FYT Proposal

3D Editing

by

XU Yanbo

CQF5

Advised by

Prof. CHEN Qifeng (陈启峰)

Submitted in partial fulfillment of the requirements for COMP 4981H

in the

Department of Computer Science

The Hong Kong University of Science and Technology

2022-2023

Date of submission: April 20, 2023

Table of Contents

1 Introduction

1.1 Overview

The tasks of image manipulation have always been important. Recently, with the advancements in AI-based image manipulation in the field of computer vision, many methods for editing in the 2D images domain have been proposed and proven effective. It is natural to extend the manipulation to the 3D domain, whose success could bring improvement in the multimedia industry and even areas such as VR, Gaming, etc.
This critical task is feasible with the success of diffusion models (DMs) [1] and text models, generating or editing an image with only textual input is possible. DMs could also be used to generate 3D models conditioned on text input, which shows the possibility of guiding 3D generation using 2D models.

1.2 Objectives

This research project aims to propose a novel NeRF-based 3D editing method using DMs as guidance and text as editing input. In the context of 3D image editing, image quality, 3D consistency, and preservation of original object characteristics are the primary focus for evaluation. As 3D models are hard to train due to the difficulty of 3D data collection, we would like to utilize a pertained 2D text-to-image diffusion model for this task.

2 Related Works

2.1 2D Image Generative Models

The editing operation on images is closely related with the task of image generation. Realistic and diverse generation of 2D images has been well studied. Based on the problem modeling, these methods could be divided into three categories, namely Generative Adversarial Network (GAN) [2], Variational Auto-Encoder (VAE) [3] and Diffusion Model (DM)[1]. GANs are capable to generate diverse and high-quality images but suffer from unstable training. On the contrary, the probabilistic-based VAE and DM are very stable, yet the generative quality of early research is lower and sampling efficiency is slower compared with GANs. These problems have been solved with the recent development of DMs.

2.2 Image Editing by Generative Models

For these generative models, their latent space and model parameters contain abundant and semantic-meaningful information about their generative data domain (the training data). Therefore, realistic and precise image editing is feasible with the guidance of a generative model. There have been many methods proposed. For example, InterfaceGAN [4] utilizes classifiers to train hyperplanes in the 2D image latent space. Thus moving in the direction orthogonal to the hyper-planes will edit the image. UniTune [5] applies a pre-trained diffusion model and fine-tunes the model to achieve editing. There are also some methods for editing in the 3D domain. For hybrid 3D generation models such as StyleSDF [6] and EG3D [7], their latent spaces are also meaningful, and it is possible to apply the techniques of 2D image editing to those methods.

2.3 Diffusion Models

Diffusion models [1, 8, 9] are inspired by the physic concept of non-equilibrium thermodynamics. The generation process is formulated as a denoising process given a random noise from the latent space, and the noise could be obtained from adding noise to the image using a Markov chain.

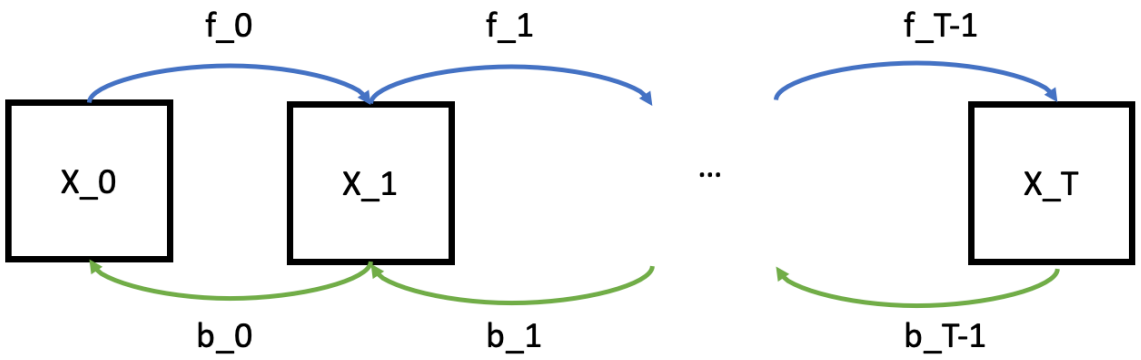

Figure 1 Diffusion Process. X_0 is the real image data sampled, X_T is the latent (noise). The generation process is the denoise process [b_T-1, …, b_0]

Given a data sampled from a real data distribution (the training data) $\mathrm{x}_0 \sim q(\mathrm{x})$, the forward process is defined as a Markov process with time step T:

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

, where $q(\mathrm{x}_{1:T}|\mathrm{x}_0) = \prod_{t=1}^{T} q(\mathrm{x}_\mathrm{t}|\mathrm{x}_{t-1})$ according to the Markov property. It could also be shown that:

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

Therefore, the forward process could be calculated efficiently that:

$$x_t = \sqrt{\alpha_t} x_0 + (1-\bar{\alpha}_t)\epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, I).$$

The generation process of an image from noise could be formulated as the backward process $q(\mathrm{x}_{\mathrm{t}-1}|\mathrm{x}_t)$ using a model p_θ, where $p_\theta(\mathrm{x}_{0:\mathrm{t}}) = p(\mathrm{x}_\mathrm{T}) \prod_{t=1}^{T} p_\theta(\mathrm{x}_{\mathrm{t}-1}|\mathrm{x}_t)$, and $p_\theta((\mathrm{x}_{\mathrm{t}-1}|\mathrm{x}_t)) = \mathcal{N}(\mathrm{x}_{\mathrm{t}-1}; \mu_\theta(\mathrm{x}_t, t), \Sigma_\theta(\mathrm{x}_t, t))$.

Also, an important fact for the reverse process is that the reverse conditional probability is tractable when conditioned on x_0 that:

$$q(\mathrm{x}_{t-1}|\mathrm{x}_t, \mathrm{x}_0) = \mathcal{N}(\mathrm{x}_{t-1}; \tilde{\mu}(\mathrm{x}_t, \mathrm{x}_0), \tilde{\beta}_t \mathrm{I}).$$

From this equation, we could finally model $\mu_\theta(\mathrm{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\mathrm{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathrm{x}_t, t))$. Then, using the variational lower bound used in VAE plus some simplification, the training objective could be written as $L_t^{\boldsymbol{simple}} = \mathbb{E}_{t\sim[1,t],\, \mathrm{x}_0,\, \epsilon_t}[|| \epsilon_t - \epsilon_\theta(\mathrm{x}_t, t) ||^2]$. Therefore, by parameterization,

$$L_t^{\boldsymbol{simple}} = \mathbb{E}_{t\sim[1,t],\, \mathrm{x}_0,\, \epsilon_t}[|| \epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathrm{x}_0 + , \sqrt{1-\bar{\alpha}_t}\,\epsilon_t,\ t) ||^2].$$

2.3.1 DDIM Inversion

As [8] stated, the reverse process could be written as:

$$x_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{x_0 - \sqrt{1-\alpha_t}\epsilon_\theta(\mathrm{x}_t, t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2} * \epsilon_\theta(\mathrm{x}_t, t) + \sigma_t\epsilon_t.$$

It could also be written as:

$$\frac{x_{t-\Delta_t}}{\sqrt{\alpha_{t-\Delta_t}}} = \frac{x_t}{\sqrt{\alpha_t}} + \left(\sqrt{\frac{1-\alpha_{t-\Delta_t}}{\alpha_{t-\Delta_t}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}}\right) * \epsilon_\theta(\mathrm{x}_t, t)$$

With enough discrete steps, $\Delta_t = T$, it is possible to invert x_0 (the origin image) from x_t(the noise generated from forward process).

2.3.2 Conditioned Diffusion Models

The origin estimator in diffusion model is conditioned only on its input in pixel space and time t, thus the generation process is not controllable, i.e., the final output is only determined by the input noise. To control the generation process, adding condition parameter to the noise estimator is a valid option. However, the conditional information should have same dimension as the noise. Stable Diffusion [9] allows conditioning from both image and text, and the generation process is defined as:

$$L_t^{LDM} = \mathbb{E}_{t\sim[1,t],\,\mathcal{E}(x_0),\,\epsilon_t}[||\,\epsilon_t - \epsilon_\theta(x_t,\ t,\ \tau_\theta(y))\,||^2]$$

, where τ_θ is a pre-trained text encoder such as clip for text-image generation or an image encoder for image conditioning. Moreover, the As text is the most natural way of expressing editing demand, we use textual description to control the editing process.

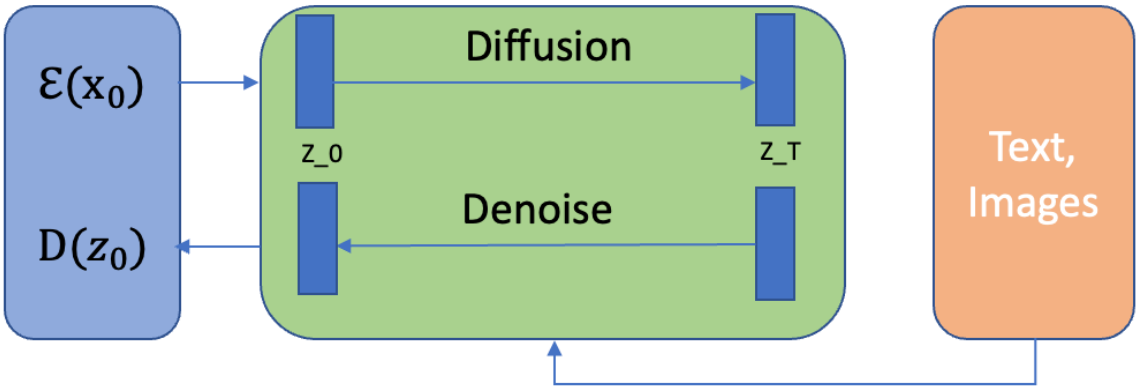

Figure 2 Pipeline of StableDiffusion. Start from a latent code Z_0 ($\mathcal{E}(x_0)$ when training). The denoise process could be conditioned by a text/image.

2.3.3 Inversion for Conditioned Diffusion Models

Although DDIM [8] could invert and reconstruct images, the deterministic nature is valid in the unconditional case. For text or image conditioned diffusion models, although images could be produced using the latent by encoding image, i.e., $z \sim \mathcal{E}(x)$, the generation process lacks specification as the origin textual/image condition is missing. This also makes editing the original images harder, as adding inaccurate textual descriptions will deviate the results from given inputs.

Textual Inversion [10] is proposed to solve this issue by optimizing a new token for the given object. Its optimization process is based on a small set of images, formulated as:

$$v_* = argmin\ \mathbb{E}_{z\sim\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\,[||\epsilon - \epsilon_\theta(x_t, t, \tau_\theta(y)||^2],$$

where v is the vector embedding of a newly added token for the object, and the weight for

diffusion model ϵ_θ and condition mapping network τ_θ are frozen. After inversion, editing could be performed by simply adding the newly added token to a text prompt.

Instead of finding the best text input for the network as [10], turning the diffusion network to fit the given input description is also a valid choice. DreamBooth [11] finetunes the generator on a small set of images with a text prompt composed of a rare token. However, the process might suffer from overfit and language drift. Thus, they propose to add a prior-preservation Loss:

$$\mathbb{E}_{x,c,\epsilon,\epsilon',t}\left[w_t||\epsilon_\theta(\alpha_t x + \sigma_t \epsilon, c) - x||_2^2 + \lambda w_{t'}||\epsilon_\theta\left(\alpha_{t'} x_{pr} + \sigma_{t'} \epsilon', c_{pr}\right) - x_{pr}||_2^2\right],$$

where c_{pr} is the caption for a general class member, and x_{pr} is the images generated by the origin model conditioned on the general caption.

Another important factor of the generation process is the null-text. The null-text is proposed by classifier-free guidance [12], which is firstly used for unconditional generation and then extrapolated with the conditional input. The null-text inversion [13] is optimizing the null-text embedding $\emptyset_t$, formulated as:

$$\min ||z_{t-1}^* - z_{t-1}(\overline{z_t}, \emptyset_t, C)||_2^2,$$

where $z_{t-1}(\overline{z_t}, \emptyset_t, C)$ is the DDIM sampling and z_{t-1}^* is the latent trajectory generated from initial DDIM process.

2.3.4 Image Editing using Conditioned Diffusion Models

Using the aforementioned inversion methods, if an image could be inverted to the latent space or text space or both, editing could be applied by modifying the inversed text prompt. For textual inversion, the unique token “<token>” could be used to represent the object, and by including this token into the prompt, we could get diverse results on the object. For example, “a photo of a <token> in the snow” is a valid token which intended to put the object in the snow. DreamBooth [11] is similar to textual inversion [10], the difference is that the new token is used to describe attribute of an object, such as “a photo of a <token> dog

on the ground". For the null-text inversion [13], no new token is added into the prompt. They relate one image to a set of null-tokens and utilize them during the generation process with edited prompt.

Instead of inverse and change prompt to edit, it is also possible to edit images in an end-to-end fashion. InstructPix2Pix [14] allows image editing with a single backward pass with the editing command. Its generator is conditioned on both image latent and textual command, and therefore its training objective becomes:

$$L = \mathbb{E}_{\mathcal{E}(x),\mathcal{E}(c_I),c_T,t,\epsilon_t}[|| \epsilon_t - \epsilon_\theta(\mathrm{z}_t, t, \mathcal{E}(c_I), c_T) ||^2].$$

Similar to [9], $\mathcal{E}$ is the image-to-latent encoder, and c_T is the condition text instruction. Here the denoising network ϵ_θ is conditioned on image latent, which serves similar purpose as the inversion process that preserves the information of origin image.

2.4 Neural Rendering

Despite the high image quality achieved by 2D generative models, viewing consistency is a challenging task. The recent development of implicit neural rendering provides a potential solution to this problem. NeRF [15] is one of the most promising methods, which represents a static scene using a 5D vector-valued function. The input is the 3D location $\mathbf{x} =$ (x, y, z) and the 2D view direction $\mathbf{d} =$ (θ, ϕ), and the output is an emitted color $\mathbf{c}$ = (r, g, b) and a volume density σ, which represents the transparency at the query point. The mapping f is approximated using a Multi-Layer Perceptron (MLP): $f: (x,\ d) \rightarrow (c,\ \sigma)$. Therefore, given the camera position, an image could be rendered by integrating the color along the ray emitted from the camera, given by:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\mathbf{r}(t)\big)\,\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

, where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$. This process is referred to as volume integration, where the volume density will be used as the weight for the color during integration. Since

the process is modeled with physical constraints, rendered results with different camera origins are view-consistent.

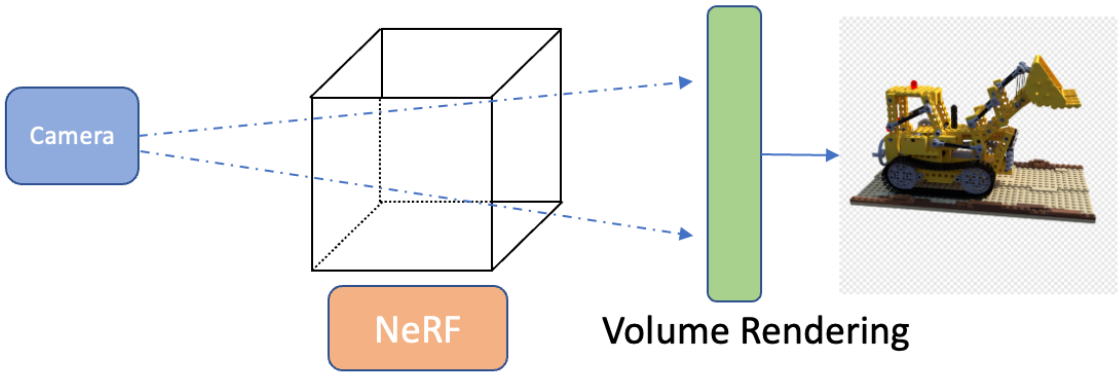

Figure 3 Volume Rendering and NeRF. NeRF represent a 3D scene using implicit weight, which could be quired from a camera position. Volume rendering will produce the final RGB image.

2.5 NeRF Editing

There are several attempts to perform editing on NeRF. [16] utilizes an unsupervised method for NeRF segmentation, thus being capable of editing those segmented part. However, as the editing lacks semantic guidance, their editing operations are confined to alteration of the colors or deletion of an object. NeRF-Editing [17] connects NeRF with mesh representation, which could be used to drive the deformation of NeRF. Although some extend of deformation could be achieved, more complex editing such as add object, change object is cannot be performed. [18] and [19] allow edits on both color and shape of an object by training a shared parameter space for a type of objects. However, for the relatively rare classes, these types of approaches are not valid. Generators such as [20, 21] could be used for 3D editing, but their capacity is limited by there generative power. So far, 3D generation is much harder than that of 2D.

3 Methodology

Our objective is to enable NeRF editing with the guidance of a pre-trained diffusion model. Additionally, we utilize textual description to control the editing process, as text is the most natural way of expressing the wanted change.

3.1 Problem Formulation

Given an input NeRF model $NeRF(\varphi)$ as a trained representation of an object or scene, together with a text description y describing the wanted editing, our objective is to optimize a new NeRF model $NeRF(\varphi_{edit})$, such that $NeRF(\varphi_{edit})$ represent the same object as $NeRF(\varphi)$ but fit the given text description.

3.2 Choice of Diffusion Models

As our objective is to utilize a textual description and guide the optimization process, given a text, the model should have the fowling quality:

$$I_{new} = Model(y, I_{old})$$

Stable Diffusion is a good option, as it could be conditioned using both image and text, and its training process is defined as:

$$L_t^{\boldsymbol{LDM}} = \mathbb{E}_{t\sim[1,t],\, \mathcal{E}(\mathrm{x}_0),\, \epsilon_t}[||\, \epsilon_t - \epsilon_\theta(\mathrm{x}_t,\ t,\ \tau_\theta(y))\, ||^2]$$

, where τ_θ is a pre-trained text encoder such as clip for text-image generation or an image encoder for image conditioning. However, during the generation process, x_t might deviate from the origin latent embedding of the input $\mathcal{E}(\mathrm{x}_0)$. Therefore, similar to text, it is possible to jointly condition the generation process using the origin image, formulated as:

$$L_t^{\boldsymbol{LDM'}} = \mathbb{E}_{t\sim[1,t],\, \mathcal{E}(\mathrm{x}_0),\, \epsilon_t}[||\, \epsilon_t - \epsilon_\theta(\mathrm{x}_t,\ t,\ \tau_\theta(y), \mathcal{E}(\mathrm{x}_0))\, ||^2],$$

where the origin image is encoded and conditioned at each time step.

3.3 Diffusion-based NeRF Optimization

Generating high quality 2D images could be easily achieved with a pre-trained text-to-image diffusion model. The optimization process of NeRF requires capture of an object from different viewpoints. However, trivially generating few images from a fixed text prompt and utilizing them to calculate the NeRF will fail, as DM produces very diverse results. Score Distillation Sampling (SDS) [22] is a solution to this problem, and it effectively leverages the semantic information in the diffusion model. SDS starts from taking gradient of NeRF parameters, described as:

$$\nabla_\varphi L(\theta, x = NeRF(\varphi)) = \mathbb{E}_{t,\epsilon}\, [w(t)(\epsilon_\theta(\mathrm{z}_t,\ t,\ \tau_\theta(y)) - \ \epsilon_t)\ \frac{\partial \epsilon_\theta(\mathrm{z}_t, t, \tau_\theta(y))}{\mathrm{z}_t} \frac{\partial x}{\partial \varphi}]$$

, where z_t is the noised image x rendered by NeRF, and $w(t) = \frac{\partial \mathrm{z}_t}{\partial x} = \ \alpha_t \mathrm{I}$. Empirically, fixing the DM parameters leads to efficient optimization results. Therefore, the gradient for the NeRF model with parameter φ is:

$$\nabla_\varphi \mathrm{L}_{SDS}(\theta, x = NeRF(\varphi)) = \mathbb{E}_{t,\epsilon}\, [w(t)(\epsilon_\theta(\mathrm{z}_t,\ t,\ \tau_\theta(y)) - \ \epsilon_t)\ \frac{\partial x}{\partial \varphi}].$$

If the model is conditioned on both image and text, then the gradient would become:

$$\nabla_\varphi \mathrm{L}_{SDS}(\theta, x = NeRF(\varphi), x' = NeRF(\varphi_{fix})) = \mathbb{E}_{t,\epsilon}\, [w(t)(\epsilon_\theta(\mathrm{z}_t,\ t,\ \tau_\theta(y), \mathcal{E}(x')) - \ \epsilon_t)\ \frac{\partial x}{\partial \varphi}].$$

Note that x' is rendered from the original NeRF input, whose weight is fixed during the optimization process of the new NeRF model. The process is shown in Figure 4.

3.4 Diffusion-guided NeRF Editing

Since DMs learn semantic information well and could be conditioned on text, image manipulation with diffusion model is possible. Given input image x_0 and its origin caption image t_0, it is desirable to generate new image x_{target} with editing instruction or edited caption t_{target}, such that x_{target} preserve qualities of origin image and could be well described by t_{target}. In the case of NeRF editing, x_0 and x_{target} are rendered from NeRF model $NeRF_0$ and $NeRF_{target}$. Although some methods utilize generated pair data to train the diffusion model for one-shot image editing, generating paired data for NeRF is much more difficult. Therefore, we adopt the optimization-based approach for our NeRF editing, which involves inversion in diffusion space flowed by editing operation.

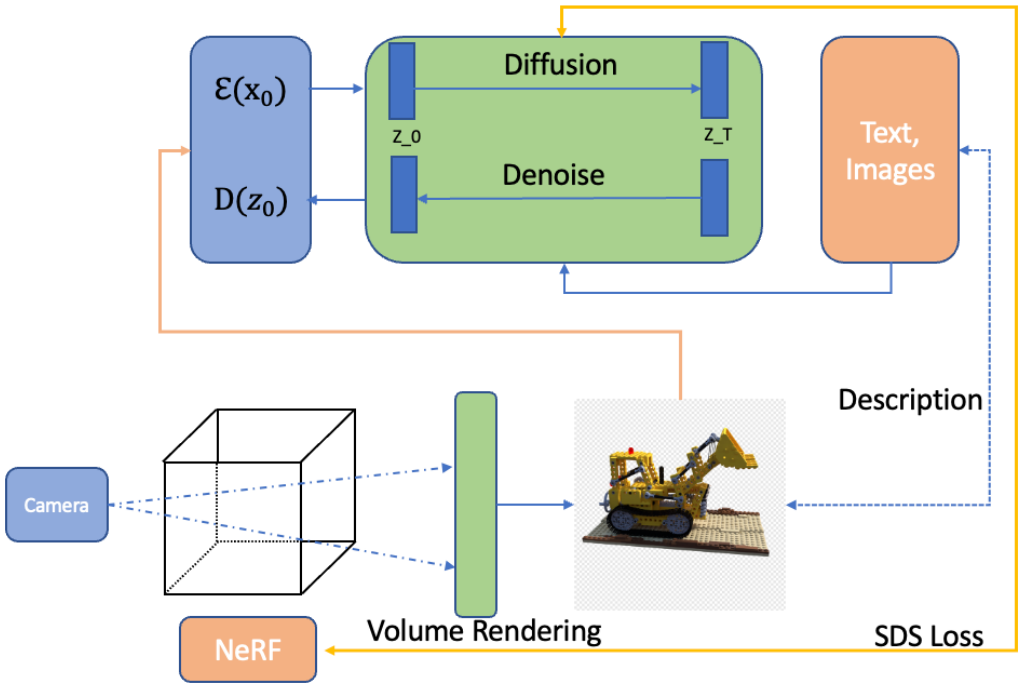

Figure 4 Dreamfusion [22] Pipeline. The NeRF will render an image, which will then be added with random noise and passed to the denoise network. The network will denoise based on its input and the condition. The gradient directly acts on the NeRF.

3.4.1 NeRF Inversion to Diffusion Space

For image diffusion inversion tasks, the inversion is mostly achieved by the inversion methods mentioned in Section 2.3.3. For our case, we utilize the $NeRF_0$ to render images from p different viewpoints into $I_0 = [x_1, \dots, x_p]$. These rendered images will be used to perform inversion. We conduct experiments on textual inversion, fine-tuning of origin model, and utilizing both. The Inversion process would link a unique token with the diffusion model, such that the model will generate images related to the object represented using $NeRF_0$ given any suitable editing command. For the model where image is used as condition at each step, no inversion is required as rendered image from $NeRF_0$ provides sufficient information.

3.4.2 NeRF Editing

With the fine-tuned model, we then optimize $NeRF_{target}$ using SDS equation given θ' and t_{target}, which can render 3D consistent images. Note that during the optimization, there are some regularization methods could be adopted to improve the editing quality, such as classifier-free guidance [12], clip distance [23], etc.

4 Experiments

4.1 NeRF Editing on Generated Models

Our assumption is that when the latent code, diffusion model and the text prompt are well aligned, editing the NeRF model could be achieved by simply changing the text prompt. In Figure 5 (c), the color could be edited by changing color description in the prompt from “yellow” to “red”. However, during the editing process, if the origin NeRF weight is not provided and the optimization is entirely based on the new prompt, the model will converge to another model Figure 5 (b). Therefore, having the origin NeRF weight as the initial point is crucial. In addition to color, editing geometric information is also feasible. Figure 5 (e)&(f) shows the result of shape editing.

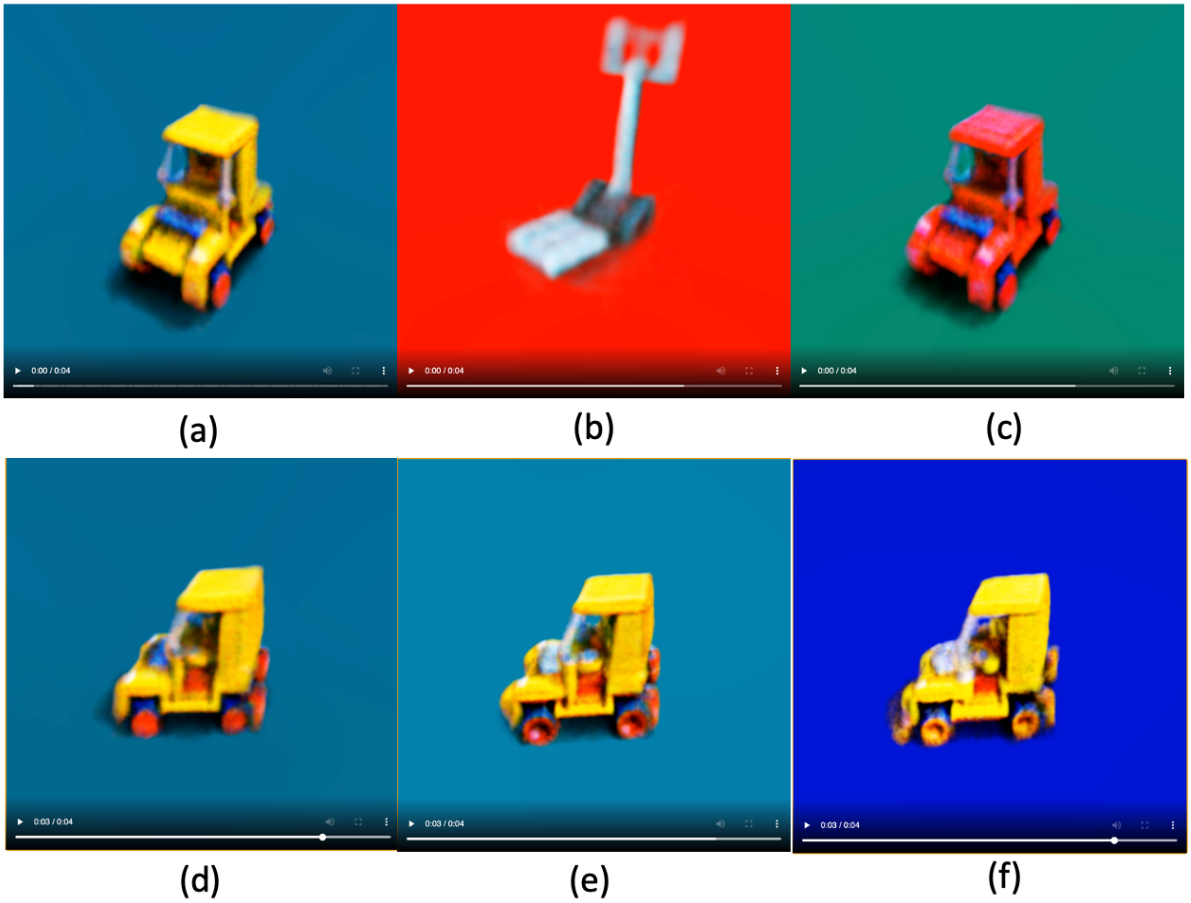

Figure 5 Editing results on generated NeRF models captioned as “A rendering of a yellow Lego shovel truck”. Image (a) and (d) are the generated results. Image (b) is the result of “A rendering of a Red Lego shovel truck” trained from start, while Image (C) is initialized from the origin Input. Image (e)

edits the larger front wheel, and image (f) tends to change the texture from Lego to cake.

4.2 Naïve Editing on Real-world Models

Although editing on generated models is relatively easy, most real-world applications take NeRF models from real data as input. In this case, an accurate text prompt is hard to obtain. Besides, the diffusion model may lack the capability of generating images in the given data domain. Figure 6 upper row shows the color editing result by simply changing prompt as in Section 4.1. For small iterations, the color could be changed while preserving its geometric information to some extent. However, as the training goes on, the edited NeRF will converge to a new model with undesirable change.

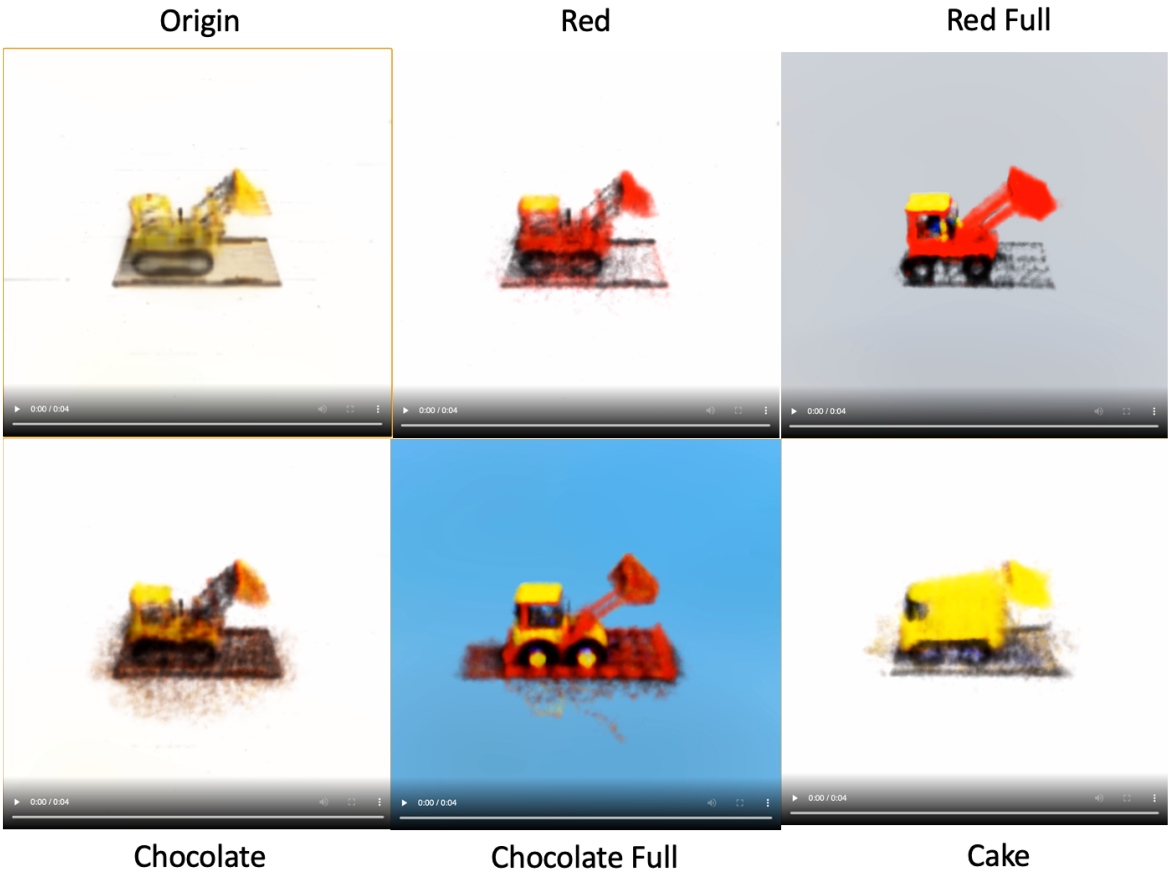

Figure 6 Direct editing on real data, with origin caption “A rendering of a yellow Lego shovel truck”. The first row edits the color. At early iterations, the model perseveres the origin model better. With full iterations, the model will converge to a different model. The below row is the early and full editing of changing texture to “chocolate”, and the last result is early iteration of changing texture to “cake”

4.3 Textual Inversion for Editing

As could be seen from section 4.2 and section 4.1, their caption is identical, yet the object generated is quite different. This is the results of language ambiguity of human language. In 2D, this problem could be solved by textual inversion that defines a unique token for the set of images. Figure 7 shows how the diffusion model interpolates the newly defined token as the training iterations increases.

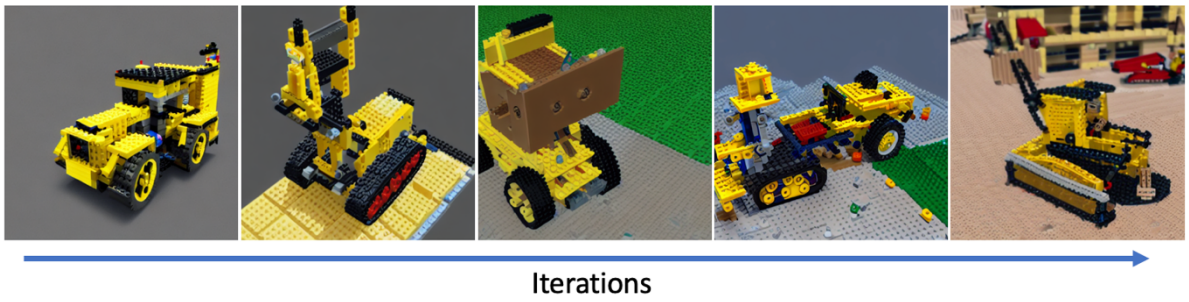

Figure 7 Results of textual inversion as during the training process. The result is generated with "A rendering of <token>, front view". It could be observed that the content changes during the training, making it hard to recognize the object. Only a high-level context of "Lego" is modeled.

The new token could then be utilized in the editing text prompt. Figure 8 shows the result after textual inversion. Compared with naively editing, the result preserves the original shape better. However, the degree of editing is still limited, which might be the result of overfitting during textual inversion. The inversion process could also fail, as in Figure 9.

Figure 8 Result of real editing after textual inversion. As observed in Figure 7, the new token could not capture the model as an instance. Therefore, the reconstruct result is not good. However, editing on cake, will change the texture. The color could not be modeled from the token, the "yellow" concept is included in the "<token>"

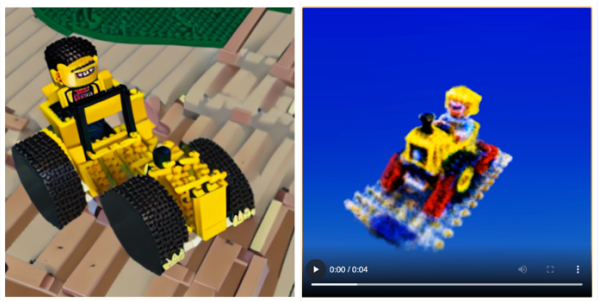

Figure 9 Failure case of textual inversion. The inversion become unstable and generate different object.

4.4 Editing with Finetuned Model

Instead of finding the text that fit the model representation, finetuning the diffusion model to fit the input prompt could customize the generator, such that the information of the origin object is preserved after editing. Figure 10 shows the result of generated 2D images by the finetuned model. The model learns to fit the origin image well. Editing in 2D could be performed by adding textual disruption to the input. Figure 10 (b) and Figure 10 (c) provides the editing result in 2D. However, the model might still overfit to the image set, as in the first row. Although the reconstruction result is good, editing can't be performed in a large scale, making the editing result similar to the origin NeRF Figure 10 (d). DreamBooth [11] proposed to utilize the class preservation loss, which encourages the diffusion model to generate diverse results. Here in Figure 10 row2, adding the result could provide larger editing scale, yet the reconstruction quality is compromised, which leads to a blurry nerf model.

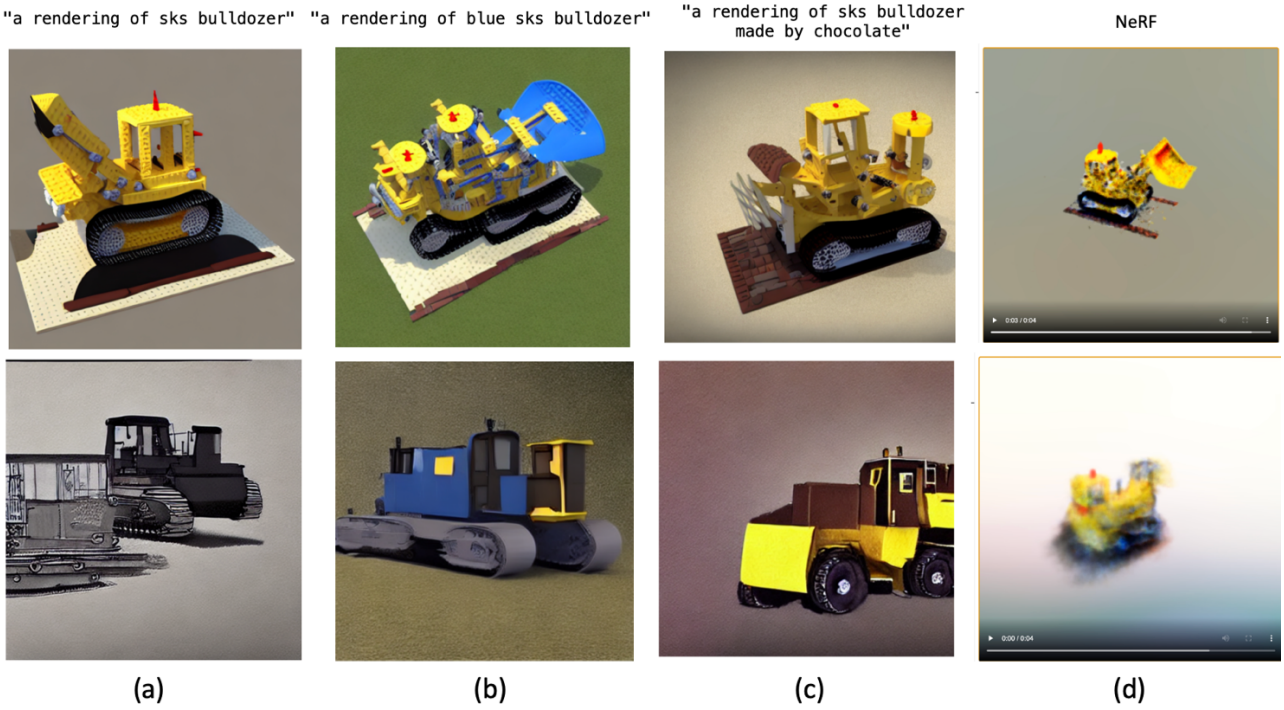

Figure 10 Result of generated 2D result after finetuning the diffusion model. The upper row is the one without class preservation loss. It captures the object attribute and reconstruct good result. However, the editing degree is low. The below row is the one with class preservation loss. Although the model is more diverse, it fails to capture the object.

4.5 Editing with Image Conditioned Model

From Section 4.1 to Section 4.4, inversion of the given object to the diffusion space or textual space is one major problem. The process of textual inversion may lead to uninterruptable result, and finetuning the diffusion model might compromise the generative ability. Additionally, the editing performance is highly dependent on the inversion process.

The problem is that during the generation process, as t increases, the generated result might deviate from the origin input latent. Instrtctpix2pix [14] adds latent embedding of the origin input to each timestep, such that the result is much more consistent with the input image.

Figure 11 Result of InstructPix2Pix under different conditions. The 2D images have desired changes but might not be 3D consistent.

Figure 11 shows the 2D results of using text instruction for the model. The model learns to preserve origin information well, while being to perform wanted editing. Although the result might not be multi-view consistent, we can utilize these images for NeRF Editing. Figure 12 shows the result of edited 3D NeRF.

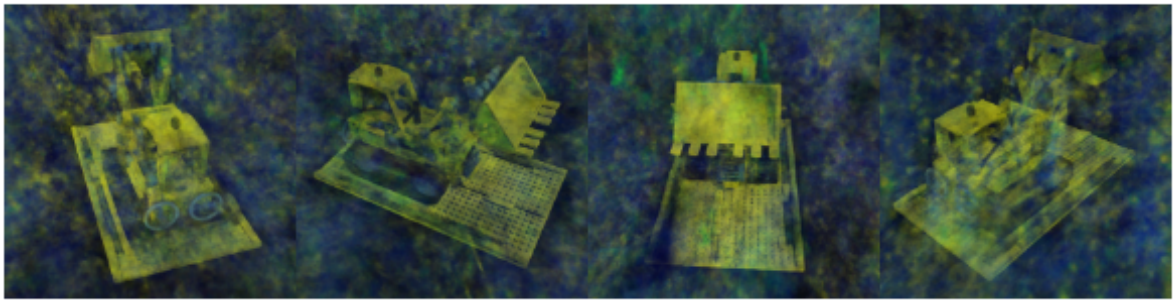

Figure 12 Result of edited 3D NeRF using caption "make it painted by van Gogh"

5 Implementation

5.1 Code

The implementation of this project is mainly based on the unofficial implementation of the Dreamfusion [22] framework. For image generation, our diffusion generator is based on StableDiffusion [9], which diffuses and denoise in the latent space instead of a trivial noise space. And the noise estimator is implemented using a U-Net structure, which suits the image inductive bias better.

To fasten the optimization approach, we utilize the instant NeRF (instant NGP [24]) as our NeRF representation. For the purpose of faster experiment, the resolution and capacity of NeRF is reduced. But the resolution could be extended for better quality.

5.2 Dataset

As our input is NeRF model, we could conduct experiments using existing NeRF models or optimize one using any set of multiview images. Here we mostly utilize the synthetic data set commonly used in the field of 3D neural rendering. We also utilized some generated result from Dreamfusion [22].

6 Project Planning

The aim of this thesis is to submit a paper to one of the top conferences.

6.1 GANTT Chart

Task	July	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Do the literature survey										
Preparing dataset										
Writing Code base										
Conducting Experiments										
Find issue and improve										
Paper Writing										

7 Required Hardware & Software

7.1 Hardware

Cluster with high-end GPU: Multi-card (8 is optimal) and each with RAM > 20G

7.2 Software

We mainly use python for our experiments on Linux clusters.

8 References

[1] Ho, J., Jain, A., & Abbeel, P.. (2020). Denoising Diffusion Probabilistic Models.

[2] I. J. Goodfellow κ.ά., 'Generative Adversarial Networks'. arXiv, 2014.

[3] Kingma, D., & Welling, M.. (2013). Auto-Encoding Variational Bayes.

[4] Shen, Y., Yang, C., Tang, X., & Zhou, B.. (2020). InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs.

[5] Valevski, D., Kalman, M., Matias, Y., & Leviathan, Y.. (2022). UniTune: Text-Driven Image Editing by Fine Tuning an Image Generation Model on a Single Image.

[6] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, και I. Kemelmacher-Shlizerman, 'StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation'. arXiv, 2021.

[7] E. R. Chan κ.ά., 'Efficient Geometry-aware 3D Generative Adversarial Networks'. arXiv, 2021.

[8] J. Song, C. Meng, and S. Ermon, 'Denoising Diffusion Implicit Models', CoRR, vol. abs/2010.02502, 2020.

[9] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, 'High-Resolution Image Synthesis with Latent Diffusion Models', arXiv [cs.CV]. 2021.

[10] R. Gal et al., 'An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion'. arXiv, 2022.

[11] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, 'DreamBooth: Fine Tuning Text-to-image Diffusion Models for Subject-Driven Generation', 2022.

[12] J. Ho and T. Salimans, 'Classifier-Free Diffusion Guidance', arXiv [cs.LG]. 2022.

[13] null-text inversion R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or, ‘Null-text Inversion for Editing Real Images using Guided Diffusion Models’, arXiv [cs.CV]. 2022.

[14] T. Brooks, A. Holynski, and A. A. Efros, ‘InstructPix2Pix: Learning to Follow Image Editing Instructions’, arXiv [cs.CV]. 2023.

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, και R. Ng, ‘NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis’. arXiv, 2020.

[16] Kobayashi, S., Matsumoto, E., & Sitzmann, V.. (2022). Decomposing NeRF for Editing via Feature Field Distillation.

[17] Yuan, Y.J., Sun, Y.T., Lai, Y.K., Ma, Y., Jia, R., & Gao, L.. (2022). NeRF-Editing: Geometry Editing of Neural Radiance Fields.

[18] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, και G. Wetzstein, ‘pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis’. arXiv, 2020.

[19] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, και G. Wetzstein, ‘Implicit Neural Representations with Periodic Activation Functions’. arXiv, 2020.

[20] J. Gu, L. Liu, P. Wang, και C. Theobalt, ‘StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image Synthesis’. arXiv, 2021.

[21] E. R. Chan κ.ά., ‘Efficient Geometry-aware 3D Generative Adversarial Networks’. arXiv, 2021.

[22] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, ‘DreamFusion: Text-to-3D using 2D Diffusion’, arXiv [cs.CV]. 2022.

[23] A. Radford et al., ‘Learning Transferable Visual Models From Natural Language Supervision’, arXiv [cs.CV]. 2021.

[24] T. Müller, A. Evans, C. Schied, and A. Keller, ‘Instant Neural Graphics Primitives with a Multiresolution Hash Encoding’, ACM Trans. Graph., vol. 41, no. 4, p. 102:1-102:15, Jul. 2022.