Prompt2NeRF-PIL: Fast NeRF Generation via Pretrained Implicit Latent

Jianmeng LIU*
HKUST
jliudq@connect.ust.hk

Yuyao ZHANG*
HKUST
yzhangkp@connect.ust.hk

Zeyuan MENG*
HKUST
zmengaf@connect.ust.hk

Yu-Wing TAI
Dartmouth College
yu-wing.tai@dartmouth.edu

Chi-Keung TANG
HKUST
cktang@cs.ust.hk

Abstract

This paper explores promptable NeRF generation (e.g., text prompt or single image prompt) for direct *conditioning and* fast *generation of NeRF parameters for the underlying 3D scenes, thus undoing complex intermediate steps while providing full 3D generation with conditional control. Unlike previous diffusion-CLIP-based pipelines that involve tedious per-prompt optimizations, Prompt2NeRF-PIL is capable of generating a variety of 3D objects with a single forward pass, leveraging a pre-trained implicit latent space of NeRF parameters. Furthermore, in zero-shot tasks, our experiments demonstrate that the NeRFs produced by our method serve as semantically informative initializations, significantly accelerating the inference process of existing prompt-to-NeRF methods. Specifically, we will show that our approach speeds up the text-to-NeRF model DreamFusion [26] and the 3D reconstruction speed of the image-to-NeRF method Zero-1-to-3 [17] by 3 to 5 times.*

1. Introduction

Recent breakthrough in text-to-visual content generation has brought unprecedented success in vision-language transfer, enabling communication and collaboration among individuals with diverse backgrounds and facilitating immersive virtual experiences, especially in the gaming and film industry. We have impressive works that bridge the gap between textual inputs and 2D visual representations: Stable Diffusion [29], DALL·E [28], and GPT-4 [24] are some prominent examples, which have propelled significant progresses in Text Inversion [7], LoRA [10], and ControlNet [40] to finetune [28, 29] to achieve their specific tasks.

However, due to the scarcity of annotated 3D data and the variety of representations, text-to-3D generation remains challenging. Many works have attempted to build their pipelines upon CLIP [27]. For example, in [11, 22] the lack of data is alleviated by using CLIP to optimize the corresponding 3D models, but purely using pretrained text-image models leads to costly optimization. CLIP-Forge [30] proposes an efficient strategy to generate 3D mesh from text without using annotated text image pairs, but the out-of-distribution performance is less satisfactory. Feature-disentangled control was incorporated in [12, 20, 36, 37] using CLIP to demonstrate interpolatable latents. With the prior of pretrained text-image diffusion models such as [29], DreamFusion [26] and Magic-3D [16] can generate 3D scenes from text prompts while using CLIP to finetune NeRF or Mesh. This approach however makes speed a severe issue, since multiple back-propagations of score distillation loss and renderings are necessary. To achieve fast 3D generation, one pioneer work [5] attempted to implicitly generate 3D scenes by regressing model parameters of a given NeRF network by a diffusion model in an unconditional manner. Despite its contributions in demonstrating the feasibility of implicit generation of NeRF parameters, this first paper uses simple datasets with lack of user control.

We propose to overcome these challenges by exploring the feasibility of *directly* generating NeRF parameters with only one single forward pass from text or image prompts, with an *optional* optimizing process (accelerated by our method) using diffusion-based methods such as [26] and [17] to improve the performance on out-of-domain tasks, where the output NeRF from our model serves as a good initialization.

Our Prompt2NeRF-PIL is a simple yet effective pretraining strategy, where an encoder-decoder structure is trained to encode NeRF parameters into an implicit latent space with desirable properties such as interpolability, and then using Transformer [35] to align the prompt semantics with the pretrained implicit latent space, which enables di-

*Co-first authors, ranked by alphabetical order of first names

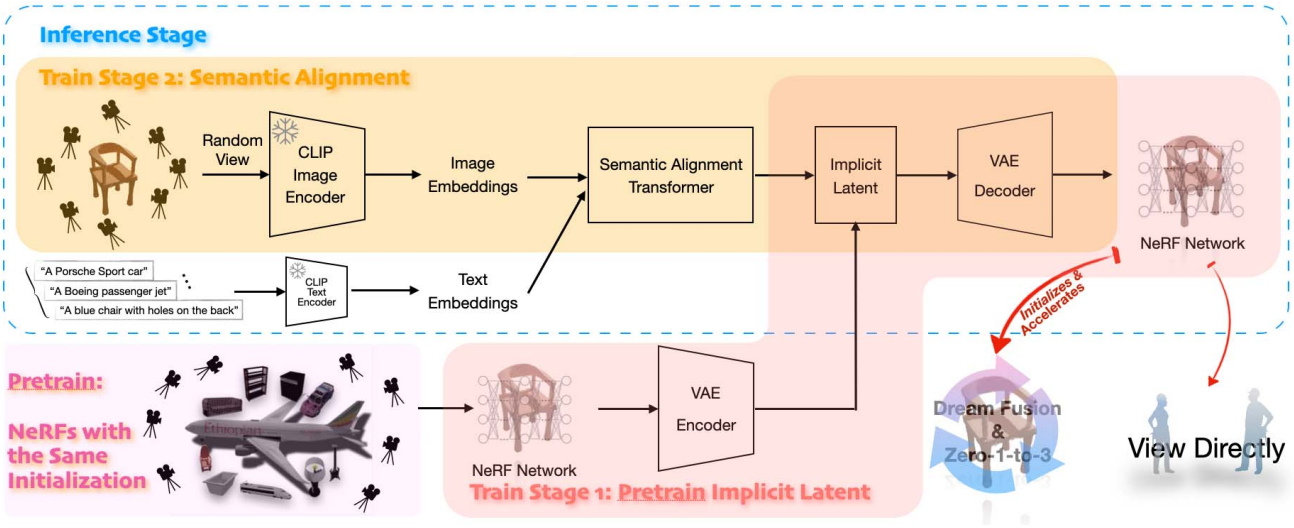


Figure 1. Prompt2NeRF-PIL's **overall pipeline**, which accepts a textual or image input with a single forward pass during inference. Two main modules are shown: a semantic alignment transformer model that maps CLIP embeddings input to a pretrained implicit latent NeRF space, and then a VAE decoder transforms it into a parameter space representing a NeRF network that can either be directly rendered or serve as a good initialization used in prior text-to-NeRF works such as DreamFusion [26] and image-to-3D works such as Zero-1-to-3 [17] to drastically reduce their generation time.

rect prediction of the NeRF parameters from prompts. Inspired by CLIP-Forge [30], we leverage image embeddings to tackle the problem of lack of text descriptions, allowing text and image prompts to be used interchangeably.

During the testing stage, our method can generate a satisfactory in-distribution object within a few seconds by a single forward pass. While for out-of-distribution generation, our method is able to provide a good initialization of the NeRF model. We demonstrate in our comprehensive experiments that such initialization can boost the convergence speed of DreamFusion [26] and Zero-1-to-3 [17] by 3 to 5 times, along with better and more consistent results in more generalized scene categories. Extensive experimental results and definitions of test scenarios (in-distribution and out-of-distribution) will be presented.

We summarize our contributions as follows:

1. As far as we know, Prompt2NeRF-PIL is the first work to enable implicit NeRF parameters generation of diverse objects from text and image prompts in a single forward pass, without any per-prompt training required.
2. Our simple yet effective method can serve as a good initialization and guidance to accelerate other main-stream text-to-3D generation methods such as [26] and image-to-3D methods such as 3D reconstruction using [17], allowing for faster NeRF generation in higher-quality.
3. Our pretrained implicit latent demonstrates the possibility of endowing the topological space of the parameters with semantics. And we proposed a new dataset for NeRF training.

2. Related Works

Neural Radiance Field. NeRF, or Neural Radiance Field introduced in [21] is a generative model that establishes a mapping from accurate camera poses to a 3D scene given a set of images. Unlike traditional representations such as voxels and meshes which highly depend on explicit 3D models, the foundation of NeRF is a radiance field, which is a function of how a batch of light rays travels through a 3D volume. NeRF can be formulated as:

$$F : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \boldsymbol{\sigma})$$

where $\mathbf{x} = (x, y, z)$, $\mathbf{d} = (\Theta, \phi)$ are respectively the location coordinate and direction in a predefined space, $\mathbf{c} = (r, g, b)$ is the color at the position, and $\boldsymbol{\sigma}$ can be thought of the probability that a light ray is occluded in 3D space. Following this novel representation, Mip-NeRF [1] alternatively uses light cones to reduce artifacts, and Bungee-NeRF [39] uses multi-resolution images to train Mip-NeRF progressively, which enables city-scale scenes rendering. Existing works also use explicit representations to obtain a hybrid NeRF. For example, Instant-NGP [23] uses a multi-resolution hash encoding structure; TensoRF [3] adopts factorizations to achieve a compact encoding space; Plenoxels [31] leverages a sparse grid for NeRF synthesis. In this experimental work, we adopt the MLP similarly done in the first NeRF work [21] as the foundation model in coarse-to-fine refinement.

Variational AutoEncoders. The Variational Autoencoder (VAE) [14] learns a low-dimensional latent space to encode the underlying distribution of the input data, thus allowing sampling from the latent to enable the generative feature. VAE demonstrates robustness in handling noisy

data, thus, in this work, we leverage its inherent properties that facilitate smooth interpolation between latent variables, so that meaningful semantic relationships can be preserved.

CLIP and CLIP-Based 3D Generation Contrastive Language-Image Pre-Training (CLIP) [27] offers a zero-shot capability that bridges the gap between the latent spaces of text and image. To date, it has been utilized for various zero-shot downstream applications [6, 15, 19, 30]. As CLIP bridges text and vision, many works [12, 20, 30, 36, 37] attempt to use it to optimize 3D models. However, despite their innovations, most of these works require a considerable amount of time to achieve satisfactory results. Recently, an image-to-3D work Zero-1-to-3 [17] tweaks Stable Diffusion [29] to adopt a vector that combines clip image embedding and pose information as input, and employs the images generated by their model as input to train NeRF model. Other methods such as [16, 26] have attempted to leverage the vast information within pretrained generative models via score distillation sampling under CLIP's guidance, by incorporating Stable Diffusion. Though having better performances on zero-shot generations, these approaches still require a long time for self-optimization and suffer from poor view consistency due to the use of 2D supervision without any initialization or view information. Therefore, aligning CLIP's embedding space with the 3D structure space is necessary.

Concurrent Works At the time of writing, we noticed one concurrent work also investigates the ability to implicitly generate NeRF structures. ATT3D [18], which adopts the InstantNGP [23] NeRF backbone, optimizes a set of prompts at the same time to save training time and share the feature between the prompts to enables unseen generation. However, the amortization direction is quite limited as the result figures shown in the paper, and that the implicit part only lies in the feature grid of InstantNGP. Other text-to-3D models such as [33, 38] aim to solve the 3D consistency problem whose setting is thus different from ours.

3. Method

Figure 1 shows Prompt2NeRF-PIL's overall architecture, which generates a NeRF from a textual or image input with merely a single forward pass during inference. Our framework consists of two main modules: an alignment model H that maps input $\mathbf{c}$ to an implicit NeRF latent space $\mathbf{z} = H(\mathbf{c})$, and then a decoder D to transform $\mathbf{z}$ back into a parameter space representing a NeRF $F = D(\mathbf{z})$. The NeRF obtained can either be directly rendered for in-distribution generation tasks, or provide a good initialization for out-of-distribution tasks to significantly accelerate the generation of prior text-to-NeRF works such as DreamFusion [26], and image-to-3D works such as Zero-1-to-3 [17].

3.1. NeRF Parameters Dataset

In the absence of prior implicit generation tasks, as noted by [5], it is necessary to develop a new dataset of NeRF parameters. To construct the dataset, we first use Blender to render 100 different camera views for each 3D scene $S_i, i \in \{1, 2, \cdots, N\}$ selected from: chair objects from Objaverse [4] and 7 other categories from ShapeNetCore [2], resulting in a total of $N = 4071$ three-dimensional scenes across 8 categories. We then train a NeRF model $F_{\Theta_i}(\cdot)$ for each instance S_i on the obtained camera views.

To reduce the training time for each NeRF, we adopt a similar approach in [5], where a simplified 7-layer MLP is used instead of the Vanilla NeRF in [21], which reduces the training time for each NeRF to around 10 minutes on a single RTX 4090 board. Then we partition and save the parameters of the MLPs $\Theta_i, i \in \{1, 2, \cdots, N\}$ into 14 components, as Θ_i consists of 7 weights $\mathbf{W}_j$ and 7 biases $\mathbf{b}_j$ for each layer $j = 1, 2, \cdots, 7$.

We randomly split the datasets into 90% training data and 10% testing data. The optimization process is halted at 15,000 iterations, as our empirical evidence demonstrates satisfactory PSNR levels (33.82 on average) can usually be achieved. Moreover, to avoid generating a prohibitive solution space of NeRF parameters, we use the same NeRF initialization for all data as inspired by [5] where, in differential geometry all 3D genus-zero objects can be deformed into the same topological sphere. We will demonstrate the effectiveness of this initialization as it yields favorable topological properties in our latent space in Sec. 5.1.

3.2. Prompt2NeRF-PIL Pipeline

Stage 1: Pretrain Implicit Latent In order to achieve a smoothly interpolatable latent representation, we first encode NeRF parameters into a latent space. Inspired by [30] that adds Gaussian noise to Autoencoder to improve robustness, we instead utilize a Variational Autoencoder (VAE [14]) that learns the latent space as a distribution to furthur enhance smoothness and robustness. Specifically, VAE first encodes the input NeRF parameter Θ_i in 14 partitions into two latent vectors $\boldsymbol{\mu}_i \in \mathbb{R}^{14\times d}, \boldsymbol{\sigma}_i \in \mathbb{R}^{14\times d}$ representing respectively the mean and variance of the distribution, where d is the dimension of the latent space. Then we randomly sample from this distribution and decode back to the reconstructed NeRF network Θ_i. Denote the Encoder and Decoder respectively as $E(\cdot;\alpha), D(\cdot;\beta)$, where α, β are the corresponding parameters. In this stage, we use Mean Square Error loss (MSE) as supervision. The whole process can be formulated as the following optimization:

$$\hat{\alpha}, \hat{\beta} = \arg\min_{\alpha,\beta} \frac{1}{N} \sum_{i=1}^{N} \left\| D\big(E(\Theta_i)\big) - \Theta_i \right\|^2 \qquad (1)$$

Stage 2: Semantic Alignment Our semantic alignment Transformer model H leverages the structure introduced in [35] that maps a semantic embedding $\mathbf{c}_i \in \mathbb{R}^d$ that represents a 3D scene to $\mathbf{z}_i \in \mathbb{R}^{14\times d}$, its corresponding NeRF parameters in the latent space obtained from VAE. We use the CLIP image encoder to obtain the embeddings from images.

Since one image cannot thoroughly represent information for one 3D scene, for each scene S_i, we randomly select one image $S_i(\boldsymbol{x}, \boldsymbol{d})$ from different viewpoints $(\boldsymbol{x}, \boldsymbol{d})$ each epoch during training to increase alignment robustness. In such a way, a rotation-invariant mapping is learned from a general representation R_i invariant to the viewpoint and rotation $(\boldsymbol{x}, \boldsymbol{d})$ chosen of each 3D scene S_i in the CLIP embeddings space, to the implicit NeRF latent space[1].

During inference, an image from an arbitrary view can be used as input for one specific scene. For text prompts, we notice that even though CLIP effectively maps images and texts into the same space, huge discrepancies still exist between their embeddings. We will show in supplementary material that for semantically highly-matched text and image pairs, the cosine similarity of their clip embeddings is still low, typically around 30% to 40%, indicating it is infeasible to directly use text CLIP embedding for inference. To get around this issue, when prompted with text, we first find the semantically nearest scene with the given text prompt in our training set and use the CLIP embedding of images of that scene instead.

4. Experiments

We will first briefly describe the datasets in Sec. 4.1, and summarize the implementation details in Sec. 4.2. Then we present the baselines and metrics we used in Sec. 4.3 and Sec. 4.4, respectively, followed by our experimental results in two aspects in Sec. 4.5 and Sec. 4.6. Specifically, we evaluate Prompt2NeRF-PIL in two settings: 1) in-distribution generation, and 2) out-of-distribution generation and acceleration for current prompt-to-NeRF works. To clarify, we define "in-distribution" as prompting Prompt2NeRF-PIL using a) images of unseen objects in the test split from our dataset, or b) texts that describe objects whose category is present in our datasets, since our training set does not contain any text description of the objects. For "out-of-distribution" generation, we use a) images from other sources whose categories are not included in the dataset, and b) texts that describe objects whose categories are not included in our dataset.

4.1. Datasets

For all our in-distribution experiments, we use the dataset built by ourselves mentioned in Section 3.1 which contains 8 categories and the ratio of the training set to the testing set is 9:1. We further modified our dataset to construct an image-NeRF parameter pair dataset, where the image is rendered from the corresponding NeRF scene.

4.2. Implementation Details

In this section, we highlight some implementation details of our method. More detailed information will be provided in supplementary materials.

[1]We include more results on this in supplementary materials.

For VAE training, we used a learning rate of 3×10^{-4} and trained for 10000 epochs, while for semantic alignment, we used a learning rate of 10^{-6} and trained for 100000 epochs until convergence. Adam optimizer [13] was used in both cases. In our experiments, we utilized the CLIP ViT-B/32 model to generate image and text embedding, thus the dimension of VAE's latent space is naturally designed to 512.

All experiments were run on NVIDIA RTX 4090 graphic cards. Unless specified, all times reported are recorded when running the experiment on one RTX 4090 card.

Figure 2. Results on **in-distribution generation from image prompts** randomly selected in test split. Each row presents one category (8 categories in total), where each pair represents the ground-truth image and the model-generated result, respectively.

4.3. Baselines

We compare our results with prior works DreamFusion [26] for text-to-NeRF tasks and Zero-1-to-3 [17] for image-to-NeRF tasks. For their implementations, we leveraged the open-source project Stable-Dreamfusion [34], a popular unofficial implementation of DreamFusion, as DreamFusion has not yet released their implementation and Stable-Dreamfusion also supports Zero-1-to-3 as guidance, while at the same time, a NeRF can be output instead of the official implementation of Zero-1-to-3 that can only synthesis novel views[2]. To ensure fairness, we replaced their NeRF network structure with ours, hence the validity of time comparison will not be undermined by different rendering times due to inconsistent NeRF structures. Moreover, we observed occasional instability in the optimization process of DreamFusion and Zero-1-to-3 using the implementation in [34]. To address this, we ran it with five different seeds for all experiments, then selected and presented the best result obtained. Despite this approach, it is important to note that we still encountered some cases where the optimization did not converge or failed to generate a scene relevant to the given prompt.

[2]Moreover, in the official implementation of Zero-1-to-3, the authors directly link the Stable-Dreamfusion's repository for 3D Reconstruction task.

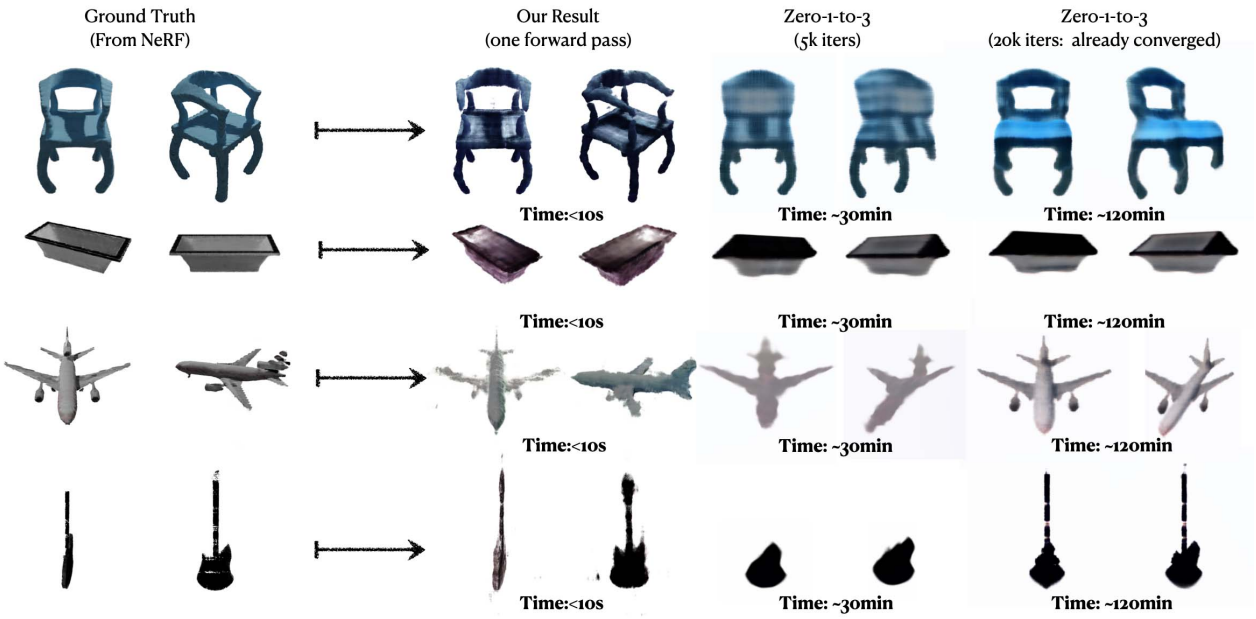


Figure 3. **In-distribution generation results from image prompts**. The first column contains the images directly rendered by our ground truth NeRFs; the second column contains the single forward inference results of our models; the third and fourth columns are the results generated by Zero-1-to-3, trained for 5k iterations and until convergence, respectively. For the blue chair and the guitar, our results are clearly of higher quality. As for the plane and the bathtub, although the quality seems less satisfying at first glance, our result is a consistent 3D solid model, while the other lookalikes are revealed as planar cardboard with little 3D consistency when viewed sideways.

4.4. Metrics

For quantitative results, we use two different metrics: Fréchet Inception Distance (FID) [9] and cosine similarity score to validate that the NeRFs generated by our model are close to ground-truth NeRFs. For FID, we compare the generated NeRFs with the ground truth NeRFs by computing the distance between the feature distributions of real and generated images rendered by corresponding NeRF using the Fréchet distance. Notice that due to inconsistent networks being used, the results may not be comparable to those in other works. Specifically, we used [32] with dim 192. Due to the lack of high-quality image captions in our dataset, we use the cosine similarity between the prompt and the embeddings of the rendered views of the generated NeRF as an analogy of the CLIP retrieval score [25] to evaluate the quality of our output.

Metrics	***Ours* *(train split)***	**Ours (test split)**	**Zero-1-to-3**
Similarity ↑	*(83.1%)*	81.5% (81.3%)	80.7%
FID ↓	*(1.334)*	4.189 (3.071)	13.525

Table 1. **Quantitative results on in-distribution image inference task.** We randomly sample 40 images and compare the CLIP embedding similarity and FID score of our method with Zero-1-to-3. The Scores in parentheses represent the evaluation of the entire train/test split, and the scores evaluated on the train split serve as an upper bound reference (in *italics*). Our method outperforms Zero-1-to-3 in higher similarity and lower FID score. Though Zero-1-to-3 also has high similarity scores, the low FID scores may indicate that it suffers from view inconsistency problems. Moreover, the performance of our method demonstrates comparable performance to the train split, indicating an effective alignment learned by our model.

4.5. Results on In-Distribution Generation

Generation from Image Prompts First, we use images on the test split as prompts to verify the in-distribution generation ability of our model. Figure 2 provides an overview of inference results of our model on test split, with 5 scenes randomly selected from each category. We then examine both qualitative and quantitative results in details as follows.

For qualitative results, we randomly sampled one scene from randomly chosen 4 categories (all cases are tested): chair, bathtub, plane, and guitar, and compared the results of our model with the performance of Zero-1-to-3 [17]. To ensure fairness, we allow Zero-1-to-3 to train until convergence, eliminating the disadvantage of lack of prior knowledge of the distribution of our dataset. From the comparison in Fig. 3, our model demonstrates a strong ability to generate in-distribution NeRF scenes from image prompts, producing much clearer and more realistic results compared to Zero-1-to-3 even with one single forward pass. Besides, it can be seen that Zero-1-to-3 sometimes produces view-inconsistent results. For example, the plane it generated lacks height when viewed from one side, and their generated guitar exhibits inconsistent shapes when viewed from the frontal and side angles.

To further evaluate performance quantitatively, we randomly selected 40 images[3] from our test split as prompts for

[3] 10% from our entire test split; we cannot afford too many since Zero-

Text Prompt	Ours	DreamFusion (5k iters.)	DreamFusion (converge)
"A green chair with armrests and a hollow back."	 Sim: 35.6% Time: $<$ 10s	 Sim: 32.7% Time: $\sim$ 30min	 Sim: 29.5% Time: $\sim$ 180min
"An F16 fighter jet."	 Sim: 28.2% Time: $<$ 10s	 Sim: 29.4% Time: $\sim$ 30min	 Sim: 28.2% Time: $\sim$ 190min
"A purple and blue sport car."	 Sim: 27.0% Time: $<$ 10s	 Sim: 22.2% Time: $\sim$ 30min	 Sim: 26.6% Time: $>$ 250min

Table 2. Comparison of **in-distribution inference result with text prompts**. For DreamFusion results, we optimized each prompt for both 5000 iterations and until convergence. Our method almost consistently achieves higher CLIP embedding cosine similarities compared to DreamFusion, and it does so in significantly shorter time frames, demonstrating the effectiveness of our method on the text inference task. For the last prompt, DreamFusion failed to converge to a satisfying result after several trials, as explained in Sec. 4.3, even though the similarity is high, indicating that sometimes CLIP guidance may mislead DreamFusion in unfavorable directions.

both our method and Zero-1-to-3, then compute the cosine similarity and FID score by comparing with ground-truth images from different views. To provide more references, we also compute these two scores on the entire test and train split using our method, where the train split result can serve as an upper-bound performance. Table 1 shows the comparison, illustrating that the performance of our model on the test split does not drastically decrease compared to its counterpart on the train split. Also, our model has higher similarity and lower FID scores compared with Zero-1-to-3, demonstrating our model's strong capability in generating view-consistent NeRFs.

Generation from Text Prompts Next, we present the generation results using text prompts. To restrict the evaluation to an in-distribution scope, all texts used in this section are designed to describe objects from the 8 categories in our dataset. Table 2 compares the cosine similarity of the text prompt and generated images, as well as the inference time of our results with DreamFusion's, where we let DreamFusion train for 5k iterations and until convergence. We can see that our method is significantly faster than DreamFusion since only one single forward pass is needed, while at the same time being able to generate scenes with almost always better quality (higher similarity) compared to DreamFusion.

1-to-3 takes a long time to optimize until convergence and sometimes even fails, as mentioned in Sec. 4.3.

4.6. Results on Out-of-Distribution Generation

As real-world prompts feature unseen geometries and attributes during the training phase, we anticipate that generating high-quality out-of-distribution 3D scenes without optimization would be impractical in our pre-trained settings. Instead, we discovered that the NeRFs generated by our model serve as good initializations for prior works that require per-prompt optimization to speed up their optimization time. In this section, we will show that with our initialization, the inference process of DreamFusion (for text-to-NeRF tasks) and the 3D generation process of Zero-1-to-3 (for image-to-NeRF tasks) will converge faster while almost always yielding better quality. Here we also provide the demo results of inferencing from text and image prompts similar to Sec. 4.5, but they are now out of distribution. Thus, in both cases, we find the semantically nearest scene with the given prompt in our training set. Other initialization methods will be explored in Sec. 5.2.

In this out-of-distribution experiment, we selected image and text prompts that describe scenes whose categories were not present in our dataset. Qualitative results are displayed through generated demo images, while quantitatively, we calculate the cosine similarity of the generated image views with the given prompt. Moreover, we report the optimization time of baselines with/without our initialization to highlight the remarkable inference acceleration provided by our initialization for both text and image prompts.

Table 3 shows the comparison of our method with baselines, using two image prompts and two text prompts[4]. We first let our model directly generate a NeRF from the given prompt in a zero-shot manner, by inferencing from the semantically nearest scene in our train split, then use the generated NeRF as an initialization to further optimize DreamFusion or Zero-1-to-3 models with the prompt until converge. The top-right inset images in the first column depict the zero-shot generation result of our model.

For comparison, we evaluate two kinds of baselines in our study: 1) Baseline (S), which is the baseline model trained from **s**cratch with the given prompt, and 2) Baseline (P), which is the baseline model first **p**re-trained to a semantically close scene with our initialization, and then optimized with the given prompt.

Our results show that ourwork converges 3 to 5 times faster than baselines on text and image prompts. Moreover, when optimized for the same iterations, as demonstrated in columns 2 and 3, we observe that both baselines only learn geometry features, while our method already learns fine textural details. By comparing the results of our method with baseline (P), we conclude that even though they are pre-trained into semantically close scenes, our method still converges much faster than the baseline (P), thus eliminating

[4]Due to space limit, more results can be found in supplementary materials.

Prompt	Ours (converge)	Zero-1-to-3 (S) (same iterations)	Zero-1-to-3 (P) (same iterations)	Zero-1-to-3 (S) (converge)	Zero-1-to-3 (P) (converge)
	 Similarity: 89.1% Iterations: 6.1k	 Similarity: 71.0% Iterations: 6.1k	 Similarity: 83.9% Iterations: 6.1k	 Similarity: 84.9% Iterations: 20.7k	 Similarity: 88.4% Iterations: 18.9k
	 Similarity: 80.8% Iterations: 7.5k	 Similarity: 73.7% Iterations: 7.5k	 Similarity: 79.8% Iterations: 7.5k	 Similarity: 78.3% Iterations: 24.6k	 Similarity: 78.6% Iterations: 22.3k

Prompt	Ours (converge)	DreamFusion (S) (same iterations)	DreamFusion (P) (same iterations)	DreamFusion (S) (converge)	DreamFusion (P) (converge)
"A brown table."	 Similarity: 32.3% Iterations: 5.5k	 Similarity: 28.1% Iterations: 5.5k	 Similarity: 32.3% Iterations: 5.5k	 Similarity: 30.9% Iterations: 24k	 Similarity: 33.3% Iterations: 20.5k
"A round fountain."	 Similarity: 31.6% Iterations: 7k	 Similarity: 29.6% Iterations: 7k	 Similarity: 26.9% Iterations: 7k	 Similarity: 30.3% Iterations: 32k	 Similarity: 30.6% Iterations: 30k

Table 3. Comparisons of **out-of-distribution inference result, using either image or text prompts**. For each prompt, we record the convergence iteration count using our initialization zero-shot, from one single forward pass (top-right inset). Scratch (S) or pre-trained (P) baselines (Zero-1-to-3 for image-to-NeRF tasks, and DreamFusion for text-to-NeRF tasks) are trained for the same number of iterations and until convergence, and their results are displayed. Our method significantly accelerates the baseline optimization, achieving a 3 to 5 times speed boost while almost consistently yielding better results. Moreover, our method provides a semantically meaningful initialization, as pretrained (P) baselines still take an extremely long time to converge though initialized with a similar scene. Due to space limit, more results can be found in supplementary materials.

the possibility that the acceleration by our method is merely due to the reason that a NeRF initialization will yield faster convergence compared to a random initialization, while at the same time, demonstrating that our method has learned a semantically meaningful alignment, thereby providing a closer initialization compared to baseline (P)[5].

[5]We include further evidence showing the semantics our model learns in supplementary materials by performing some interpolation experiments.

5. Ablation Study

5.1. Good Topology of NeRF Parameter Space

This section verifies that using the same (our) initialization while training NeRFs leads to better topological structures. Figure 4 visualizes the pairwise Principal Component Analysis (PCA) result of the trained NeRF parameters, where the parameters within each class are clustered with different centers. The intra-class affinity and inter-class separability, manifested as the blue and red clusters, are clearly visible.

Instead, if NeRFs are trained with random initializations,

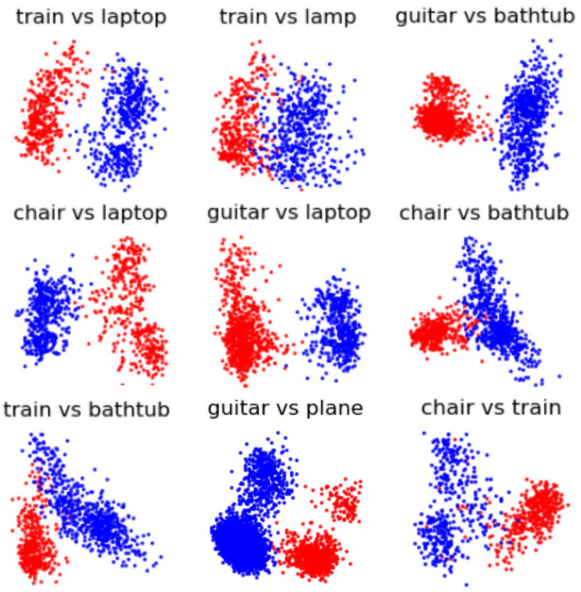


Figure 4. Sample pairwise PCA of our trained NeRF parameters. The NeRF parameters are clearly separated among different classes, and clustered within the same class.

such property will not occur. Take the chair and plane's PCA plot as an example, Figure 5 illustrates that in the left plot (with random initialization), two classes are mixed together, while in the right plot (with the same initialization), the NeRF weights are separated apart for the two categories. These trained parameters can thus result in a good manifold for our pretrained implicit latent space for better interpolation within a given class as well as latent alignment in later steps.

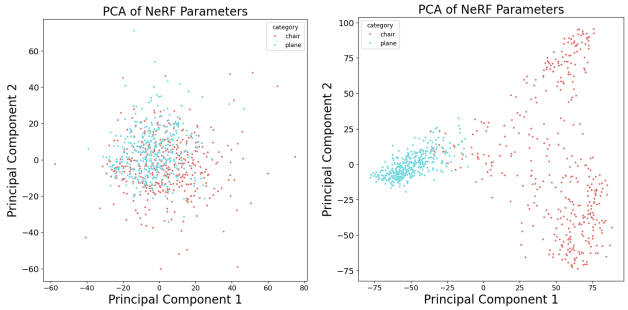


Figure 5. The PCA plots of plane and chair NeRF parameters with (right) and without (left) using the same initialization, where the left plot fails to separate the NeRF parameters for two categories.

5.2. How to initialize Dreamfusion

To determine the most effective initialization method for our out-of-distribution generation experiment that accelerates Dreamfusion [26] or Zero-1-to-3 [17], we explore two options: 1) Ours (NRS): use the NeRF inferenced from the semantically **n**ea**r**e**s**t image on our dataset to the prompts, which is extensively used in Sec. 4 and 2) Ours (DI): use the NeRF **di**rectly generated from the prompts. We evaluate the performance of both methods on text-to-NeRF tasks in terms of visual quality, cosine similarity between generated images and text prompts, as well as convergence speed. Using the per-prompt optimization of DreamFusion from scratch as a baseline, this comparison will allow us to determine which initialization method yields optimal results.

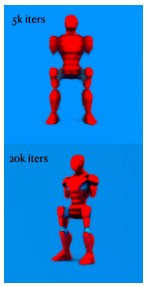
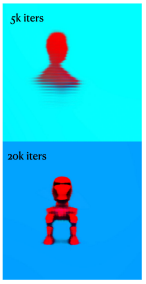
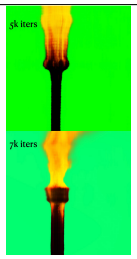
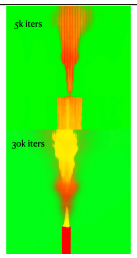
Text Prompt	Ours (NRS)	Ours (DI)	DreamFusion (Scratch)
"Iron man on a blue chair"	 Similarity: 31.74%	 Similarity: 29.44%	 Similarity: Failed
"A burning torch"	 Similarity: 31.47%	 Similarity: 31.01%	 Similarity: 32.20%

Table 4. This experiment qualitatively demonstrates two results of comparison on DreamFusion trained from scratch (baseline) against training with our two **initialization** methods. The top images demonstrate the results after 5k iterations; the bottom are trained either to converge or capped at maximum iterations (20k for our methods, 40k for baseline). We can see that our method (NRS) produces good results in only 5k iterations, and converges a lot earlier than other methods.

We designed 10 prompt texts that include both some extensions of our dataset categories, such as "Ironman on a blue chair", and some entirely out-of-distribution categories, such as "Thor's hammer" and "A burning torch". For example, in Tab. 4, using the Ironman prompt, we can see that DreamFusion initialized with NeRFs generated using the semantically closest image results in earlier convergence (similar quality at 5k iterations to direct initialization at 20k iterations) and higher semantic similarity, than with the text prompt than the result initialized from NeRFs generated directly from texts. However, both methods are much faster and produce more satisfying results than optimizing DreamFusion from scratch (last column of the figure), which fails to converge even trained to 40k iterations with multiple tries. More results will be presented in the supplementary materials.

6. Conclusion

Concurrent to contemporary works investigating implicit NeRF generation indicating its potential high impact, this experimental paper explores alternatives to conventional NeRF generation where typical input consists of multiple 2D images, as a springboard transitioning into promptable NeRF generation (e.g., text prompt or single image prompt) for *direct* conditioning and *fast* generation of NeRF parameters for the underlying 3D scenes, tapping into promptable generative NeRF model. From our experiments, we hope Prompt2NeRF-PIL will ignite future works in faster 3D scene generation and easy-to-use 3D content creation.

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 2

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022. 2

[4] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects, 2023. 3

[5] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. *arXiv preprint arXiv:2303.17015*, 2023. 1, 3

[6] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen. Clip2video: Mastering video-text retrieval via image clip, 2021. 3

[7] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. 1

[8] Robert A. Gonsalves. Using openclip for image search and automatic captioning, 2023. 1

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. 5

[10] 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, 2021. 1

[11] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 1

[12] Wonbong Jang and Lourdes Agapito. Codenerf: Disentangled neural radiance fields for object categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12949–12958, 2021. 1, 3

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 4

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 3, 1

[15] Jie Lei, Linjie Li, Luowei Zhou, Zhe Gan, Tamara L. Berg, Mohit Bansal, and Jingjing Liu. Less is more: Clipbert for video-and-language learning via sparse sampling, 2021. 3

[16] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 1, 3

[17] 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, 2023. 1, 2, 3, 4, 5, 8

[18] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. Att3d: Amortized text-to-3d object synthesis. *arXiv preprint arXiv:2306.07349*, 2023. 3

[19] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval, 2021. 3

[20] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 1, 3

[21] 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, 2020. 2, 3, 1

[22] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 1

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 3

[24] OpenAI. Gpt-4 technical report, 2023. 1

[25] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. In *NeurIPS Datasets and Benchmarks*, 2021. 5

[26] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1, 2, 3, 4, 8

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1, 3

[28] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 1

[29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 1, 3

[30] Aditya Sanghi, Hang Chu, Joseph G. Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation, 2022. 1, 2, 3

[31] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[32] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, 2020. Version 0.3.0. 5

[33] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 3

[34] Jiaxiang Tang. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion. 4

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 1, 3

[36] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 1, 3

[37] Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 1, 3

[38] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 3

[39] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pages 106–122. Springer, 2022. 2

[40] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 1

Prompt2NeRF-PIL: Fast NeRF Generation via Pretrained Implicit Latent

Supplementary Material

I. Rotation-invariant Mapping

As mentioned in Sec. 3.2, here, we demonstrate that by randomly selecting images from different viewpoints during the semantic alignment training, our model can learn a rotation-invariant mapping, i.e., from a general semantic representation invariant to viewpoint and rotation to the implicit NeRF [21] parameter space.

The presented findings can be observed in Fig. 6. The upper section of the figure exhibits eight images, each representing inference results obtained from randomly selected viewpoints. These images demonstrate a high degree of similarity, indicating consistent outcomes across different viewpoints. Conversely, in the lower section of the figure, only the first image, which derived from the training viewpoint, depicts satisfactory results. However, when inferring from alternative viewpoints, the generated outcomes lack consistency and fail to align with the desired expectations. To ensure fairness, the two checkpoints we used for comparison were trained to similar accuracies, with the fixed-view checkpoint having a slightly higher accuracy since it is more stable during training.

II. Low CLIP Similarity on Highly-matched Image-Text Pairs

As we have mentioned in Sec. 3.2, although CLIP [27] effectively maps texts and images onto the same space, semantically highly matched image and text pairs still receive low (typically around 30% to 40%) cosine similarity, hence making it unlikely for our model trained on pure image embeddings to receive text embeddings directly during inference stage.

We noticed that some prior works such as [8] have already demonstrated and discussed such a phenomenon, and in this section, we prove that this holds true on our dataset as well. Figure 7 shows some examples of image and text pairs, where the cosine similarities of each pair are calculated and displayed in a heat map. We may observe that the pairs on the diagonal are semantically highly matched, yet only receive similarities of around 30%.

III. Implementation Details

This section provides more implementation details of our model, as an extension of Sec. 4.2

VAE implementation detail As detailed in Sec. 3.2, our NeRF network employs 14 distinct tokens for representation. To facilitate this, we employ 14 independent Variational Autoencoders (VAEs) [14] to learn a separate implicit latent space for each token. Each VAE is comprised of an encoder and a decoder, each of which is composed of a 3-layer Multilayer Perceptron (MLP). Two linear layers are added between each encoder and decoder to learn a mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\Sigma}$ for each token, thereby emulating a Gaussian distribution $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ representing each implicit latent space, subsequently samples an input to the decoder.

Semantic Alignment implementation detail We utilized the Transformer [35] model to attain semantic alignment, where we regard the semantic clip embedding $\mathbf{c} \in \mathbb{R}^d$ as the source sequence and its corresponding NeRF parameter in our pretrained latent space $\mathbf{z} \in \mathbb{R}^{14\times d}$ as the target sequence. In the training phase, a zero-valued beginning-of-sequence ([BOS]) token is appended before the target sequence, and the prediction is performed sequentially. During the inference stage, akin to the training phase, only a zero-valued [BOS] token is provided at the beginning as the target sequence. The Transformer model predicts the next target token based on the provided target sequence (only [BOS] at this stage) and the source sequence. The predicted token is then concatenated after the previous target sequence, and the Transformer model continues to predict the next token, and this process is repeated until all 14 target sequence tokens have been predicted.

IV. More Results for Out-of-distribution Inference

In this section, we provide more out-of-distribution inference results, from both image and text prompts, as an extension to Tab. 3 in main paper.

Table 5 shows four more out-of-distribution inference results from image prompts that fall out of our eight categories, and Tab. 6 shows four corresponding results from text prompts. We can observe the 3 to 5 times significant acceleration that our initialization brings to the baseline models, either Zero-1-to-3 [17] for the image-to-NeRF task or DreamFusion [26] for the text-to-NeRF task.

V. Interpolation Results

We continue the discussion in Sec. 4.6 to demonstrate that our method has learned a semantically meaningful alignment by displaying some interpolation results, thereby providing a valid explanation of why the initializations given by our method yield much faster convergence compared with the pre-trained baseline (P), which is also trained to a semantically close scene. The results in Fig. 8 demonstrate that we can get not only smooth interpolation between objects with a single feature, like colors, but also a smooth morph between two objects with a difference in both geometry and color.

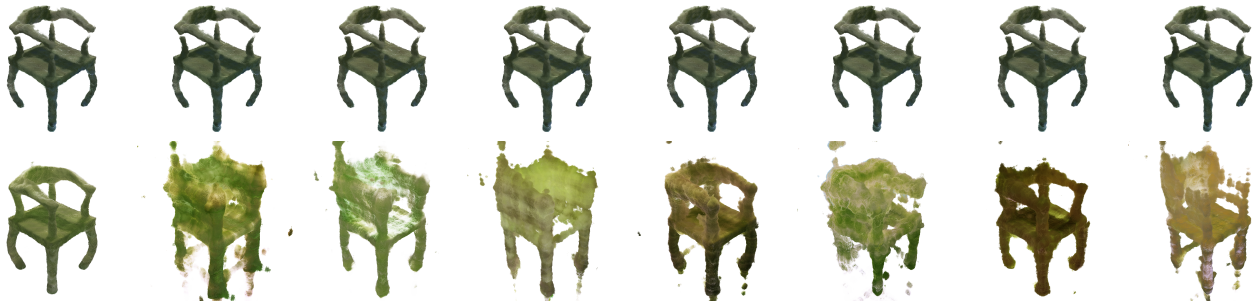


Figure 6. Comparison of inference results when using randomly-selected views (top) and fixed view (bottom) during semantic alignment. The results in column i, $i = 1, 2, \cdots 8$ are obtained by using the clip embedding of the i-th view during inference. Notice that in the fixed-view method, the 1st view is selected during training, hence we can observe the result in column 1 is much better than the other columns for fixed-view method. On the contrary, no matter which view is used in inference, the method trained using randomly-selected views can always produce satisfying and consistent results.

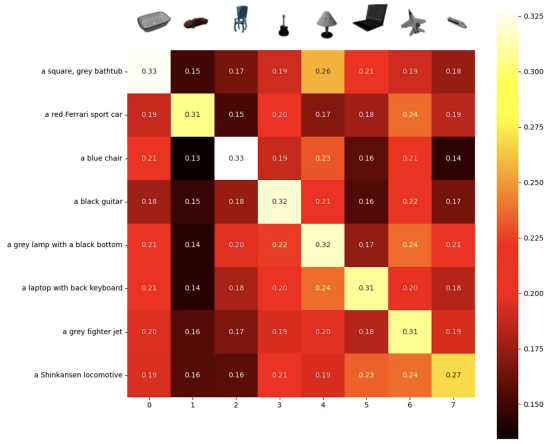


Figure 7. Result of some clip embedding cosine similarity of text description and image in our train set. Semantically highly matched pairs are placed on the diagonal.

VI. Comparisons on Convergence Speed

As previous results only provide the number of iterations required for convergence when demonstrating the effective acceleration ability of our initialization, here we include examples of the curves comparing the optimization process of the baseline model (Zero-1-to-3 for image-to-NeRF tasks and DreamFusion for text-to-NeRF tasks) with/without our initialization.

Specifically, after training every 100 iterations (or one epoch), we render m images $I_1^{(i)}, I_2^{(i)}, \cdots, I_m^{(i)}$ for epoch i, from viewpoints evenly divided from 360°, and compute the pixel-wise mean-squared difference D_i between the m images rendered in the i-th epoch and the previous $(i-1)$-th epoch. This metric can be formulated as

$$D_i = \sum_{j=1}^{m} \left\| I_j^{(i)} - I_j^{(i-1)} \right\|_2^2, \quad \text{for } i > 1$$
$$D_1 = 0$$

where by taking the difference of two images I we mean the pixel-wise difference.

We then consider the optimization process is converged if $D_i < \epsilon$ for continuous 20 epochs.

Figure 9 shows two examples of the convergence curves with $m = 8$ and $\epsilon = 0.01$, using image prompt and text prompt, respectively. We may clearly observe the acceleration that our initialization brings to the baseline.

VII. Video & Multi-view Results of Generated Scenes

In this section, we provide some video results[6] of the NeRF scene we generated, as well as the 8 multi-view images used to generate the videos, demonstrating that our implicit generation method yields satisfying view-consistent NeRF scenes.

We randomly select some samples from our test split and display them in Figs. 10 and 11. For out-of-distribution image or text inference results, we display them in Fig. 12 and Fig. 13, respectively.

[6] will release later

Prompt	Ours (converge)	Zero-1-to-3 (S) (same iterations)	Zero-1-to-3 (P) (same iterations)	Zero-1-to-3 (S) (converge)	Zero-1-to-3 (P) (converge)
	 Similarity: 69.6% Iterations: 12.6k	 Similarity: 68.1% Iterations: 12.6k	 Similarity: 71.3% Iterations: 12.6k	 Similarity: 68.4% Iterations: 41.2k	 Similarity: 68.3% Iterations: 40k
	 Similarity: 93.5% Iterations: 4k	 Similarity: 84.0% Iterations: 4k	 Similarity: 90.1% Iterations: 4k	 Similarity: 92.0% Iterations: 20k	 Similarity: 92.9% Iterations: 17.9k
	 Similarity: 85.8% Iterations: 6.9k	 Similarity: 78.6% Iterations: 6.9k	 Similarity: 74.1% Iterations: 6.9k	 Similarity: 81.7% Iterations: 20.1k	 Similarity: 86.3% Iterations: 21.2k
	 Similarity: 94.8% Iterations: 10.7k	 Similarity: 89.7% Iterations: 10.7k	 Similarity: 90.3% Iterations: 10.7k	 Similarity: 94.1% Iterations: 34.7k	 Similarity: 93.8% Iterations: 33.4k

Table 5. More examples of **out-of-distribution inference result, using image prompts**. For each prompt, we record the convergence iteration count using our initialization zero-shot, from one single forward pass (top-right inset). Scratch (S) or pre-trained (P) Zero-1-to-3 are respectively trained for the same number of iterations and until convergence with their respective results displayed above. Our method significantly accelerates the Zero-1-to-3's optimization, achieving 3 to 5 times speed boost while almost consistently yielding better results. Moreover, our method provides a semantically meaningful initialization, as pretrained (P) Zero-1-to-3 still takes an extremely long time to converge though initialized with a similar scene.

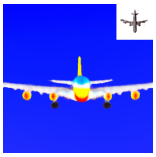
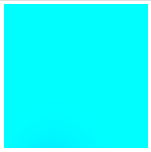
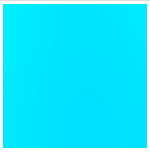
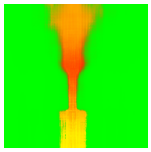
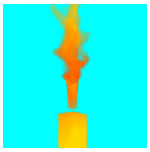
Prompt	Ours (converge)	DreamFusion (S) (same iterations)	DreamFusion (P) (same iterations)	DreamFusion (S) (converge)	DreamFusion (P) (converge)
"A colorful Boeing passenger plane."	 Similarity: 31.5% Iterations: 7.5k	 Similarity: 29.6% Iterations: 7.5k	 Similarity: 28.6% Iterations: 7.5k	 Similarity: 33.6% Iterations: 30k	 Similarity: 33.6% Iterations: 31.5k
"Iron man on a blue chair."	 Similarity: 28.9% Iterations: 5k	 Similarity: 23.4% Iterations: 5k	 Similarity: 26.1% Iterations: 5k	 Similarity: 23.2% Iterations: 40k	 Similarity: 31.0% Iterations: 15.5k
"A burning torch."	 Similarity: 32.2% Iterations: 7k	 Similarity: 30.1% Iterations: 7k	 Similarity: 29.7% Iterations: 7k	 Similarity: 32.3% Iterations: 30k	 Similarity: 31.4% Iterations: 19k
"A basketball."	 Similarity: 30.4% Iterations: 7.4k	 Similarity: 26.1% Iterations: 7.4k	 Similarity: 24.1% Iterations: 7.4k	 Similarity: 27.3% Iterations: 18k	 Similarity: 27.4% Iterations: 19.5k

Table 6. More examples of **out-of-distribution inference result, using text prompts**. For each prompt, we record the convergence iteration count using our initialization zero-shot, from one single forward pass (top-right inset). Scratch (S) or pre-trained (P) DreamFusion are respectively trained for the same number of iterations and until convergence, with their respective results displayed above. Our method significantly accelerates the DreamFusion's optimization, achieving 3 to 5 times speed boost while almost consistently yielding better results. Moreover, our method provides a semantically meaningful initialization, as pretrained (P) DreamFusion still takes an extremely long time to converge though initialized with a similar scene.

Figure 8. Interpolation results, with two objects differ only in color (left) and differ both in color and shape (right). Each time we slightly increase the weight of the first object for interpolation, decrease the weight of the second object by 0.01, and organize the results row by row, which allows us to observe a smooth interpolation result.

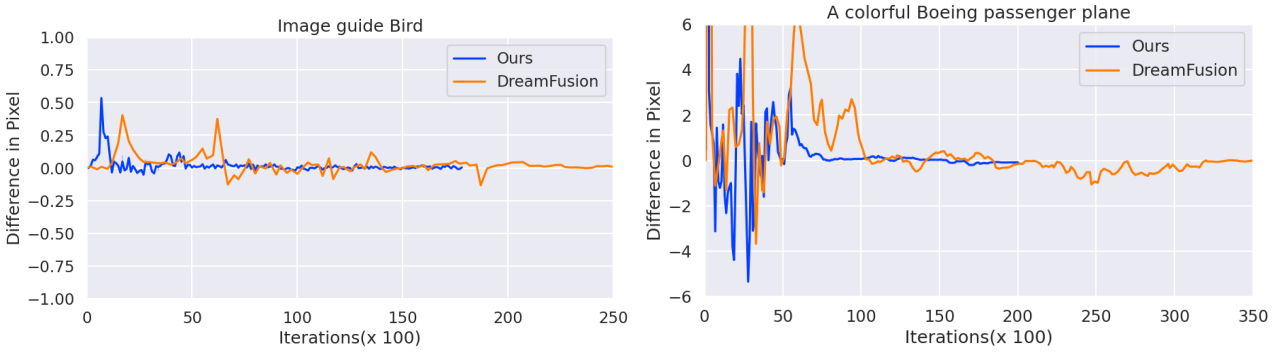


Figure 9. Comparisons of optimization convergence curves with (Ours) / without (DreamFusion) our initialization, when using image (left) and text (right) prompts for inference.

Figure 10. **Multi-view results of in-distribution inference from image prompts**. Samples are randomly selected from our test split. For each scene, we display the 8 camera views generated by our NeRF obtained.

Figure 11. (*Cont.*) **Multi-view results of in-distribution inference from image prompts**.

Figure 12. **Multi-view results of out-of-distribution inference from image prompts**. We reuse the samples shown in the experiment results section in out-of-distribution inference. We display the 8 camera views generated by our NeRF obtained for each scene. Prompts used are the same in previous experiment results.

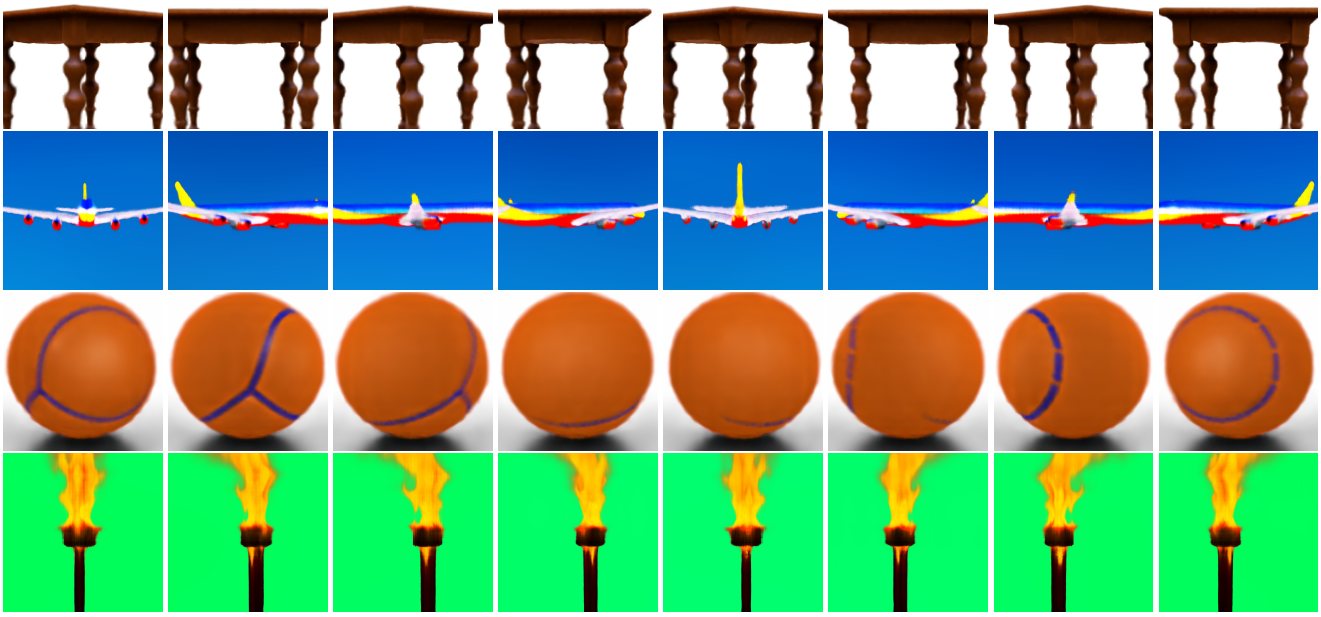


Figure 13. **Multi-view results of out-of-distribution inference from text prompts**. We reuse the samples shown in the experiment results section in out-of-distribution inference. We display the 8 camera views generated by our NeRF obtained for each scene. Prompts used (from top to bottom): “A brown table.”, “A colorful Boeing passenger plane.”, “A basketball.”, “A burning torch.”