Improved 3D scene reconstruction with diffusion models

Semester project

Clément Jambon
cjambon@student.ethz.ch
ETH Zürich

Supervisor: Prof. Dr. M. Pollefeys
Co-supervisor: PhD Student Silvan Weder
17.02.2023 - 28.06.2023

Computer Vision and Geometry Lab

June 24, 2023

Abstract

Neural Radiance Fields (NeRF) have recently revolutionized the task of 3D reconstruction and novel view synthesis. Nevertheless, the quality of their reconstructions remains highly dependent on the good coverage of the input images which is hardly met when casually capturing a scene with a handheld device without careful instructions. The consequence is the multiplication of artifacts and the incapacity to recover out-of-distribution viewpoints. On the other hand, recent large-scale generative models, namely diffusion models (e.g. *Stable Diffusion*), have proven to model complex and diverse 2D priors, providing both good data coverage and sample quality. We thus propose to lift the latter to 3D in order to perform "neural extrapolations" of partial pre-trained NeRF scenes. To tackle this problem, we mostly investigate three state-of-the-art approaches: Feature Fields, Score Distillation Sampling (SDS) and feature-guided conditioning of diffusion models. Through carefully conducted experiments, we show that these methods come with strong limitations tied to the architecture of diffusion models, the inherent biases of large-scale training and the text-to-image cross-modality paradigm. We conclude by suggesting that stronger geometric priors need to be provided to these "generic" models through the use of specialized probabilistic adapters leveraging the partial reconstructions of the target scenes.

Contents

Chapter 1

Introduction

Despite decades of research, 3D reconstruction from 2D images remains a challenging task. One of the main difficulty resides in the unconstrained nature of the problem and the limited 2D priors that input images, and inherently viewpoints, provide us with (e.g. fixed resolution, sparse coverage, etc). This makes recovering both geometry and appearance arduous. Recently, *Neural Radiance Fields* (NeRF) [66] have shown promising improvements by introducing a parametrization based on *Implicit Neural Representations* (INR) and by using *Volume Rendering* as a differentiable rendering process. However, the resulting reconstructions are still highly dependent on both the quality of the input signal and the distribution of the training viewpoints, which hampers their wide adoption, especially with existing handheld devices.

To tackle these issues, previous approaches have been proposed. A range of works has focused on reparameterization of either geometry (e.g. contractions [114, 6, 128]) or appearance (e.g. material models [109]). Making use of structured priors and additional signals such as depth was shown to yield significant improvements in scene-scale reconstructions [21, 84]. Domain-specific priors have also been introduced, either as class-based conditioned models [31, 55], from high-dimensional feature volumes [125, 13] or through local patch-based refinement [117, 85]. Finally, a large number of heuristics and regularizations have been proposed to fix canonical NeRF reconstruction artifacts (e.g. floaters) [70, 74, 90]. However, these methods all fall short of generalization.

In the 2D domain, a new generative machine learning paradigm has emerged: *Diffusion Models*. The latter have shown groundbreaking semantic and conceptual capabilities. Nevertheless, their utilization has so far mostly been confined to *text-to-image* (T2I) applications, with recent works extending them to impressive, yet limited text-to-3d capabilities[76, 113, 54]. Our goal shares similarities with some of these approaches in the sense that we investigate a common challenge: how can we lift the 2D priors of large-scale vision models (e.g. *Stable Diffusion* [87], *Imagen* [91]) in a 3D multi-view consistent manner? However, contrary to them, we aim at performing "neural extrapolations" (or even "hallucinations") from an existing distribution of images representing real 3D scenes rather than sampling them directly through their structuring conditioning signal, namely text. We insist on two assumptions which we chose to focus on. The first is **the use of large-scale pre-trained and generalist 2D text-to-image models without expensive re-training or fine-tuning**. The second is the nature of the sought "extrapolations": we do not want to perform super-resolution upsampling and/or recover high-frequency details but rather **large baseline "extrapolations"** even if it means trading off reconstruction accuracy.

As we show in multiple experiments, this task is particularly challenging as there is a huge domain gap between the training distribution of such models and the specific distribution of viewpoints used to train an arbitrary NeRF scene. We unfortunately do not have a final solution to tackle this problem but provide novel insights on the capabilities, internal representations and intrinsic biases of large-scale diffusion models. To do so, we explored three main directions: "perceptual" feature fields, *Score Distillation Sampling* (SDS) [76] and inversion with feature-guided conditioning. We hope that the insights we propose will help guide future successful approaches.

Chapter 2

Related works

As we experimented with many approaches to tackle our problem, we must cover a significant amount of related work. In order to keep things concise, we leave foundational related works to chapter 3.

Text-to-3d. *DreamFusion* [76], *Score Jacobian Chaining* [113] concurrently introduced *text-to-3d* object (or scene) generation from a textual prompt by distilling 2D diffusion priors through an optimization process called *Score Distillation Sampling* (SDS). This technique has since then been applied to various submodalities including vector graphics [44], textures [83] and semantically stylized text [43]. These show how powerful and generic the method is. Nevertheless, as it is a particularly ill-posed problem, SDS requires high text guidance to converge. This results in low-frequency and highly saturated reconstructions, missing concepts, object-centered generation and prompt bias among which the *Janus problem* [76] is the most striking illustration. Several solutions have been proposed: *Magic3D* [54] introduces a two-stage process to recover high-frequency details thanks to differentiable mesh optimization, Hong et al. [40] and Armandpour et al. [3] propose to debias prompts in a 3D-aware manner, *Set-the-Scene* [18] and *Locally Conditioned Diffusion* [75] compose multiple SDS processes, and *Prolific-Dreamer* [116] and *HiFA* [132] enable high-fidelity sampling by introducing respectively a *Variational Score Distillation* approach and a multistep annealed sampling strategy with an effective reparameterization. Nonetheless, these models rely almost exclusively on textual inputs and often make use of textual inversion or fine-tuning to sample from 2D images [20, 122, 79, 100]. Unfortunately, text cannot capture the full appearance and geometrical details of the visual world and fine-tuning is prone to catastrophic forgetting and overfitting.

Few-shot NeRFs. Generalizable NeRFs can be divided into two main categories. The first one makes use of high-dimensional pixel-aligned feature volumes usually extracted from the training images with a CNN encoder [125, 13, 115] or a transformer [98]. The latter are then reprojected in the target view and aggregated either via a pooling operation [125], using a *ray transformer* [115, 98] or additional filtering operations [16]. The second class builds on category-level priors that are usually derived and learned from synthetic datasets. These class-conditioned models [118] are usually performant, but their success relies on the clean, yet restricted, data distribution they build on and thus fail to generalize to real and complex scene scenarios.

Iterative approaches. Multiple works have concurrently proposed to start from a text-conditioned generated image and progressively generate a mesh by carefully inpainting disocclusions and estimating and filtering depth [27, 42]. Instead of building explicitly the geometry, other approaches construct trajectories online and in an autoregressive way [82, 10]. However, being iterative, all these methods are very sensitive to abrupt changes of distributions and thus are either restricted to specialized types of scenes (e.g. rooms [42], more general closed environments [27], outdoor aerial scenes [10]) or constrained linear trajectories [27].

Local priors. Rather than building on global data priors which are expensive to collect, especially in quality. A line of works has recently suggested injecting local priors through specialized models. These can be either 2D (e.g. GANs [85] or Diffusion Models [120]) or 3D [117]. However, being local, these priors cannot address more ambitious "neural extrapolations" and lack contextualization w.r.t. the content and the structure of the scene.

Personalization. In order to close the domain gap between the learned (and inherently biased towards the training data) priors of large-scale diffusion models, many approaches have been proposed to specialize the latter to specific concepts and appearances. *Textual Inversion* [28] proposes to capture the visual appearance and semantic of an object inside a token whose embedding is learned differentiably given a set of reference images. However, this conditioning is bottlenecked by the low expressivity of text and the interference due to its contextualization within the input sentence. As a consequence, further extensions have been introduced that optimize "tail" token embeddings [32], learn multiple embeddings across different ranges of the noise schedule [19], for the different layers of the denoising U-Net [111] or both at the same time with a hypernetwork [2]. These approaches still struggle to disentangle multiple concepts within the same images (e.g. background or other object bleeding). To this extent, *Break-A-Scene* [4] introduces a mask diffusion loss and a cross-attention loss. Another ongoing debate within the community lies between high-quality, yet expensive, fine-tuning [89] against more efficient and lightweight approaches [103, 49].

Conditioning. In order to sample diffusion models close to a target 2D/3D domain, we can make use of various additional signals including depth, poses, normal maps, etc. These signals can be used to condition the sampling process explicitly via adapters or hypernetworks such as *T2I-Adapter* [68] or *ControlNet* [129]. Another line of work extends classifier-free guidance [39] and re-formulates conditioning as an additional guidance at each step of the sampling process which can support any arbitrary constraint loss [5]. As shown by Liu et al. [56], this can be explained by interpreting diffusion models as *Energy-based models* [51] and has been applied to various types of conditioning signals including color [29], layout [14], faces [5] and complex editing scenarios [24].

Internal representations of diffusion models. At a high level, as diffusion models are trained to approximate the score of a data distribution, they are highly biased towards their training data and majority modes [92, 107]. At a finer level, it has recently been shown that the internal representations of diffusion models capture strong semantic and appearance properties which have been leveraged to perform tasks such as open-vocabulary segmentation [123] and semantic image correspondences [127, 101, 60, 35]. More relevant to our project, *Prompt-to-Prompt* [36] shows that we can use cross-attention maps to perform prompt-guided edits while preserving the layout and appearance of the image. This approach has been extended to perform layout-based or shape-based generation [14, 61, 72], nurse neglected concepts [12, 119], improve compositionality [26], reduce incorrect "visual bindings" [80] and inject additional conditioning [29]. More recently, some works have proposed to manipulate more complex internal representations, namely self-attention maps [8, 41] or even residual feature maps [106, 24].

Chapter 3

Background

In this chapter, we start by introducing Neural Radiance Fields on which our problematization relies and then move on to describing the foundational aspects and formalism behind diffusion models.

3.1 NeRF

Neural Radiance Fields (NeRF) [66] represent a scene as a function f_θ with trainable parameters θ, mapping a 3D position $\mathbf{x}$ and a 3D direction $\mathbf{d}$ to a corresponding RGB color $\mathbf{c}$ and scalar density σ. Images are rendered by approximating the volumetric rendering equation through quadrature. More precisely, NeRF uses volumetric ray-marching [62] along a ray $\mathbf{r}$ according to

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i \quad \text{with} \quad T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{3.1}$$

where samples are taken along the ray $\mathbf{r}$ with intervals δ_i. This differentiable formulation enables gradients to smoothly propagate from image space to the enclosed volume of the scene and allows to learn the appearance and geometry of complex scenes solely from posed images without the need for additional signals. Since there has been an explosion in the volume of NeRF literature in the last few years, we refer to the excellent surveys that have recently been published for a comprehensive review [102, 121]. In our report, we rely mostly on the two recent frameworks *NerfAcc* [53] and *Nerfstudio* [99] for our experiments.

3.2 Diffusion models

Diffusion models were initially introduced to the field of machine learning by Sohl-Dickstein et al. [93] and later revisited in *Denoising diffusion probabilistic modeling* [37]. As the field is evolving fast and already vast, we restrict ourselves to a description of 2D diffusion models only. Images are first mapped to a symmetric domain (namely $[-1, 1]$ or in a latent space as we will describe soon).

A *forward process* or *diffusion process* is then used to progressively turn an initial unaltered image z_0 into a gaussian noise distribution z_T. More precisely, given a noise schedule $(\beta_t)_{t\in[1,T]}$ where $0 < \beta_1 < \ldots < \beta_T$, this process can be defined as a *fixed* markov chain where $p(z_t|z_{t-1}) = \mathcal{N}(z_t; \sqrt{1-\beta_t}z_{t-1}, \beta_t I)$. Interestingly, we can see from this definition that z_t can be expressed directly as a function of z_0 by factorizing the individual forward steps with $p(z_t|z_0) = \mathcal{N}(z_t; \sqrt{\alpha_t}z_0, (1-\alpha_t)I)$ where $\alpha_t = \prod_{i=1}^{t}(1-\beta_t)$. By introducing $\sigma_t^2 = (1-\alpha_t)$, this can be rewritten as

$$z_t = \sqrt{\alpha_t}z_0 + \sigma_t\epsilon_t \text{ where } \epsilon_t \sim \mathcal{N}(0, I) \tag{3.2}$$

The key idea behind diffusion models is to choose the noise schedule denoted equivalently by $(\beta_t)_{t\in[1,T]}$ or $(\sigma_t)_{t\in[1,T]}$ such that $\sigma_T^2 \approx 1$ and thus $p(z_T|z_0) \approx \mathcal{N}(z_T; 0, I)$. In other words, we progressively map a complex data distribution to a simple tractable 2D gaussian distribution $\mathcal{N}(z_T; 0, I)$ from which it is easy to sample. Note that the choice of a proper noise schedule is essential to fully leverage this framework [46, 15].

Now comes the interesting part! By conditioning on z_0, z_{t-1} is then tractable and follows precisely a gaussian distribution w.r.t. z_t (assuming small enough steps). Ho et al. [37] show that we can thus approximate $p(z_{t-1}|z_t)$ with a neural network with a set of weights θ such that $p_\theta(z_{t-1}|z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(x_t, t), \beta_t I)$ by maximizing in a variational way the *Evidence Lower Bound* (ELBO) (note that variance can also be learned in theory but yields another ELBO). In practice, they show that the problem can be reparameterized by learning ϵ_θ instead of μ_θ and results in the following loss:

$$L(\theta) = \mathbb{E}_{t\sim\mathcal{U}([1,T]),\epsilon_t\sim\mathcal{N}(0,I)} \left[w(t)\|\epsilon_t - \epsilon_\theta(z_t; t)\|^2\right] \tag{3.3}$$

In plain English, the diffusion model is trained by taking a step t along the noise schedule $[1, T]$, sampling a random noise vector $\epsilon_t \sim \mathcal{N}(0, I)$, mixing the initial input z_0 with this noise to obtain z_t and predicting the noise that was injected with a neural network $\epsilon_\theta(z_t; t)$ conditioned on t. Note that, in practice, $w(t)$ is usually chosen to be 1 [37]. Finally, after training, sampling can easily be performed by reversing this markov chain using the approximate $p_\theta(z_{t-1}|z_t)$ with *ancestral sampling.* We summarize the process in algorithm 1.

Algorithm 1 DDPM sampling

```
1: $z_T \sim \mathcal{N}(0, I)$
2: for t=T,..., 1 do
3:     $\epsilon_t \sim \mathcal{N}(0, I)$ if $t > 1$ else $\epsilon_t = 0$
4:     $z_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(z_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \cdot \epsilon_\theta(z_t; t)\right) + \sigma_t\epsilon_t$
5: end for
6: return $z_0$
```

As highlighted by Luo [59], diffusion models can be interpreted as *Hierarchical Variational Autoencoders* where the decoding process for each latent z_t is only conditioned on the "previous" latent z_{t+1}. This formulation provides a nice intuition as to how "information" flows during the decoding/denoising process. Additionally, diffusion models are closely related to score-matching models [110, 96] since learning the noise ϵ_θ can be shown to be equivalent to learning the score function of the noisy marginal distributions: $\epsilon_\theta(z_t; t) \approx -\sigma_t \nabla \log p(z_t)$. Song et al. [97] showed that both models can be unified by interpreting the forward diffusion process as a Stochastic Differential Equation (SDE) and the denoising process as a reverse SDE. This interpretation has given rise to a myriad of optimizations of the denoising process and an ODE relaxation, both inspired from the literature on numerical solvers.

Most interesting to us is a specific variant of this ODE formulation, namely *Denoising Diffusion Implicit Models* (DDIM) [94]. DDIM proposes to break the markov chain assumption during sampling by conditionally building on an estimate of z_0 at each step. By removing the dependency to the joint distribution of the diffusion trajectory, this enables a sampling strategy where some steps can be skipped and the amount of injected noise at each step can be adjusted and, in the limit, removed:

$$z_{t-1} = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \cdot z_t + \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}\right) \cdot \epsilon_\theta(z_t; t) \tag{3.4}$$

This formulation has, since then, been widely adopted by the community as it allows to reduce the number of sampling steps (usually from 1000 to only 50) without trading off too much quality. However, we would like to draw the attention of the reader that with this deterministic strategy, errors are progressively accumulated along sampling trajectories and its efficacy thus rely on the amount of error generated at each step!

3.3 Guidance

As introduced in the previous paragraph, diffusion models allow to sample a given distribution in an unconditional way. Yet, we are usually interested in sampling images given a specific signal, e.g. a text prompt. To this extent, Dhariwal et al. [23] introduce *classifier guidance*. The idea can be derived directly from Bayes rule. If we want to sample z conditioned on an arbitrary class c, we can simply write $p(z|c) \propto p(z)p(c|z)^{\gamma}$ where γ can be interpreted as an inverse temperature term and is generally called the *guidance scale*. By noting that in diffusion models we sample from the score of the (approximated) probability distribution $p_\theta(z_t)$, we can rewrite the previous equation accordingly:

$$\nabla_{z_t} \log p_\theta(z_t|c) = \underbrace{\nabla_{z_t} \log p_\theta(z_t)}_{\text{(a)}} + \gamma \underbrace{\nabla_{z_t} \log p_\theta(c|z_t)}_{\text{(b)}} \tag{3.5}$$

where (a) is learned by the diffusion model as presented before and (b) can be approximated by a classifier. Unfortunately, this requires such a classifier to be trained on noisy estimates of the data. To adress this point, *classifier-free guidance* (CFG) [39] proposes to treat $\nabla_{z_t} \log p_\theta(c|z_t)$ as the gradient of an implicit classifier $p_\theta^{\text{imp}}(c|z_t) \propto \frac{p_\theta(z_t|c)}{p_\theta(z_t)}$ such that equation 3.5 becomes

$$\nabla_{z_t} \log p_\theta(z_t|c) = \nabla_{z_t} \log p_\theta(z_t) + \gamma \left[\nabla_{z_t} \log p_\theta(z_t|c) - \nabla_{z_t} \log p_\theta(z_t)\right] \tag{3.6}$$

In practice, equation 3.6 is implemented by training ϵ_θ conditionally on the class c and an unconditional null class/embedding $\emptyset$ in a *dropout* fashion. During sampling, $\epsilon_\theta(z_t; t, c)$ can thus be replaced by $\hat{\epsilon}_\theta(z_t; t, c)$ where

$$\hat{\epsilon}_\theta(z_t; t, c) = \epsilon_\theta(z_t; t, \emptyset) + \underbrace{\gamma \left[\epsilon_\theta(z_t; t, c) - \epsilon_\theta(z_t; t, \emptyset)\right]}_{\text{guidance term}} \tag{3.7}$$

Note that *classifier-free guidance* requires two evaluations of the diffusion model. Furthermore, increasing the *guidance scale* helps in finding modes of the data distribution. However, it comes with some costs (on which we shall elaborate further down in this report) among which high saturation and high contrast are common examples. To prevent the latter, Saharia et al. [91] introduces a *dynamic thresholding* operation.

The classifier guidance equation 3.5 suggests a flexible extension. As highlighted by previous works [124, 56, 5, 24], $\nabla_{z_t} \log p_\theta(c|z_t)$ can be interpreted as the gradient of an arbitrary energy function $g(z_t; t, c)$, which enables the use of various regularizations inside the diffusion process itself. $g(z_t; t, c)$ can be an arbitrary image space loss [5], a loss conditioning internal feature maps of the diffusion model [24] or even a diffusion model itself [56, 124] (by interpreting the latter as a specific *Energy Based Model* [51]). We end this paragraph by writing down a more general form of guidance.

$$\hat{\epsilon}_\theta(z_t; t, c) = \epsilon_\theta(z_t; t, \emptyset) + \underbrace{\gamma \left[\epsilon_\theta(z_t; t, c) - \epsilon_\theta(z_t; t, \emptyset)\right]}_{\text{CFG}} + \underbrace{\sigma_t \sum_{i=1}^{n} \gamma_i \nabla_{z_t} g_i(z_t; t, c)}_{\text{energy-based guidance}} \tag{3.8}$$

3.4 Diffusion model architectures

In this paragraph, we briefly review existing diffusion model architectures and their specificities, especially *Latent Diffusion Models* [87] on which most of the experiments presented in the next chapters are based.

Most denoising networks share the same backbone implemented as a *U-Net* [88], which is composed of several *ResNet* blocks [34] following a u-shape downsampling/upsampling path where the downsampling half is often called the *encoder* and the upsampling one the *decoder*. Note that this choice can be confusing with the VAE counterparts used in *Latent Diffusion Models* [87] as introduced further away. This nested architecture permits the extraction of information at different scales and has been shown to be particularly efficient to extract semantic concepts. As diffusion models are meant to be generative, the *decoder* is usually implemented with more layers than the *encoder* (e.g. *Stable Diffusion* has 6 encoding blocks, 1 bottleneck block and 9 decoding blocks).

In order to support text-guidance, these models often make use of pre-trained text encoders, which can be BERT [22], T5 [78] or CLIP text encoders [77]. The prompts are first tokenized, mapped to *token embeddings*, contextualized thanks to the text encoder to give *text embeddings* and injected in multiple blocks of the U-Net thanks to a cross-attention mechanism [108]. More formally, let l denote the index of a residual block in the U-Net and e the text embeddings, the features from the previous $(l-1)$ block first go through the l-th residual block and yield intermediate features $f^{(l)}$. The latter are then projected to yield a query matrix $Q^{(l)} = P_Q^{(l)}(f^{(l)})$ while text embeddings are projected to a key matrix $K^{(l)} = \phi_K^{(l)}(e)$ and value $V^{(l)} = \phi_V^{(l)}(e)$ matrix thanks to learned linear projections $\phi_Q^{(l)}, \phi_K^{(l)}, \phi_V^{(l)}$. This results in the attention map:

$$A^{(l)} = \text{Softmax}\left(\frac{Q^{(l)}{K^{(l)}}^T}{\sqrt{d}}\right) \tag{3.9}$$

where $(A^{(l)})_{i,j}$ quantifies how much the (contextualized) token j attends to pixel i and d is the dimension resulting from the projection operation. The output of the cross-attention layer is then $A^{(l)}V^{(l)}$. Additionally, some Diffusion Models (e.g. *Stable Diffusion*) make use of an additional self-attention layer which follows the same definition except that the text embedding e is replaced by the activated feature maps themselves.

As the diffusion framework requires to disentangle complex concepts and modes in the high-dimensional space of images, two main architectures have emerged:

- **Latent Diffusion Models** (LDMs) [87] reduce the dimensionality of the input images by compressing them thanks to a *Variational autoencoder*. In other words, the initial image x_0 is mapped to a latent $z_0 = \mathcal{E}(x_0)$ thanks to an encoder $\mathcal{E}$, the diffusion process is then carried out in this lower-dimensional space and the result is decoded thanks to a decoder $\mathcal{D}$. In practice, *Stable Diffusion* maps $512 \times 512 \times 3$ images to $64 \times 64 \times 4$ latents using a VAE that is trained using an adversarial loss inspired from *VQGANs* [25] (please refer to appendix A for a more complete description).
- **Cascaded Diffusion Models** [38], among which *Imagen* [91] and *DeepFloyd IF* [1] are examples, split the complexity of image generation in a hierarchy by first training a diffusion model to operate at low resolution (usually 64×64) followed by upsampling super-resolution models conditioned on the result of the base model (usually $64 \times 64 \rightarrow 256 \times 256$ followed by $256 \times 256 \rightarrow 1024 \times 1024$). As we will discuss further down in this report, these models tend to perform better at modeling the modes of the training data because the compression is intrinsic (image size) and representations are learned directly by the diffusion model: LDMs learn a VAE independently of the diffusion process thus making assumptions on the nature of information within the input data!

As we shall deal with both the image space and the latent space (in the LDM sense) in the forthcoming chapters, we lift any ambiguities by denoting by z_t the signal in the space where the diffusion process actually takes place (this can be either the image space for cascaded diffusion models[38] or the VAE latent space for latent diffusion models[87]) and x_t the corresponding representation purely in image space. In other words, for image space diffusion $x_t = z_t$ and for latent-based diffusion $x_t = \mathcal{D}(z_t)$.

Chapter 4

Distilling LDM features

In this chapter, we detail our first attempt at leveraging diffusion models in the context of "neural extrapolations". The latter was motivated by the apparent compressive power of the LDM latent space (in our case *Stable Diffusion*) and a recent generalization of NeRF, namely *Feature fields* [48, 104]. We start by introducing the latter, present our proposed method and explain the mechanisms behind its failure.

4.1 Feature fields

With the success of NeRF at learning color in a volumetric sense, it was not surprising to see that other signals could be optimized volumetrically. *Feature Field Distillation* [48] and *Neural Feature Fusion Fields* [104] concurrently proposed to train an additional neural branch of the NeRF to predict high-dimensional features from self-supervised 2D image feature extractors, namely CLIP-LSeg [52] and DINO [9]. In other words, a feature vector $\mathbf{f}(\mathbf{x})$ (usually view-independent) is learned in addition to density $\sigma(\mathbf{x})$ and color $\mathbf{c}(\mathbf{x}, \mathbf{d})$ by using the same volume rendering formulation. This branch can be trained jointly with RGB [48] or in a second stage using the guidance of the predicted volumetric density [104].

As shown in both works [48, 104], this process is effective and enables both 3D segmentation and query-based decomposition and editing. The latter was later improved by Goel et al. [30] using image processing methods. Furthermore, by making use of patches and multiple scales, *LERF* [45] recently extended the process to image-text features thus enabling pixel-aligned queries from text prompts.

4.2 Our method

We first proposed to leverage *Feature Fields* by distilling pixel-aligned 4-dimensional features from the latent space of *Stable Diffusion v1.4*. To avoid any interference that may be due to the *a priori* perceptual nature of these features, we first learned the geometry (i.e. the density field) using the RGB signal in the input images and then used it as an "oracle" to distill LDM features. Additionally, to avoid any information bottleneck, we propose to train a completely independent *dual network* i.e., not sharing the same backbone for $\mathbf{f}$, $\mathbf{c}$ and σ contrary to what has been done in previous works [104, 48]. This *dual network* has the same capacity as the vanilla network NeRF MLP, namely 8 layers of size 256 with a skip connection. We also investigated the impact of modeling a sharp density distribution (using the regularizer introduced by *Mip-NeRF 360* [6]) and making the features view-dependent (i.e., $\mathbf{f}(\mathbf{x}, \mathbf{d})$).

In table 4.1, we present the reconstruction results using both PSNR and a perceptual metric, namely LPIPS [131], on a subset of the LLFF dataset [65]. Each metric is computed on a validation set of the corresponding scene. We denote as *simple* the feature branch with a shared 256-dimensional layer from previous works [104, 48] and *dual* our *dual network* approach. We also compare the reconstructions from distilled features (i.e., after decoding with $\mathcal{D}$) with what we would obtain from the corresponding ground truth features ("GT encoded/decoded" in the table); in other words, we simply encode/decode

	PSNR				LPIPS			
	fern	trex	flower	benchflower	fern	trex	flower	benchflower
NeRF baseline	26.55	28.08	27.98	24.18	0.143	0.109	0.116	0.151
Distilled *simple*	17.78	**17.21**	18.09	**16.08**	0.590	0.548	0.577	0.659
Distilled *dual*	**17.88**	17.13	18.18	15.86	0.455	0.458	0.477	**0.527**
Distilled *dual* w. dist loss	17.85	17.07	18.19	15.82	0.456	0.461	0.478	0.529
Distilled *dual* w. direction	17.72	**17.21**	**18.29**	15.60	**0.450**	**0.452**	**0.475**	0.516
GT encoded/decoded	22.21	23.29	24.69	18.57	0.131	0.109	0.158	0.219
Distilled fine-tuned	20.84	21.48	22.04	18.87	0.330	0.303	0.354	0.468
GT enc/decoded fine-tuned	16.54	16.12	18.91	15.72	0.283	0.260	0.280	0.361

Table 4.1: We observe that distilling *Stable Diffusion* pixel-aligned features using volume rendering, either in a feature branch relying on the same intermediate activations as RGB and density branches (*simple*) or as an independent *dual network* (*dual*), yields poor overall reconstructions across several scenes from the LLFF dataset. Fine-tuning the VAE decoder on the training set of images mitigates the effect but hinders dramatically the generalization capabilities of the corresponding latent space as shown in the last row.

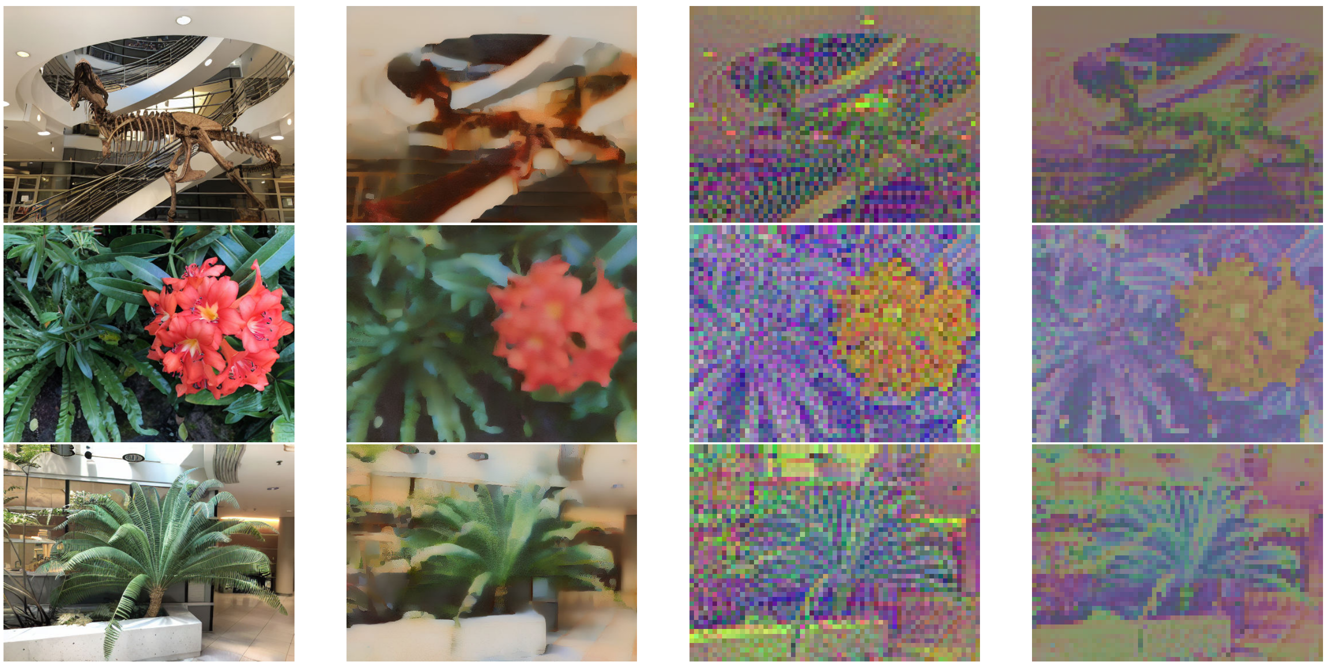

(a) Ground Truth encode/decoded (b) Distilled features decoded (c) Ground truth features (d) Distilled features

Figure 4.1: Results of the LDM feature distillation on the LLFF dataset. Note that for the distilled features, we only visualize the first 3 channels of the corresponding latent space.

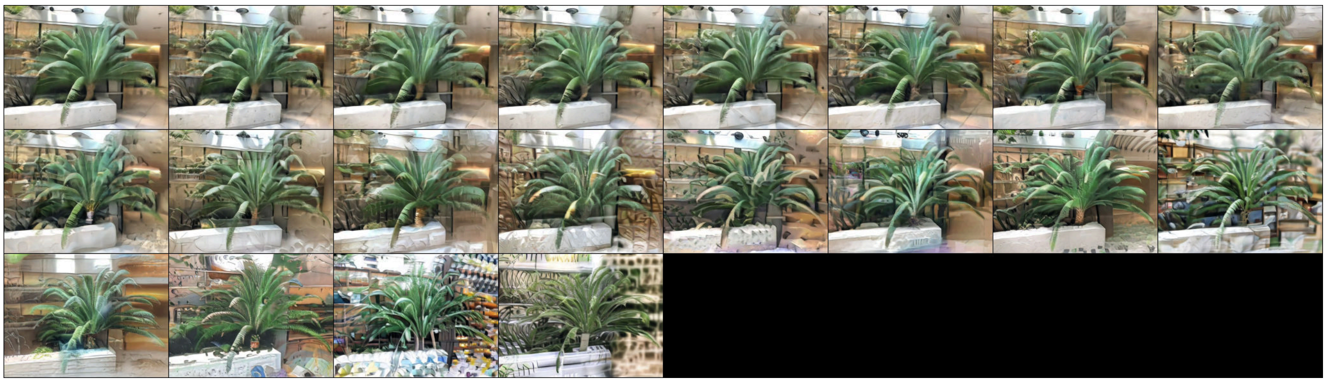


Figure 4.2: In this experiment, we start from the image in the top-left hand corner and proceed to a noise/denoise operation as introduced in *SDEdit* [63] with increasing amount of noise from left to right. The resulting features are then decoded with the fine-tuned decoder trained on NeRF distilled features. Note that in this experiment, there is no NeRF or distilled features directly involved in the sampling process!

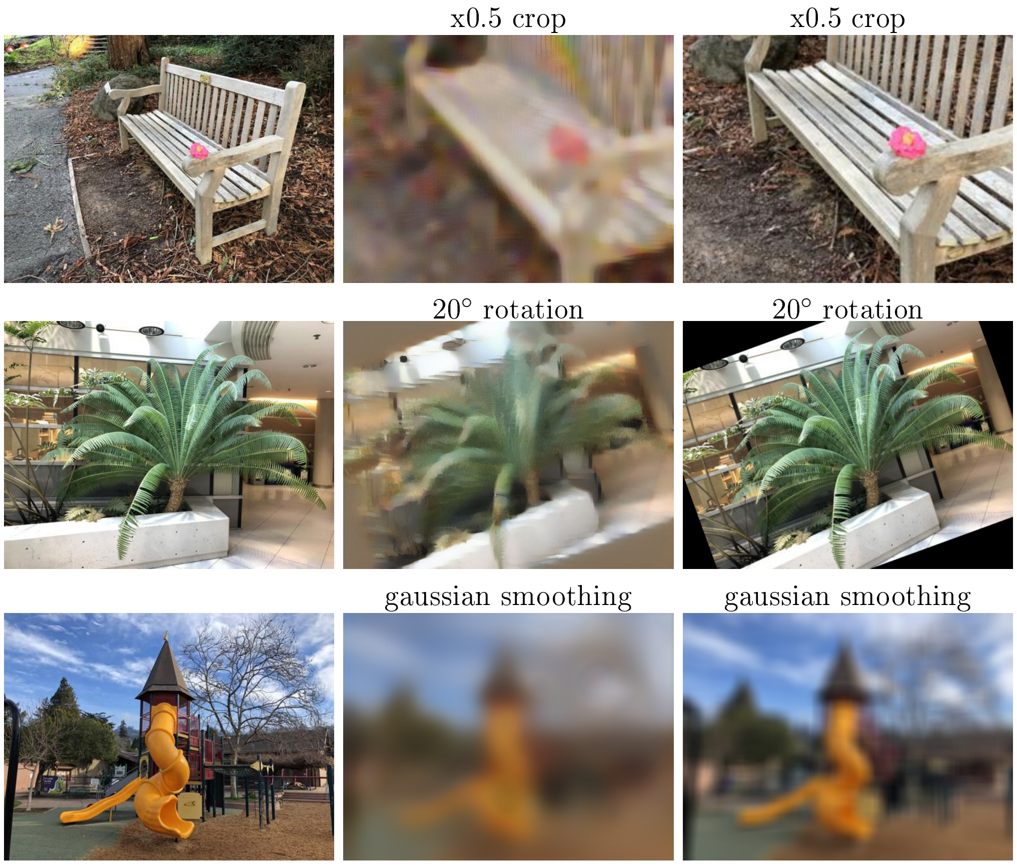
the validation images $\mathcal{D}(\mathcal{E}(x_0))$. In Fig. 4.1, we visualize these results. We can clearly observe that only low-frequency details are recovered.

Additionally, we tried to fine-tune the decoder $\mathcal{D}$ on a per-scene manner by rendering the training images with our distilled features and regressing their decoded reconstructions to match their RGB references. As shown in the last two rows of table 4.1, this significantly improved both the absolute and perceptual metrics but like any fine-tuning it came at the cost of severe overfitting on the regressed distribution, completely removing the generalization capabilities of the decoder as shown in the last column where we run the same encode/decode operation on the validation features, this time with the fine-tuned decoder. Worse, as the diffuser U-Net assumes the latents $z_0, z_1, \dots, z_T$ to live in a homogeneous space, fine-tuning the decoder causes the U-Net to produce (after diffusion) completely misaligned samples as shown in Fig 4.2.

4.3 "Lost in Translation"

With the primary observations of the previous paragraph, we now attempt to explain the reasons behind the unsatisfactory phenomenon. We start by providing the key intuition: volume rendering as done according to equation 3.1 proceeds by mixing features from different viewpoints that map to the same location in a pixel-wise manner. However, *Stable Diffusion*'s VAE relies on convolutions to establish the features that will be blended. Unfortunately, these convolutions operate on a fixed, limited and box-sampled receptive field to produce a single feature. In other words, each feature that the encoder produces depends on the surrounding aligned pixels in the original RGB image. This implies two things: 1. simply applying non-translational operations (i.e., out of the domain of invariances of the CNN backbone of the VAE) like scaling or rotations will result in different features, 2. different viewpoints result in different neighborhood for each pixel (e.g. disocclusions) especially as the baseline becomes large between them. Both observations cause the features to be blended during volume rendering, thus resulting in a significant loss of information (which explains that only low-frequency details are recovered). We now proceed to justifying these two observations with two dedicated experiments.

Non-invariance to transformations. We first show that applying simple 2D transformations in the latent space results in flawed RGB reconstructions when feeding the corresponding transformed latent to the decoder. Note that this is not too big of a surprise as it is deeply tied to the nature of convolutional networks, but this experiment provides a nice *sanity check*. More precisely, we proceed by encoding an original image using the encoder, performing a transformation of the corresponding latent in the latent space, decoding it and comparing the reconstruction with the equivariant transformation in image-space. Fig. 4.3 illustrates the process and some of its results. We experiment with cropping, rotation and gaussian smoothing. Note that in both Fig. 4.3 and Tab. 4.2 the results are shown for bilinear resampling after the transformations; we experimented with hard nearest resampling too, but this resulted in even worse

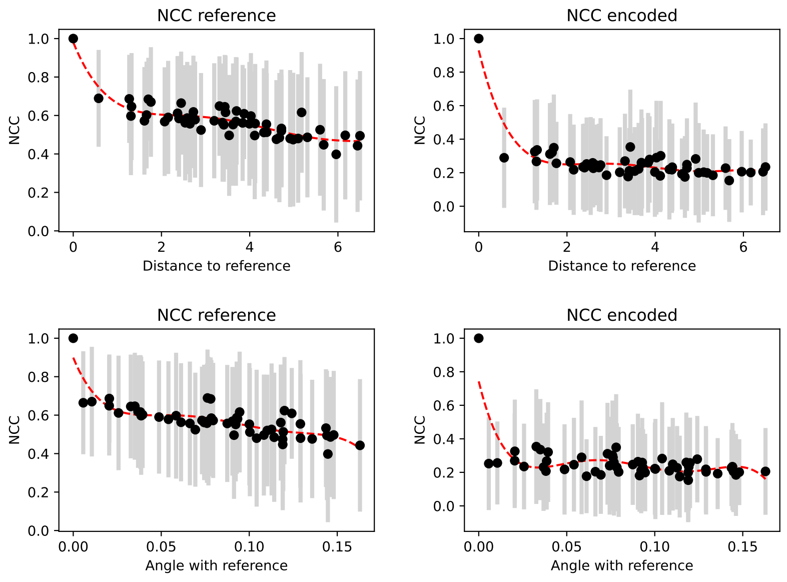


(a) Ground Truth image (b) Decoded transformed latent (c) Transformed GT image

Figure 4.3: In this experiment, we start from an image in the column (a), encode it using *Stable Diffusion*'s encoder, perform a non-translational transformation in its latent space and decode it using the decoder (b). We then compare the result with the equivariant transformation in image space (c).

		Identity	Crop			Rotation				Gaussian filter		
			x0.25	x0.5	x0.75	5°	10°	20°	45°	$\sigma = 4$	$\sigma = 8$	$\sigma = 16$
PSNR	fern	22.86	19.72	20.42	20.16	17.30	15.95	14.46	13.07	28.25	25.40	22.41
	flower	25.37	19.43	19.89	20.45	18.29	16.66	14.94	13.25	28.93	21.74	20.18
	benchflower	20.05	17.06	17.59	18.10	16.69	15.52	14.15	12.82	25.66	24.57	25.66
	playground	22.66	17.87	19.19	19.87	17.62	16.17	14.60	13.23	27.82	24.60	22.44
LPIPS	fern	0.115	0.663	0.616	0.468	0.383	0.420	0.478	0.518	0.483	0.478	0.427
	flower	0.153	0.546	0.587	0.512	0.470	0.488	0.511	0.528	0.479	0.483	0.479
	benchflower	0.204	0.701	0.653	0.552	0.511	0.517	0.552	0.583	0.635	0.708	0.595
	playground	0.149	0.574	0.579	0.481	0.430	0.444	0.480	0.520	0.543	0.498	0.429

Table 4.2: Performing transformations (cropping, rotation, and gaussian filtering) in the latent space of *Stable Diffusion* and comparing the decoded reconstructions with the equivariantly transformed RGB shows that the latent space is hardly robust to geometrical transformations. Note that gaussian filtering results yield better pixel-wise and perceptual metrics than the identity (RGB ground truth vs. $\mathcal{D}(\mathcal{E}(x_0))$) as we compare against the transformed (and thus gaussian filtered) RGB image.



(a) RGB space (b) Latent space

Figure 4.4: Average *Normalized Cross Correlations* (NCC) for matched patches in RGB (a) and latent space (b) as a function of distance (first row) and angle (second row) for the *trex* scene of the LLFF dataset.

results. Interestingly, the artifacts in the reconstruction that we obtain are consistent with the renderings that we showed in Fig. 4.1 in the previous paragraph. This suggests that this is the very same precise phenomenon that is causing the partial failure of our distillation process.

Local patch matching. We hypothesize that due to both the adversarial training procedure inspired from *VQGAN* [25] and the convolutional nature of *Stable Diffusion*'s VAE, the latter encodes information directly in the spatial layout of the resulting latent feature maps to maximize "perceptual compression". To validate this hypothesis, we propose a simple experiment. We start from a scene of the LLFF dataset [65]. We select a "reference" image e.g. "most in the middle" (which is arguably reasonable for the LLFF dataset as the distribution of images is well-behaved), find, match and filter features with all the other images of the dataset in a pair-wise manner. We extract patches for each match in both RGB space and the latent space (with equivariant patch sizes for the comparison to make sense) and we compute the average (with variance) *Normalized Cross Correlation* (NCC) of these patches as a function of either the distance or the angle with the reference image. Fig. 4.4 shows our results on the *trex* scene of the LLFF dataset. In the original RGB space, we can observe an overall decreasing trend in the correlations between patches as angle and distance vary. On the other hand, the latent space shows rather decorrelated latents no matter which views we compare.

4.4 Further comments and discussions

The experiments in the previous paragraph not only confirm our initial intuitions but also shed light on the more general limitations of feature distillation within NeRFs. The two key takeaways are:

1. to know how the features we wish to distill are "built": both structurally, e.g. resulting from a CNN backbone, and implicitly, e.g. trained in an adversarial way to compress perceptual information.
2. what the target application is: for segmentation, only hard volumetric boundaries may be sought and thus "smoothed" features may still do the job. However, in our case, for visual reconstruction, it is essential to recover all aspects of the encoded signals which is unfortunately lost in the blending.

At this stage, our conclusions may also be arguable in two ways:

1. Why cannot we just learn view-dependent features to address the problem? As shown in Tab. 4.1, despite slightly improving the results, this is insufficient to fully address the problem as it is not

only related to the viewing direction but also the box-sampling operation of the image formation process. A *discretized frustrum-dependent* parameterization could be an interesting approach, of which *LERF*'s [45] scale parameterization might be the closest existing analog we can think of. Yet, it is not clear how this could be achieved practically (especially for the specific case of the box-sampling step of the pixel image formation process) and, first and foremost, introducing additional parameters might just lead to overfitting the input images without any chance of generalizing to new viewpoints.

2. *IBRNet* [115] and *pixelNeRF* [125] both rely on frustrum-aligned CNN features that are merged across views in a non-translational manner. Yet, it seems to actually help in providing the NeRF backbone with additional information. In their case, it is essential to note that the features are trained in a completely end-to-end manner. More precisely, these features are trained jointly with the pooling operation: multi-view consistency is thus inherently embedded in their construction! On the contrary, we tried to build on pre-existing features trained to pseudo-maximize the amount of visual information in a purely independent and 2d manner.

Chapter 5

Score Distillation Sampling

The initial and core motivations behind our experiment to distill VAE-like features in a volumetric way were:

1. *Stable Diffusion* was the only true model whose weights were available at the start of the project. Since then, non-LDM models like *DeepFloyd IF* have been released.
2. Being a latent diffusion model, we needed a way to leverage its denoising power in a multi-view consistent way. More concretely, this would have enabled us to sample new out-of-distribution views, which could have been refined by injecting the powerful priors of such models thanks to a partial noise/denoise operation as introduced in *SDEdit* [63].

Unfortunately, as we presented in the previous section, this was hardly tractable due to the intrinsic (convolutional) and "frozen" nature of the investigated features. We then turned ourselves to a more implicit form of distillation, namely *Score Distillation Sampling* (SDS) [76]. To this extent, we first start by reviewing the latter and then highlight the limitations that we discovered through carefully conducted experiments.

5.1 Background

Score Distillation Sampling (SDS) was recently introduced by *DreamFusion* [76] and *Score Jacobian Chaining* [113] for the task of text-to-3d generation. As its name suggests, it proposes to distill within a differentiable renderer (e.g. a NeRF) the 2D priors of a pre-trained conditional diffusion model by leveraging the denoising score matching objective introduced in equation 3.3.

To do so, SDS builds on the following assumption: the probability of rendering a 3D model or scene is proportional to the expected probability of this object sampled at different viewpoints through a text-conditioned diffusion model. We can already see in this formulation that this will raise two main problems. The first will directly affect the task of *text-to-3D* generation because, by the same equation 3.3, large-scale 2D diffusion models are estimators of the probability distributions they are trained on and not all angles of the same object are equally represented in the training data. In other words, they are biased towards majority modes and categories and as a consequence, the model will try to reproduce them more predominantly when sampling arbitrary viewpoints. This gives rise to the *Janus problem* whereby models optimized with SDS show several faces. The second builds on the first one by noting that to leverage SDS for "neural extrapolations", we will need to bring the distribution of the diffusion model close to the one of the scene we are initially provided with!

Formally, let $g_\phi : \xi \mapsto x_0$ an arbitrary differentiable renderer that maps rendering parameters ξ (e.g. camera pose) to a rendering $x_0 = g_\phi(\xi)$. SDS proceeds by taking noisy estimates of these renderings through the forward noising markov chain $x_t = \sqrt{\alpha} x_0 + \sigma_t \varepsilon_t$ and differentiating the loss in equation 3.3

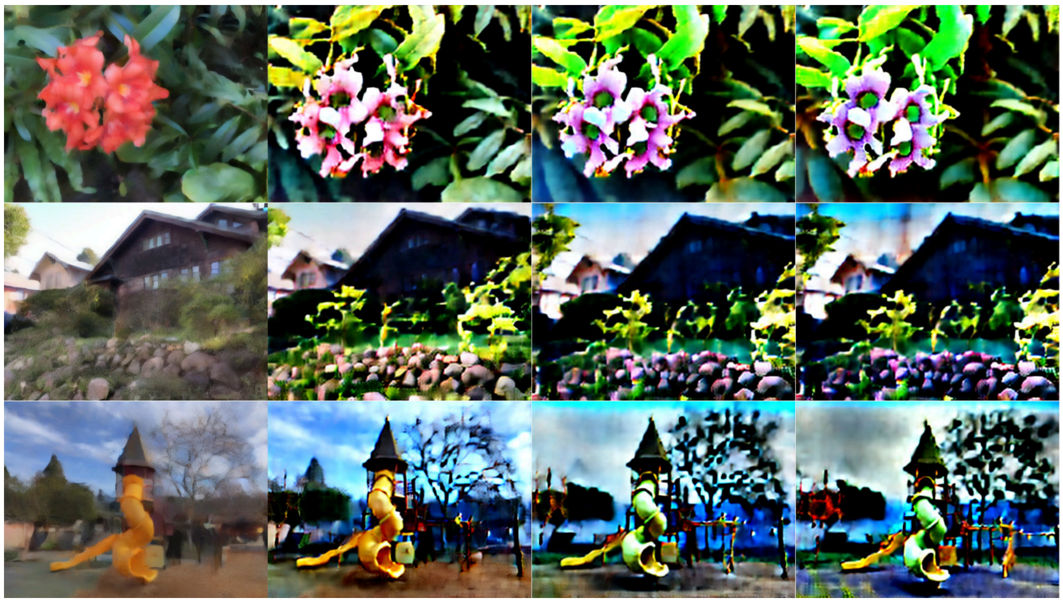


Figure 5.1: We apply SDS to our previously distilled scenes (cf. chapter 4). From left to right, we display the initial pre-trained scene (decoded with the VAE) and the progressively optimized versions of it. The prompts we used were, from top to bottom: "*a DSLR photo of a flower in a garden*", "*a DSLR photo of a wooden house with a garden, outdoor*", "*a DSLR photo of a playground, outdoor, sunny*".

w.r.t. the parameters of the renderer this time:

$$\nabla_\phi L = \mathbb{E}_{t\sim\mathcal{U}([1,T]),\epsilon_t\sim\mathcal{N}(0,I)}\left[w(t)\underbrace{(\epsilon_\theta(x_t;t)-\epsilon_t)}_{\text{Noise Residual}}\underbrace{\frac{\partial\epsilon_\theta(x_t;t)}{\partial x_t}}_{\text{U-Net}}\underbrace{\frac{\partial g_\phi}{\partial\phi}}_{\text{Renderer}}\right] \tag{5.1}$$

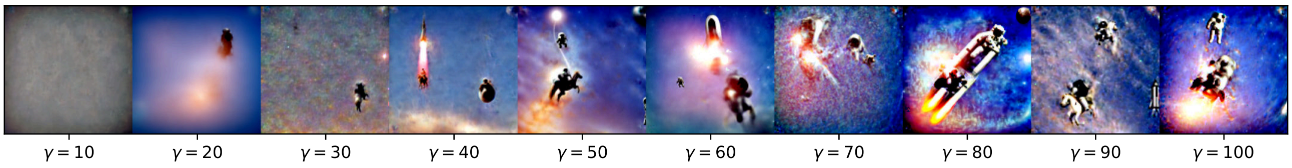
The next "trick" behind SDS is to drop the gradient of the U-Net in the expression above. In the original paper, Poole et al. [76] argue that this is motivated by the expensive nature of such a computation. However, a more valid argument may be that such a gradient is very "noisy" and would cause the supervision signal to be hardly reliable. This yields the now famous SDS loss:

$$\nabla_\phi L_{\text{SDS}} = \mathbb{E}_{t\sim\mathcal{U}([1,T]),\epsilon_t\sim\mathcal{N}(0,I)}\left[w(t)(\epsilon_\theta(x_t;t)-\epsilon_t)\frac{\partial g_\phi}{\partial\phi}\right] \tag{5.2}$$

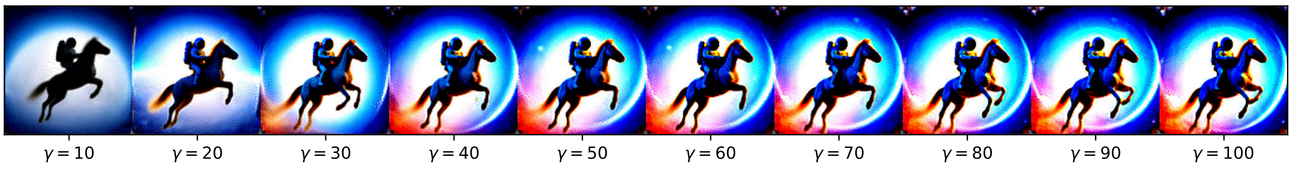
For the reason mentioned above, the problem is strongly under-constrained! We can already guess that the regression signal in equation 5.2 is very brittle: we learn the scene parameter by sampling an arbitrary time-step, predicting the noise from a noisy estimate of our scene and we backpropagate with SGD. Yes, that is a combination of low-information and hardly robust signals! Furthermore, to that, we must add the conflicts induced by sampling different viewpoints with the inherent bias of the diffusion model that would tend to align the views of an object to their most common ones in the training data. Therefore, *Score Distillation* requires a particularly high text guidance usually of the order 100 for a 3D differentiable renderer like NeRF which is more that one order of magnitude higher than its usual value during sampling (between 2.5 and 7.5). As shown theoretically by Poole et al. [76], the *SDS* loss can be seen as the reverse KL-divergence of the differential renderer w.r.t. the learned score function of the diffusion model. This means that SDS proceeds in a *mode-seeking* way and explains partially (or intuitively) why at high guidance, high-frequency details are lost, prompt ambiguities are unresolved (e.g. some concepts of the prompt are not recovered) and colors diverge towards high saturation.

5.2 First experiments

Score Distillation inside Feature Fields. To highlight some of the issues we just mentioned, we start by running score distillation in our distilled scenes. Note that as we make use of a *Latent Diffusion*



(a) without augmentations



(b) with augmentations

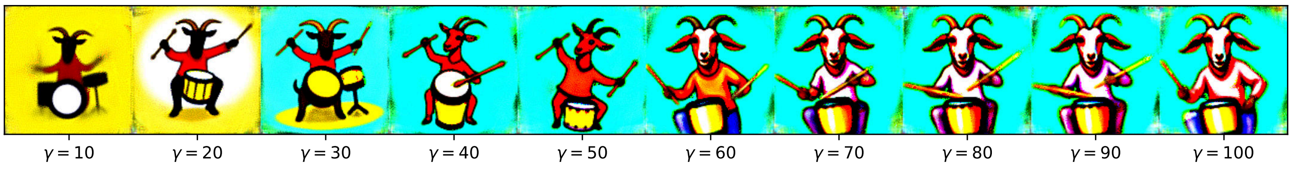
Figure 5.2: With *DeepFloyd IF*, using random augmentations is primordial to find consistent modes of the data, even at very high guidance. In both cases, we used the prompt "*a picture of an astronaut riding a horse*".

Model, we perform SDS directly in the VAE latent space. This is the approach recently chosen by *Latent-NeRF* [64]. As we highlighted in the previous chapter this means that we will suffer from the very same feature smoothing but this will give us some intuitions of the main challenges that there is to tackle. As *Score Distillation* relies fundamentally on text guidance, we manually forge a prompt describing each scene (see the caption of Fig. 5.1). We first observed that optimizing the density (in addition to the VAE features) resulted in complete collapse during optimization and thus we show in Fig. 5.1 results where only the latent feature field was optimized. As can be observed, the scenes progressively converge towards a high saturated mode. Nonetheless, the resulting colors seem consistent with the semantic of the objects they are representing.

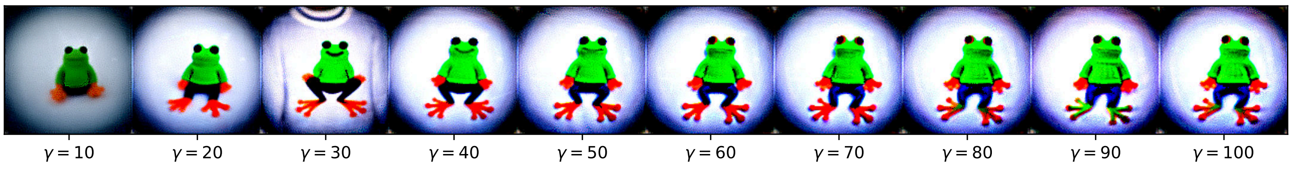
5.3 Exploring the parameter space of SDS

We then proceed to an exploration of the parameter space of SDS by introducing a 2D toy renderer. Instead of using a NeRF as the differentiable renderer in equation 5.2, we simply train a 2D grid of pixels constrained to stay within $[0, 1]$ by a clipping operation. This very simple setup allows us to precisely isolate the impact of each parameter in the image formation process as we present and explain below. Note that initially, all the experiments given below were conducted with *Stable Diffusion v2.1* as *DeepFloyd IF* was released in the last weeks of the project. For the results with *DeepFloyd IF*, we used the largest model available namely *DeepFloyd/IF-I-XL-v1.0* available on Hugging Face and only focused on the first 64×64 stage of this cascaded model. In all experiments (except when specified), we used a 128×128 renderer and proceeded to random augmentations in the form of arbitrary rotations between $[-30°, 30°]$ and scaling between $[0.6, 1.2]$.

- **Random augmentations:** *DreamFusion* [76] emphasized the importance of random augmentations. The first one was, of course, random camera sampling to avoid overfitting to a specific viewpoint. Yet, perhaps more importantly, they showed how crucial using a *shading* model with random light sampling was to fully recover both fine and multi-view consistent details. In our 2D example, we observe that introducing the random augmentations listed above is also essential to the image optimization process as shown for different guidance weights in Fig. 5.2. As pointed out by *Vector-Fusion* [44], this is essentially connected to the absence of a proper initialization. We indeed observed that using a seeding image helps the process converge towards a consistent final rendering.
- **Guidance:** Even for the simpler and better-conditioned scenario of a 2D renderer, we observe that in order to find modes of the data, it is particularly essential to use high text guidance. Fig. 5.3



(a) "*a picture of a goat playing the drums*"



(b) "*a DSLR photo of a frog wearing a sweater*"

Figure 5.3: Using high guidance is essential to reach consistent and sought modes of the data but even so, some concepts tend to be lost or confused due to the heavily mode-seeking behavior of SDS (see the ambiguity with the sweater around $\gamma = 30$ in (b))

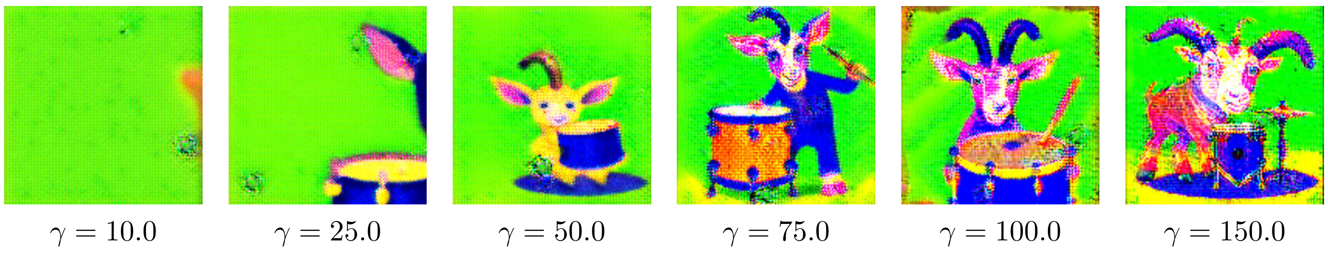
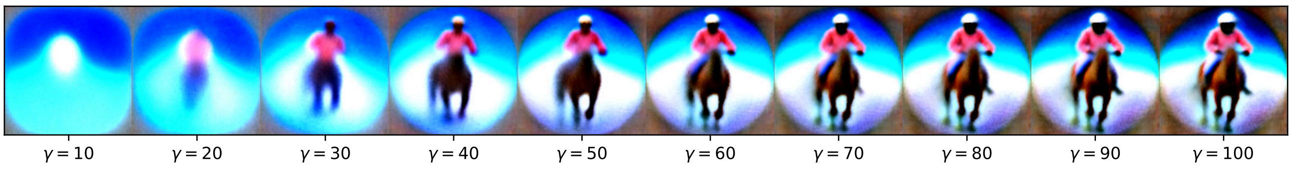
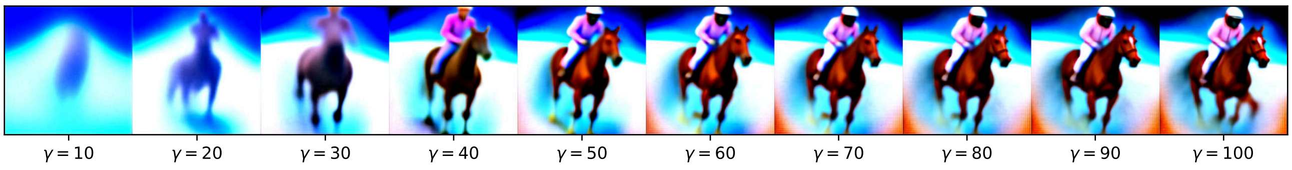


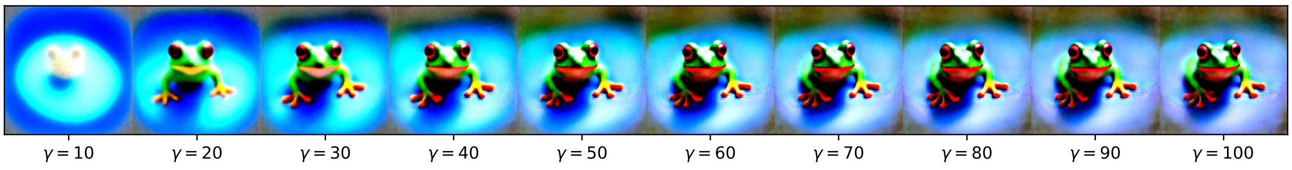
Figure 5.4: Trying to perform SDS with *Stable Diffusion* by differentiably optimizing an image in RGB space, thus requiring backpropagation through the VAE encoder, requires extremely high guidance for convergence and is prone to various artifacts.



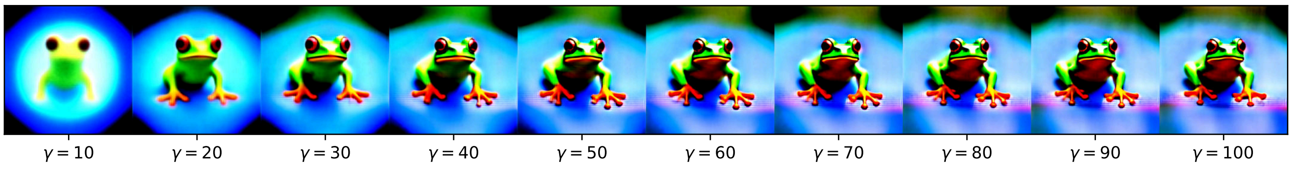
(a) "*a picture of an astronaut riding a horse*" with random augmentations



(b) "*a picture of an astronaut riding a horse*" without random augmentations

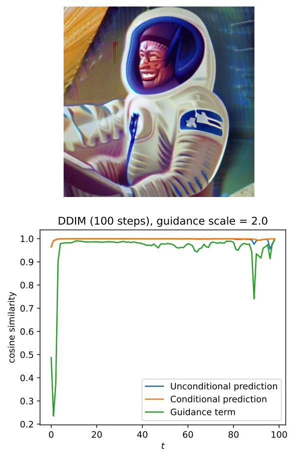


(c) "*a DSLR photo of a frog wearing a sweater*" with random augmentations

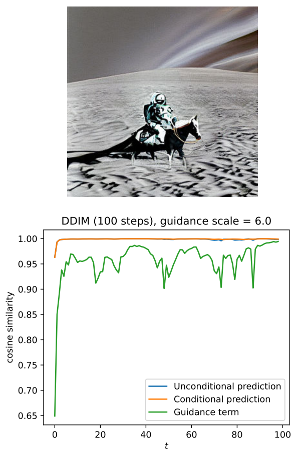


(d) "*a DSLR photo of a frog wearing a sweater*" without random augmentations

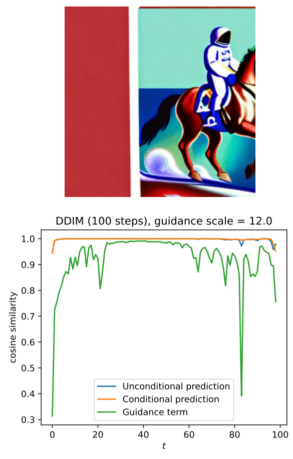
Figure 5.5: Performing SDS directly in the latent space of *Stable Diffusion* alleviates most of the issues highlighted in Fig. 5.4. However, as the latent space is not robust to transformations, using random augmentations results in over-smoothing as shown in (a, c) compared to (b, d).



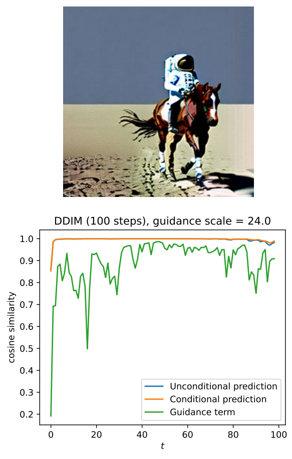
(a) $\gamma = 2.0$



(b) $\gamma = 6.0$



(c) $\gamma = 12.0$



(d) $\gamma = 24.0$

Figure 5.6: Similarity trajectories of the conditional/unconditional noise estimates and the guidance term during DDIM sampling (100 steps) for different guidance scales. The generated images are shown above each plot.

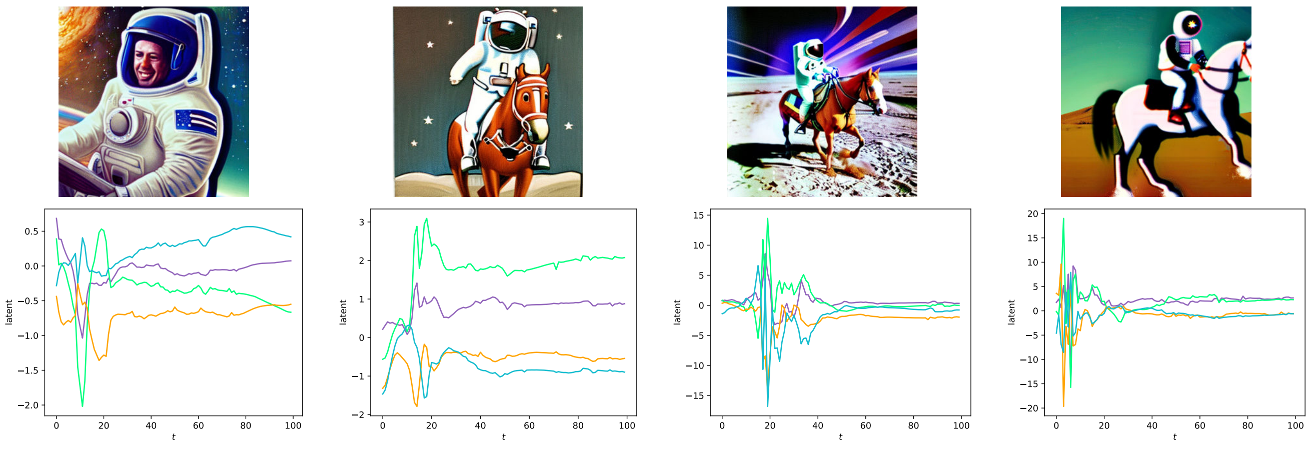
shows that at very low guidance, SDS does not converge to valid modes of the data. As it increases, the overall shape becomes distinguishable and using higher levels allows us to recover finer details. As discussed below, this is also highly dependent on the choice of parameterization and the diffusion model used to perform distillation.

- **Parameterization:** When using *Stable Diffusion*, we compare two different approaches. The first, rather naive, one consists in parameterizing the renderer as an RGB image that we then feed to the VAE encoder. As shown on Fig. 5.4, the addition of an intermediate network (the encoder) introduces strong instabilities. This has two main consequences: 1. the problem becomes even more unconstrained, requiring higher guidance to converge and thus producing very high saturation, 2. it comes with severe artifacts that are closely related to the representations 2D CNNs have been shown to exhibit [71]. These two observations strongly corroborates the previously mentioned argument that the Jacobian of large neural networks (in this case the VAE) are often very noisy. On the other hand, parameterizing the renderer in the latent space directly makes the optimization more stable as shown in Fig. 5.5. This might explain the choices of *Latent-NeRF* [64] to train a NeRF in the latent space of *Stable Diffusion* and to use of a linear approximation of the latent-to-RGB mapping (way better-conditioned than the large decoder!). However as we highlighted in the previous chapter, latent features are not robust to non-translational transformations and using random augmentations as proposed above causes over-smoothing (see the difference in Fig. 5.5).

5.4 Digging further

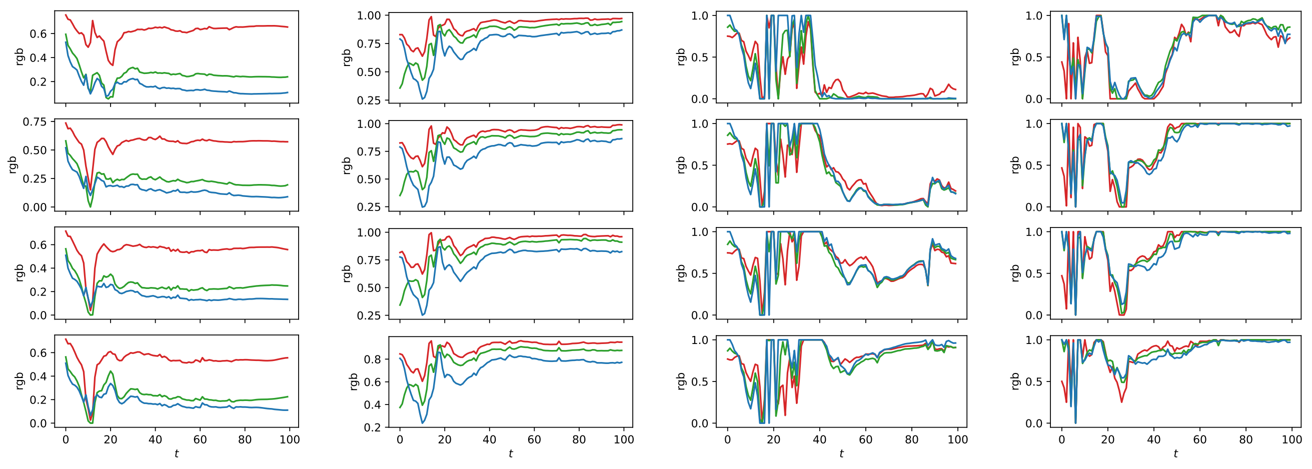
Having explored the many parameters of score distillation on a toy 2D renderer example, we now seek to explain some of the observed phenomena.

We start by investigating the effect of the guidance term in equation 3.7. Inspired by the methodology proposed by *EDICT* [112], we show in Fig. 5.6 how the unconditional noise estimate, the conditional noise estimate and the corresponding guidance term evolve as a function of t. Note that the plot is to be read from right to left (to stay consistent with the $z_0, \ldots, z_T$ notation) when doing sampling since we start from a noise vector z_T and progressively denoise it towards $\hat{z}_0$. More precisely, we perform standard text-guided generation using DDIM sampling and record the cosine similarities between each term and its previous value at each time step $t \in [1, T]$ (in Fig. 5.6, we use $T = 100$ steps). This provides us with a visualization of the "smoothness" of sampling trajectories. Interestingly, these plots show that even at low guidance scales, the classifier-free guidance term is very noisy across the denoising schedule and the phenomenon gets worse as we increase the strength. As highlighted in the seminal publication [39], this term is only a "pseudo-gradient" derived by analogy to classifier guidance but doesn't inherit the



(a) $\gamma = 5.0$ (b) $\gamma = 10.0$ (c) $\gamma = 25.0$ (d) $\gamma = 50.0$

Figure 5.7: Point-wise trajectories within the latent space of *Stable Diffusion* for different guidance scales.



(a) $\gamma = 5.0$ (b) $\gamma = 10.0$ (c) $\gamma = 25.0$ (d) $\gamma = 50.0$

Figure 5.8: Pixel trajectories in RGB space of *Stable Diffusion* for different guidance scales (mapping to the same point in the latent space as shown in Fig. 5.7). The corresponding generated images are shown on top.

smooth properties of "*a gradient-based adversarial attack on an image classifier*". Consequently, pushing its value particularly high as required by the unconstrained nature of the SDS scenario will unavoidably introduce additional instabilities and large steps in pixel-space. This causes not only high saturation (as we discuss below) at the pixel level but also concept forgetting as only the most prominent direction(s) can be selected due to the "noisy" guidance signal.

We elaborate on this last observation by showing that due to the large variations introduced by the irregular classifier-free guidance term at every step, image-space pixels are progressively steered towards the boundary of their domain of definition, namely $[0, 1]$. To do so, we follow standard DDIM sampling (again with $T = 100$) and we record point-wise trajectories in the latent space of *Stable Diffusion* and plot the 4 channels at different guidance scale in Fig. 5.7. First, we can see that increasing guidance makes the trajectories fluctuate over larger ranges of values which meets the result of our previous experiment. More interestingly, we map these trajectories to their corresponding ones in RGB space. As *Stable Diffusion* upsamples one point in its latent space to $8 \times 8 = 64$ pixels, we arbitrarily sample a subset of these pixels and show their trajectories in RGB space in Fig. 5.8. This time, we can see that at high guidance, pixel channel values tend to collapse in degenerate configurations at the boundary of their output domain, namely 0 and 1. Despite resulting from sampling and not SDS, this experiment provides the intuition behind the highly saturated and contrasted results of *Score Distillation*. Furthermore, this phenomenon is very likely to be exacerbated by the non-bijective nature between the space where diffusion operates and the final output image (pixels are clipped to $[-1, 1]$ and remapped to $[0, 1]$ following the sampling process). Note that a theoretical solution to this problem has recently been proposed under the name of *Reflected Diffusion Models* [57].

As *Score Distillation* is an optimization process, we investigated the use of several regularizers to mitigate the saturation effects: a hue regularizer, a saturation regularizer and differentiable histogram matching. Unfortunately, none of them turned out to be effective against the observed color drift.

Chapter 6

Inversion and conditioning

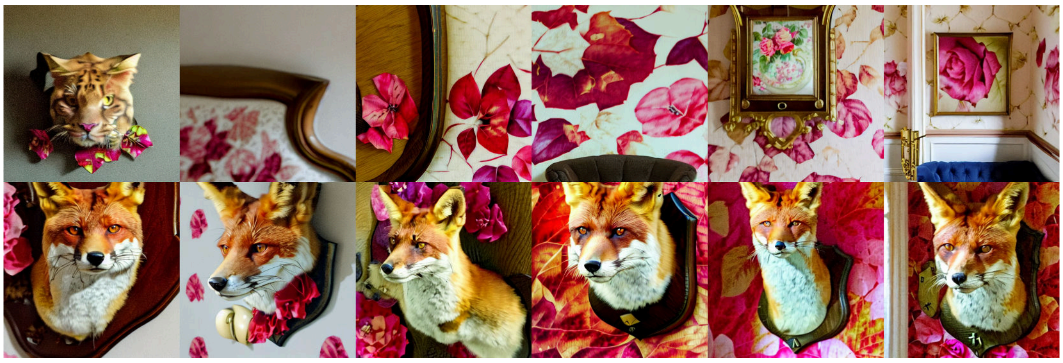
As highlighted in the previous chapter, by the time we completed the experiments described above, *Score Distillation Sampling* appeared too noisy and thus hardly reliable for the task of robust 3D "neural extrapolations" within a NeRF. Inspired by the recent success of multiple techniques to customize and edit images generated with diffusion models, we decided to shift to a more "direct" approach rather than the highly unstable optimization process of SDS.

More concretely, *Instruct-NeRF2NeRF* [33] recently shown that using an instruction-based *image-to-image* diffusion model, namely *InstructPix2Pix* [7], to sample variations of 2D views from a pre-trained NeRF and using these alternative renderings to refine the NeRF, we could produce multi-view consistent edits of 3D volumetric scenes. We thus hypothesized that if we managed to leverage a large-scale diffusion models to sample novel views relatively close to our scene distribution, we could iteratively inject them as new training views in a *progressive refinement* manner. However, the crux of the matter resided in **closing the domain gap between our scene and the priors of such models**. To do so, we investigated three main approaches which we describe thereafter, namely *Textual Inversion*, *DDIM inversion* and *feature-guided conditioning*.

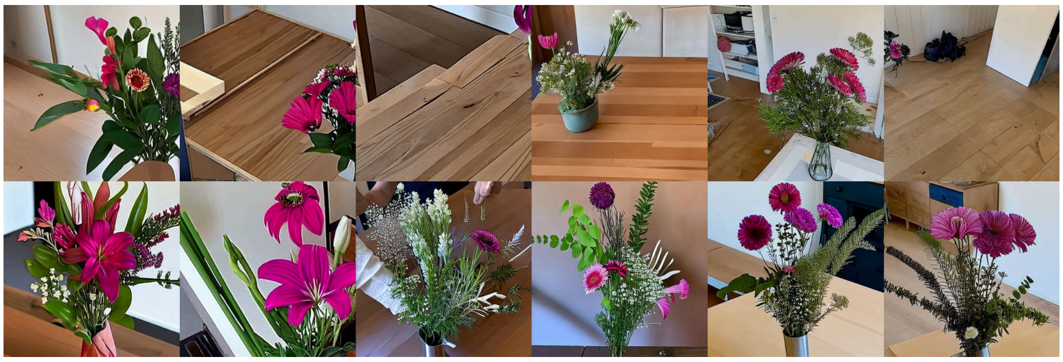
6.1 Textual Inversion

Textual Inversion [28] (TI) aims at customizing text-conditional generation without fine-tuning the diffusion model itself. To do so, it builds on the fact that, for text-conditional models, the conditioning signal c in equation 3.7 is usually computed by tokenizing a text prompt y in a sequence of tokens $t_1, \dots, t_p$ and mapping each token to a corresponding *token embedding* $\text{emb}(t_1), \dots, \text{emb}(t_p) \in \mathbb{R}^{d_t}$ where d_t is the token embedding dimension. These embeddings are then concatenated as $\mathbf{X} = [\text{emb}(t_1), \dots, \text{emb}(t_p)]$ and fed to a text-encoder $\mathcal{T}(\mathbf{X})$ which contextualizes each token embedding into a *text embedding* so that the final conditioning matrix is given by $c = \mathcal{T}(\mathbf{X}) = [e_1, \dots, e_p]$ where $e_1, \dots, e_p \in \mathbb{R}^{d_e}$ are the resulting text embeddings and d_e their dimension. Given a specific object, TI proposes to initialize a new arbitrary token $< S_* >$ (usually using a token embedding semantically close to it) and to learn its weights using equation 3.3 applied to a set of reference images and by sampling (with random text augmentations) sentences of the form "*a picture of a* $< S_* >$. This provides an efficient and lightweight solution but comes with several limitations which we investigate next.

The first challenge is to be able to disentangle an object from its environment. In real scenes, individual objects lie on a support and are surrounded by an environment that may change (more or less) from one viewing angle to another. One option to disentangle objects as proposed concurrently by *Break-a-scene* [4] might be to extract robust enough masks and to learn a token embedding as described above on purely extracted objects. However, for the purpose of our "neural extrapolations" and in light of the inherent contextualization of the objects in the scene that we are trying to refine, we argue that it might be necessary, if not essential, to learn a pseudo-contextualized token embedding of the object. To do so, we propose to extract (through a set of cropping/resizing operations) a subset of images depicting purely the object and an additional set of images of its close surroundings. We then learn two token



(a) *fox* scene

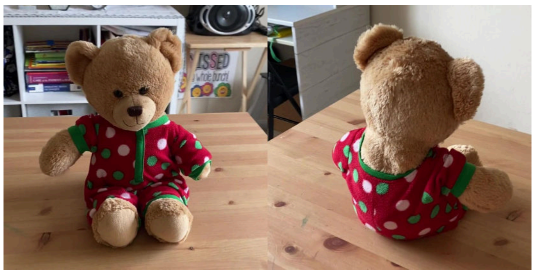


(b) *bouquet* scene

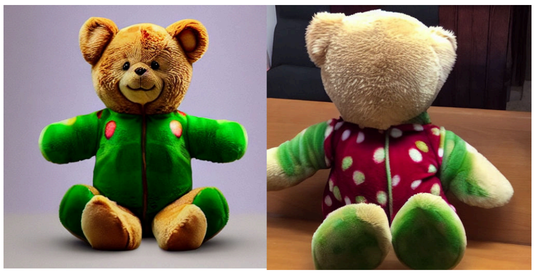
Figure 6.1: Progressive results of our proposed disentangled *Textual Inversion* on the *fox* and *bouquet* scenes. For each scene, the upper row (resp. lower row) was obtained with DDIM sampling and the prompt "*a picture of a [room/scene]* $< S^b_* >$" (resp. "*a picture of* $< S^o_* >$ *in a [room/scene]* $< S^b_* >$"). From left to right, we show the progressive stages of the optimization.

embeddings $< S^o_* >$ and $< S^b_* >$ representing respectively the target object and its direct surroundings. The second token $< S^b_* >$ can be thought of as an absorbing token to disentangle the scene from the relevant concept. We optimize both in a *dropout* fashion by randomly sampling pictures of the object with sentences (with random augmentations) of the form "*a picture of* $< S^o_* >$ *in a [room/scene]* $< S^b_* >$" and cropped out images with prompts of the form "*a picture of a [room/scene]* $< S^b_* >$". The rest follows the methodology and parameters introduced by TI [28], and we build our implementation on *Stable Diffusion v2.1*. Fig 6.1 shows the result of this procedure on the *fox* scene of *Instant-NGP* [69] and the *bouquet* scene of *LERF* [45] where the displayed images are re-sampled from scratch with the optimized tokens. As can be seen between the upper row and the lower row, our method manages to (partially) extract the object from its background while also capturing parts of its surroundings within the absorbing token.

Unfortunately, as the *bouquet* scene highlights, it becomes particularly challenging to extract embeddings when the object is covered in a 360 degree manner as multiple views tend to conflict with the



(a) Training samples from the *teddybear* scene



(b) Results of our disentangled textual inversion

Figure 6.2: Performing textual inversion on a complex 360 distribution of views of a single, yet isolated, object results in intra-entanglement as conflicting views "merge" multiple parts of the learned concept.

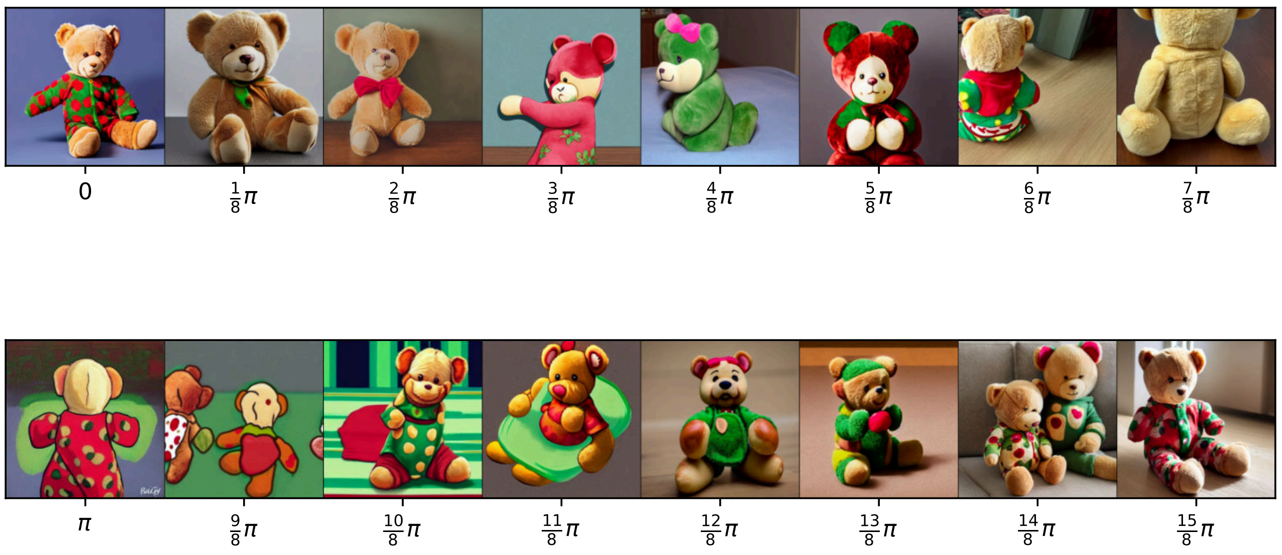


Figure 6.3: Sampled views using our view-dependent *Textual Inversion* introduced in equation 6.1 on the *teddybear* scene with $m = 6$ bins and $k = 3$ interpolated neighbors

concept that we seek to learn. In other words, as we show with an additional example in Fig. 6.2 where we use the *teddybear* scene from the *Co3D* dataset [81], trying to embed the representation of a single object within a single token embedding produces "merging" artifacts. For example, the back of the head is shown in a front-facing manner or four legs are generated. To address this, we draw inspiration from *Multiresolution Textual Inversion* [19] and learn view-dependent tokens. The latter proposes to learn tokens as a function of a subset of the noise schedule. In other words, instead of sampling $t \in [1, T]$ in equation 3.3, they optimize a specialized token $< S_*(t_{\text{fixed}}) >$ by sampling $t \in [1, t_{\text{fixed}}]$. Note that we give a very simplified description of the more exhaustive parameterizations they introduce. We push this approach further and propose to optimize a text embedding conditioned on the azimuthal angle around an object for a 360-degree scene. To make this approach tractable, we propose to learn m token embeddings $< S_*(\theta_1) >, \ldots, < S_*(\theta_m) >$ for evenly distributed angles $\theta_1, \ldots, \theta_m$. To learn a continuous embedding $< S_*(\theta) >$ at any angle θ, we propose to interpolate these binned embeddings using *knn* nearest neighbors and weight their embeddings by a renormalized inverse angular distance weighting. In other words,

$$\text{emb}(< S_*(\theta) >) = \sum_{i \in \text{knn}(\theta)} a_i \cdot \text{emb}(< S_*(\theta_i) >) \text{ where } a_i = \frac{1/|\theta_i - \theta|}{\sum_{j \in \text{knn}(\theta)} 1/|\theta_j - \theta|} \tag{6.1}$$

We experimented with many values for m and the number of nearest neighbors and settled on $m = 6$ and $knn = 3$ for the results shown in Fig. 6.3. Note that similarly to NeTI [2], we could have also optimized a small hypernetwork conditioned on the azimuthal angle. This simple experiment provides us with multiple insights. First, despite sampling (and binning) uniformly images around the object, not all angles are equally reconstructed. More specifically, only the front-facing points of views seem to fully recover the actual appearance of the teddy bear. Additionally, side-facing samples hardly match the angles they should be representing. Both observations hint at the potential bias in the representational landscape of the diffusion model. Put differently, *Stable Diffusion* was very likely exposed to many front-facing pictures of teddy bears at training time and thus, there exists a large (in terms of volume) and relatively well-behaved weight landscape that gradients can navigate through when performing TI. On the other hand, the rest of the angles are far from the actual data distribution modeled by the network, and we thus navigate through multiple adjacent low-capacity and noisy modes, which would explain the cartoon style around π. Note that, with this in mind, it would be interesting to investigate the properties of the optimized latents because recent works [4, 2] have shown that TI tends to overfit and produce non-compliant CLIP token embeddings and have thus suggested either explicit or implicit regularization (e.g,

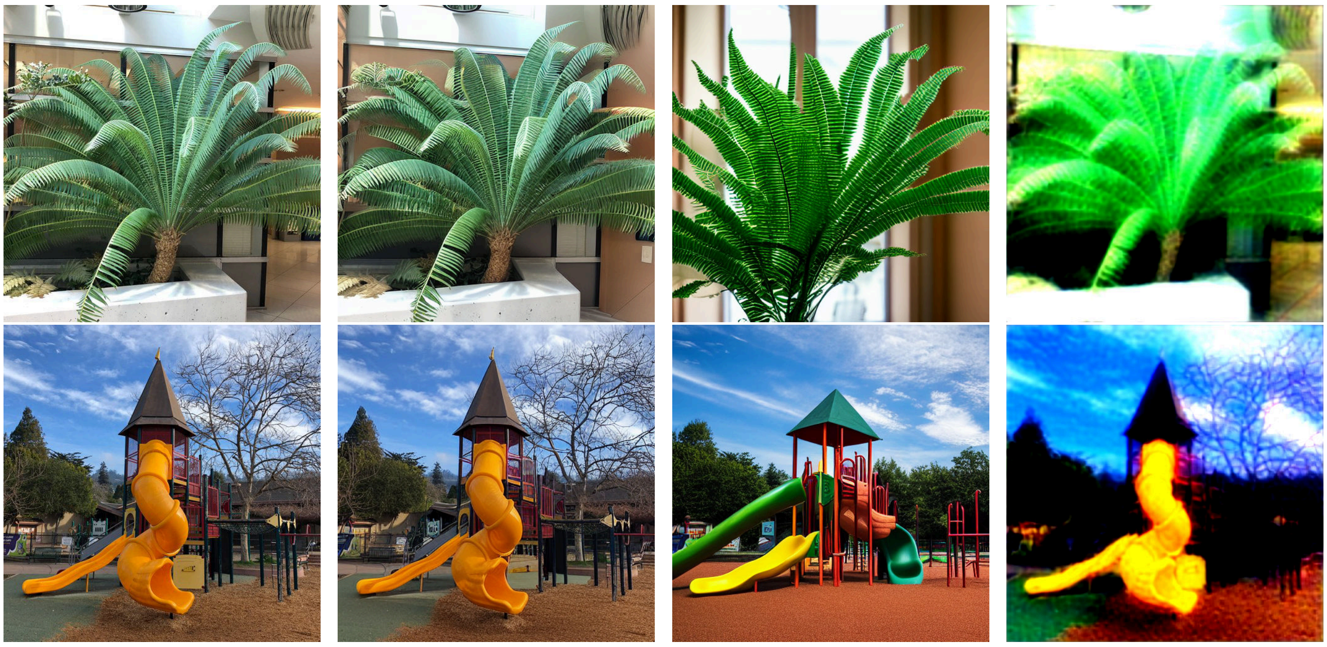

(a) Original image (b) Low-low inversion (c) Low-high inversion (d) High-high inversion

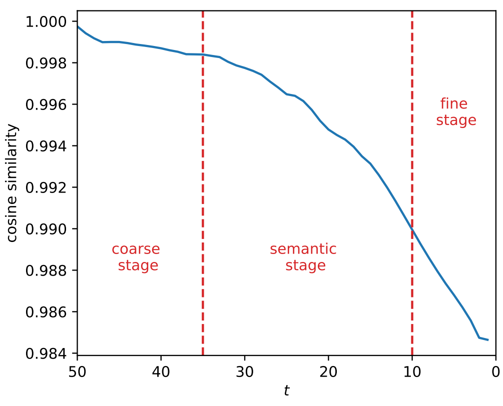
Figure 6.4: We investigate the effects of guidance scale during both inversion and reconstitution for an image of the *fern* scene (resp. *playground* scene) with the prompt "*a DSLR picture of a big fern in a hall*" (resp. "*a DSLR photo of a playground, outdoor, sunny*"). *Low-high* means that low guidance ($\gamma = 1.0$) was applied during the forward inversion process and high guidance ($\gamma = 7.5$) during the reverse reconstruction process.

renormalization, contextualized residual text embeddings and cross-attention map regularizations).

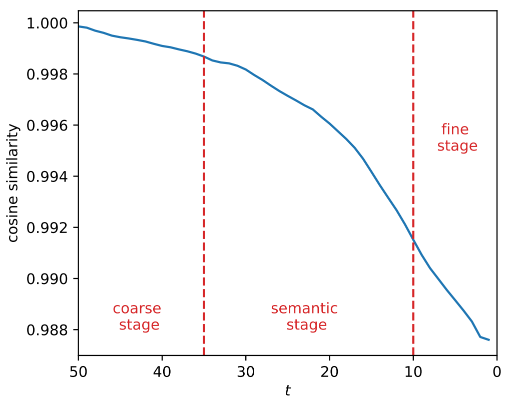
6.2 DDIM inversion & null-text optimization

As we presented above, *Textual Inversion* comes with significant challenges that are inherent to its conditioning capacity: the text embeddings are only fed through cross-attention in a coarse-grained manner and have to be robust to multiple appearance augmentations. As such they can only capture concepts in a rather global manner. We thus propose, in this paragraph, to explore another direction, namely DDIM inversion, to better align the reconstruction thanks to the partial estimate of the scene that we might already have from an out-of-distribution viewpoint. DDIM inversion [94] builds on the ODE formulation of equation 3.4 and the observation that, up to the (non-negligible) accumulated error over the sampling trajectories, DDIM defines a one-to-one mapping between the latent $z_T \sim \mathcal{N}(0, I)$ and the original image $z_0 \sim p(z)$. In other words, from an image z_0, we can reverse equation 3.4 and recover the corresponding latent. In the original publication, Song et al. [94] show that spherical interpolations of different z_T latents give consistent generated images. Unfortunately, as highlighted by follow-up works [73, 130], both DDIM inversion and its corresponding interpolation does not perform well in the context of large-scale diffusion models for which modes of the data appear to be more entangled.

One direct consequence of the previous observation is that, at high-guidance, DDIM inversion performs badly as it steers the trajectories away from actual modes of the data and introduces a significant amount of error following the large ill-conditioned steps of classifier-free guidance. As shown in Fig. 6.4d, using high-guidance for inversion (and reconstruction) results in poor reconstruction of the inverted image. On the other hand, low-guidance inversion yields a moderately faithful reconstruction (Fig. 6.4b) but poor editability. Using high-guidance during the reverse process boosts edits but at the cost of reconstruction faithfulness. Inspired by the pivotal tuning approach introduced for GANs [86], *Null-text inversion* [67] proposes to leverage the stability and reconstruction power of low-guidance inversion by first inverting the source image z_0 at low-guidance yielding a pivot trajectory $z_0 = z_0^*, z_1^*, \ldots, z_T^*$ and then progressively optimizing unconditional embeddings $(\emptyset_t)_{t\in[1,T]}$ for each time step in equation 3.7 in a contiguous way starting from $\emptyset_T$ all the way down to $\emptyset_1$. In Fig. 6.5, we show how according to this scheme, the optimized

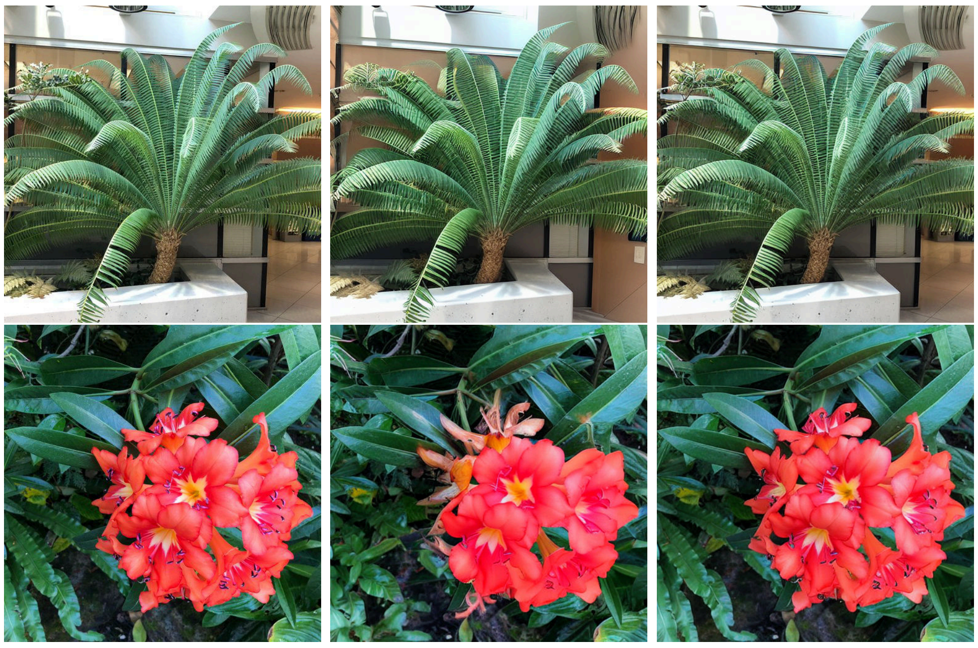


(a) *fern* scene



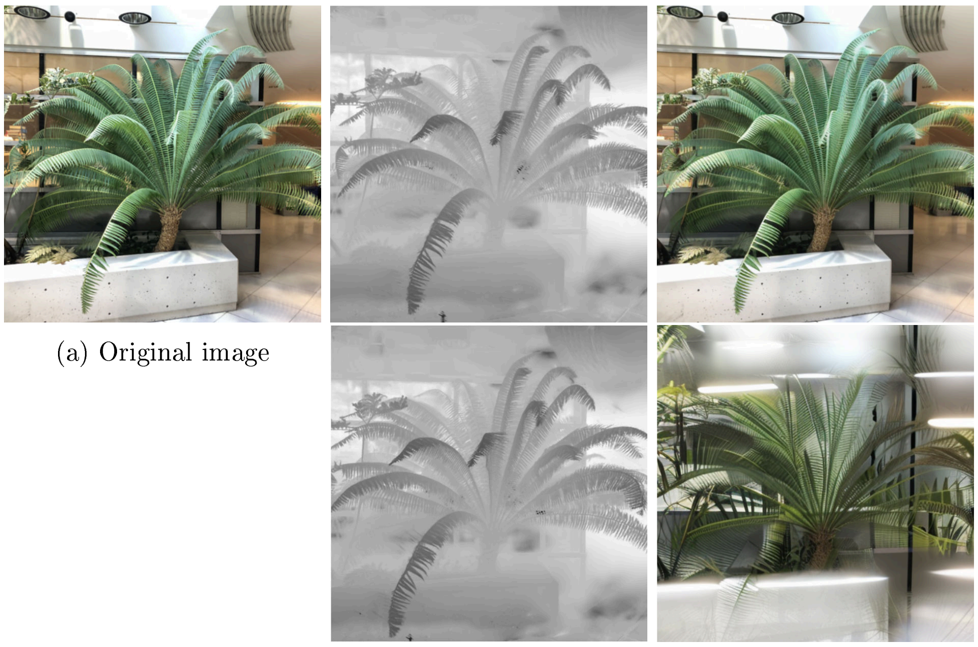
(b) *playground* scene

Figure 6.5: We show how the optimized null-text embeddings progressively deviate from the vanilla ones on two images extracted from the *fern* and *playground* scenes of the LLFF dataset. Note how the similarity starts to deviate once we leave the coarse stage where only the layout of the image is being formed.



(a) Original image (b) Low guidance inversion (c) Inversion with null-text optimization

Figure 6.6: We investigate the benefits of null-text optimization compared to vanilla low-guidance inversion (b). (c) shows that null-text optimization allows to faithfully recover more details of the input image. Note that we used similar prompts as Fig. 6.4.



(a) Original image

(b) Conditioning depth

(c) Reconstructions from the inversion with optimized null-text embeddings

Figure 6.7: In an attempt to leverage null-text optimization across novel viewpoints, we optimize a sequence of null-text embeddings on an initial image (a) conditioned on depth using *ControlNet* (top of (b)). This yields a particularly good reconstruction for the original image (top of (c)) but fails to produce a satisfactory reconstruction (bottom of (c)) when conditioned on a novel depth map (bottom of (b)).

null-text embeddings progressively deviate from the original null-text embedding $\emptyset$ as a function of t. Note especially how the similarity drops around 15 steps. This hints at the fact that differentiating perceptual details (which we wish to recover) start being injected at this stage during sampling. This corroborates recent findings on the different stages of the image formation process which we try to plot on Fig. 6.5: *coarse/layout*, *content/semantic* and *fine/detail* stages. All things considered, optimizing the unconditional embedding allows to overfit the corresponding appearance thus resulting in a more faithful reconstruction as shown in Fig. 6.6.

In practice, the optimized $(\emptyset_t)_{t\in[1,T]}$ are used to steer conditional editing of real images (using another editing prompt and techniques such as *Prompt-to-Prompt* [36]) while preserving their original appearance. However, this is not what we want to achieve in this project. We initially hypothesized that these unconditional embeddings when used with the depth conditioned variant of *Stable Diffusion* would be robust to the task of synthesizing new views by conditioning on depth estimates from close, yet different, viewpoints. Unfortunately, as shown in Fig. 6.7, this resulted in poorly satisfying and diverging results. We suggest that this is due to the strong connections between the optimized null-text embeddings and their seeding latent. To justify this, we propose to jointly optimize a set of embeddings $(\emptyset_t)_{t\in[1,T]}$ for a batch of 5 images from close viewpoints. Fig. 6.10 shows that this does not converge which we hypothesize is due to two complementary points: 1. inversion trajectories (and thus the resulting latents) are highly decorrelated from one viewpoint to another (which makes sense considering the restricted domains of invariances of CNNs), 2. optimizing null-text embeddings is yet another overfitting approach where degrees of freedom are simply provided though the text-conditioning channels of diffusion models. To justify the first point, we proceed to a simple additional experiment. In Fig. 6.8, we show how various corruptions in an image translate in deviations in the inversion trajectories. If image-space corruptions behave quite well (sometimes even improving the similarity when inverting!), geometric ones result in a significant decorrelation and, as highlighted in Fig. 6.9, this degradation drops very quickly. Note that for the latter, we also experimented with equivariant transformations in the latent space but observed no significant improvement. Regarding the second point, we invite the reader to look at line (6) of

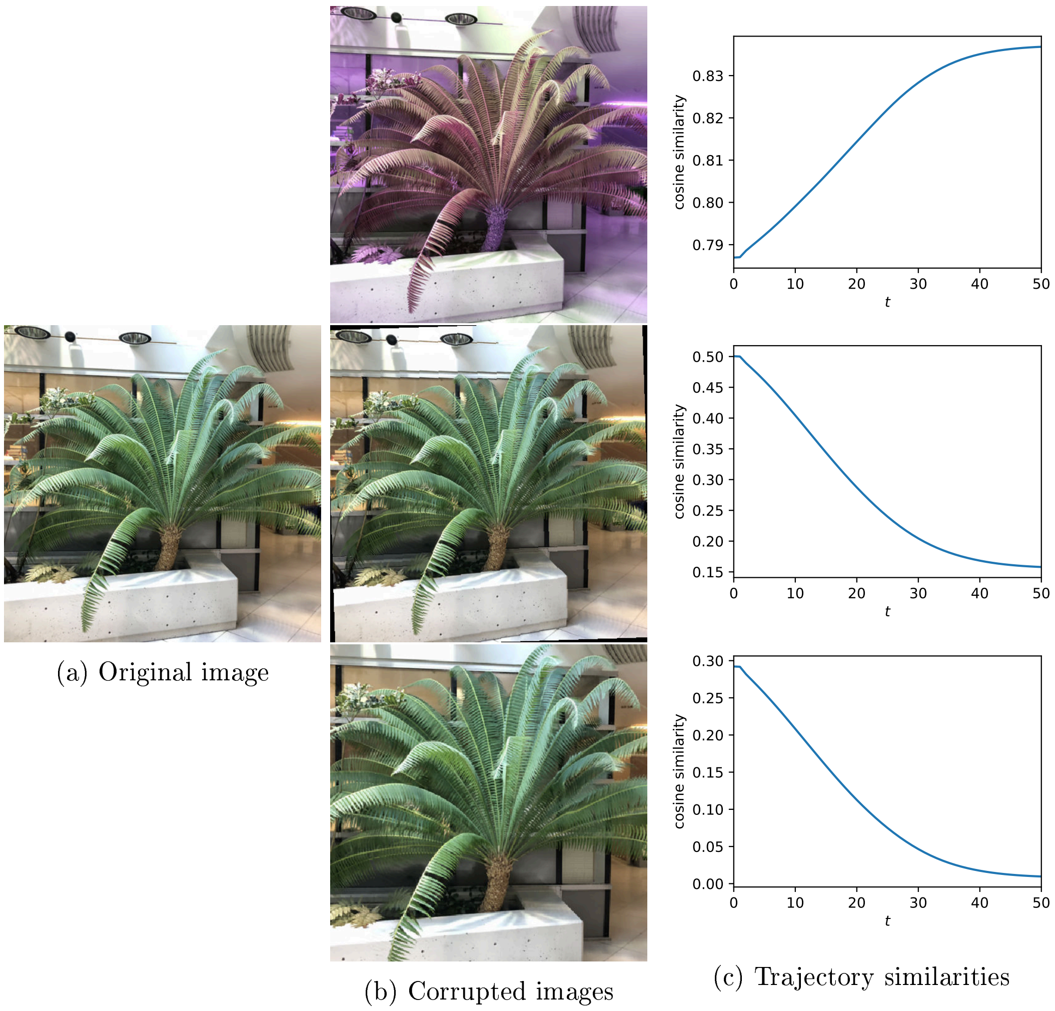


Figure 6.8: Starting from the initial image (a), we proceed to various corruptions/augmentations (from top to bottom: *colorjittering*, *rotations*, *scaling*) (b) and plot the cosine similarity between the original noise estimate and the corrupted ones during the inversion of both the original and corrupted images (c)

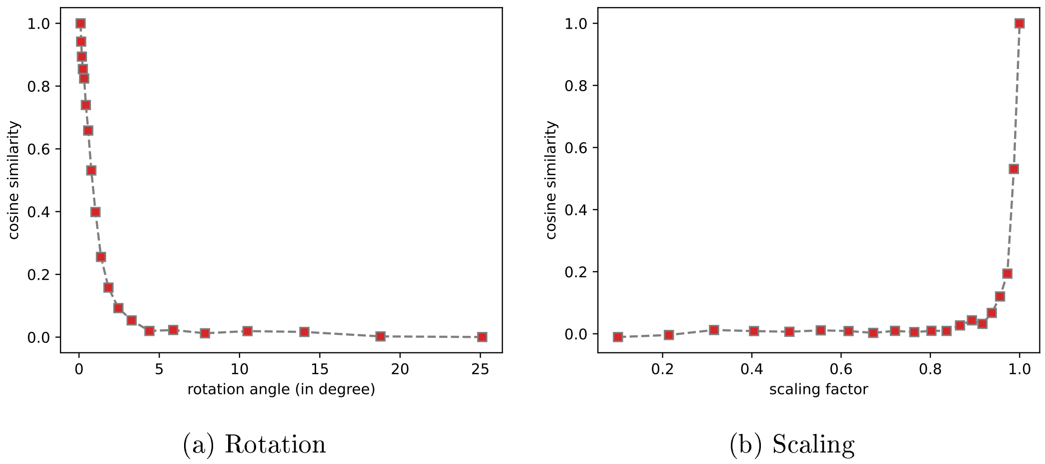


Figure 6.9: Using the same methodology as Fig. 6.8, we report the cosine similarity betwen the final inverted latent z_T in the original image and the corrupted one as a function of the scale of corruptions (i.e., either angle or scaling factor). Consistently with the domain of invariances of CNN, this breaks completely and very rapidly and suggests why null-text optimization is not robust to batched optimization and varying conditioning from different viewpoints.

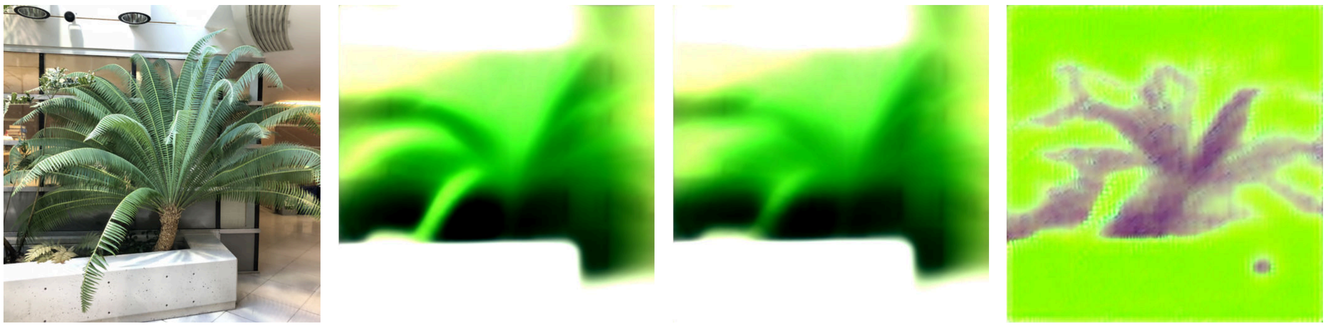

(a) Reference sample image from the training batch (b) Reconstruction 1 (c) Reconstruction 2 (d) Reconstruction 3

Figure 6.10: Starting from a batch of nearby viewpoints among which (a) is an example, we jointly optimize null-text embeddings on the whole batch. (b-c) show some of the completely corrupted resulting inversions.

algorithm 2 in Appendix C and the gradient update to obtain $\emptyset_t$ from $\emptyset_{t-1}$. We can easily picture that this gradient will fluctuate across different trajectories (especially further down in the schedule, i.e. close to $\emptyset_1$, as shown in Fig. 6.5) and thus will produce poorly exploitable embeddings.

6.3 Feature-guided conditioning

Adapter-based conditioning. Using adapters to condition generation with other signals works particularly well, especially with *ControlNet* [129]. Unfortunately, as these adapters are only regressed to generate images given (out-of-context) depth maps, multi-view consistency is far from satisfied. In other words, a slight change in camera viewpoint or a different starting latent $z_T \sim \mathcal{N}(0, I)$ lead to a completely different generated image. Fig. 6.11 shows some examples of renderings generated using *ControlNet* on *Stable Diffusion* with depth maps obtained from a pre-trained NeRF and with a specialized token learned by disentangled TI as described in paragraph 6.1. These results suggest that we need to inject semantic and conceptual priors both at finer granularities and in a more geometrically-consistent way. One way of proceeding could be to train diffusion models that are explicitly conditioned on pose [11] or that use more complex geometric priors such as "epipolar attentions" [105]. However, as we stated in our introduction, this requires to have correspondingly labelled data and deviates from the constraints of the project as it implies training a new specialized model.

We thus explore a more "accessible" and "hacky" way of conditioning generation during sampling and propose to focus directly on the internal representations of diffusion models, namely feature maps. Following the recent trends in diffusion-based editing, we start with coarser features, namely cross-attention map, and progressively move towards more expressive, yet sensitive, feature maps (self-attention maps and raw residual feature maps). As the landscape of feature-map extraction, manipulation and injection is (already) relatively complex, we propose in appendix B a survey of the existing approaches by the time this report was submitted.

Cross-attention maps. *Prompt-to-Prompt* [36] recently showed that, during text-conditioned image generation, cross-attention maps are spatially correlated to the token they correspond to. We show some examples in Fig 6.12 where we can see that the cross-attention maps attend (roughly) to the regions of the image where the corresponding tokens will eventually be generated. Hertz et al. [36] thus propose to build on this observation to edit a T2I-generated image by either manipulating the prompt and/or the cross-attention maps while preserving the original composition and structure. Motivated by this work, we decided to experiment with this signal to sample new views from a partial NeRF scene. Yet a major challenge abode: how can we lift this 2D prior in 3D? Inspired by our experiments with feature fields, we proposed to learn multiple feature fields corresponding to the cross-attention maps.

To do so, we start by providing a prompt describing the scene (e.g. "*a picture of a bouquet on a table in a living room*" or with textual inversion, "a picture of a $< S_* >$ on a table in a living room"), we

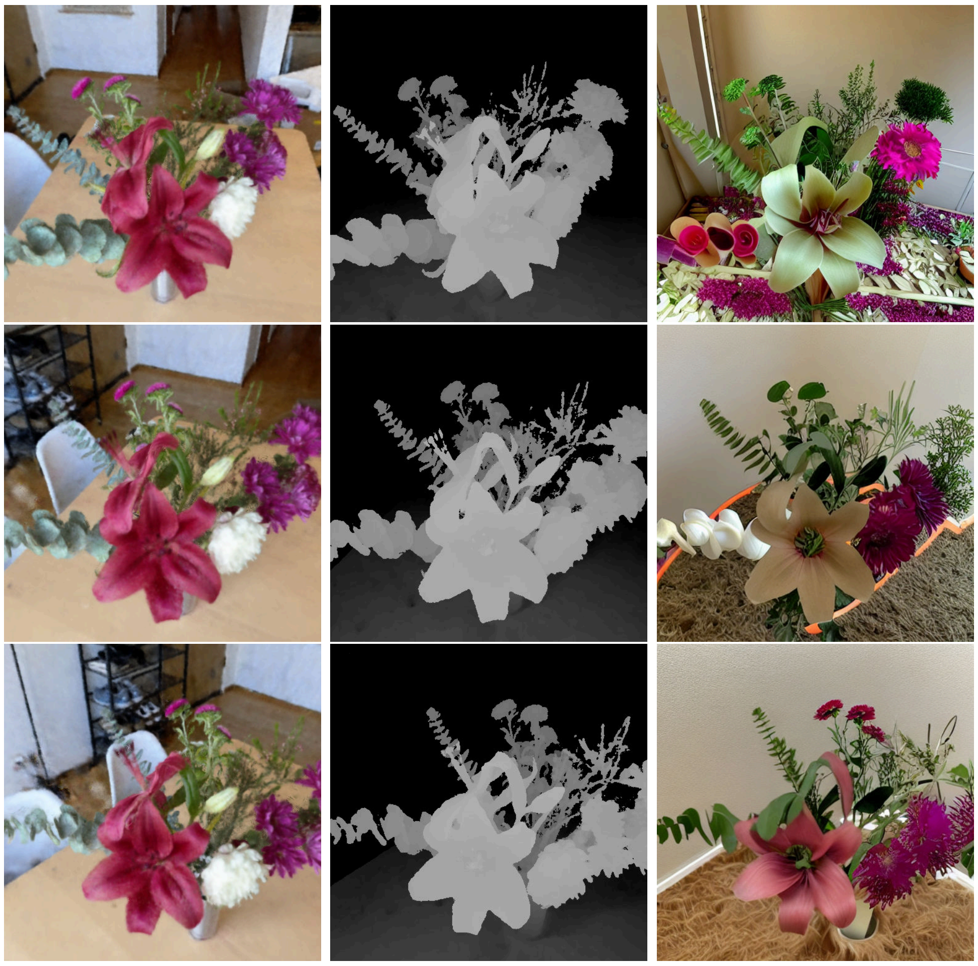
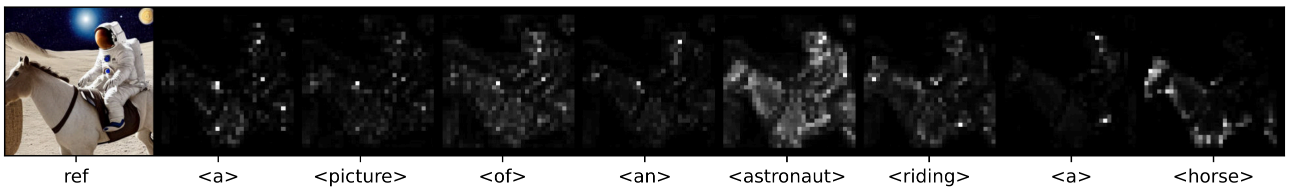
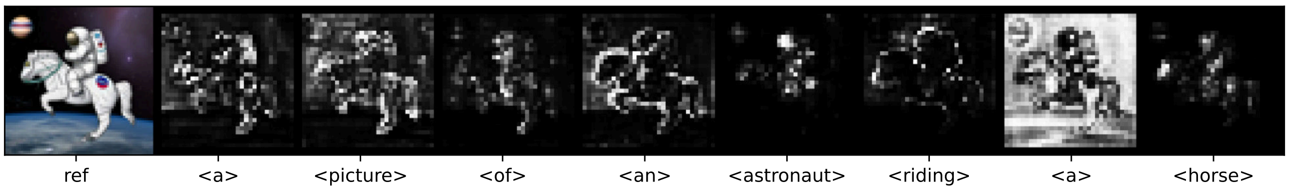

(a) Reference RGB (b) Conditioning depth (c) Generated image

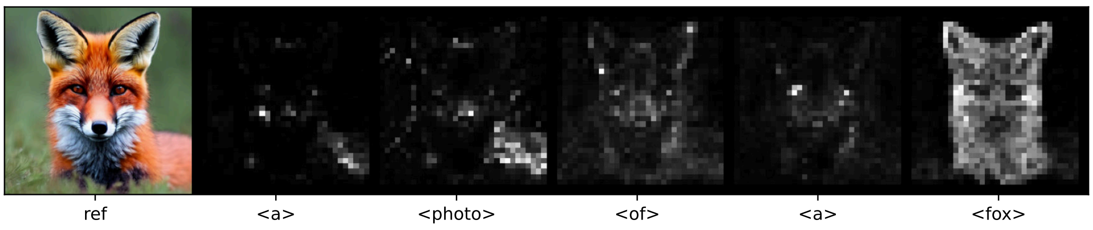
Figure 6.11: Conditioning the sampling process from depth maps (b) using *ControlNet*, even with a properly optimized token, produces results that are far from faithful and hardly multi-view consistent.



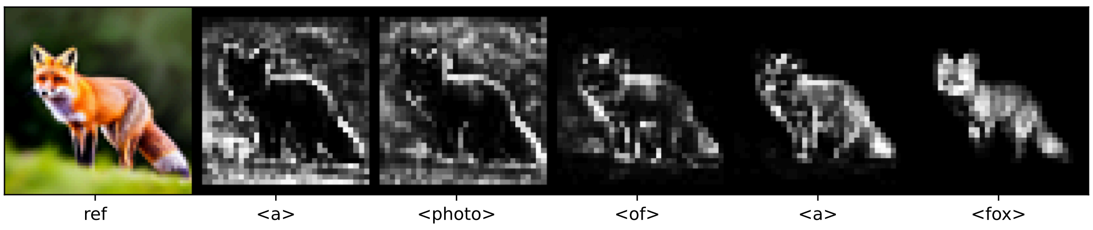
(a)



(b)

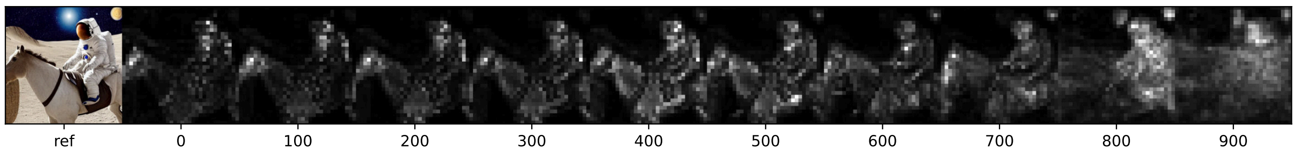


(c)

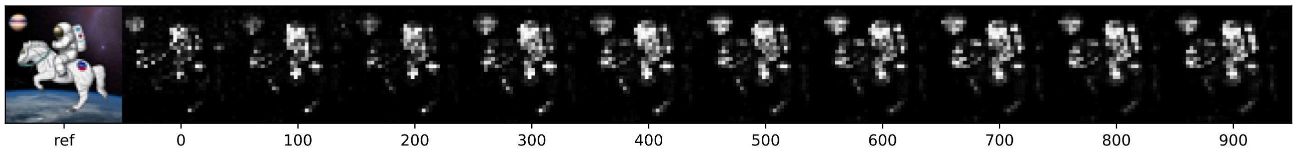


(d)

Figure 6.12: Examples of cross-attention maps extracted during sampling on both *Stable Diffusion* (a, c) and *DeepFloyd IF* (b, d). In all examples, the cross-attention maps were taken from the upsampling part of the U-Net and averaged over all heads of the corresponding layer. For *DeepFloyd IF*, we show only maps from the first stage, and we used the soft-thresholding operation of *self-guidance diffusion* [24] (described in Appendix B) in order to obtain cleaner maps.



(a)



(b)

Figure 6.13: We apply the same methodology as Fig. 6.12 but instead of focusing on a single timestep t, we show the evolution of one of the cross-attention maps of the "<astronaut>" token for both *Stable Diffusion* (a) and *DeepFloyd IF* (b)

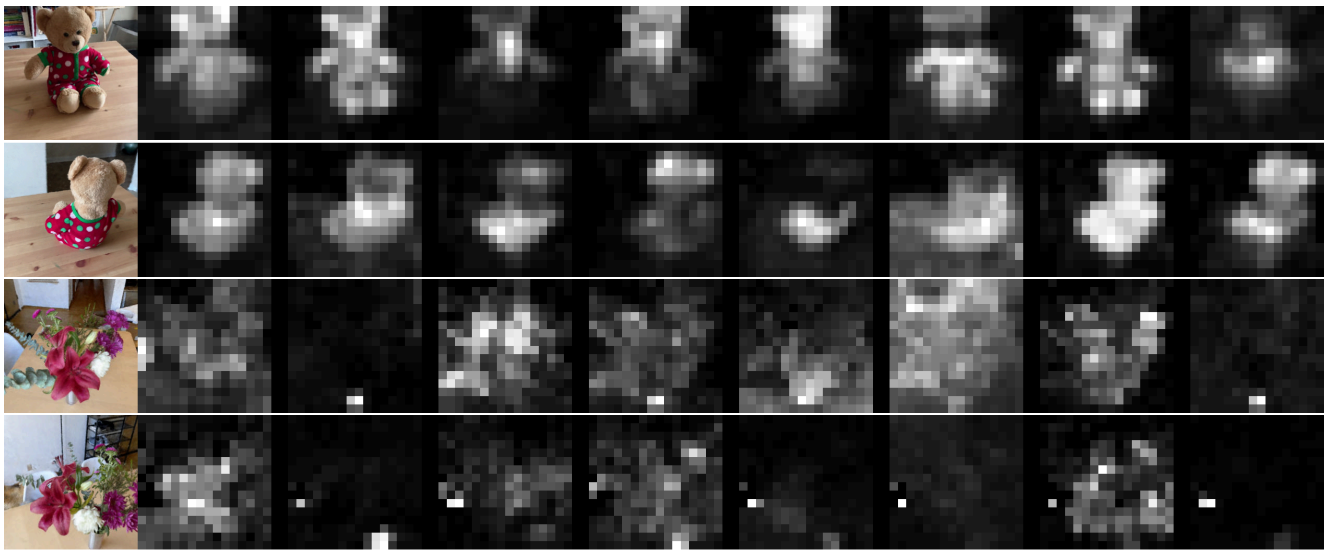
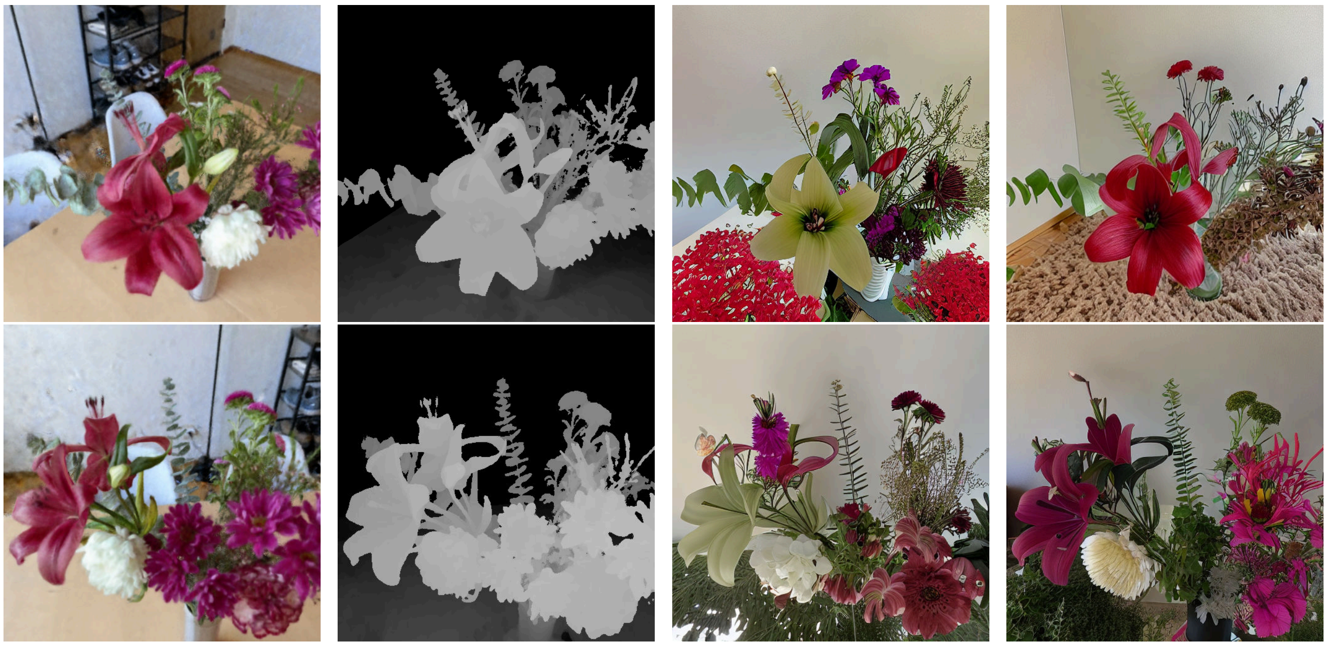


Figure 6.14: We train multiple feature fields for the cross-attention maps (across multiple heads and layers) corresponding to object tokens on the *bouquet* and *teddy*. As can be seen from the NeRF renderings above, our distilled cross-attention maps are 3D consistent with the underlying geometry and layout of the target objects. In other words, we observe that contrary to the perceptual latent features described in chapter 4, cross-attention maps are more amenable to the distillation process.



(a) NeRF RGB estimate (b) NeRF depth estimate (c) Reconstruction without the cross-attention maps (d) Reconstruction with the cross-attention maps

Figure 6.15: We use our NeRF distilled cross-attention maps to condition sampling from *Stable Diffusion*. Solely conditioning on depth (b) using *ControlNet* and an optimized token yields a rather out-of-distribution appearance (c). On the other hand, injecting the cross-attention maps (d) enables a more faithful reconstruction. Note how the colors of the flowers and the per-flower specificities are partially recovered.

use DDIM inversion (at low-guidance!) and extract the cross-attention maps for the relevant tokens (e.g. $<$ *bouquet* $>$, $< S_* >$, $<$ *table* $>$) using the *forward inversion-based* scheme described in Appendix B. Building on the existing implementation of *LERF* [45] in the general-purpose framework *Nerfstudio* [99], we learn multiple feature fields associated to each head of multiple cross-attention layers. Note that it would have been too expensive to train one feature field branch for every token, for every head, for every cross-attention layer and at each time-step $t \in [1, T]$ of DDIM sampling (usually 50 steps). Fortunately, we observe in Fig. 6.13 that cross-attention maps are consistent across multiple time-steps (except late in the schedule where they are too noisy). We thus, select a subset of the noise schedule, namely $[0, 700]$ and average temporally the corresponding maps. The resulting maps are then distilled in 3D using volume rendering guided by the fixed density of the RGB pre-trained NeRF. As attention maps are relatively low-resolution, we add an additional gaussian smoothing operation when rendering them to avoid aliasing. Surprisingly, this overall approach turned out be successful. As highlighted in Fig. 6.14 and contrary to the perceptually-compressed latent space of *Stable Diffusion*, this signal is robust enough so that we can learn a 3D distilled version of it. Beyond the scope of this project, this result is to be put into perspective with recent attempts to learn semantic correspondences [127, 101, 60, 35] or segmentation maps [123] by using internal representations of diffusion models.

With these 3D attention maps at our disposal, we use them along with our disentangled token optimized as described in section 6.1. More precisely, we recover a partial depth estimate from our pre-trained NeRF, render the cross-attention maps across all heads for the relevant channels and inject the latter while performing DDIM sampling with depth-conditioning with *ControlNet*. As shown on Fig 6.15, using the cross-attention maps helps in disentangling the concept from the rest of the scene and with the help of the specialized token seems to enable the synthesis of consistent flowers (which *ControlNet* cannot by default as it only hallucinates the depth), thus allowing a more faithful reconstruction. Note that, as suggested in previous works such as *Asyrp* [50], we found it beneficial to inject noise during DDIM sampling as illustrated in Fig. 6.16. Still, as nothing is done in that direction, we do not recover the background correctly. Additionally, this process remains very stochastic and simply choosing another starting latent $z_T \sim \mathcal{N}(0, I)$ produces completely different results for the same conditioning.

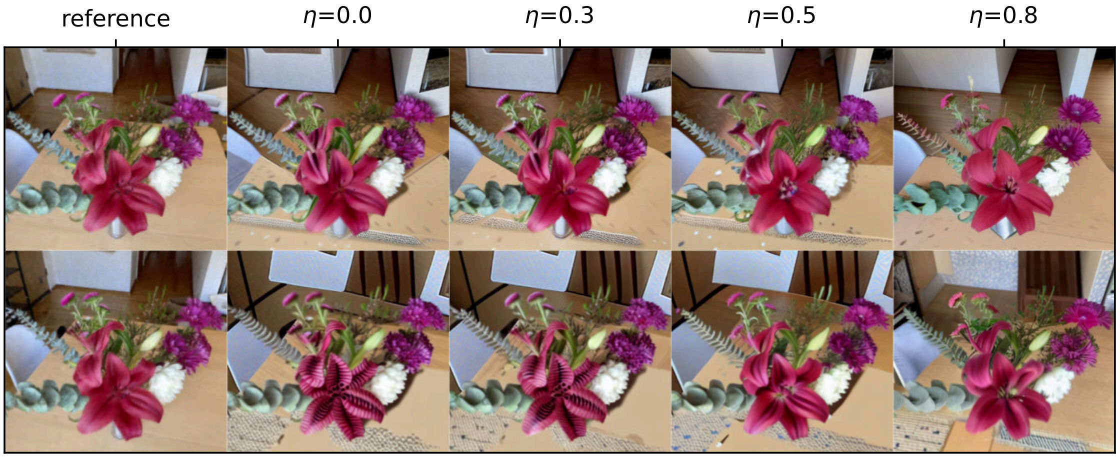


Figure 6.16: Adding some amount of noise $\eta \in [0, 1]$ during DDIM sampling produces smoother and more realistic results when injecting our distilled cross-attention maps. We hypothesize that this helps relax the error that is progressively accumulated during DDIM sampling due to the misalignment between the injected cross-attention maps and the singular novel trajectory we synthesize.

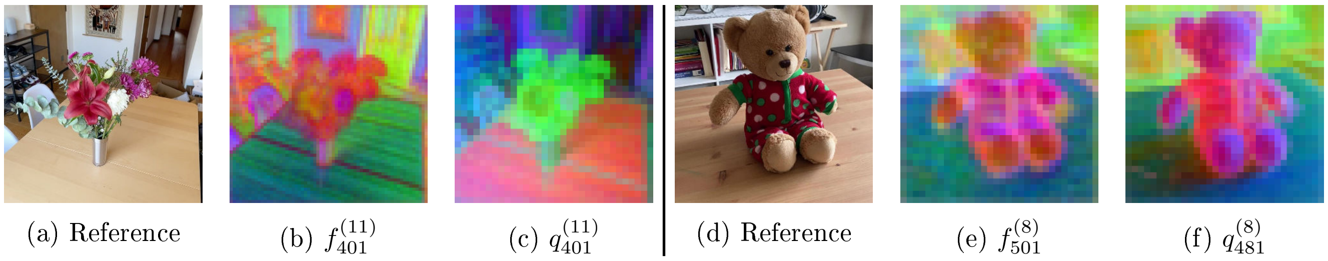
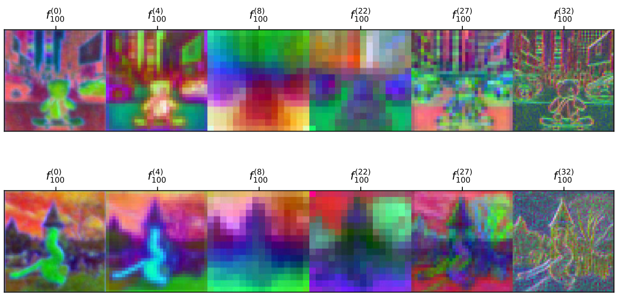


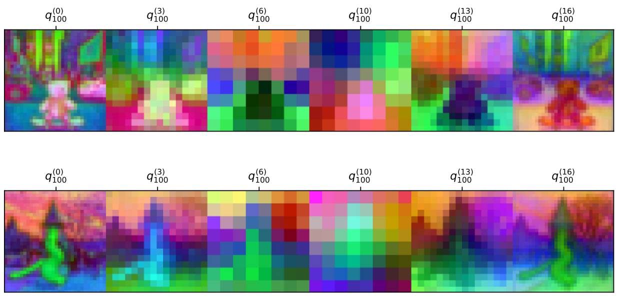
Figure 6.17: We investigate the internal representations of several layers of *Stable Diffusion* by performing a PCA on their corresponding feature maps. Note how the directions are clearly related to objects or parts. (b, c) (resp. (e, f)) were obtained by performing DDIM inversion (i.e., *reverse inversion-based* as defined in Appendix B) on an extract (a) (resp. (d)) of the *bouquet* (resp. *teddy*) scene.

Self-attention maps and raw feature maps. The two partial conclusions above highlight the too weak and concept-level conditioning that cross-attention provides us with. Differently put, it can hardly do better than *Textual Inversion* as it is inherently functionally bounded by "which token to inject at which location in the generated image". This claim is to be slightly qualified though, especially considering that multiple cross-attention maps are injected per-head and thus allow sub-concepts to be expressed. This might explain why we are able to recover flowers with consistent colors and appearance in Fig. 6.15. Yet, this invites us to consider a stronger conditioning signal. *Prompt-to-Prompt* [36] already hinted at the use of self-attention maps to recover the appearance in addition to the structure but found it complicated to solve side effects such as the "appearance leakage" between the edited object and its background when manipulating the former. Further works have mitigated this effect by carefully choosing which layers to inject [106, 24] and masking the injected self-attention using mask derived from thresholded cross-attention [8].

Attracted by the surprisingly convincing results of these efforts, we explore their approaches. Unfortunately, in our specific novel view synthesis case, we face additional challenges. First, these features are very expressive and thus highly view-correlated which precludes distilling them within a NeRF. To justify this argument, we turn to the structure of the residual blocks of *Stable Diffusion* and the more recent *DeepFloyd IF*. For the former each block is made of a residual layer followed by a self-attention layer and a cross-attention layer, whereas for the latter the self-attention and cross-attention are fused. Note that for *DeepFloyd IF*, this description is the result of our exploration of the provided codebase as its descriptive paper has not been released yet. From this, it seems unclear why injecting raw activated feature maps $f^{(l)}_t$ should even work as they correspond to residual features in the ResNet blocks of the

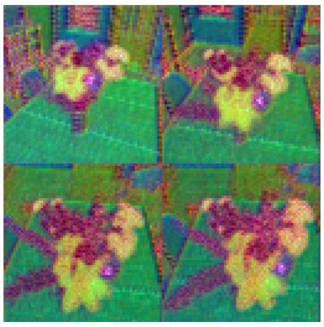


(a) Residual $f_t^{(l)}$ feature maps

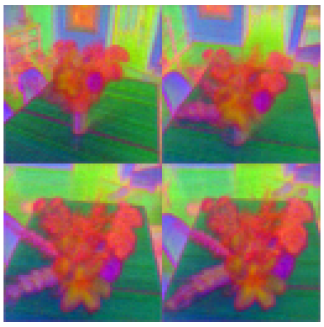


(b) Self-attention $q_t^{(l)}$ *query* maps

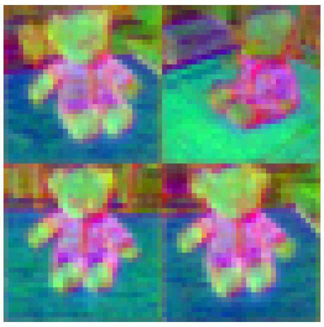
Figure 6.18: Analyzing the feature maps within different blocks of the U-Net of *DeepFloyd IF* exposes the different types of representations that the network builds. Note that similarly to cross-attention maps, we observe a good temporal consistency of these feature maps. The image in the first row was generated from the prompt "*a teddy bear on a skateboard in times square*" while the second row was obtained by performing the *noise-based* feature extraction technique described in Appendix B.



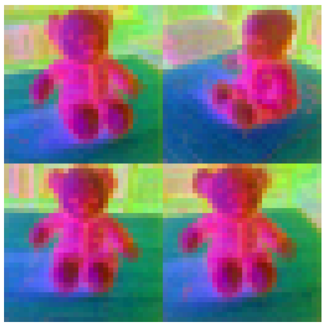
(a) $f_{401}^{(11)}$ on a subset of *bouquet*



(b) $q_{401}^{(11)}$ on a subset of *bouquet*



(c) $f_{401}^{(8)}$ on a subset of *teddy*



(d) $q_{401}^{(8)}$ on a subset of *teddy*

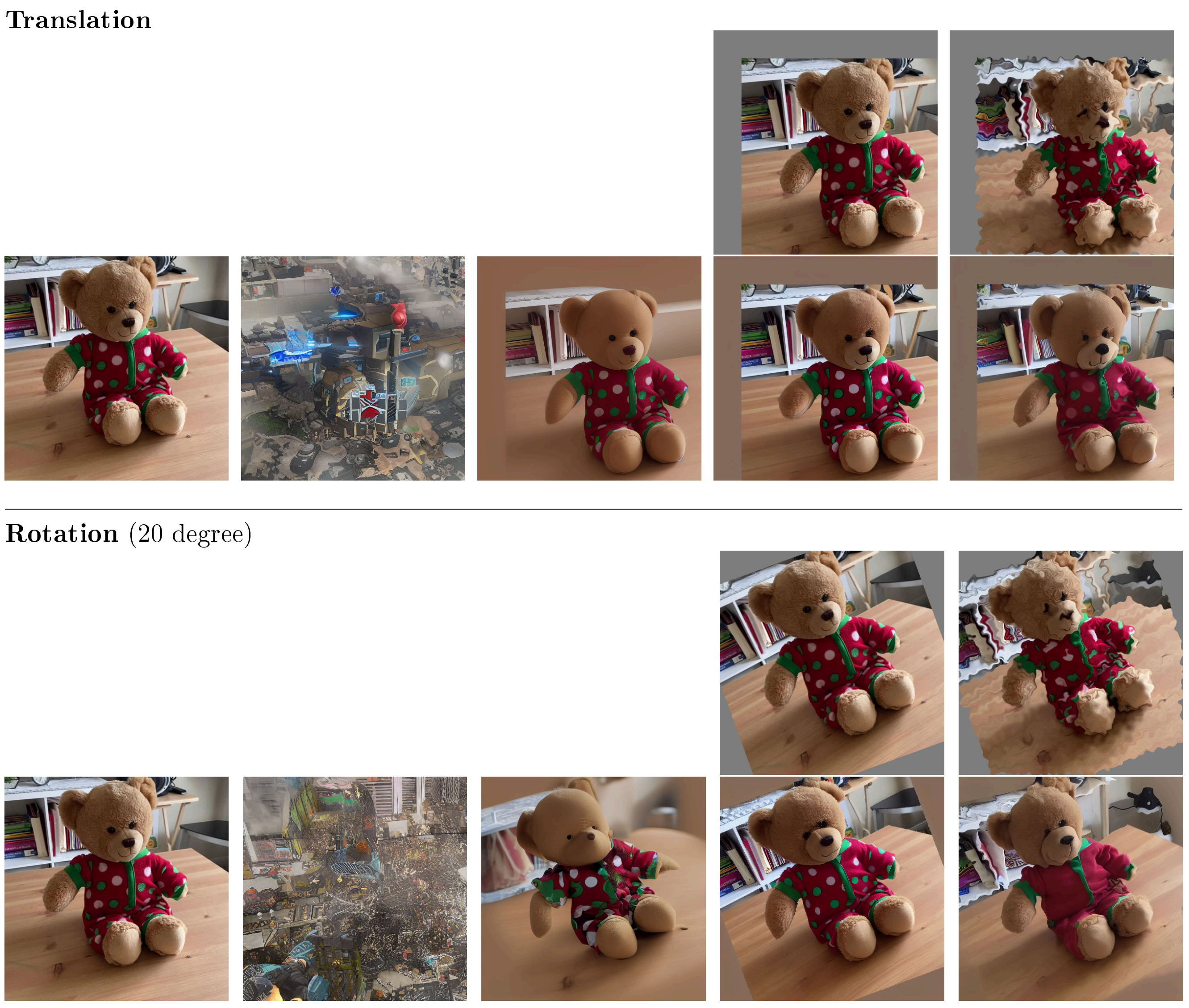
Figure 6.19: To evaluate the genericity of internal feature maps, we propose to run a PCA *jointly* on several images from multiple viewpoints. We observe that across all layers, this yields (perceptually) consistent "segments".

U-Net. One explanation might be that they are injected early during DDIM sampling starting from the same seed latent $z_T \sim \mathcal{N}(0, I)$ sampled "at random" when performing T2I generation or resulting from the inversion process when using real images. As a consequence, it builds on the assumption that DDIM sampling trajectories can be correlated early during the integration. However, as we observed when prototyping and as we shall expose further down, constraining the trajectories too long or too strongly makes the sampling process diverge and collapse in inconsistent modes. This is to connect with the theory behind DDIM inversion: it is an approximation and each integration step introduces an amount of error which is propagated down the whole trajectory. We start by doing PCA on activated feature maps $f_t^{(l)}$ and projected key $k_t^{(l)}$ and query $q_t^{(l)}$ self-attention feature maps across different layers of the U-Net of both *Stable Diffusion* and *DeepFloyd IF*. Fig. 6.17 shows different feature maps (residual and self-attention) extracted from *Stable Diffusion* at different layers and timesteps. Furthermore, Fig. 6.18 suggests that different layers carry different types of information, namely deeper layers seem to capture semantic concepts while extreme layers (both the encoder and decoder of the U-Net) capture finer appearance details which justify their recent use to regularize appearances [24, 106]. However, as we wish to "resample" these features from other viewpoints, we must ensure that, to a certain degree, they are consistent across multiple views. Fig. 6.19 shows that performing a joint PCA on features from 4 different viewpoints yields the same principal directions as a single PCA thus suggesting that they are robust in a multi-view consistent manner.

The second challenge, and not the least, relies on the ability to actually re-inject them when sampling from a new viewpoint. As we highlighted that these features are tied to a specific viewpoint, the key difficulty resides in the choice of the transformed latent. To investigate that, we devise a simple experiment. Rather than modeling complex viewpoint deformations, we start by extracting features in a reference image, we transform the resulting maps using *simple* translation/rotation/scaling (the two last being the most interesting as they go out of the domain of invariance of CNNs), we use a strategy to choose a new latent (see discussion thereafter) and run DDIM sampling using the transformed feature maps following the procedure of *PnP-Diffusion*. Fig. 6.20 shows that, unsurprisingly, injecting feature maps with either the original (untransformed) latent or a random one results in completely corrupted images. This makes sense as, in this case, they are decorrelated and the argument about early injections and DDIM trajectories given above applies. More interestingly, equivariantly transforming the latent gives visually plausible results. However, as we push the transformation out of the domain of invariance of CNNs, this progressively falls apart.

At this stage, we might consider (and we experimented with that too) that last layers may be close to RGB and thus more robust to non-translational transformations. This is the approach recently taken by *Self-guidance Diffusion* [24]. Unfortunately, in their case, this component is part of a more constrained setup where additional and more structured regularizations are used, namely positions, shapes and sizes of objects. This builds on heavy manual preparation and the assumption that the model is able to disentangle concepts based on the input text prompt. Unfortunately, we observe in many cases that, for both *Stable Diffusion* and *DeepFloyd IF*, this is far from the case. For the former, we suggest that it is due to the poor compositionality of the CLIP text encoder [126] while for the latter, despite the use of a large T5 encoder analogous to *Imagen*, we hypothesize that it might be a side effect of the hybrid self-attention/cross-attention mechanism that they introduced. Such *visual binding* failure cases have recently been investigated [80, 26], but the proposed mending techniques are often complex to carry out.

In conclusion, manipulating feature maps requires to explore a very complicated optimization landscape. Furthermore, it often requires a lot of manual pre-processing, fine-tuning and engineering. In other words, there is no general recipe! As a consequence, it seems difficult to apply it directly to our "neural extrapolations" efforts. Note that we provide in appendix B additional insights regarding feature extraction on real images, inversion and feature-guided guidance.



(a) Reference (b) Random latent (c) Initial inverted latent transformed equivariantly (d) Transformed reference with re-inversion (e) Transformed and corrupted reference with re-inversion

Figure 6.20: With the extracted features as described in this section, we proceed to a re-injection process after performing a transformation (top row translation, bottom row: rotation) of the corresponding feature maps. The main challenge lies in the choice of the latent to use: (b) using a random latent doesn't take into account the residual nature of these feature, (c) performing an equivariant transformation of the latent works for transformations in the domain of invariance of CNN (namely translation) but does not generalize to rotations, (d) shows, as a *sanity check* (thus not realistic!), that inverting the equivariantly transformed original image and using the corresponding latent with the injected transformed feature maps faithfully recovers the image (suggesting that there is a good alignment and no obvious interferences), (e) extends (d) by corrupting the transformed image that is to be inverted and shows that feature maps can help recover some of the missed details but this becomes more limited when performing more aggressive transformations (e.g. rotations).

Chapter 7

Conclusion and discussions

Throughout this project, we investigated large-scale diffusion models and tried to lift the powerful 2D representations that they seem to capture in order to perform "neural extrapolations" within partial NeRF reconstructions. To do so, we decided to rely on two recently released models, namely *Stable Diffusion* and *DeepFloyd IF*. This task proved to be particularly arduous due to the intrinsic parameterizations and limitations of such models.

Indeed, we can summarize the challenges we faced into three main categories:

- At the time we write this report, large-scale diffusion models essentially build on a directed *text-to-image* cross-modality. Of course, recent works (e.g. *ControlNet*) have proposed to provide control through additional signals but they still require a text prompt to guide the entire process. This entails that they model the data manifold as a function of text and sampling a specific distribution can hardly be achieved in a 2D manner, let alone a 3D multi-view consistent way. As we observed throughout our experiments with *Textual Inversion* and (null-text) *DDIM inversion*, bridging the text-2D gap still relies heavily on the ability of the prompt to capture the context and concepts of the scene thus highlighting the critical underlying information bottleneck. Overfitting may be a sample-wise solution (see our results with *null-text inversion*) but comes at the cost of generalization which is central to the task we sought to tackle.
- By investigating multiple optimization strategies, we observed that diffusion models have a strong bias to model data close to their training data. This observation, of course, is not surprising as they are formally trained to approximate the score of the underlying probability distribution, thus recovering more accurately majority modes while neglecting long tail minority samples. This makes the above-mentioned task particularly challenging. More practically, this explains why trying to learn visual representations close to a specific scene is particularly complicated, either by inversion or through score distillation, as the optimization landscape is intractably unstable when it comes to these out-of-distribution representations. *Score distillation* also sheds light on the dependency on the "cleanliness" and disentanglement of the modelled data. Put differently, as the training corpus gets larger, diffusion models have a tendency to "mix" the different modes of the data, thus making the task of recovering a specific representation within their parameterization challenging, if not impossible.
- Last but not least, we observe by breaking down their internal activations (e.g. attention maps, feature maps, etc.) that these models build impressive geometrically and semantically-aware representations in a 2D manner. However, the latter seem to remain limited to a purely text-based conceptualization of geometry and semantic. Furthermore, their intrinsic convolutional parameterization, well-suited to the task of learning structure from 2D data, comes as a severe obstacle when trying to move to the 3D realm.

The question then becomes: what should we do then? Does it mean we need to directly learn 3D priors from data that might be very expensive to collect? We argue in favor of an intermediate solution that builds on recent observations between pre-training, generalization and geometrical priors. The fact that large-scale diffusion models are biased towards their training corpus and modality does not mean that they don't build generic priors at both coarse and fine levels of granularity, on the contrary. The key difficulty resides in finding a way to steer the latter towards the specific distribution of a given 3D scene. To this extent, we suggest drawing inspiration from recent energy-based frameworks [124] that propose to *adapt* the probabilistic prior of large pre-trained models by using smaller adapter models trained on a specific target domain, which for us would be the representations of a target scene or a small set of target scenes. This, of course, presupposes to keep the dependency on text but, contrary to these work and in light of the success of pose-guided T2I models [105], nothing prevents these small probabilistic adapters from being solely or additionally conditioned on geometrical signals. In other words, this would allow us to leverage the "global" priors of large-scale models as powerful generative regularizers while following the per-scene distributions acquired through the training of these adapters. To whomever read this report and would be interested in further investigating this direction, our door and minds are open...

Bibliography

[1] Stability AI. DeepFloyd IF. https://deepfloyd.ai/deepfloyd-if/.

[2] Yuval Alaluf, Elad Richardson, Gal Metzer, and Daniel Cohen-Or. A neural space-time representation for text-to-image personalization, 2023.

[3] Mohammadreza Armandpour, Huangjie Zheng, Ali Sadeghian, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *arXiv preprint arXiv:2304.04968*, 2023.

[4] Omri Avrahami, Kfir Aberman, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. Break-a-scene: Extracting multiple concepts from a single image. *arXiv preprint arXiv:2305.16311*, 2023.

[5] Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models, 2023.

[6] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022.

[7] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023.

[8] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing, 2023.

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

[10] Lucy Chai, Richard Tucker, Zhengqi Li, Phillip Isola, and Noah Snavely. Persistent nature: A generative model of unbounded 3d worlds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20863–20874, June 2023.

[11] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. GeNVS: Generative novel view synthesis with 3D-aware diffusion models. In *arXiv*, 2023.

[12] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models, 2023.

[13] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021.

[14] Minghao Chen, Iro Laina, and Andrea Vedaldi. Training-free layout control with cross-attention guidance. *arXiv preprint arXiv:2304.03373*, 2023.

[15] Ting Chen. On the importance of noise scheduling for diffusion models, 2023.

[16] Yuedong Chen, Haofei Xu, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Explicit correspondence matching for generalizable neural radiance fields, 2023.

[17] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models, 2022.

[18] Dana Cohen-Bar, Elad Richardson, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Set-the-scene: Global-local training for generating controllable nerf scenes, 2023.

[19] Giannis Daras and Alexandros G. Dimakis. Multiresolution textual inversion, 2022.

[20] 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)*, pages 20637–20647, June 2023.

[21] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised NeRF: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[23] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[24] Dave Epstein, Allan Jabri, Ben Poole, Alexei A. Efros, and Aleksander Holynski. Diffusion self-guidance for controllable image generation. *arXiv preprint arXiv:2306.00986*, 2023.

[25] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[26] Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. Training-free structured diffusion guidance for compositional text-to-image synthesis. *arXiv preprint arXiv:2212.05032*, 2022.

[27] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. *arXiv preprint arXiv:2302.01133*, 2023.

[28] 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.

[29] Songwei Ge, Taesung Park, Jun-Yan Zhu, and Jia-Bin Huang. Expressive text-to-image generation with rich text. *arXiv preprint arXiv:2304.06720*, 2023.

[30] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and P.J. Narayanan. Interactive Segmentation of Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[31] 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.

[32] Inhwa Han, Serin Yang, Taesung Kwon, and Jong Chul Ye. Highly personalized text embedding for image manipulation by stable diffusion. *arXiv preprint arXiv:2303.08767*, 2023.

[33] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. 2023.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[35] Eric Hedlin, Gopal Sharma, Shweta Mahajan, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. Unsupervised semantic correspondence using stable diffusion, 2023.

[36] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. 2022.

[37] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[38] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *arXiv preprint arXiv:2106.15282*, 2021.

[39] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[40] Susung Hong, Donghoon Ahn, and Seungryong Kim. Debiasing scores and prompts of 2d diffusion for robust text-to-3d generation, 2023.

[41] Susung Hong, Gyuseong Lee, Wooseok Jang, and Seungryong Kim. Improving sample quality of diffusion models using self-attention guidance. *arXiv preprint arXiv:2210.00939*, 2022.

[42] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models, 2023.

[43] Shir Iluz, Yael Vinker, Amir Hertz, Daniel Berio, Daniel Cohen-Or, and Ariel Shamir. Word-as-image for semantic typography, 2023.

[44] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. *arXiv*, 2022.

[45] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. *arXiv preprint arXiv:2303.09553*, 2023.

[46] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21696–21707. Curran Associates, Inc., 2021.

[47] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[48] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[49] Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *CVPR*, 2023.

[50] Mingi Kwon, Jaeseok Jeong, and Youngjung Uh. Diffusion models already have a semantic latent space, 2023.

[51] Yann Lecun, Sumit Chopra, Raia Hadsell, Marc Aurelio Ranzato, and Fu Jie Huang. *A tutorial on energy-based learning.* MIT Press, 2006.

[52] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022.

[53] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox, 2023.

[54] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[55] Kai-En Lin, Lin Yen-Chen, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *WACV*, 2023.

[56] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models, 2023.

[57] Aaron Lou and Stefano Ermon. Reflected diffusion models, 2023.

[58] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022.

[59] Calvin Luo. Understanding diffusion models: A unified perspective, 2022.

[60] Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. *arXiv*, 2023.

[61] Wan-Duo Kurt Ma, J. P. Lewis, W. Bastiaan Kleijn, and Thomas Leung. Directed diffusion: Direct control of object placement through attention guidance, 2023.

[62] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[63] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022.

[64] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. *arXiv preprint arXiv:2211.07600*, 2022.

[65] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[66] 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.

[67] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models, 2022.

[68] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, 2023.

[69] 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.

[70] Michael Niemeyer, Jonathan T. Barron, Ben Mildenhall, Mehdi S. M. Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5480–5490, June 2022.

[71] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[72] Dong Huk Park, Grace Luo, Clayton Toste, Samaneh Azadi, Xihui Liu, Maka Karalashvili, Anna Rohrbach, and Trevor Darrell. Shape-guided diffusion with inside-outside attention. *arXiv*, 2022.

[73] Yong-Hyun Park, Mingi Kwon, Junghyo Jo, and Youngjung Uh. Unsupervised discovery of semantic latent directions in diffusion models, 2023.

[74] Julien Philip and Valentin Deschaintre. Radiance field gradient scaling for unbiased near-camera training, 2023.

[75] Ryan Po and Gordon Wetzstein. Compositional 3d scene generation using locally conditioned diffusion, 2023.

[76] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

[77] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[78] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[79] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Ben Mildenhall, Nataniel Ruiz, Shiran Zada, Kfir Aberman, Michael Rubenstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. Dreambooth3d: Subject-driven text-to-3d generation. 2023.

[80] Royi Rassin, Eran Hirsch, Daniel Glickman, Shauli Ravfogel, Yoav Goldberg, and Gal Chechik. Linguistic binding in diffusion models: Enhancing attribute correspondence through attention map alignment, 2023.

[81] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021.

[82] Xuanchi Ren and Xiaolong Wang. Look outside the room: Synthesizing a consistent long-term 3d scene video from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[83] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes, 2023.

[84] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[85] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Ganerf: Leveraging discriminators to optimize neural radiance fields, 2023.

[86] Daniel Roich, Ron Mokady, Amit H. Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 42(1), aug 2022.

[87] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and BjÃ¶rn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[88] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[89] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

[90] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J. Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses, 2023.

[91] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[92] Dvir Samuel, Rami Ben-Ari, Simon Raviv, Nir Darshan, and Gal Chechik. It is all about where you start: Text-to-image generation with seed selection, 2023.

[93] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[94] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[95] Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*, 2023.

[96] Yang Song and Stefano Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[97] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[98] Mukund Varma T, Peihao Wang, Xuxi Chen, Tianlong Chen, Subhashini Venugopalan, and Zhangyang Wang. Is attention all that neRF needs? In *The Eleventh International Conference on Learning Representations*, 2023.

[99] 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.

[100] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior, 2023.

[101] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023.

[102] 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, 2020.

[103] Yoad Tewel, Rinon Gal, Gal Chechik, and Yuval Atzmon. Key-locked rank one editing for text-to-image personalization. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[104] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022.

[105] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. Consistent view synthesis with pose-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16773–16783, June 2023.

[106] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1921–1930, June 2023.

[107] Soobin Um and Jong Chul Ye. Don't play favorites: Minority guidance for diffusion models, 2023.

[108] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[109] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022.

[110] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[111] Andrey Voynov, Qinghao Chu, Daniel Cohen-Or, and Kfir Aberman. P+: Extended textual conditioning in text-to-image generation. 2023.

[112] Bram Wallace, Akash Gokul, and Nikhil Naik. Edict: Exact diffusion inversion via coupled transformations. *arXiv preprint arXiv:2211.12446*, 2022.

[113] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, June 2023.

[114] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. *CVPR*, 2023.

[115] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[116] 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.

[117] Frederik Warburg*, Ethan Weber*, Matthew Tancik, Aleksander Hołyński, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. 2023.

[118] 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.

[119] Qiucheng Wu, Yujian Liu, Handong Zhao, Trung Bui, Zhe Lin, Yang Zhang, and Shiyu Chang. Harnessing the spatial-temporal attention of diffusion models for high-fidelity text-to-image synthesis, 2023.

[120] 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)*, pages 4180–4189, June 2023.

[121] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond, 2022.

[122] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Yi Wang, and Zhangyang Wang. Neurallift-360: Lifting an in-the-wild 2d photo to a 3d object with 360° views. 2022.

[123] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. ODISE: Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. *arXiv preprint arXiv: 2303.04803*, 2023.

[124] Mengjiao Yang, Yilun Du, Bo Dai, Dale Schuurmans, Joshua B. Tenenbaum, and Pieter Abbeel. Probabilistic adaptation of text-to-video models, 2023.

[125] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[126] Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations*, 2023.

[127] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. 2023.

[128] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.

[129] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[130] Qinsheng Zhang, Molei Tao, and Yongxin Chen. gDDIM: Generalized denoising diffusion implicit models. In *The Eleventh International Conference on Learning Representations*, 2023.

[131] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[132] Junzhe Zhu and Peiye Zhuang. Hifa: High-fidelity text-to-3d with advanced diffusion guidance, 2023.

Appendix A

Stable Diffusion & DeepFloyd IF

In this chapter, we further review the internal specificities of the two models that we used in our experiments, namely *Stable Diffusion* and *DeepFloyd IF*.

A.1 Stable Diffusion

VAE. As jointly training the auto-encoder and the diffusion U-Net is particularly ill-defined and thus intractable. *LDM* [87] advocates the use of a two-stage training scheme where the VAE is first trained as a latent variable model and the diffusion model is then trained in the resulting representation space. The corresponding VAE is directly adapted from *VQGAN* [25] and is made of stacked convolutional residual blocks with downsampling (resp. upsampling) blocks for the encoder (resp. decoder). As a consequence, the latent space conserves a 2D layout, which makes it amenable to manipulations like the ones we introduce in chapter 4. In practice, this latent space has a lower resolution (downsampled 4 times in most available implementations and thus yields 64×64 images) and a different number of channels (4 in commonly used implementations). To learn a perceptually compressive latent space, *LDM* builds again on *VQGAN* and, in addition to a perceptual reconstruction loss (with LPIPS), uses an adversarial loss where a patch-based discriminator is optimized to discriminate the reconstructions. Additionally, they propose two different regularizations: the first is a rather standard KL-divergence term to force the learned distribution to stay close to a standard zero-mean gaussian [47], the second is an adaptation of the vector quantization scheme introduced in *VQGAN* which constructs a large discrete codebook of latent vectors.

U-Net. The U-Net architecture introduced by *LDM* [87] builds on the standard U-Net architecture by stacking residual blocks where each encoder block is augmented by a self-attention layer followed by a cross-attention layer. Note that as already discussed in section 3.4, *Stable Diffusion* implements 6 encoding/downsampling blocks, 1 bottleneck block and 9 decoding/upsampling blocks.

A.2 DeepFloyd IF

While implementing the experiments of chapter 6 on the HuggingFace implementation of the model, we observed that it is mostly a standard *cascaded diffusion model* [38] (i.e. similar to *imagen* [91]). However, it uses a fused self-attention and cross-attention mechanism. We hypothesize that this was made to restrict the number of weights of the model and thus make it more accessible while retaining the advantages of both mechanisms. Yet, as highlighted in section 6.3, we noticed that it is particularly subject to "visual binding" failure and "concept leakage". This choice of implementation might be partially responsible for that. Finally, contrary to *imagen*, *DeepFloyd*'s super-resolution models do implement self-attention, but in a fused way as discussed earlier.

Appendix B

Feature injection: a short survey & discussions

In the following, we denote as feature maps any of cross-attention maps, self-attention maps and residual feature maps. Tab. B.1 summarizes the methods we describe below in related works.

Extracting features. When performing *text-to-image* sampling, it is direct that feature maps are to be extracted at each step of ancestral sampling i.e., when evaluating $\epsilon_\theta(z_t; t)$ at line (4) of algorithm 1. However, when dealing with real images, it is more ambiguous how these should be extracted. We identify three strategies:

- "*forward inversion-based*": the feature maps are extracted during the inverse process of DDIM inversion i.e., from z_0 to z_T.
- "*reverse inversion-based*": the feature maps are extracted during the reverse reconstruction following DDIM inversion i.e., from z_T to $\hat{z}_0$. This is the approach taken by *PnP-Diffusion* [106].
- "*noise-based*" where the features do not depend on the trajectory. More precisely, for each $t \in [1, T]$, we draw $z_t = \sqrt{\alpha_t} z_0 + \sigma_t \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, I)$. Note that considering our discussions on the relationship between the DDIM inversion trajectory and the residual nature of raw feature maps, this approach can only work if we consider an entirely decorrelated injection strategy i.e., relying on pure DDPM sampling! We provide additional insights on that in the following **DDPM inversion** paragraph.

Manipulating features. After their extraction, features are sometimes post-processed to either "clean" them or produce lower-dimensional properties that can later be constrained (see paragraph **Injecting features**):

- Cross-attention maps are often binarized to produce masks that will help disentangle an object from its background when performing edits or injecting other feature maps [8]. This can usually be performed naively using a hard thresholding operation. However, as suggested by *self-guidance diffusion*, it might be beneficial to rather apply a soft-thresholding operation to steer apart noisy intermediate values:

  $$\begin{aligned} A^{\text{thres}} &= \mathbf{normalize}(\text{sigmoid}(s \cdot \mathbf{normalize}(A) - 0.5)) \\ \text{where } \mathbf{normalize}(X) &= \frac{X - \min_{h,w}(X)}{\max_{h,w}(X) - \min_{h,w}(X)} \end{aligned} \tag{B.1}$$

  Note that we applied this operation for the cross-attention maps extracted from *DeepFloyd IF* in Fig. 6.12 and Fig. 6.13 (we used $s = 10$ as suggested originally in the paper too).
- When edited, cross-attention maps can be either clipped [36], renormalized [29] or filtered [12].
- *Self-guidance diffusion* [24] also suggested the use of cross-attention maps to compute more general concept-centric (and thus token-centric) properties like a "center of mass" or size. Let k be a target

Paper	**Model**	**Layers**	**Injection schedule**	**Injection method**	**Post-processing**
Prompt-to-Prompt [36]	*imagen*	all cross-attention maps	early schedule	direct injection	values clipped when scaling
Attend-and-Excite [12]	*Stable Diffusion*	16×16 cross-attention maps	$t \in [25, 50]$ for DDIM(50)	guidance-based to produce minimal per-token activation	gaussian filter
Layout Control [14]	*Stable Diffusion*	bottleneck and early upsampling blocks cross-attention maps	$t \in [40, 50]$ for DDIM(50)	both direct and guidance-based injection	renormalization implicitly specified by the guidance energy formulation
Rich Text-to-Image [29]	*Stable Diffusion*	all cross-attention maps except the first encoder and last decoder blocks	unclear	direct and guidance-based depending on the types of edits	renormalization
Spatial-Temporal Attention [119]	*Stable Diffusion*	all cross-attention maps	entire schedule	mixed injection with optimized blending weights using a CLIP-based loss	
MasaCtrl [8]	*Stable Diffusion*	projected self-attention key and value feature maps of bottleneck and early upsampling blocks	entire schedule except early steps	direct injection with foreground/background masking to avoid leakage	
PnP-Diffusion [106]	*Stable Diffusion*	bottleneck raw residual feature maps and early upsampling feature maps	$t \in [25, 50]$ for DDIM(50)	direct injection	
Self-guidance diffusion [24]	*imagen*	raw residual feature maps of the penultimate upsampling layer	early schedule and then alternating between injection/no injection	guidance-based	multiple regularization on per-object properties

Table B.1: Summary of the feature injection strategies in related works

token and $A_{h,w,k}$ be the value of pixel h, w in the corresponding cross-attention map at an arbitrary layer l, these two properties can be approximated as follows. Note that, for clarity, we omit the timestep index t and channels c in the previous equation; for the latter, in practice we sum over all channels and the renormalization step in the following equation takes care of the rest.

$$\mathbf{centroid}(k) = \frac{1}{\sum_{h,w} A_{h,w,k}} \begin{bmatrix} \sum_{h,w} w \cdot A_{h,w,k} \\ \sum_{h,w} h \cdot A_{h,w,k} \end{bmatrix} \tag{B.2}$$

$$\mathbf{size}(k) = \frac{1}{HW} \sum_{h,w} w \cdot A_{h,w,k} \tag{B.3}$$

Unfortunately, when prototyping with *DeepFloyd IF*, we were not able to properly disentangle objects because its cross-attention maps tend to be noisy or bound to unwanted concepts as discussed earlier in section 6.3.

Injecting features. Once extracted, there exists three main strategies for re-injecting feature maps:

- A *direct* approach where the feature maps are simply replaced at specific steps of the reverse sampling process [36, 106]. As we observed throughout our experiments, this is particularly unstable as it doesn't account for the error and misalignment throughout the sampling trajectory. In other words, with DDIM sampling, it works well in early stages but if used too long, it will eventually diverge. This is particularly critical as, in our case, we were not only interested in recovering the layout and shapes of images but also their appearance. However, as highlighted by multiple works [17, 50], later stages of the noise schedule are mostly responsible for the generation of the latter properties. This justifies the use of smoother mechanisms.
- A *mixed* approach where cross-attention maps are blended in a weighted fashion. Note that the corresponding blending weights can smoothly be optimized using for example a per-object CLIP-based loss as done in *Spatial-Temporal Attention* [119].
- A *guidance-based* approach where a loss or energy is defined in order to smoothly force the feature maps generated during sampling to match extracted, post-processed and potentially edited ones. More precisely, equation 3.8 provides a general framework for that approach which can be seen as a form of *implicit regularization*. In practice, this can be done either explicitly on a per-pixel basis with a L2 loss [29] or L1 loss [24], using more heuristic energies [14] or by taking into account per-token overall activations [12] or interactions in-between tokens [80]. Note that as suggested by *Universal Guidance* [5], it may be beneficial to break per-step updates into multiple projection sub-steps in a similar fashion as the *re-noise/denoise* operation originally proposed for inpainting in *RePaint* [58] or using Monte Carlo approximations [95].

Furthermore, another key design choice is the schedule of the injection. In other words, most methods use dense and early injection intervals while some add relaxed injection schedules in a *dropout* fashion [24].

DDPM inversion & self-guidance. At this stage, we remained rather elusive regarding the inversion process used to perform feature injection as, a priori, it seems obvious that feature maps are particularly connected to a specific inverted trajectory, either when sampling from text and thereafter editing, or when performing DDIM inversion from a real image. However, a recent work has shown that we can actually leverage DDPM and the correlations between noise vectors injected at each step to perform another form of inversion [37]. This is particularly interesting because it addresses a problem that we pointed out multiple times in this report: DDIM sampling and inversion are particularly sensitive to divergence! On the other hand, by building on DDPM and stochasticity, we can prevent dramatic "mode collapse" (when DDIM diverges it is not a real mode collapse though...). Unsurprisingly and concurrently, *self-guidance diffusion* [24] has recently built on the idea of completely getting read of the inversion formulation to perform edits that are still highly consistent in appearance. Intuitively, we interprete its success as a carefully tuned regression/regularization approach: sampling is conditioned solely via energy/gradient based guidance on both global properties (e.g. position, shape, size, etc.) and appearance properties (penultimate feature maps). The latter act as a powerful way to steer the sampling process towards a target 2D appearance in a "bottleneck" fashion. However, doing so via pure gradient-based guidance, the denoising network can fully navigate the space of possible representations through DDPM without hard constraints and thus "deadlocking". Note that the fact that the chosen feature maps are the penultimate ones is central to avoid the invariance "locking" problem we faced several times throughout this project: these feature maps can be seen as almost (or an augmented version of) RGB colors and are thus potentially more robust to deformations.

Appendix C

Additional derivations & algorithms

C.1 Null-text optimization

Algorithm 2 Null-text optimization

```
 1: Perform DDIM inversion at low-guidance ($\gamma = 1.0$) and collect $z_0^*, \ldots, z_T^*$
 2: Set the guidance to a higher value (e.g. $\gamma = 7.5$)
 3: Initialize $\bar{z}_T \leftarrow z_T^*$, $\emptyset_T \leftarrow \emptyset$
 4: for t=T,..., 1 do
 5:     for j=0,..., N-1 do
 6:         $\emptyset_t \leftarrow \emptyset_t - \eta \nabla_\emptyset \|z_{t-1}^* - z_{t-1}(\bar{z}_t; \emptyset_t, c)\|_2^2$
 7:     end for
 8:     $\bar{z}_{t-1} \leftarrow z_{t-1}(\bar{z}_t, \emptyset_t, c), \emptyset_{t-1} \leftarrow \emptyset_t$
 9: end for
10: return $\bar{z}_t; (\emptyset_t)_{t=1}^T$
```

where $z_{t-1}(\bar{z}_t; \emptyset_t, c) = \sqrt{\frac{\alpha_{t-1}}{\alpha_t}} \cdot \bar{z}_t + \left(\sqrt{\frac{1}{\alpha_{t-1}} - 1} - \sqrt{\frac{1}{\alpha_t} - 1}\right) \cdot \hat{\epsilon}_\theta(z_t; \emptyset_t, c)$ and
$\hat{\epsilon}_\theta(z_t; \emptyset_t, c) = \epsilon_\theta(z_t; t, \emptyset_t) + \gamma \left[\epsilon_\theta(z_t; t, c) - \epsilon_\theta(z_t; t, \emptyset_t)\right]$