RefinedFields: Radiance Fields Refinement for Planar Scene Representations

Anonymous authors
Paper under double-blind review

Abstract

Planar scene representations have recently witnessed increased interests for modeling scenes from images, as their lightweight planar structure enables compatibility with image-based models. Notably, K-Planes have gained particular attention as they extend planar scene representations to support in-the-wild scenes, in addition to object-level scenes. However, their visual quality has recently lagged behind that of state-of-the-art techniques. To reduce this gap, we propose RefinedFields, a method that leverages pre-trained networks to refine K-Planes scene representations via optimization guidance using an alternating training procedure. We carry out extensive experiments and verify the merit of our method on synthetic data and real tourism photo collections. RefinedFields enhances rendered scenes with richer details and improves upon its base representation on the task of novel view synthesis. Our code is publicly available as open-source.

1 Introduction

(a) RefinedFields Rendering (b) K-Planes (c) RefinedFields (d) GT

Figure 1: **Qualitative Results.** Given images of the Trevi fountain from Phototourism (Jin et al., 2020), as well as a pre-trained model (Rombach et al., 2022), our method leverages the pre-trained model and refines K-Planes with finer details that are under-represented when optimizing the same K-Planes on the images alone.

To draw novel objects and views, humans often rely on a blend of **cognition** and **intuition**, where the latter is built on a large prior acquired from a long-term continuous exploration of the visual world. Nevertheless, ablating one of these two elements results in catastrophic representations. On the one hand, humans find it particularly difficult to draw a bicycle based solely on this preconceived prior (Gimini, 2016). However, once

one photograph is observed, drawing novel views of a bicycle becomes straightforward. On the other hand, drawing monuments and complex objects based solely on observed images, and with no preconceived notions of geometry and physics, is also non-trivial. In computer vision, recent methods tackling object generation and novel view synthesis generally focus on either the former or the latter ablation.

The first class of methods approaches novel view synthesis by learning scenes through rigorous **cognition**, as in dense observations of captured images. Although classic methodologies like structure-from-motion (Hartley & Zisserman, 2004) and image-based rendering (Shum et al., 2008) have previously tackled this problem, the field has recently seen substantial advancements thanks to stereo reconstruction techniques (Wang et al., 2024), as well as implicit neural representation methods (Mildenhall et al., 2020) which this paper focuses on. Particularly, planar scene representations such as Tri-Planes (Chan et al., 2022) have recently gained significant attention (Shue et al., 2023; Lan et al., 2024) for their lightweight planar structure that allows for a seamless integration with image-based models. Recent methods extend NeRFs (Martin-Brualla et al., 2021) and Tri-Planes (Fridovich-Keil et al., 2023, K-Planes) to support learning from unconstrained "in-the-wild" photo collections by enabling robustness against illumination variations and transient occluders. These representations, however, do not learn any prior across scenes as they are trained from scratch for each scene. This means that these representations are learned in a closed-world setting, where the information scope is limited to the training set at hand. Despite the wide interest in planar scene representations, their visual quality currently falls short behind recent state-of-the-art methods (Kulhanek et al., 2024; Xu et al., 2024). Our goal is to integrate pre-trained networks into the training framework of K-Planes, with an aim to improve their rendering performance.

The second class of methods tackles novel view synthesis and object generation by learning and leveraging priors over images and scenes, reminiscent of drawing from insights and **intuition**. These methods have also recently witnessed accelerated advancements. Recent works leverage pre-trained networks to achieve Object Generation (OG) (Poole et al., 2023; Metzer et al., 2023) and Novel View Synthesis (NVS) (Yu et al., 2021; Jain et al., 2021; Liu et al., 2023; Melas-Kyriazi et al., 2023). Particularly, Liu et al. (2023) achieve NVS from single images only by simply fine-tuning a pre-trained latent diffusion model (Rombach et al., 2022). This proves pivotal significance related to large-scale pre-trained vision models, as it shows that, although trained on 2D data, these models learn a rich geometric 3D prior about the visual world. Nevertheless, as these pre-trained models alone have no explicit multi-view geometric constraints, their use for 3D applications is usually prone to geometric issues (e.g. geometric inconsistencies, multi-face Janus problem, content drift issues (Shi et al., 2023, Figure 1)). This class of methods has not yet been explored to enhance implicit models representing in-the-wild scenes, as leveraging priors over such representations is not evident.

RefinedFields proposal. Our work builds on the previous discourse and aims to enhance planar scene modeling by leveraging pre-trained networks. To this end, we present an alternating training algorithm that integrates a novel *scene refining* stage, which aims to propose a better optimization initialization by utilizing a pre-trained model. We adopt K-Planes (Fridovich-Keil et al., 2023) as a base scene representation, which extend Tri-Planes to support in-the-wild scenes, in addition to object-level scenes. Liu et al. (2024) highlight that features in planar scene representations resemble projected scene images, a finding we also corroborate in Appendix B, and exploit to improve upon K-Planes. Specifically, RefinedFields *refines* planar scene representations by projecting them onto the space of representations inferable by a pre-trained network, which pushes K-Planes features to more closely resemble real-world images. To do so, we build on the seamless integration of K-Planes with image-based models and present an alternating training procedure that iteratively switches between optimizing a K-Planes representation on images from a particular dataset, and fine-tuning a pre-trained network to output a new conditioning leading to a refined version of this K-Planes representation. Overall, this procedure guides the optimization of a particular scene, by leveraging not only the training dataset at hand but also the rich prior lying within the weights of the pre-trained model, which is a first for in-the-wild scene modeling.

We conduct extensive quantitative and qualitative evaluations of RefinedFields. We show that our method improves upon K-Planes with richer details in scene renderings. We prove via ablation studies that this added value indeed comes from the fine-tuned prior of the pre-trained network. Figure 1 illustrates the improvements our method showcases on the *Trevi fountain* scene from Phototourism (Jin et al., 2020).

Table 1: **Related work overview.** RefinedFields leverages a pre-trained prior (Rombach et al., 2022) on the scene representation to refine K-Planes, our underlying scene representation, utilized for novel view synthesis in the wild (NVS-W).
†Non exhaustive, other works with characteristics similar to NeRF-W exist. Refer to Section 2 for more details.

	No 3D supervision	Pre-trained prior	Geometric consistency	In-the-wild scene modeling	Underlying representation	Task
NFD (Shue et al., 2023)	✗	✗	✓	✗	Tri-Planes	OG
3DGen (Gupta et al., 2023)	✗	✗	✓	✗	Tri-Planes	
Latent-NeRF (Metzer et al., 2023)	✓	✓	✓	✗	NeRF	
DreamFusion (Poole et al., 2023)	✓	✓	✓	✗	NeRF	
NeRF (Mildenhall et al., 2020)	✓	✗	✓	✗	NeRF	NVS
RealFusion (Melas-Kyriazi et al., 2023)	✓	✓	✓	✗	NeRF	
DiffusioNeRF (Wynn & Turmukhambetov, 2023)	✓	✗	✓	✗	NeRF	
NerfDiff (Gu et al., 2023)	✗	✓	✓	✗	Tri-Planes	
3DiM (Watson et al., 2023)	✗	✗	✓	✗	—	
Zero-1-to-3 (Liu et al., 2023)	✗	✓	✗	✗	—	
NeRF-W (Martin-Brualla et al., 2021)	✓	✗	✓	✓	NeRF	NVS-W†
WildGaussians (Kulhanek et al., 2024)	✓	✗	✓	✓	3DGS	
Splatfacto-W (Xu et al., 2024)	✓	✗	✓	✓	3DGS	
K-Planes (Fridovich-Keil et al., 2023)	✓	✗	✓	✓	K-Planes	
RefinedFields (ours)	✓	✓	✓	✓	K-Planes	

2 Related Work

RefinedFields achieves geometrically consistent novel view synthesis, that can also be applied in-the-wild, by leveraging K-Planes, and a large-scale pre-trained network. Our method is the first to satisfy all of these attributes, as summarized in Table 1. In this section, we develop the various preceding works from which our method takes inspiration.

Neural representations. Neural rendering (Tewari et al., 2020) has seen significant advancements since the introduction of Neural Radiance Fields (Mildenhall et al., 2020, NeRF). At its core, NeRF learns a scene by fitting the weights of a neural network on posed images of said scene. This subsequently enables the reconstruction of the scene thanks to volume rendering (Kajiya & Herzen, 1984). Subsequent to NeRFs, various scene representations have been introduced (Chen et al., 2022a; Müller et al., 2022; Kerbl et al., 2023). Particularly, Chan et al. (2022) introduce Tri-Planes, a planar scene representation serving as a middle ground between implicit and explicit representations, enabling a faster learning of scenes. Tri-Planes have been widely adopted in recent works (Shue et al., 2023; Lan et al., 2024; Wang et al., 2023b), as their planar structure enables a seamless integration with image-based models.

Neural representations in-the-wild. Subsequent to NeRFs, several techniques emerged to extend the NeRF setup to "in-the-wild" unconstrained photo collections (Jin et al., 2020, Phototourism) plagued by illumination variations and transient occluders. This added variability makes learning a scene particularly challenging, as surfaces can exhibit significant visual disparities across views. NeRF-W (Martin-Brualla et al., 2021) addresses the challenge of novel view synthesis in-the-wild (NVS-W) by modeling scene lighting through appearance embeddings, and transient occluders through an additional transient head. Splatfacto-W (Xu et al., 2024) proposes a Nerfstudio (Tancik et al., 2023) implementation of 3D Gaussian Splatting (Kerbl et al., 2023, 3DGS) that extends standard 3DGS to support in-the-wild scene modeling via appearance

embeddings. WildGaussians (Kulhanek et al., 2024) also proposes using 3DGS with appearance embeddings for in-the-wild scene modeling, but additionally predicts uncertainty masks to exclude transient occluders from the loss computation. It is important to note that although this work adopts a pre-trained network, it is solely used for the computation of the loss masks on the data, and not as a prior on the scene representation. This approach is orthogonal to our approach of leveraging a prior to directly improve upon a specific scene representations, as it acts on the data and not the scene representation itself. Fridovich-Keil et al. (2023) present K-Planes, which modify and extend Tri-Planes (Chan et al., 2022) to in-the-wild scenes thanks to learnable appearance embeddings, similarly to Martin-Brualla et al. (2021). Our work aims to improve upon K-Planes representations by extending their training beyond closed-world setups using pre-trained networks. Note that other works also tackle NVS-W: Chen et al. (2022b) model scene lighting through appearance embeddings and transient occluders through transient embeddings, Yang et al. (2023) leverage interactive information across rays to mimic the perception of humans in-the-wild, and Chen et al. (2024) separate static and transient components by utilizing heuristics-guided segmentation. However, these works limit their training and evaluations to downscaled versions of Phototourism, which makes their results not directly comparable with ours.

Priors in neural representations. The integration of pre-trained priors for downstream tasks has emerged as a prominent trend, as they enable the effective incorporation of extrinsic knowledge into diverse applications. For neural representations, priors have been utilized for few-shot scene modeling (Yu et al., 2021; Jain et al., 2021; Deng et al., 2023; Gao et al., 2024) as well as generative tasks (Shue et al., 2023; Poole et al., 2023; Watson et al., 2023; Gao et al., 2024) for object generation and novel view synthesis. In this realm, denoising diffusion probabilistic models (Ho et al., 2020; Rombach et al., 2022) have recently gained particular attention for their application as plug-and-play priors (Graikos et al., 2022). They have been utilized in various domains such as super-resolution (Wang et al., 2023a) and more specifically novel view synthesis. Liu et al. (2023) fine-tune Stable Diffusion (Rombach et al., 2022), a pre-trained latent diffusion model for 2D images, to learn camera controls over a 3D dataset and thus performing NVS by generalizing to other objects. These results hold paramount value, as they highlight the rich 3D prior learned by Stable Diffusion, even though it has only been trained on 2D images. This however comes with geometric inconsistency issues across views, as a pre-trained model alone has no explicit multi-view geometric constraints. In this work, we aim to leverage a pre-trained model to enhance a volumetric scene representation, which inherently adheres to consistency constraints, hence mitigating multi-view consistency issues. The utilization of such priors to enhance in-the-wild scene representation has been an unexplored area of research.

3 Method

To guide the optimization of planar scene representations with extrinsic signals, we learn a scene through two alternating stages, as illustrated in Figure 2. *Scene fitting* optimizes our K-Planes representation $\mathbf{P}_\gamma$ to reproduce the images in the training set, as traditionally done in neural rendering techniques. *Scene refining* finetunes a pre-trained network to this K-Planes representation, and then infers a new one $\mathbf{P}_\varepsilon$, which will subsequently be corrected by scene fitting. The main idea behind this is that we use our 3D implicit model $\mathbf{P}_\gamma$ for optimizing the scene to the available information in the training set and adhere to essential geometric constraints, and then project this scene representation on the set of scenes inferable by the pre-trained network, making it closer to natural images. In this section, we detail each stage and elucidate the intuition behind our method.

3.1 Scene Fitting

The goal at this stage is to fit a scene, adhering to pre-defined geometric constraints, from posed RGB images. To fit the scene, we adopt the K-Planes representation (Fridovich-Keil et al., 2023). As such, this stage corresponds to optimizing K-Planes to fit a scene, from which we adapt the code.

K-Planes are compact 3D model representations applicable to static scenes, "in-the-wild" scenes (scenes with varying appearances), and dynamic scenes. These models allow for fast training and rendering, while

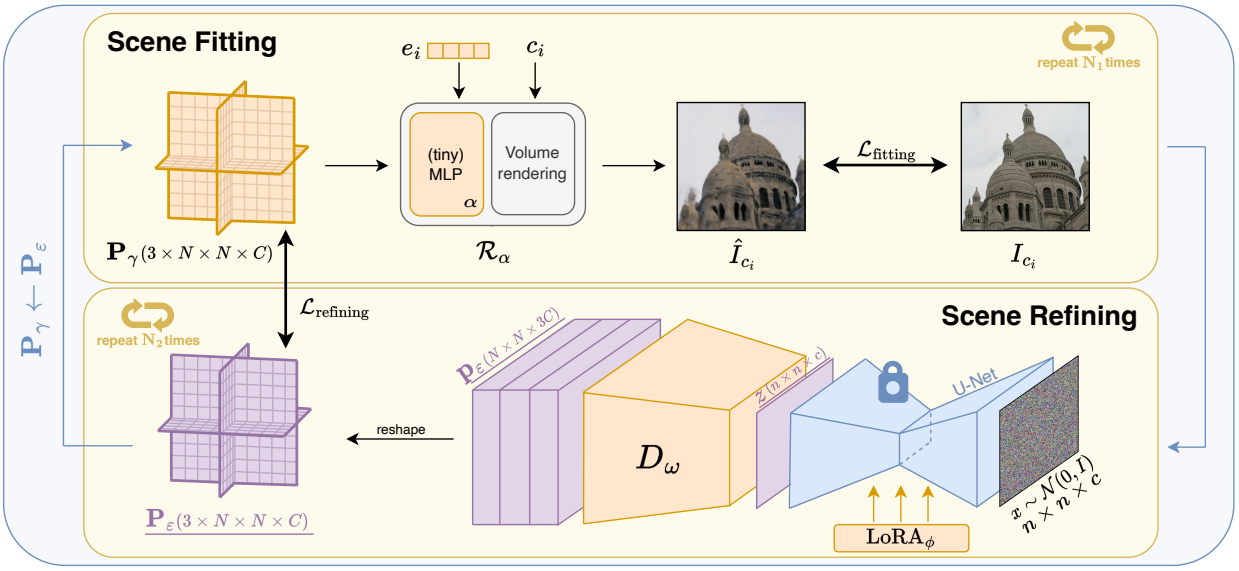


Figure 2: **Scene learning procedure.** The K-Planes $\mathbf{P}_\gamma$, the MLP with trainable parameters α, and the appearance embeddings e_i are learned during scene fitting. The LoRA parameters ϕ as well as the decoder D_w are learned during scene refining. The pre-trained U-Net is frozen. Assets in violet and underlined are intermediate results. At each iteration, new planes $\mathbf{P}_\varepsilon$ are inferred and assigned to $\mathbf{P}_\gamma$, which are then corrected by scene fitting.

maintaining low-memory usage. K-Planes model a d-dimensional scene with $k = \binom{d}{2}$ planes, which represent the combinations of every pair of dimensions. This structure makes K-Planes compatible with a multitude of neural network architectures, and more particularly image-specialized network architectures. This enables K-Planes inference by minimally tweaking image architectures. For a static 3D scene, $k = 3$ and the planes represent the xy, xz, and yz planes. These planes, each of size $N \times N \times C$, encapsulate features representing the density and view-dependent colors of the scene.

The K-Planes model $\mathbf{P}_\gamma$ is originally randomly initialized. The first goal of scene fitting is then to correct this random initialization to fit the training set. Note that the first iteration of scene fitting is especially particular, since it is starting with a randomly initialized scene, as opposed to a *proposed* scene, as we describe in Section 3.2.

To render the 3D scene from K-Planes, as done by Mildenhall et al. (2020) and Fridovich-Keil et al. (2023), we cast rays from the desired camera position through the coordinate space of the scene, on which we sample 3D points. We decode the corresponding RGB color for each 3D point $\mathbf{q} = (i, j, k)$ by normalizing it to $[0, N)$ and projecting it onto the $k = 3$ planes, denoted as $\mathbf{P}_\gamma^{(xy)}, \mathbf{P}_\gamma^{(xz)}, \mathbf{P}_\gamma^{(yz)}$:

$$f^{(h)}(\mathbf{q}) = \psi(\mathbf{P}_\gamma^{(h)}, \pi^{(h)}(\mathbf{q})) \ , \tag{1}$$

where $h \in \mathbf{H} = \{xy, xz, yz\}$, $\pi^{(h)}(\mathbf{q})$ projects $\mathbf{q}$ onto $\mathbf{P}_\gamma^{(h)}$, and ψ denotes bilinear interpolation on a regular 2D grid.

These features are then aggregated using the Hadamard product to produce a single feature vector of size M:

$$f(\mathbf{q}) = \prod_{h \in \mathbf{H}} f^{(h)}(\mathbf{q}) \ . \tag{2}$$

To decode these features, we adopt the hybrid formulation of K-Planes (Fridovich-Keil et al., 2023). Two small Multi-Layer Perceptrons (MLPs), g_σ and g_{RGB}, map the aggregated features as follows:

$$\begin{aligned} \sigma(\mathbf{q}), \hat{f}(\mathbf{q}) &= g_\sigma(f(\mathbf{q})) \ , \\ c(\mathbf{q}, \mathbf{d}) &= g_{\text{RGB}}(\hat{f}(\mathbf{q}), \gamma(\mathbf{d})) \ , \end{aligned} \tag{3}$$

where $\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{L-1}\pi p),\ \cos(2^{L-1}\pi p))$ is the positional embedding of p. g_σ maps the K-Planes features into density σ and additional features $\hat{f}$. Subsequently, g_{RGB} maps $\hat{f}$ and the positionally-encoded view directions $\gamma(\mathbf{d})$ into view-dependent RGB colors. This enforces densities to be independent of view directions.

These decoded RGB colors are then used to render the final image thanks to ray marching and integrals from classical volume rendering (Kajiya & Herzen, 1984), that are practically estimated using quadrature:

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))c_i \ , \\ T_i &= \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \ , \end{aligned} \tag{4}$$

where $\hat{C}(\mathbf{r})$ is the expected color, T_i is the accumulated transmittance along the ray, and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples.

3.2 Scene Refining

This section presents the core of our method, which consists of proposing better optimization initializations for the scene fitting stage. Given a fitted scene representation $\mathbf{P}_\gamma$, this stage consists of learning this fitted implicit representation and proposing a new *refined* representation $\mathbf{P}_\varepsilon$. Formally, this stage consists of projecting our K-Planes $\mathbf{P}_\gamma$ on the set $\mathbb{Q}$ of K-Planes inferable by a low-rank fine-tuning of the pre-trained model:

$$\mathbf{P}_\varepsilon = \underset{P \in \mathbb{Q}}{\arg\min}\ \|\mathbf{P}_\gamma - P\|_2^2 \ . \tag{5}$$

As K-Planes feature channels show similar structure to images (Appendix B), this projection pushes the K-Planes to be even more similar in structure to real images, more particularly to orthogonal projections of the scene on the planes. Figures 5 to 7 illustrate a comparison between feature planes at the end of our optimization and those of standard K-Planes. RefinedFields leads to feature planes exhibiting sharper details, which ultimately lead to more refined details in scene renderings, as proven by our experiments (Figure 3 and Table 3).

To provide scene refining with a rich prior, we employ a large-scale pre-trained latent diffusion model, as these networks exhibit great performances as priors for downstream tasks, and share similar properties to our planar representation, both in terms of shape and distribution (Appendix B). More particularly, we adopt Stable Diffusion (Rombach et al., 2022, SD) for its proven performances for downstream 3D (Liu et al., 2023) and 2D (Wang et al., 2023a) tasks. Thus, we integrate the U-Net $\mathbf{SD}_\phi$ and the decoder $\mathbf{D}_\omega$ into our pipeline, and treat the K-Planes as $3C$-channel $N \times N$ images. We also replace the last layer of the decoder $\mathbf{D}_\omega$ with a randomly initialized convolutional layer (with no bias), to take into account the shape of the K-Planes. We then fine-tune the pre-trained model using the fitted K-Planes $\mathbf{P}_\gamma$, and infer refined K-Planes $\mathbf{P}_\varepsilon = \mathbf{D}_w(\mathbf{SD}_\phi(x))$ where $x \sim \mathcal{N}(0, I)$ is sampled once at the beginning of the training. Note that this is different from the multi-step generation process of diffusion model inference, as we only apply the inference at the last time-step of our diffusion model. This is key as our goal here is not to learn distributions over scenes and sample them for generation, but to adapt the pre-trained network and leverage the information already learned within its weights to infer K-Planes closest to representing the scene at hand.

To achieve the fine-tuning of our pre-trained network, a significant challenge presents itself: due to the sheer size of Stable Diffusion, it would be too costly to fine-tune all of its trainable parameters. Moreover, as we only want to modulate priors embedded into the pre-trained network, we look for an alternative to doing full fine-tuning. To circumvent these constraints, we adopt Low-Rank Adaptation (Hu et al., 2022, LoRA), a simple yet effective parameter-efficient fine-tuning method that has proven great transfer capabilities across modalities and tasks (Fan et al., 2023; Lee et al., 2023; Zeng & Lee, 2023). LoRA's relatively minimal design works directly over weight tensors, which means that it can be seamlessly applied to most model architectures. Furthermore, LoRA does not add any additional cost at inference, thanks to its structural re-parameterization design. To achieve this, Hu et al. (2022) inject trainable low-rank decomposition matrices

Algorithm 1 Alternating training algorithm.

Input: N_{epochs}, N_1, N_2, N, C, n, c, $\mathcal{I} = \{I_{c_i}, c_i\}$, $\mathcal{R}_\alpha$, $\mathbf{D}_w$, $\mathbf{SD}_\phi$, optimizer
$x \leftarrow$ standard-gaussian(n, n, c)
$\mathbf{P}_\gamma \leftarrow$ standard-gaussian$(N, N, 3C)$
for N_{epochs} steps **do**
  // scene fitting
  for N_1 steps **do**
    $\gamma, \alpha \leftarrow$ optimizer.step$(\mathcal{L}_{\text{fitting}}(\mathbf{P}_\gamma, \mathcal{I}))$
  end for
  // scene refining
  for N_2 steps **do**
    $\mathbf{P}_\varepsilon \leftarrow \mathbf{D}_w(\mathbf{SD}_\phi(x))$
    $\omega, \phi \leftarrow$ optimizer.step$(\mathcal{L}_{\text{refining}}(\mathbf{P}_\varepsilon, \mathbf{P}_\gamma))$
  end for
  $\mathbf{P}_\gamma \leftarrow \mathbf{P}_\varepsilon$
end for

into each layer of a frozen pre-trained model. Let $\mathbf{W}_0$, $\mathbf{b}_0$ be the frozen pre-trained weights and biases, and x be the input. Fine-tuning a frozen linear layer $f(x) = \mathbf{W}_0 x + \mathbf{b}_0$ comes down to learning the low-rank decomposition weights $\Delta\mathbf{W} = \mathbf{BA}$:

$$f(x) = (\mathbf{W}_0 + \Delta\mathbf{W})x + \mathbf{b}_0 \tag{6}$$

where $\mathbf{W}_0, \Delta\mathbf{W} \in \mathbb{R}^{d\times k}$; $\mathbf{B} \in \mathbb{R}^{d\times r}$; $\mathbf{A} \in \mathbb{R}^{r\times k}$; and the rank $r \ll \min(d, k)$.

Thus, to implement scene refining, we fine-tune the LoRA parameters ϕ modulating the pre-trained U-Net, as well as the decoder's parameters ω, on the fitted scene $\mathbf{P}_\gamma$. Subsequently, we query the U-Net with Gaussian noise x, decode its intermediary output latent z with $\mathbf{D}_\omega$, and infer $\mathbf{p}_\varepsilon$ that is reshaped into a new *refined* scene $\mathbf{P}_\varepsilon$. Finally, $\mathbf{P}_\varepsilon$ is proposed to *scene fitting* as an improved initialization to be optimized. For an in-depth inspection of the feature planes, we refer the reader to Appendix B.

3.3 Training

We define an alternating training procedure rotating between scene fitting and scene refining, as described above, and as illustrated in Figure 2.

For *scene fitting*, we train the K-Planes model as proposed by Fridovich-Keil et al. (2023). We use spatial total variation regularization to encourage smooth gradients. This is applied over all the spatial dimensions of each plane in the representation:

$$\mathcal{L}_{\text{TV}}(\mathbf{P}) = \frac{1}{|C|N^2} \sum_{c,i,j} (\|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i-1,j}\|_2^2 + \|\mathbf{P}_c^{i,j} - \mathbf{P}_c^{i,j-1}\|_2^2) \ . \tag{7}$$

For scenes with varying lighting conditions (e.g. *in-the-wild* scenes as in the Phototourism dataset (Jin et al., 2020)), an M-dimensional appearance vector e_i is additionally optimized for each image. This vector is then passed as input to the MLP color decoder g_{RGB} at the rendering step $\mathcal{R}_\alpha$. Hence, the training objective for scene fitting is written as:

$$\min_{\alpha,\gamma} \mathcal{L}_{\text{fitting}} \triangleq \|\mathcal{R}_\alpha(\mathbf{P}_\gamma, \mathbf{C}) - I_{\mathbf{C}}\|_2^2 + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}}(\mathbf{P}_\gamma) \ , \tag{8}$$

where $\mathcal{R}_\alpha$ represents the K-Planes rendering procedure (i.e. ray marching, feature decoding via a small MLP with trainable parameters α, and volume rendering), $\mathbf{P}_\gamma$ are the K-Planes with trainable parameters γ and $I_{\mathbf{C}}$ is a ground truth RGB image with camera position $\mathbf{C}$.

As for the *scene refining* phase, we optimize the decoder parameters w as well as the LoRA parameters ϕ, modulating the frozen U-Net weights, on the fitted scene $\mathbf{P}_\gamma$. Thus, the fine-tuning objective for *scene*

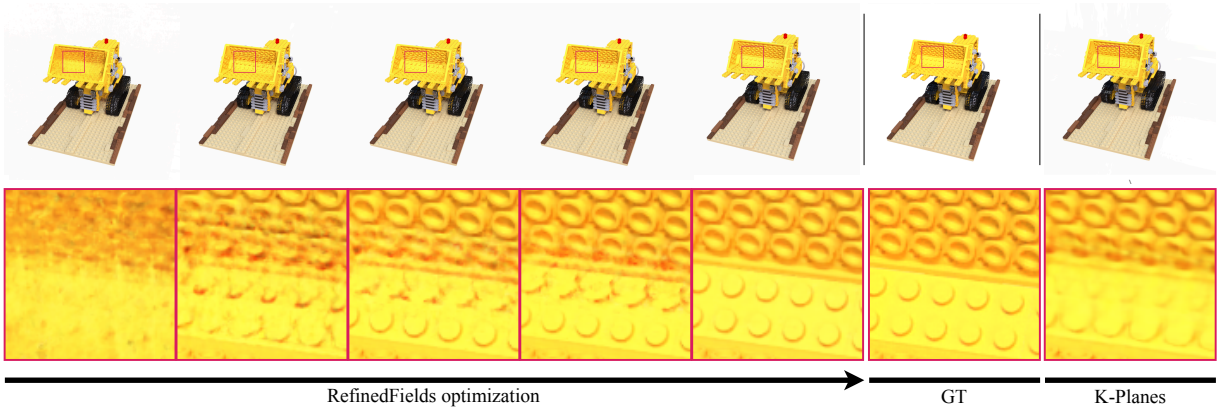


Figure 3: **Case study.** Qualitative results on the Lego scene from the NeRF synthetic dataset (Mildenhall et al., 2020) showcasing the optimization progression on RefinedFields, and a comparison with the ground truth and K-Planes. The training set is constrained to 50% of its initial size for both RefinedFields and K-Planes. RefinedFields refines the K-Planes representation enabling the proper reconstruction of details in the scene. At the end of optimization, the Mean Squared Error (MSE) for RefinedFields is 3.46×10^{-4}, while the one for K-Planes is 4.36×10^{-4}.

refining is written as:

$$\min_{w,\phi} \mathcal{L}_{\text{refining}} \triangleq \|\mathbf{D}_w(\mathbf{SD}_\phi(x)) - \mathbf{P}_\gamma\|_2^2 \ , \tag{9}$$

where $x \sim \mathcal{N}(0, I)$ is fixed during scene refining, $\mathbf{SD}_\phi$ is the frozen Stable Diffusion model modulated by LoRA with trainable parameters ϕ, and $\mathbf{D}_w$ is the latent K-Planes decoder. After this optimization, $\mathbf{P}_\gamma$ is reassigned as $\mathbf{D}_w(\mathbf{SD}_\phi(x))$ and passed to scene fitting. Note that, thanks to the alternating nature of our training and the absence of bias in the decoder's convolutional layers, this optimization does not overfit the model to produce exactly $\mathbf{P}_\gamma$, which is key as it would lead to resuming scene fitting from exactly the same point.

At the end of the alternating training procedure, we save the refined and corrected representation $\mathbf{P}_\gamma$ for rendering and testing. We refer the reader to Algorithm 1 for an overview of our training procedure.

4 Experiments

We start by assessing RefinedFields via an experiment on a case study. We then evaluate RefinedFields on synthetic scenes (Mildenhall et al., 2020) and real-world Phototourism (Jin et al., 2020) scenes, where we showcase the improvements our method exhibits relative to our K-Planes base representation. Quantitative results can be found in Tables 2 and 3, where we report for each experiment the Peak Signal-to-Noise Ratio (PSNR) for pixel-level similarity, the Structural Similarity Index Measure (SSIM) for structural-level similarity, and the Learned Perceptual Image Patch Similarity (Zhang et al., 2018, LPIPS) for perceptual similarity. RefinedFields demonstrates an improved performance compared to K-Planes on the task of novel view synthesis. For a further look, experimental details including hyperparameters and more dataset details can be found in Appendix A. Additional qualitative results on synthetic and real-world scenes are available in Appendix C.

4.1 Datasets

We evaluate our method similarly to prior work (Mildenhall et al., 2020; Martin-Brualla et al., 2021; Fridovich-Keil et al., 2023) in novel view synthesis, by adopting the *Real Synthetic 360°* dataset (Mildenhall et al., 2020) for synthetic scenes and the same three scenes of cultural monuments from the Phototourism dataset (Jin et al., 2020) for real-world scenes: *Brandenburg Gate*, *Sacré Coeur*, and *Trevi Fountain*. Additional dataset details can be found in Appendix A.1.

Table 2: **Quantitative results.** Results on static synthetic scenes (Mildenhall et al., 2020). The **bold** and underlined entries respectively indicate the best and second-best results. Dashes denote values that were not reported in prior work. Our method outperforms K-Planes, our main baseline, on the task of novel view synthesis for synthetic scenes.

	PSNR (↑)								
	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (Mildenhall et al., 2020)	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65	31.00
TensoRF (Chen et al., 2022a)	35.76	26.01	**33.99**	**37.41**	36.46	**30.12**	34.61	30.77	33.14
Plenoxels (Fridovich-Keil et al., 2022)	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62	31.71
INGP (Müller et al., 2022)	35.00	**26.02**	33.51	37.40	36.39	29.78	**36.22**	31.10	**33.18**
K-Planes (Fridovich-Keil et al., 2023)	34.98	25.68	31.44	36.75	35.81	29.48	34.10	30.76	32.37
K-Planes-SS (Fridovich-Keil et al., 2023)	33.61	25.27	30.92	35.88	35.09	28.83	33.01	30.04	31.58
RefinedFields (ours)	**35.77**	25.94	32.45	37.08	**36.47**	29.39	34.77	**31.41**	32.91

	SSIM (↑)								
	Chair	Drums	Ficus	Hotdog	Lego	Materials	Mic	Ship	Mean
NeRF (Mildenhall et al., 2020)	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856	0.947
TensoRF (Chen et al., 2022a)	**0.985**	0.937	**0.982**	**0.982**	0.983	**0.952**	0.988	0.895	**0.963**
Plenoxels (Fridovich-Keil et al., 2022)	0.977	0.933	0.976	0.980	0.975	0.949	0.985	0.890	0.958
INGP (Müller et al., 2022)	—	—	—	—	—	—	—	—	—
K-Planes (Fridovich-Keil et al., 2023)	0.983	**0.938**	0.975	**0.982**	0.982	0.950	0.988	0.897	0.962
K-Planes-SS (Fridovich-Keil et al., 2023)	0.974	0.932	0.971	0.977	0.978	0.943	0.983	0.887	0.956
RefinedFields (ours)	**0.985**	0.937	0.980	0.981	**0.984**	0.945	**0.989**	**0.903**	**0.963**

4.2 Implementation Details

For a fair comparison, we take similar experimental settings in scene fitting to Fridovich-Keil et al. (2023). However, due to the nature of our scene refining pipeline, we limit the implementation in our case to a single-scale K-Planes of 512×512 resolution, in contrast to the multi-scale approach taken by Fridovich-Keil et al. (2023) where $N \in \{64, 128, 256, 512\}$. The number of channels in each plane remains the same ($C = 32$). Moreover, throughout all the experiments, we consider the hybrid implementation of K-Planes, where plane features are decoded into colors and densities by a small MLP. As for the scene refining pipeline, we apply no modification to the U-Net in Stable Diffusion. Yet, we replace the last layer of the decoder D_ω with a new convolutional layer (without bias) to account for the shape of the K-Planes. Hence, the dimensions used in scene refining (Figure 2) are: $N = 512$, $C = 32$, $n = 64$, and $c = 4$. For an in-depth look at our frameworks and hyperparameter settings, we refer the reader to Appendices A.2 and A.3.

4.3 Evaluations

Baselines. For both synthetic and in-the-wild scenes, we primarily compare our method to our K-Planes baseline, highlighting the improvements it brings to planar scene representations. Note that, for a fair comparison, and to assess the added value of our refining pipeline with respect to our base representation (Section 4.2), we also include a single-scale ablation of K-Planes with $N = 512$ (dubbed K-Planes-SS). In order to provide a more comprehensive perspective, we also illustrate the NVS performances of other recent works that do not employ planar scene representations for synthetic (Mildenhall et al., 2020; Chen et al., 2022a; Fridovich-Keil et al., 2022; Müller et al., 2022) and in-the-wild (Mildenhall et al., 2020; Martin-Brualla et al., 2021; Xu et al., 2024; Kulhanek et al., 2024) scenes.

Comparisons. We start by testing our method against K-Planes on a **case study** consisting of the Lego scene from the NeRF synthetic dataset. Here, we train both methods on half of the training set as to deliberately produce a lower-quality fitted scene. As illustrated in Figure 3 our method refines K-Planes and exhibits better quantitative and qualitative results thanks to our scene refining pipeline.

Table 3: **Quantitative results.** Results on three real-world datasets from Phototourism (Jin et al., 2020). Our method shows notable improvements compared to K-Planes on the task of NVS-W.
†Results from public implementation (Aoi, 2022) reproduced by Fridovich-Keil et al. (2023).

	Brandenburg Gate			Sacré Coeur			Trevi Fountain		
	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)	PSNR (↑)	SSIM (↑)	LPIPS (↓)
NeRF	18.90	0.8159	0.231	15.60	0.7155	0.291	16.14	0.6007	0.366
NeRF-W†	21.32	—	—	19.17	—	—	18.61	—	—
Splatfacto-W	26.87	**0.9320**	**0.124**	22.53	**0.8760**	**0.158**	22.66	**0.7690**	**0.224**
WildGaussians	**27.77**	0.9270	0.133	**22.56**	0.8590	0.177	**23.63**	0.7660	0.228
K-Planes	25.49	0.8785	0.224	20.61	0.7735	0.265	22.67	0.7139	0.317
K-Planes-SS	24.48	0.8629	0.242	19.86	0.7419	0.312	21.30	0.6627	0.355
RefinedFields-noFinetuning	25.39	0.8834	0.206	21.41	0.8059	0.239	22.54	0.7324	0.291
RefinedFields-noPrior	25.42	0.8822	0.214	21.17	0.7978	0.248	22.16	0.7251	0.291
RefinedFields (ours)	26.64	0.8869	0.206	22.26	0.8176	0.228	23.42	0.7379	0.284

Figure 4: **Qualitative results.** Results on three scenes from Phototourism (Jin et al., 2020). Our method refines K-Planes and leads to richer and finer details in scene renderings.

We then apply our method to learn synthetic and in-the-wild scenes. In this case, RefinedFields improves upon not only K-Planes-SS but also K-Planes on the task of NVS (Tables 2 and 3). As a result, it enhances the performance of planar scene representations and brings them closer to recent works, which highlights the value of scene refining. Particularly, for synthetic scenes, RefinedFields improves upon K-Planes and sometimes even outperforms other state-of-the-art methods. For in-the-wild scenes, RefinedFields also improves K-Planes performances in a notable way. However, recent state-of-the-art methods based on Gaussian Splatting architectures (Kerbl et al., 2023) continue to outperform the refined planar representations in terms of overall quality. Figures 1 and 4 show qualitative comparisons of RefinedFields with K-Planes, showing the visual improvements brought by our refining pipeline, which brings finer details to monuments in the Phototourism scenes. Further qualitative results on synthetic and in-the-wild scenes can be found in Appendix C. We also present an inspection of K-Planes features learned by our method and K-Planes in Appendix B.

It is important to note that we do not compare our NVS metrics with other recent works (Chen et al., 2022b; Yang et al., 2023; Zhang et al., 2025). This is because these methods are trained and evaluated on downscaled versions of the Phototourism dataset, which reduces the prominence of fine details in the ground truth for monument structures. As their NVS metrics are computed relative to these ground truth, their results are not comparable to ours. Although other works do compare NVS metrics of various methods across different resolutions, we refrain from this practice as it is not an accurate comparison.

As presented, RefinedFields utilizes an alternating training procedure and a pre-trained prior to refine scene representations, leading to richer details in rendered images. While our method demonstrates promising results, this however comes with a training time increase as compared to our base representation, as our K-Planes feature projection via alternating training leads to the repeated fine-tuning of both scene fitting

and scene refining pipelines, which takes overall around 80 hours on a single NVIDIA A100 GPU. We leave the optimization of training time for future work.

4.4 Ablations

To justify our choices and explore further, we compare our in-the-wild results (Table 3) to results from two main ablations of our method. **RefinedFields-noFinetuning** is a variation of our method without LoRA fine-tuning. Here, we consider the same exact pipeline (frozen U-Net, same decoder configuration), except that we don't modulate the weights of the frozen U-Net with LoRA. This means that the prior is kept intact and no fine-tuning is done. This is to assess the role that LoRA finetuning of the pre-trained prior plays in our pipeline. Note that the decoder is still fine-tuned in this ablation, as additionally freezing it would lead to a random constant optimization initialization $\mathbf{P}_\varepsilon$, which would defeat the purpose of the refining stage. **RefinedFields-noPrior** ablates the prior of Stable Diffusion by randomly re-initializing all U-Net weights while leaving all other elements of the scene-refining pipeline intact. This ablation is done to evaluate the importance of the prior, and to verify that the observed refinements are not entirely resulting from alternate training. Note that ablating the entire scene refining pipeline leads back to the **K-Planes-SS** setting. As illustrated in Table 3, we consistently obtain worse results during the ablation study as compared to our full model, thus demonstrating the value of the pre-trained prior and of LoRA finetuning. This proves the importance of extending scene learning beyond closed-world settings, as our K-Planes projection is done via an optimization conditioning coming from a large image prior that a reasonably sized training set cannot fully capture, especially in-the-wild.

5 Conclusion

In this paper, we introduce RefinedFields, a method that refines K-Planes representations by using a pre-trained prior and an alternate training procedure. Extensive experiments show that RefinedFields exhibits notable improvements on the task of novel view synthesis compared to its K-Planes baseline. In concluding this study, several avenues of future work emerge as we consider this work to be a first stepping-stone in improving planar scene representations via conditioning with extrinsic signals. This includes the exploration of approaches to achieve this conditioning other than optimization guidance, and the application of scene refining on other representations.

Impact Statement

This paper presents work that enhances the construction of high-quality neural representations. As such, the risks associated with our work parallel those of other neural rendering papers. This includes but is not limited to privacy and security concerns, as our method is trained on a dataset of publicly captured images, where privacy-sensitive information (e.g. human faces, license plate numbers) could be present. Hence, similarly to other neural rendering approaches, there is a risk that such data could end up in the trained model if the employed datasets are not properly filtered before use. Furthermore, as our work utilizes Stable Diffusion as prior, it inherits any problematic biases and limitations this model may have.

References

Al Aoi. nerf_pl. https://github.com/kwea123/nerf_pl/tree/nerfw, 2022. Accessed: 2023-10-25.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient Geometry-Aware 3D Generative Adversarial Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16123–16133, June 2022.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *European Conference on Computer Vision (ECCV)*, 2022a.

Jiahao Chen, Yipeng Qin, Lingjie Liu, Jiangbo Lu, and Guanbin Li. NeRF-HuGS: Improved Neural Radiance Fields in Non-static Scenes Using Heuristics-Guided Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19436–19446, June 2024.

Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated Neural Radiance Fields in the Wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12943–12952, June 2022b.

Congyue Deng, Chiyu Max Jiang, Charles R. Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, and Dragomir Anguelov. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20637–20647, June 2023.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2305.16381*, 2023.

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields Without Neural Networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5501–5510, June 2022.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12479–12488, June 2023.

Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole. CAT3d: Create anything in 3d with multi-view diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=TFZlFRl9Ks.

Gianluca Gimini. Velocipedia. https://www.gianlucagimini.it/portfolio-item/velocipedia/, 2016. Accessed: 2023-10-25.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion Models as Plug-and-Play Priors. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14715–14728. Curran Associates, Inc., 2022.

Jiatao Gu, Alex Trevithick, Kai-En Lin, Josh Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. NerfDiff: Single-image View Synthesis with NeRF-guided Distillation from 3D-aware Diffusion. In *International Conference on Machine Learning*, 2023.

Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. doi: 10.1017/CBO9780511811685.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5885–5894, October 2021.

Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching Across Wide Baselines: From Paper to Practice. *International Journal of Computer Vision*, 129(2):517–547, oct 2020. doi: 10.1007/s11263-020-01385-0.

James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, 1984.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

Jonas Kulhanek, Songyou Peng, Zuzana Kukelova, Marc Pollefeys, and Torsten Sattler. WildGaussians: 3D Gaussian Splatting In the Wild. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Yushi Lan, Fangzhou Hong, Shuai Yang, Shangchen Zhou, Xuyi Meng, Bo Dai, Xingang Pan, and Chen Change Loy. LN3Diff: Scalable Latent Neural Fields Diffusion for Speedy 3D Generation. In *ECCV*, 2024.

Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*, 2023.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9298–9309, October 2023.

Ying-Tian Liu, Yuan-Chen Guo, Guan Luo, Heyi Sun, Wei Yin, and Song-Hai Zhang. PI3D: Efficient Text-to-3D Generation with Pseudo-Image Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19915–19924, June 2024.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7210–7219, June 2021.

Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. RealFusion: 360deg Reconstruction of Any Object From a Single Image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8446–8455, June 2023.

Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12663–12673, June 2023.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020.

Thomas Müller. tiny-cuda-nn, 4 2021. URL https://github.com/NVlabs/tiny-cuda-nn.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation, 2023.

J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3D Neural Field Generation Using Triplane Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20875–20886, June 2023.

H.Y. Shum, S.C. Chan, and S.B. Kang. *Image-Based Rendering*. Springer US, 2008. ISBN 9780387326689.

Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. State of the Art on Neural Rendering. *Computer Graphics Forum (Eurographics '20): State of the Art Reports*, 39(2):701 – 727, May 2020.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models, 2022. Accessed: 2023-10-25.

Jianyi Wang, Zongsheng Yue, Shangchen Zhou, Kelvin CK Chan, and Chen Change Loy. Exploiting Diffusion Prior for Real-World Image Super-Resolution. In *arXiv preprint arXiv:2305.07015*, 2023a.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, June 2024.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, and Baining Guo. RODIN: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4563–4573, June 2023b.

Daniel Watson, William Chan, Ricardo Martin Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel View Synthesis with Diffusion Models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jamie Wynn and Daniyar Turmukhambetov. DiffusioNeRF: Regularizing Neural Radiance Fields With Denoising Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4180–4189, June 2023.

Congrong Xu, Justin Kerr, and Angjoo Kanazawa. Splatfacto-w: A nerfstudio implementation of gaussian splatting for unconstrained photo collections. *arXiv preprint arXiv:2407.12306*, 2024.

Yifan Yang, Shuhai Zhang, Zixiong Huang, Yubing Zhang, and Mingkui Tan. Cross-Ray Neural Radiance Fields for Novel-View Synthesis from Unconstrained Image Collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15901–15911, October 2023.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4578–4587, June 2021.

Yuchen Zeng and Kangwook Lee. The expressive power of low-rank adaptation. *arXiv preprint arXiv:2310.17513*, 2023.

Dongbin Zhang, Chuming Wang, Weitao Wang, Peihao Li, Minghan Qin, and Haoqian Wang. Gaussian in the wild: 3d gaussian splatting for unconstrained image collections. In *European Conference on Computer Vision*, pp. 341–359. Springer, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

A Experimental Details

A.1 Datasets

Synthetic dataset. For synthetic renderings, we adopt the *Real Synthetic 360°* dataset from NeRF (Mildenhall et al., 2020). This dataset consists of eight path-traced scenes containing objects exhibiting complicated geometry and realistic non-Lambertian materials. Each image is coupled with its corresponding camera parameters. Consistently with prior work, 100 images are used for training each scene and 200 images are used for testing. All images are at 800×800 pixels.

In-the-wild dataset. For in-the-wild renderings, we adopt the Phototourism dataset (Jin et al., 2020) which is commonly used for in-the-wild tasks. This dataset consists of multitudes of images of touristic landmarks gathered from the internet. Thus, these images are naturally plagued by visual discrepancies, notably illumination variation and transient occluders. Camera parameters are estimated using COLMAP (Schönberger & Frahm, 2016). All images are normalized to [0,1]. We adopt three scenes from Phototurism: *Brandenburg Gate* (1363 images), *Sacré Coeur* (1179 images), and *Trevi Fountain* (3191 images). Testing is done on a standard set that is free of transient occluders.

A.2 Frameworks

We make use of multiple frameworks to implement our method. Our Python source code (tested on version 3.7.16), based on PyTorch (Paszke et al., 2019) (tested on version 1.13.1) and CUDA (tested on version 11.6), is publicly available as open source. We also utilize Diffusers (von Platen et al., 2022) and Stable Diffusion (tested on version 1-5, main revision). K-Planes also adopt the *tinycudann* framework (Müller, 2021). We run all experiments on a single NVIDIA A100 GPU.

A.3 Hyperparameters

A summary of our hyperparameters for synthetic as well as in-the-wild scenes can be found in Table 4.

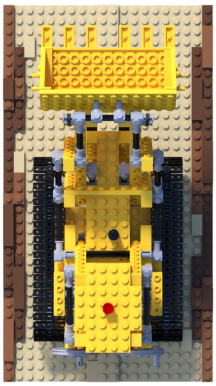
B Feature Planes Inspection

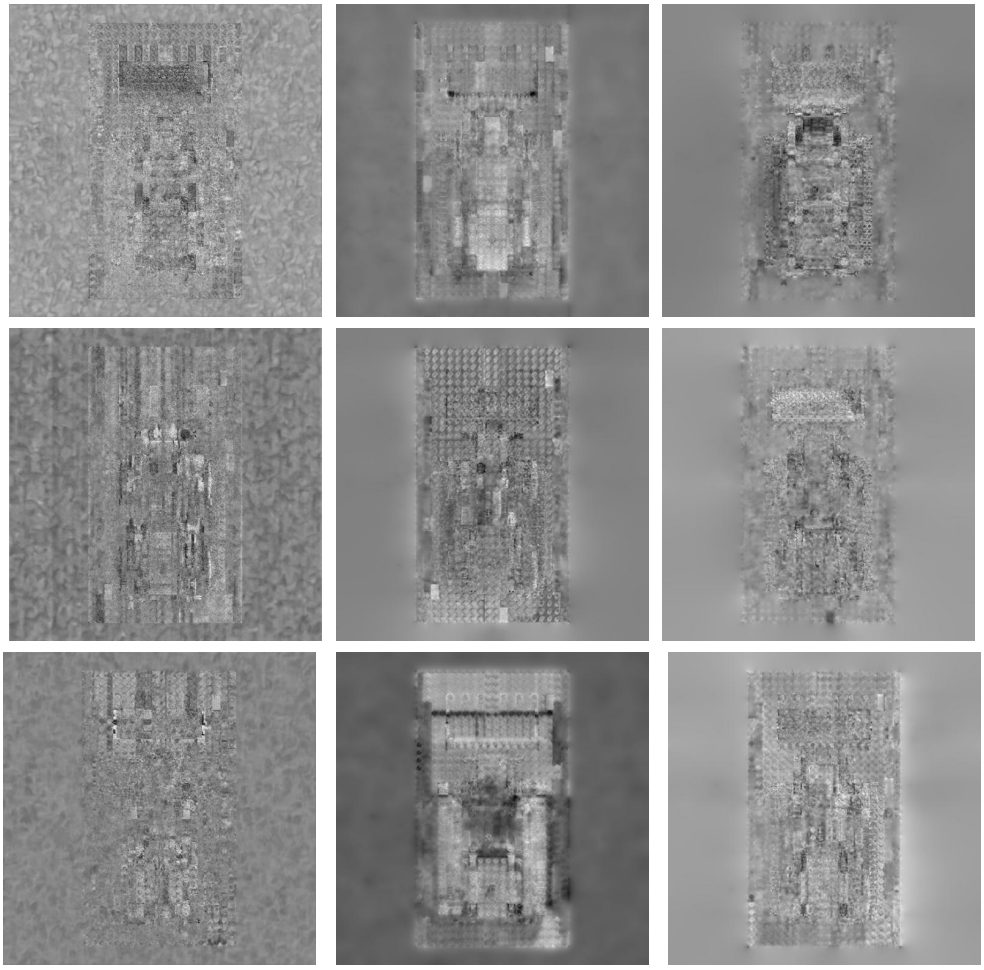
In this section, we present a visual inspection of K-Planes feature planes within different contexts. Figures 5 to 7 each correspond to one out of the three orthogonal planes. Each element in Figures 5 to 7 presents a single feature plane, picked randomly from the C feature planes. Figures 5a, 6a and 7a represent the state of the planes at an intermediate stage of the RefinedFields optimization process. Figures 5b, 6b and 7b represent the state of the planes at the end of the RefinedFields optimization. Figures 5c, 6c and 7c represent the planes at the end of the K-Planes-SS optimization (no refining is done in this case).

Two noteworthy observations emerge. First, as seen in Figures 5 to 7, K-Planes feature planes are very similar in structure to real images. In fact, these feature planes depict orthogonal projections of the scene onto the planes. These findings are especially compelling, as they justify the appropriate choice of Stable Diffusion as the pre-trained prior for the refining stage, and provide insight onto the quantitative and qualitative results showcased by our method. Second, a comparison between columns [5b, 6b and 7b] and [5c, 6c and 7c] highlights the impact scene refining has on the feature planes themselves, as planes 5b, 6b and 7b exhibit details that are more similar in structure to the scene, and that are sharper than planes 5c, 6c and 7c.

C Supplementary Results

We present below additional qualitative results for in-the-wild scenes (Figures 8 to 10) and synthetic scenes (Figures 11 and 12).



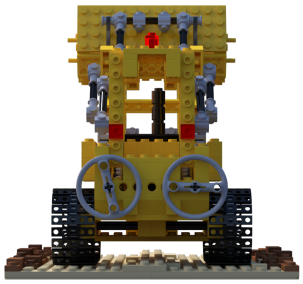


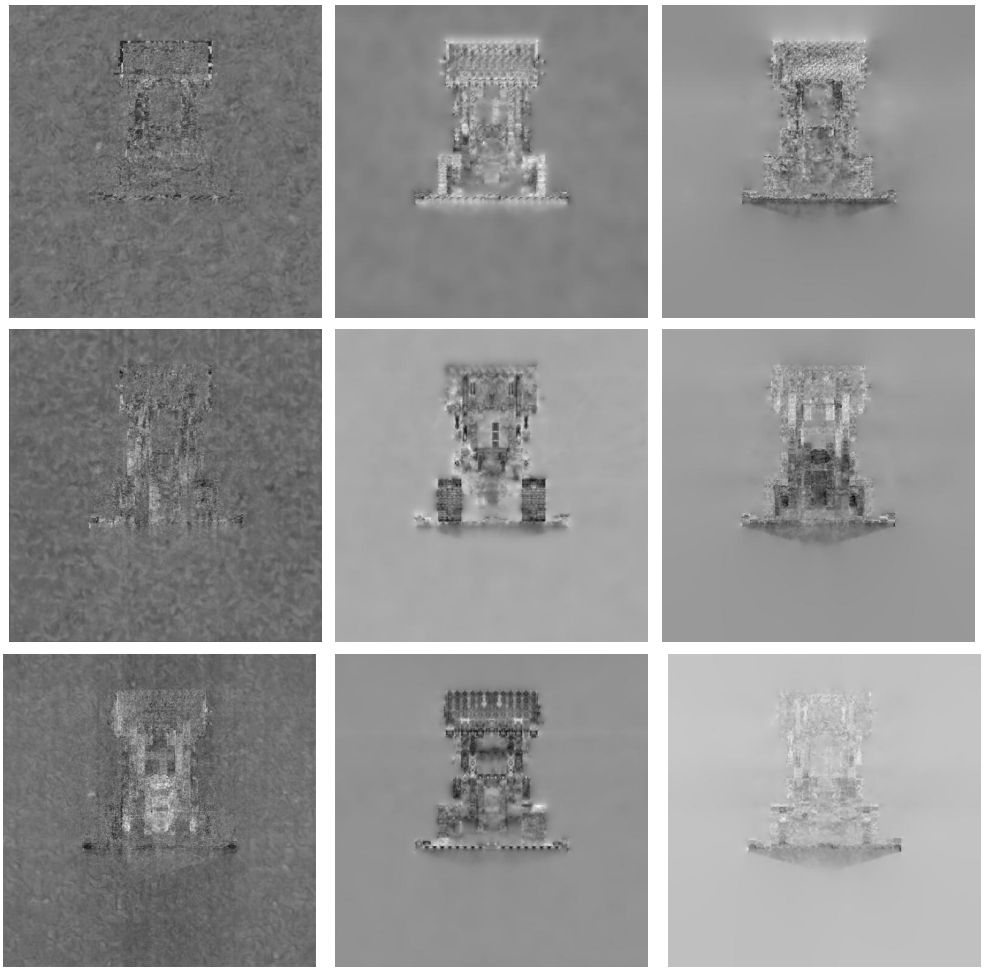
(a) RefinedFields (In-Process)

(b) RefinedFields (Final)

(c) K-Planes

Figure 5: **Feature planes inspection.** Visualization of the (xy) K-Planes feature planes during the RefinedFields optimization process (5a), at the end of the RefinedFields optimization (5b), and a comparison with vanilla K-Planes-SS (5c). Feature planes within the (xy) K-Planes are picked randomly.



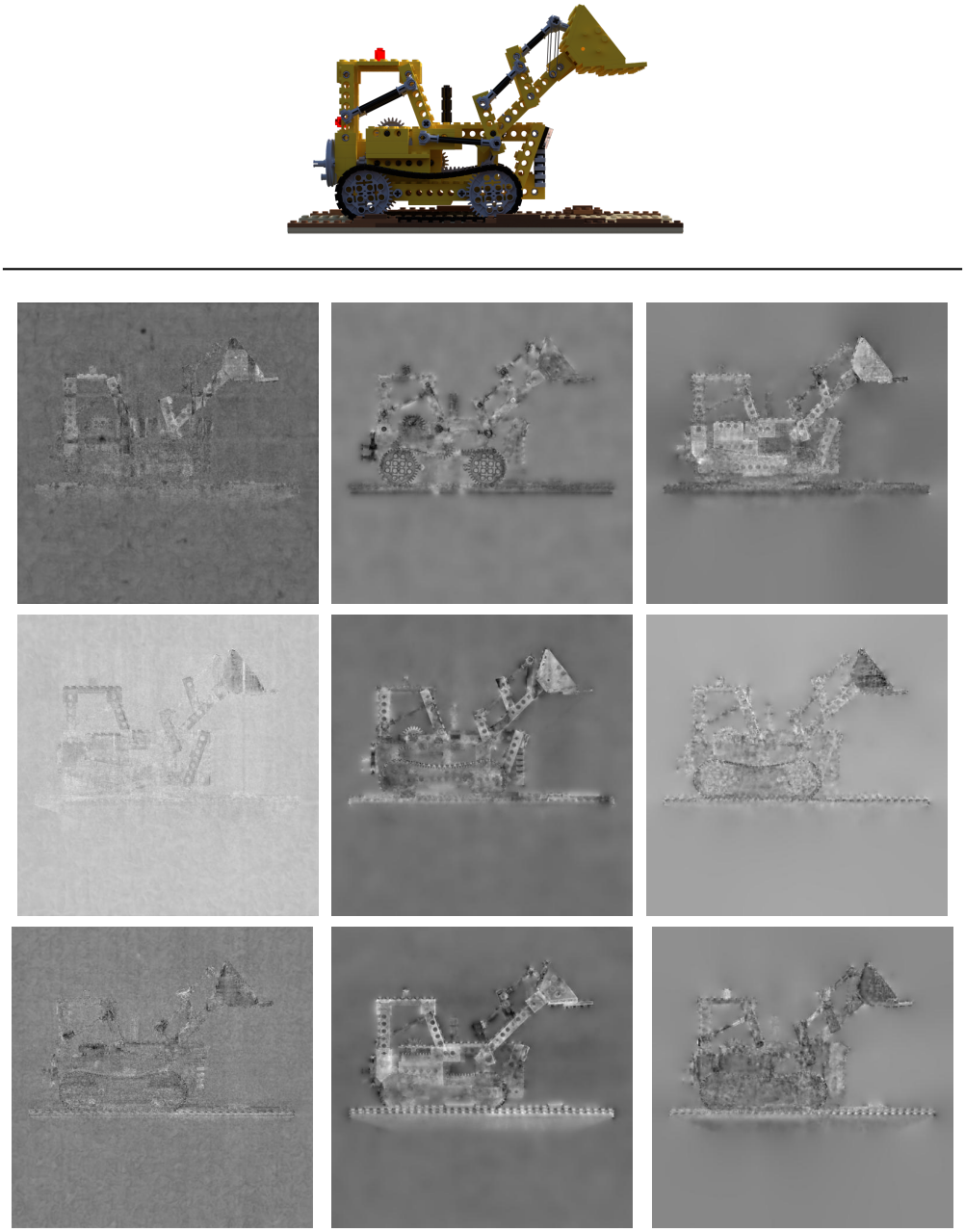


(a) RefinedFields (In-Process)

(b) RefinedFields (Final)

(c) K-Planes

Figure 6: **Feature planes inspection.** Visualization of the (xz) K-Planes feature planes during the RefinedFields optimization process (6a), at the end of the RefinedFields optimization (6b), and a comparison with vanilla K-Planes-SS (6c). Feature planes within the (xz) K-Planes are picked randomly.



(a) RefinedFields (In-Process)

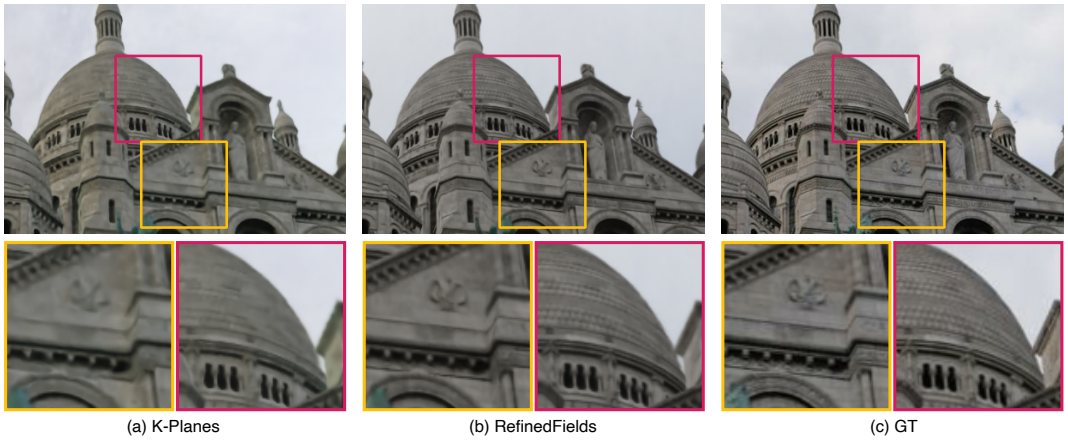
(b) RefinedFields (Final)

(c) K-Planes

Figure 7: **Feature planes inspection.** Visualization of the (yz) K-Planes feature planes during the RefinedFields optimization process (7a), at the end of the RefinedFields optimization (7b), and a comparison with vanilla K-Planes-SS (7c). Feature planes within the (yz) K-Planes are picked randomly.

(a) K-Planes (b) RefinedFields (c) GT

Figure 8: **Qualitative results.** Results on the *Brandenburg Gate* scene from Phototourism (Jin et al., 2020).



(a) K-Planes (b) RefinedFields (c) GT

Figure 9: **Qualitative results.** Results on the *Sacré Coeur* scene from Phototourism (Jin et al., 2020).

(a) K-Planes (b) RefinedFields (c) GT

Figure 10: **Qualitative results.** Results on the *Trevi Fountain* scene from Phototourism (Jin et al., 2020).

Figure 11: **Qualitative results.** RefinedFields results on the NeRF Synthetic scenes.

Figure 12: **Ground Truth Renderings.** Ground truth images from the NeRF Synthetic dataset.

Table 4: **Hyperparameters.** A summary of the hyperparameters used to train our model. Appearance optimizations only apply for in-the-wild training.
*Note that this parameter is taken differently from Fridovich-Keil et al. (2023), as we only work with single-scale planes. We consider the highest plane resolution from the multi-scale approach taken by Fridovich-Keil et al. (2023).

Hyperparameter	Value
Epochs (N_{epochs})	200 (synthetic) 20 (*Sacré Coeur*) 20 (*Brandenburg Gate*) 10 (*Trevi Fountain*)
Fitting iterations (N_1)	30000
Refining iterations (N_2)	3000
Batch size	4096
Optimizer	Adam
Scheduler	Warmup Cosine
K-Planes Learning Rate	0.01
LoRA Learning rate	0.0001
LoRA rank (r)	4
SD latent resolution	64
SD channel dimension	4
SD prompt	“ ”
Number of planes	3
K-Planes resolution*	512
K-Planes channel dimension	32
Epochs Appearance Optimization	10
Appearance embeddings dimension	32
Appearance learning rate	0.1 (*Sacré Coeur*) 0.1 (*Trevi Fountain*) 0.001 (*Brandenburg Gate*)
Appearance batch size	512