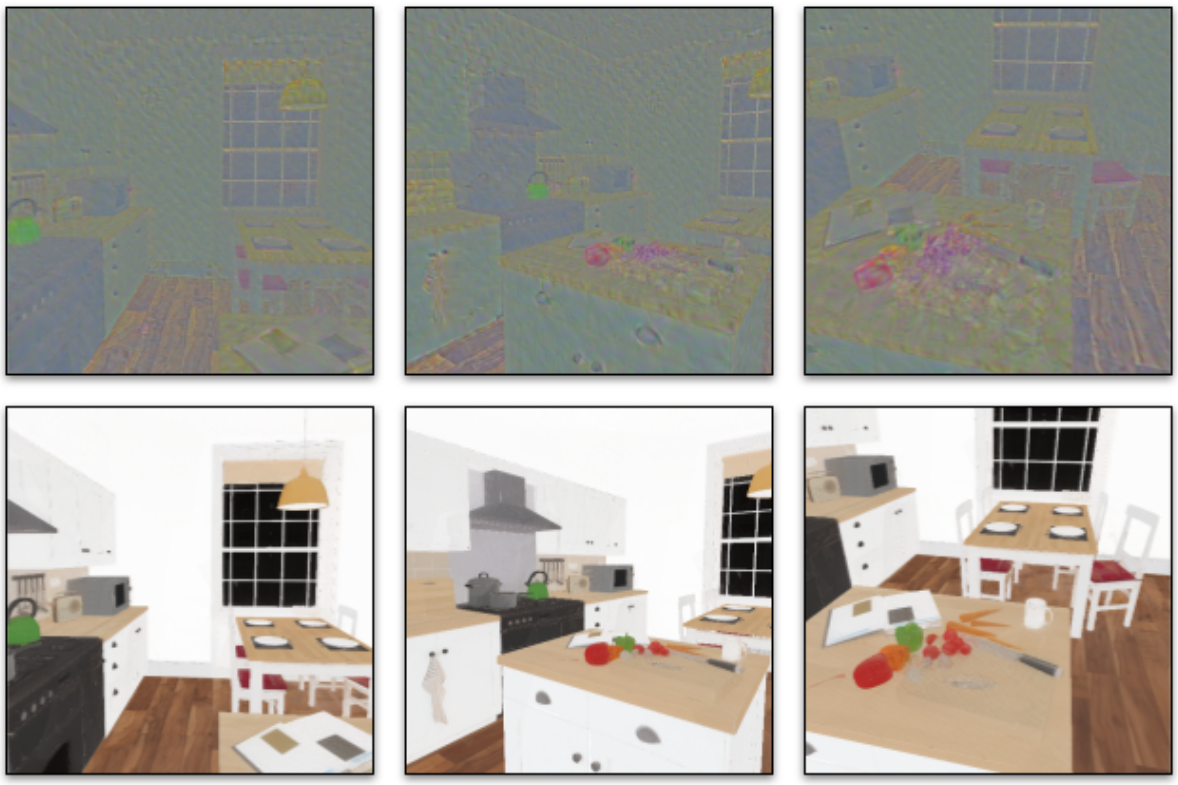


Figure 5.14.: **Results optimized with latent-space loss (above) compared to results optimized with image-space loss (below)**. Optimizing with the ordinary SDS latent-space objective yields unfeasible convergence in the current pipeline setup. The latent-space objective could be leveraged after extensive hyperparameter search, but theoretically would still yield results suboptimal compared to an image-space objective, due to the latent compression mechanism.

5.5.3. Visibility Weighting

As detailed in section 4.3.1, the core problem the implemented visibility-based weighting aims to solve is that the number of times a geometric point of the 3D scene is seen by the pre-established camera poses can greatly vary, which results in more frequently seen vertexes to have a larger influence on the loss propagated to the target texture map. The goal of the weighting scheme is to enforce correct texture values for less seen vertexes.

The schema successfully mitigated the previously observed seaming artifact in the generated texture maps, as seen in figure 5.15.It also ensured that the gradients derived from vertices less seen get properly propagated, enforcing correct texture parameters across the entire 3D geometry.

Figure 5.15.: **Results optimized without and with utilizing view-based weighting**. A view-based weighting of the loss was implemented to counter seaming artifacts observed in the generated texture maps. The weighting schema also enhanced the overall visual fidelity of the result by enforcing correct coloring on less-seen geometric points of the scene. Further features and mathematic explanation of the view-based weighting schema is detailed in section 4.3.1

5.5.4. Timestep annealing

In such a texture distillation task it commonly yields better results to utilize timestep annealing, a strategy where different steps in the optimization grant different available ranges for sampling the diffusion timestep. The initially implemented annealing strategy allows the full timestep sampling range initially. After a number of iterations in the optimization process, it essentially halves the available range to only include the lower half. This guides the model to first learn both coarse global structure smaller details by supplying both high and low timestep t values. After the annealing, the supplied low t values encourage the model to

further enhance high-frequency texture details. In usual industry pipelines, such annealing is leveraged to prevent burn-in artifacts in the output results.

Empirical testing found however that applying such annealing on the timestep schedule introduces significant artifacts in the output of the model, considerably degrading visual fidelity. More specifically, the generated texture map consisted of more shaded colors, and the pipeline optimized diffuse colors into transparent regions of the scene such as windows, as illustrated in figure 5.16.

Figure 5.16.: **Results optimized utilizing timestep annealing**. Contrary to the expected results, timestep annealing introduced considerable artifacts in the output. Visual fidelity significantly decreased thanks to the optimized texture map consisting of lower hue colors. Furthermore, transparent regions of the 3D scene, such as the window visible in the left and middle images, were mapped to regions of the albedo texture map with color, producing diffuse colors on expected transparent surfaces.

5.5.5. Image-Space Dropout

As introduced in section 4.3.2, the proposed pipeline leverages an Image-Space Dropout to emulate the stochasticity of path-tracing used in PBR rendering pipelines. The fundamental concept is that by setting the loss to zero on a subset of pixels, the condition of the view ray not intersecting any light source can effectively be simulated. Whether the loss is dropped on a given pixel is defined by retention probability p_{keep}. Extensive search was run on p_{keep} to determine the best dropping probability. However, as illustrated in figure 5.17, decreasing p_{keep} significantly degrades the visual fidelity of the output and the quantitative evaluation detailed in section 5.3. This ultimately led to the decision of keeping $p_{keep} > 0.9$, to keep its effects minimal.

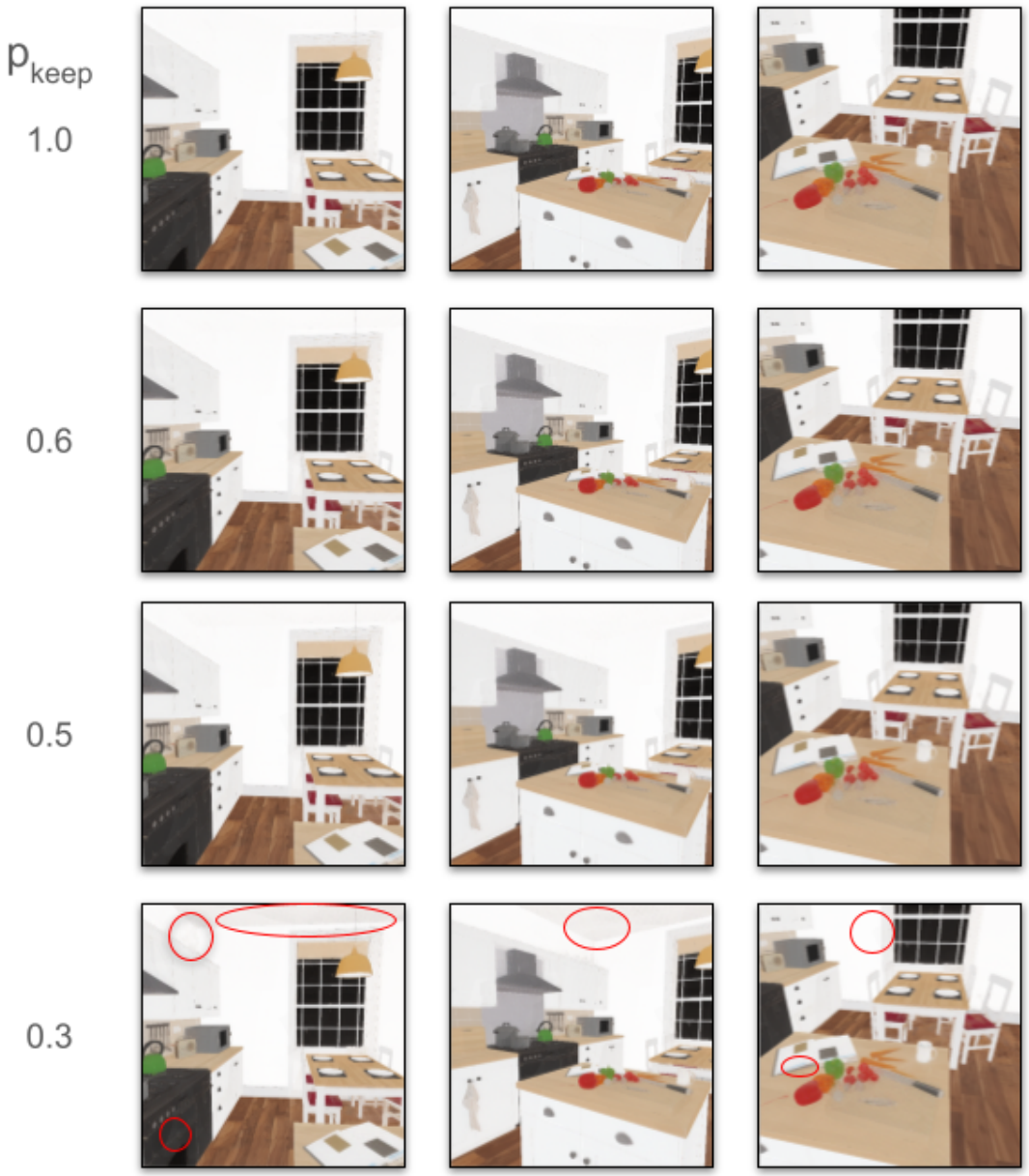


Figure 5.17.: **Sample renders of the albedo material using differing values of retention probability p_{keep} for the Image-Space Dropout**. Contrary to the expected results, implementing Image-Space Dropout degraded both output quality both in qualitative and quantitative assessment. Significant artifacts can be seen at $p_{keep} = 0.3$. Subsequently, the Image-Space Dropout was ultimately discarded.

5.6. Limitations and future work

Significant limitation is the inability of the model to physically accurately reconstruct material properties with view-dependent effects, namely metallic and roughness material attributes. This is an inherent restriction of the leveraged monocular diffusion model, which exhibits limited quality due to its exclusion of a physics-based inverse rendering formulation. One possible solution is a per object aggregation on the rendering loss. Evaluating the error over areas of geometry instead of relying on per-pixel difference should provide viable regularization.

The disjoint optimization of the individual material maps raises further limitations. As discussed in section 5.3, it forfeits implementing an accurate rerendering loss impossible. One potential resolution is a disentangled rendering loss. In this approach, instead of calculating the loss by the simultaneously trained material maps, the predicted map for the currently optimized material and ground-truth maps of all other properties are taken.

As detailed in the appendix section A, utilizing SDS introduces inherent limitations such as oversmoothing and oversaturation. Two possible solutions are investigated to mitigate these problems, namely V-parameterization and re-evaluation of possible VSD application.

Lastly, a view-consistent way of noise generation is planned. The core concept is that by generating a global noise entity and referencing parts of it specific to the respective camera pose used in the current optimization setup the diffusion model can further stabilize training convergence and exploit global information on the consistent noise pattern.

6. Conclusion

We have presented a novel optimization approach for 3D material map reconstruction for indoor environments. Our approach leverages a pre-trained monocular material estimator in a Score Distillation Sampling (SDS) framework, which distills the predictions of the estimator into visually plausible and geometrically consistent 3D texture maps. Contrary to other SDS applications however, the proposed pipeline shifts the supervision signal from latent-space to image-space in order to leverage larger objective resolution. Subsequently, the model incorporates an image-space regularization term, more specifically a visibility-based weighting scheme to enforce balanced optimization between frequently seen and obscured areas of the scene, removing significant visual artifacts stemming from non-uniform optimization trajectory. The plausibility of model results are evaluated with qualitative and quantitative approaches, confirming that the pipeline outperforms current state-of-the-art methods in albedo estimation. Furthermore, we use a rasterization-based optimization method, which is computationally less expensive than the path-tracing leveraged in the aforementioned approaches. In the ablation, extensive hyperparameter search yielding in the current superior performance is detailed, such as choice of learning rate, guidance scales and texture representation.

Disentanglement of specular material properties remains a challenge, an inherent limitation of the leveraged diffusion backbone. As highlighted in comparisons, the pipeline falls behind baselines in the accurate estimation of metallic and roughness parameters. This opens up future research paths, such as implementing a hybrid approach bridging physically based inverse rendering and the current pipeline setup, or multi-view consistent noise generation to improve consistency.

Despite this, material parameters generated by our approach show visual plausibility in material editing and relighting use-cases, as shown in the applications. Consequently, the pipeline is immediately applicable to post-processing tasks in digital entertainment, simulation and real-world visualization.

A. Score Distillation Sampling versus Variational Score Distillation

Initially, the proposed pipeline was developed leveraging Variational Score Distillation (VSD), a generalization of SDS aiming explicitly model the sampling distribution. VSD requires computing and storing an additional set of gradients and intermediate tensors, commonly resulting in visually more plausible results in generation tasks. However, this increases memory consumption substantially, regularly not fitting on consumer-grade GPUs.

Vanilla SDS has a relatively straightforward objective function leveraging a simple MSE term, where the score calculated by a pre-trained 2D diffusion model is used to compute the gradients of the texture generation network with respect to the input 3D representation, as detailed in section 4.2. Determining the gradient only involves backpropagating through the generation network and the 2D scoring model. On the other hand, VSD aims to explicitly model the sampling distribution of 3D parameters instead of a single set of parameters, utilizing a more sophisticated objective function which minimizes the Kullback-Leibler divergence between the distribution of the true diffusion model and that of the generated 3D content. This necessitates the introduction of an auxiliary network used to estimate densities or divergence or utilizing lightweight adapters like Low-Rank Adaptation (LoRA) modules. On a conceptual level, LoRA freezes the weight matrices of the original diffusion model and injects smaller, trainable matrices called rank-decomposition matrices; making LoRA is memory-efficient compared to a full network adaptation. Nevertheless, the introduction of any additional modules necessitates the extension of both forward and backpropagation passes to include them, which implies the calculation and storage of the intermediate tensors of the auxiliary activations, significantly increasing the memory footprint.

Furthermore, common implementation of VSD utlizies particle-based variational inference, where a particle means a single set of 3D parameters. To improve diversity, VSD works with multiple particles, resulting in a linear increase in memory usage for each particle. In this sense, SDS is just a special case of VSD, as shown by Wang et al ([46])

To counter the problem, initially the calculating precision of the model was reduced to half, using 16-bit floating point numbers instead of the standard 32-bit floating point numbers. While this allowed the model to be run without further modification, it raised the issue of vanishing gradients. Optimizing a material texture map can yield particularly small gradients, which can vanish if the required precision level is not granted. This results in a texture map of unusually vivid hue, unfavorable for subsequent use.

Subsequently, different versions of model offloading methods were utilized. Using an offloading approach, the actively not used components or parts of components of the model are offloaded from the CPU to the GPU. This reduces memory usage to the absolute minimum

required to train the model. However, training was still unfeasible on the available hardware.

Finally, a regression from the initially utilized Variational Score Distillation (VSD) to SDS was found to be plausible solution to the problem. While this theoretically decreases output visual fidelity, empirical testing found the generated material maps to be still plausible quality.

B. Camera Normalization

Both the Rendering Resources ([70]) and the ScanNet++ ([71]) dataset provide their own respective set of pre-established camera poses. Direct application is however unfeasible in both cases, due to different conventions being utilized, necessitating a normalization.

As discussed in section 3.1.1, during the image formation process the extrinsic parameters of the camera are used in a rigid body transformation from World Space to Camera Space. There are however multiple common conventions varying in regards of axis orientation, e.g. which axis points up; and handedness. The handedness of a coordinate system means the orientation of its third axis, typically the Z-axis, relative to the other two, X and Y axes. An incorrect processing of the camera extrinsic parameters results in skewed or inverted projections, not suitable for use as conditioning, as it leads to a misalignment between the conditioning and the projected UV texture maps.

B.1. Camera Pose Normalization in Rendering Resources

No evidence on the used convention was found in the Rendering Resources, and common convention translation, such as inverting the Y and Z axes, would yield no better results. As empirical testing was not deemed suitable, a transformation logic based on vector analysis was derived, as detailed in algorithm 1.

Algorithm 1 Compute Camera View Transform

Require: $M_{c2w} \in \mathbb{R}^{4\times4}$ ▷ Read in from dataset
Ensure: $R \in \mathbb{R}^{3\times3}, T \in \mathbb{R}^3$

$eye \leftarrow M_{c2w}[0..2, 3]$
$eye \leftarrow Unsqueeze(eye, 0)$

$R_{raw} \leftarrow M_{c2w}[0..2, 0..2]$

$\mathbf{v}_{up} \leftarrow [0, 1, 0]$
$\mathbf{v}_{fwd} \leftarrow [0, 0, 1]$

$up \leftarrow \mathbf{v}_{up} \cdot R_{raw}^{\top}$
$at \leftarrow eye + (\mathbf{v}_{fwd} \cdot R_{raw}^{\top})$

$(R, T) \leftarrow LookAtViewTransform(eye, at, up)$
return R, T

The aim of the algorithm is to exploit the already available $LookAtViewTransform$ logic, whose output translation **T** and rotation **R** matrices can be used to set the camera in the correct position and orientation of the respective image capture. To achieve this, the $LookAtViewTransform$ requires an initial position *eye*, the direction to look at *at* and the vector pointing upwards up. The *eye* vector and a raw rotation matrix R_{raw} can easily be extracted from the camera transformation matrix provided by the dataset, as its first 3×3 matrix and its last column vector. Next, the $\mathbf{v}_{up}$ and $\mathbf{v}_{fwd}$ vectors used in the rendering framework of the pipeline are initialized as $(0, 1, 0)$ and $(0, 0, 1)$ respectively. The up vector suitable for the $LookAtViewTransform$ is calculated by rotating $\mathbf{v}_{up}$ by the rotation of the camera transformation R_{raw}. Similarly, the looking direction *at* is calculated by first rotating $\mathbf{v}_{fwd}$ by R_{raw} and then translating this vector to have its origin at the camera position *eye*.

Utilizing the $LookAtViewTransform$ removes the need of discerning the coordinate system convention used by the dataset, as it robustly computes the necessary transformation from the required parameters. As the dataset did not specify custom camera intrinsic parameters, acquiring only the extrinsic parameters ensures pixel-perfect alignment between the latent renderings and the conditioning images.

B.2. Camera Pose Normalization in ScanNet++

Similarly, the Scannet++ dataset uses a camera coordinate system convention and denormalizing different than the rendering pipeline. Due to this custom denormalization, the normalization detailed in algorithm 1 cannot be directly applied.

In this case, renormalization of the extrinsic parameters can be done by re-orthogonalization and axis inversion of the respective camera pose. During testing it was identified that inverting

both the X and Z axes is necessary to align the coordinate systems. It may be noted that in such convention transformations, it is always crucial to invert an even number of axes to preserve its handedness, as the transformation matrix only keeps its determinant in this case. Only after this renormalization can we apply the transformation detailed in algorithm 1.

B.3. Camera Intrinsic Normalization in ScanNet++

The ScanNet++ dataset is a collection of real-world data, meaning the intrinsic parameters of the used camera must be taken into account when processing the provided image captures. Similarly to its extrinsic parameters, the dataset utilizes custom normalizing of the camera intrinsics, resulting in skewed or inverted renderings, or renderings of vanishingly small field of view, when leveraging the raw parameters. The way to renormalize them is by scaling with half the minimum image dimension, to maintain the correct aspect ratio in NDC space, as detailed in algorithm 2.

Algorithm 2 Camera Intrinsic Normalization

Require: $M_{intrinsics} \in \mathbb{R}^{3\times3}$; $w, h \in \mathbb{R}$ ▷ Read in from dataset
Ensure: Normalized focal length f_{norm} and principal point p_{norm}

$f_x \leftarrow M_{intrinsics}[0,0]$
$f_y \leftarrow M_{intrinsics}[1,1]$
$c_x \leftarrow M_{intrinsics}[0,2]$
$c_y \leftarrow M_{intrinsics}[1,2]$

$f \leftarrow [f_x, f_y]$
$p \leftarrow [c_x, c_y]$
$S_{wh} \leftarrow [w, h]$

$s \leftarrow \min(w, h)/2.0$
$c_0 \leftarrow S_{wh}/2.0$

$f_{norm} \leftarrow f/s$
$p_{norm} \leftarrow -(p - c_0)/s$

return f_{norm}, p_{norm}

The scale factor s is the half of the smaller image dimension, used to scale focal lengths. c_0 is the center of the image as half of both dimensions, and is used to first shift the principal point p, which is then normalized by scale factor s. The negative sign accounts for the coordinate origin shift from the top-left to the center.

C. Hyperparameter Tuning

Firstly, the learning rate was analyzed. Observation of the loss curves indicated that the convergence could be sped up, to decrease the number of iterations necessary to propagate high-frequency details from the conditioning images to the target texture maps. Consequently, the learning rate was progressively increased by a whole magnitude, from 0.001 to 0.01. As expected, this resulted in higher convergence speeds, while still not degrading the stability of training. This is assumed to be a property of the loss landscape being broader, as a larger learning rate can effectively fill in unobserved regions of the target texture map faster leveraging the diffusion prior in the early stages, letting the model avoid stagnating in blurry local minimas. This observation allowed the number of training iterations to be reduced to almost half of the original value; from 30′000 steps to only 17′000. Graphs detailing the training convergence are documented in figure C.1. This behavior suggests that the loss landscape for this specific texture optimization is relatively convex or possesses a "flat minimum," where the magnitude of the step size primarily affects speed rather than the quality of the final solution.

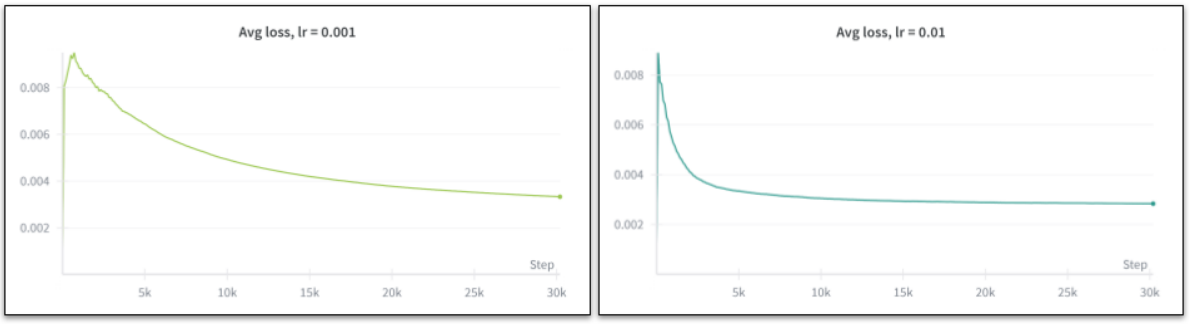


Figure C.1.: **Graphs detailing the average loss achieved during optimizing the albedo texture map for Benedikt Bitterli's Country Kitchen scene**. The graph on the left showcases a training run with the learning rate set to 0.001, while the one on the right has the value set to 0.01, a result of progressive hyperparameter tuning. As apparent on the right graph, raising the learning rate speeds up training convergence considerably while not sacrificing stability, allowing the training to be stopped after around the 17′000th timestep, essentially cutting optimization time in half.

Other hyperparameters analyzed were the lower end of the range of the diffusion time-schedule and the initialization data of the target texture map. Experiments showed that switching the texture initialization to utilize a zero-tensor instead of the original random

valued tensor has significant effect on training stability, with considerably faster convergence but no significant improvement in output fidelity. Detrimental effect over training convergence speed was observed from the starting point of the diffusion time-schedule t_{start} and no change on the visual fidelity of the results. Increasing this value from the default $t_{start} = 0.2$ increased initial error and provided slightly slower convergence. Inherently, increasing t_{start} also narrows down the solution domain. Therefore, the default $t_{start} = 0.2$ is used. In figure C.2, qualitative and quantitative evaluations are shown on the effects of texture initialization and time-schedule start t_{start}.

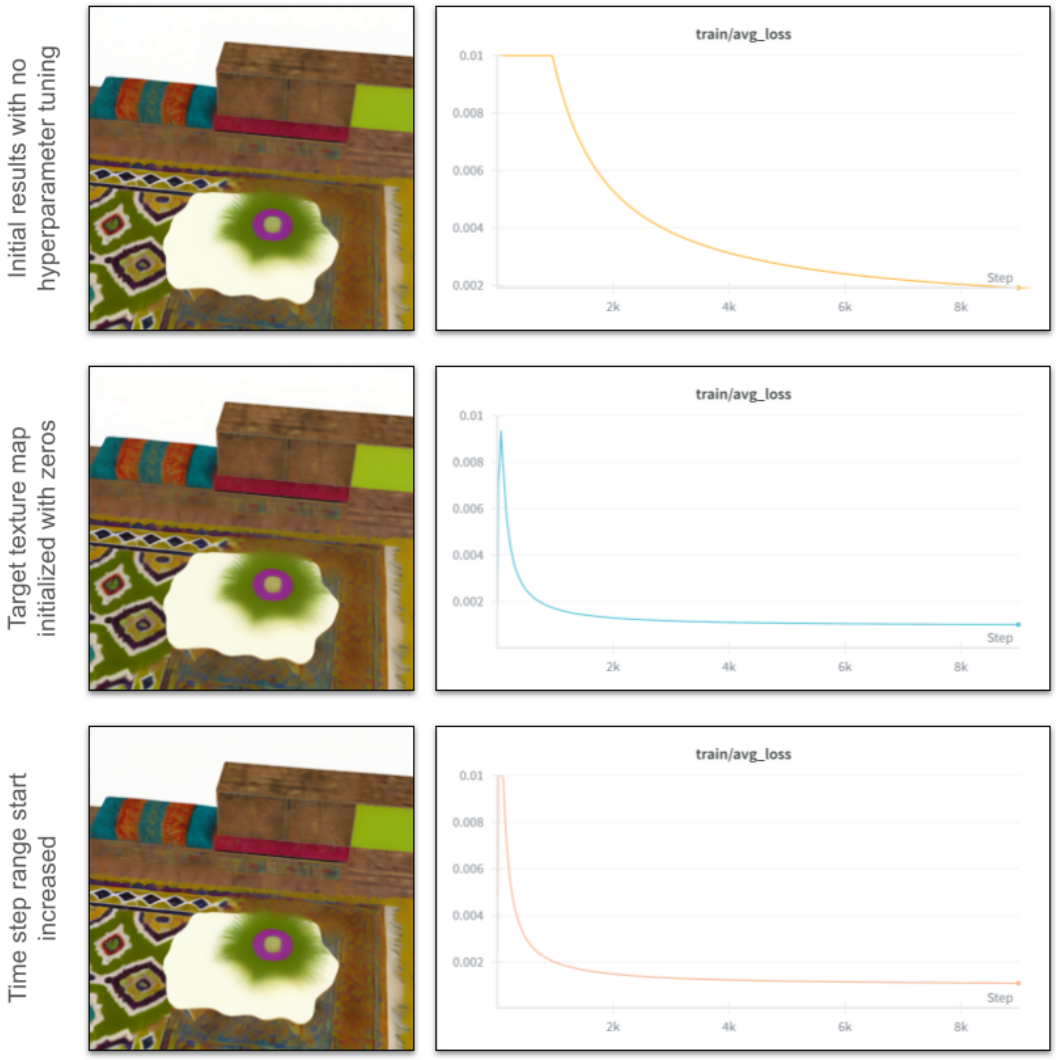


Figure C.2.: **Qualitative and quantitative assessment of the effect of tuning texture initialization and the lower end of the diffusion timestep range t_{start} on optimizing albedo material on a test scene**. Initializing the target material maps to zero instead of random values yielded significantly faster training convergence but no increase in visual fidelity. Increasing t_{start} slightly degraded the speed of the optimization and produced initially higher errors.

D. Underlying Texture Representation

Following other state-of-the-art approaches, a number of different texture representation such as hashgrids or simple RGB maps were experimented with. While hashgrids can provide higher quality especially in the domain of higher frequency textures, the proposed pipeline opts to use a simple RGB representation and utilizes no further Multilayer Perceptrons during rendering process, to ensure compatibility with standard rendering pipelines while still yielding plausible results. This led to a significant architectural incompatibility being discovered in initial implementation concerning the anchor embedding module. Anchor embedding is a mechanism addressing the challenge of multi-view consistency by associating each geometric vertex of the 3D scene with a local texture coordinate mapped to a canonical embedding vector. Subsequently, the differentiable rendering process leverages these local texture coordinates by bilinearly interpolating the anchor embeddings. This ensures that the changes made to a single spatial location are applied across all visible camera views, enforcing spatial coherence. However, it was determined that the incompatibility was specific to the interaction between the anchor embedding mechanism and the chosen rendering approach and underlying RGB texture representation. Consequently, the anchor embedding was permanently disabled to ensure pipeline stability and convergence, without harshly sacrificing the quality of the generated output.

Bibliography

[1] J. Lin, X. Yang, M. Chen, Y. Xu, D. Yan, L. Wu, X. Xu, L. XU, S. Zhang, and Y.-C. Chen. *Kiss3DGen: Repurposing Image Diffusion Models for 3D Asset Generation*. 2025. arXiv: 2503.01370 [cs.GR]. URL: https://arxiv.org/abs/2503.01370.

[2] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang. *MVDream: Multi-view Diffusion for 3D Generation*. 2024. arXiv: 2308.16512 [cs.CV]. URL: https://arxiv.org/abs/2308.16512.

[3] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. *DreamFusion: Text-to-3D using 2D Diffusion*. 2022. arXiv: 2209.14988 [cs.CV]. URL: https://arxiv.org/abs/2209.14988.

[4] A. Nichol, H. Jun, P. Dhariwal, P. Mishkin, and M. Chen. *Point-E: A System for Generating 3D Point Clouds from Complex Prompts*. 2022. arXiv: 2212.08751 [cs.CV]. URL: https://arxiv.org/abs/2212.08751.

[5] D. Z. Chen, H. Li, H.-Y. Lee, S. Tulyakov, and M. Nießner. *SceneTex: High-Quality Texture Synthesis for Indoor Scenes via Diffusion Priors*. 2023. arXiv: 2311.17261 [cs.CV]. URL: https://arxiv.org/abs/2311.17261.

[6] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or. "TEXTure: Text-Guided Texturing of 3D Shapes". In: *ACM SIGGRAPH 2023 Conference Proceedings* (2023). URL: https://api.semanticscholar.org/CorpusID:256597953.

[7] D. Z. Chen, Y. Siddiqui, H.-Y. Lee, S. Tulyakov, and M. Nießner. *Text2Tex: Text-driven Texture Synthesis via Diffusion Models*. 2023. arXiv: 2303.11396 [cs.CV]. URL: https://arxiv.org/abs/2303.11396.

[8] L. Wu, R. Zhu, M. B. Yaldiz, Y. Zhu, H. Cai, J. Matai, F. Porikli, T.-M. Li, M. Chandraker, and R. Ramamoorthi. *Factorized Inverse Path Tracing for Efficient and Accurate Material-Lighting Estimation*. 2023. arXiv: 2304.05669 [cs.CV]. URL: https://arxiv.org/abs/2304.05669.

[9] C.-H. Lin, J.-B. Huang, Z. Li, Z. Dong, C. Richardt, T. Li, M. Zollhöfer, J. Kopf, S. Wang, and C. Kim. *IRIS: Inverse Rendering of Indoor Scenes from Low Dynamic Range Images*. 2025. arXiv: 2401.12977 [cs.CV]. URL: https://arxiv.org/abs/2401.12977.

[10] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. "Microfacet Models for Refraction through Rough Surfaces." In: *Rendering techniques* 2007 (2007), 18th.

[11] B. Burley and W. D. A. Studios. "Physically-based shading at disney". In: *Acm siggraph*. Vol. 2012. 2012. vol. 2012. 2012, pp. 1–7.

[12] E. Garces, C. Rodriguez-Pardo, D. Casas, and J. Lopez-Moreno. *A Survey on Intrinsic Images: Delving Deep Into Lambert and Beyond*. 2021. arXiv: 2112.03842 [cs.CV]. URL: https://arxiv.org/abs/2112.03842.

[13] H. G. Barrow and J. M. Tenenbaum. "RECOVERING INTRINSIC SCENE CHARACTERISTICS FROM IMAGES". In: 1978. URL: https://api.semanticscholar.org/CorpusID:14892007.

[14] E. H. Land and J. J. McCann. "Lightness and retinex theory". In: *Journal of the Optical society of America* 61.1 (1971), pp. 1–11.

[15] R. Grosse, M. Johnson, E. Adelson, and W. Freeman. *Ground truth dataset and baseline evaluations for intrinsic image algorithms*. Nov. 2009. DOI: 10.1109/ICCV.2009.5459428.

[16] B. Horn. "Determining Lightness from Image". In: *Computer Graphics and Image Processing* 3 (Dec. 1974), pp. 277–299. DOI: 10.1016/0146-664X(74)90022-7.

[17] C. Careaga and Y. Aksoy. "Intrinsic Image Decomposition via Ordinal Shading". In: *ACM Trans. Graph.* (2023).

[18] C. Careaga and Y. Aksoy. "Colorful Diffuse Intrinsic Image Decomposition in the Wild". In: *ACM Transactions on Graphics* 43.6 (Nov. 2024), pp. 1–12. ISSN: 1557-7368. DOI: 10.1145/3687984. URL: http://dx.doi.org/10.1145/3687984.

[19] S. Bell, K. Bala, and N. Snavely. "Intrinsic images in the wild". In: *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301. DOI: 10.1145/2601097.2601206. URL: https://doi.org/10.1145/2601097.2601206.

[20] H. Chung, J. Kim, and J. C. Ye. *Diffusion models for inverse problems*. 2025. arXiv: 2508.01975 [cs.LG]. URL: https://arxiv.org/abs/2508.01975.

[21] A. Nichol and P. Dhariwal. *Improved Denoising Diffusion Probabilistic Models*. 2021. arXiv: 2102.09672 [cs.LG]. URL: https://arxiv.org/abs/2102.09672.

[22] Z. Zeng, V. Deschaintre, I. Georgiev, Y. Hold-Geoffroy, Y. Hu, F. Luan, L.-Q. Yan, and M. Hašan. "RGBX: Image decomposition and synthesis using material- and lighting-aware diffusion models". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. SIGGRAPH '24. ACM, July 2024, pp. 1–11. DOI: 10.1145/3641519.3657445. URL: http://dx.doi.org/10.1145/3641519.3657445.

[23] K. Perlin. "An image synthesizer". In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), pp. 287–296. ISSN: 0097-8930. DOI: 10.1145/325165.325247. URL: https://doi.org/10.1145/325165.325247.

[24] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, S. Worley, W. Mark, and J. Hart. *Texturing and Modeling: A Procedural Approach: Third Edition*. Dec. 2002, pp. 1–687.

[25] A. Efros and T. Leung. "Texture Synthesis by Non-parametric Sampling". In: *Proceedings of the IEEE International Conference on Computer Vision* 2 (Mar. 2000).

[26] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. "Graphcut textures: image and video synthesis using graph cuts". In: 22.3 (July 2003), pp. 277–286. ISSN: 0730-0301. DOI: 10.1145/882262.882264. URL: https://doi.org/10.1145/882262.882264.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: https://arxiv.org/abs/2103.00020.

[28] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski. *StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery*. 2021. arXiv: 2103.17249 [cs.CV]. URL: https://arxiv.org/abs/2103.17249.

[29] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff. *VQGAN-CLIP: Open Domain Image Generation and Editing with Natural Language Guidance*. 2022. arXiv: 2204.08583 [cs.CV]. URL: https://arxiv.org/abs/2204.08583.

[30] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[31] M. Arjovsky, S. Chintala, and L. Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML]. URL: https://arxiv.org/abs/1701.07875.

[32] T. Karras, S. Laine, and T. Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: 1812.04948 [cs.NE]. URL: https://arxiv.org/abs/1812.04948.

[33] G. Metzer, E. Richardson, O. Patashnik, R. Giryes, and D. Cohen-Or. *Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures*. 2022. arXiv: 2211.07600 [cs.CV]. URL: https://arxiv.org/abs/2211.07600.

[34] Y. Siddiqui, J. Thies, F. Ma, Q. Shan, M. Nießner, and A. Dai. *Texturify: Generating Textures on 3D Shape Surfaces*. 2022. arXiv: 2204.02411 [cs.CV]. URL: https://arxiv.org/abs/2204.02411.

[35] X. Zeng, X. Chen, Z. Qi, W. Liu, Z. Zhao, Z. Wang, B. Fu, Y. Liu, and G. Yu. *Paint3D: Paint Anything 3D with Lighting-Less Texture Diffusion Models*. 2023. arXiv: 2312.13913 [cs.CV]. URL: https://arxiv.org/abs/2312.13913.

[36] S. R. Marschner. "Inverse rendering for computer graphics". AAI9839924. PhD thesis. USA, 1998. ISBN: 0591937395.

[37] S. Choi, H.-G. Chung, Y. Jeon, G. Nam, and S.-H. Baek. *A Real-world Display Inverse Rendering Dataset*. 2025. arXiv: 2508.14411 [cs.GR]. URL: https://arxiv.org/abs/2508.14411.

[38] T. Moi, A. Cibicik, and T. Rølvåg. "Digital twin based condition monitoring of a knuckle boom crane: An experimental study". In: *Engineering Failure Analysis* 112 (2020), p. 104517. ISSN: 1350-6307. DOI: https://doi.org/10.1016/j.engfailanal.2020.104517. URL: https://www.sciencedirect.com/science/article/pii/S1350630719315742.

[39] F. M. Chaudhry, J. Ralli, J. Leudet, F. Sohrab, F. Pakdaman, P. Corbani, and M. Gabbouj. *Deep-BrownConrady: Prediction of Camera Calibration and Distortion Parameters Using Deep Learning and Synthetic Data*. 2025. arXiv: 2501.14510 [cs.CV]. URL: https://arxiv.org/abs/2501.14510.

[40] S. Wu, S. Basu, T. Broedermann, L. V. Gool, and C. Sakaridis. *PBR-NeRF: Inverse Rendering with Physics-Based Neural Fields*. 2025. arXiv: 2412.09680 [cs.CV]. URL: https://arxiv.org/abs/2412.09680.

[41] X. Yan, J. Xu, Y. Huo, and H. Bao. *Neural Rendering and Its Hardware Acceleration: A Review*. 2024. arXiv: 2402.00028 [cs.GR]. URL: https://arxiv.org/abs/2402.00028.

[42] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. P. A. Lensch. *NeRD: Neural Reflectance Decomposition from Image Collections*. 2021. arXiv: 2012.03918 [cs.CV]. URL: https://arxiv.org/abs/2012.03918.

[43] H. Kato, D. Beker, M. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. *Differentiable Rendering: A Survey*. 2020. arXiv: 2006.12057 [cs.CV]. URL: https://arxiv.org/abs/2006.12057.

[44] X. Wei, Z. Liu, P. Li, and Y. Luximon. *SIR: Multi-view Inverse Rendering with Decomposable Shadow Under Indoor Intense Lighting*. 2025. arXiv: 2402.06136 [cs.CV]. URL: https://arxiv.org/abs/2402.06136.

[45] K. Kim, A. Torii, and M. Okutomi. "Multi-view Inverse Rendering Under Arbitrary Illumination and Albedo". In: *European Conference on Computer Vision*. 2016. URL: https://api.semanticscholar.org/CorpusID:46180918.

[46] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. *ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation*. 2023. arXiv: 2305.16213 [cs.LG]. URL: https://arxiv.org/abs/2305.16213.

[47] P. Kocsis, V. Sitzmann, and M. Nießner. *Intrinsic Image Diffusion for Indoor Single-view Material Estimation*. 2024. arXiv: 2312.12274 [cs.CV]. URL: https://arxiv.org/abs/2312.12274.

[48] T. Möller and E. Haines. *Real-Time Rendering, Second Edition*. Ak Peters Series. Taylor & Francis, 2002. ISBN: 9781568811826. URL: https://books.google.de/books?id=9YhRAAAAMAAJ.

[49] E. Hemayed. "A survey of camera self-calibration". In: *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, 2003.* 2003, pp. 351–357. DOI: 10.1109/AVSS.2003.1217942.

[50] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 22.11 (2000), pp. 1330–1334.

[51] M. Pharr, W. Jakob, and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN: 0128006455.

[52] F. E. Nicodemus. "Directional Reflectance and Emissivity of an Opaque Surface". In: *Appl. Opt.* 4.7 (July 1965), pp. 767–775. DOI: 10.1364/AO.4.000767. URL: https://opg.optica.org/ao/abstract.cfm?URI=ao-4-7-767.

[53] C. Zhou, A. Sztrajman, G. Rainer, F. Zhong, F. Gokbudak, Z. Guo, W. Xia, R. Mantiuk, and C. Oztireli. *Physically Based Neural Bidirectional Reflectance Distribution Function*. 2024. arXiv: 2411.02347 [cs.GR]. URL: https://arxiv.org/abs/2411.02347.

[54] D. P. Greenberg. "A framework for realistic image synthesis". In: *Commun. ACM* 42.8 (Aug. 1999), pp. 44–53. ISSN: 0001-0782. DOI: 10.1145/310930.310970. URL: https://doi.org/10.1145/310930.310970.

[55] E. Veach. "Robust monte carlo methods for light transport simulation". AAI9837162. PhD thesis. Stanford, CA, USA, 1998. ISBN: 0591907801.

[56] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. "Diffusion Models: A Comprehensive Survey of Methods and Applications". In: *ACM Computing Surveys* 56 (Sept. 2023). DOI: 10.1145/3626235.

[57] J. Ho, A. Jain, and P. Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG]. URL: https://arxiv.org/abs/2006.11239.

[58] J. Luo, L. Yang, Y. Liu, C. Hu, G. Wang, Y. Yang, T.-L. Yang, and X. Zhou. *Review of diffusion models and its applications in biomedical informatics*. 2025. PMID: 2006.11239. URL: https://pubmed.ncbi.nlm.nih.gov/41121196/.

[59] X. Wang, Z. He, and X. Peng. "Artificial-Intelligence-Generated Content with Diffusion Models: A Literature Review". In: *Mathematics* 12.7 (2024). ISSN: 2227-7390. DOI: 10.3390/math12070977. URL: https://www.mdpi.com/2227-7390/12/7/977.

[60] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. *Flow Matching for Generative Modeling*. 2023. arXiv: 2210.02747 [cs.LG]. URL: https://arxiv.org/abs/2210.02747.

[61] T. Xie, Y. Zhu, L. Yu, T. Yang, Z. Cheng, S. Zhang, X. Zhang, and C. Zhang. *Reflected Flow Matching*. 2024. arXiv: 2405.16577 [stat.ML]. URL: https://arxiv.org/abs/2405.16577.

[62] A. Karnewar, A. Vedaldi, D. Novotny, and N. Mitra. *HoloDiffusion: Training a 3D Diffusion Model using 2D Images*. 2023. arXiv: 2303.16509 [cs.CV]. URL: https://arxiv.org/abs/2303.16509.

[63] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. *DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation*. 2024. arXiv: 2309.16653 [cs.CV]. URL: https://arxiv.org/abs/2309.16653.

[64] Y. Guo, X. Zuo, P. Dai, J. Lu, X. Wu, Y. Yan, S. Xu, X. Wu, et al. "Decorate3d: text-driven high-quality texture generation for mesh decoration in the wild". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 36664–36676.

[65] J. Ho and T. Salimans. *Classifier-Free Diffusion Guidance*. 2022. arXiv: 2207.12598 [cs.LG]. URL: https://arxiv.org/abs/2207.12598.

[66] T. Nguyen-Phuoc, C. Li, S. Balaban, and Y.-L. Yang. *RenderNet: A deep convolutional network for differentiable rendering from 3D shapes*. 2019. arXiv: 1806.06575 [cs.CV]. URL: https://arxiv.org/abs/1806.06575.

[67] D. P. Kingma and M. Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. ISSN: 1935-8245. DOI: 10.1561/2200000056. URL: http://dx.doi.org/10.1561/2200000056.

[68] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV]. URL: https://arxiv.org/abs/1505.04597.

[69] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2022. arXiv: 2112.10752 [cs.CV]. URL: https://arxiv.org/abs/2112.10752.

[70] B. Bitterli. *Rendering resources*. https://benedikt-bitterli.me/resources/. 2016.

[71] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai. *ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes*. 2023. arXiv: 2308.11417 [cs.CV]. URL: https://arxiv.org/abs/2308.11417.