MSc thesis in Geomatics

3D building model edit with generative AI

Yingxin Feng

2024

MSc thesis in Geomatics

3D building model edit with generative AI

Yingxin Feng

June 2024

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Nail Ibrahimli
	Dr. Ken Arroyo Ohori
Co-reader:	Dr. Liangliang Nan

Abstract

Generative AI is developing rapidly and has shown great potential in generating and editing images with text prompts. It has achieved partial success in the challenging 3D model edits of common objects. However, there is a lack of attention in the building domain, which already faces certain limits in the 2D space. 3D building models automatic edit has wide application potential, especially for renovation, concept comparison, and large-scale scenes. Therefore, the thesis aims to edit the building models effectively through texture creation and geometric changes with generative AI by evaluating the existing 3D edit pipelines and putting forward practical solutions based on them.

The thesis first examines the current methods. Latent-paint, Text2Tex and X-Mesh are chosen as the representative existing explicit representation (mesh) based pipelines for their relatively satisfying performance in common objects and the different ideas contained. Modifications are made based on them to improve output quality from the aspects of the control module and edit module. For the first aspect, attempts include adding the view specification text prompt to the original X-Mesh and using the image as the control for X-Mesh. Image control X-Mesh is also further experimented with placing higher attention on the input image view and editing the key semantic parts of the building separately. For the latter aspect, modifications include freezing sampled vertices geometry and combining X-Mesh and Text2Tex.

Results show that Latent-Paint mainly creates texture with only major colour and lacks details. Text2Tex generally generates fine textures for inputs with richer depth information. X-Mesh creates textures and edits geometry jointly. It can improve the fidelity of the input mesh to some extent but suffers from noise problems. Image control X-Mesh generates more realistic results and outperforms the text control pipeline. Combining Text2Tex with X-Mesh takes advantage of Text2Tex, which can generate smoother and more realistic textures compared to X-Mesh in some cases. The combination is especially recommended for low-fidelity mesh of small or medium size. The user study also shows that the combination of X-Mesh and Text2Tex generates the most favoured results while image control X-Mesh ranks second. There are still limitations in geometric deformation scope, fidelity, generalizability and computational efficiency in the proposed 3D edit pipelines. The thesis succeeds in proving the application potential of 3D edit pipelines in the building domain and obtaining high-quality results by modifying existing methods.

Acknowledgements

I want to express my sincere gratitude to my mentors Nail and Ken. You both gave me many valuable suggestions throughout the process, encouraged me and helped me figure out new possibilities when I became pessimistic about the current results. To Nail, you introduced me to the exciting generative AI world and shared many useful resources and tips. You helped me out when I ran into confusing bugs with my codes and provided very detailed comments for the thesis. To Ken, you engaged a lot in this topic and were always ready to help. You offer ideas from broader aspects and structured comments for the thesis. And thanks to Liangliang for the in-depth questions and suggestions from new angles, and Azarakhsh for being supportive in my graduation process.

I am grateful to my parents for supporting me throughout the journey. Though far apart, you still share the milestones in my study. And to my friends, I would say that you share my delight and stress and keep me positive towards the thesis and beyond. With you, I have had many interesting and meaningful moments during the challenging study process. Finally thanks to all the people who helped me do the user study.

Contents

List of Figures

List of Tables

Acronyms

1. Introduction

1.1. Motivation

Generative AI is based on deep learning and can produce new content following user demand [Bandi et al., 2023]. Such models have been developed rapidly recently and have proved to be applicable in different types of content including text, audio, image, 3D model and video, and various domains, like translation and writing, design and gaming [Gozalo-Brizuela and Garrido-Merchán, 2023]. Successful AI models can automatically generate diverse and realistic results under easy-to-use instructions, though at the cost of relatively high computation cost. The 3D building model edit is a future field for the application of generative AI models [Haque et al., 2023].

In the 2D domain, studies in image generative AI models have achieved satisfying results (see Figure 1.1) and attracted attention from beyond the academic community. Popular image-generation models like text-to-image Stable Diffusion [Rombach et al., 2021] can generate high-quality images for general objects in either realistic or artistic styles and image-edit models like InstructPix2Pix [Brooks et al., 2022] can perform identity-preserved edits for common scenes.

Figure 1.1.: Successful image generation and edit samples: Stable Diffusion can generate realistic images of common objects and InstructPix2Pix can add simple elements and preserve the identity of the scene

There is also fast-emerging research focusing on applying generative AI in the 3D domain. Edit of 3D models with text prompts is a popular branch for its wide application potential and low requirement for prior experiences. Possible application cases include editing figures with different postures and appearances for animation and creating multiple slightly different scenes by editing objects inside for gaming. Despite its broad application prospects, the generative edit in 3D space is a tough task due to the limitation of available 3D datasets,

computation resources and the complex nature of 3D models. It is even more difficult compared to generation tasks as it is required to preserve the identity of the input [Koo et al., 2023]. Ongoing research is striving to improve the quality of 3D model editing.

The majority of existing 3D edit pipelines are based on the 2D pre-trained models to alleviate the problems caused by the lack of data and computational resources. Though papers have claimed their pipelines can successfully deal with common objects, the displayed samples are usually limited to certain types like figures and daily necessities. Many cases only display partial views, with part of the objects edited instead of the entire ones from 360 degrees. Some pipelines focus more on editing scenes and pay less attention to the geometry [Haque et al., 2023; Song et al., 2023; Dong and Wang, 2023]. In general, existing 3D editing pipelines can only handle objects or scenes within a certain scope and have limited geometric editing capabilities.

Editing 3D building models manually is time-consuming and requires domain knowledge and certain experiences. Thus, dealing with this task automatically with generative AI can improve efficiency and have rich application scenarios. Compared to 3D building generation, model edits can maintain the basic structure of input model geometry. Thus it is especially useful for comparing different design styles for the same region or renovation cases. For example, it can be used in fast concept comparison in the early design stage. It can also be applied in large-scale urban planning, where the structure of the original building blocks needs to be preserved.

However, there is a lack of 3D edit trials in the building domain. Buildings are complex and unique and thus more challenging for editing. The precise geometry is also more significant for building models. It leads to doubts about the achievable quality of building edits with generative AI using the existing pipelines. As most 3D edit pipelines are based on 2D pre-trained models, their limitations are inevitably inherited. There are limitations in the training dataset of 2D pre-trained models in the building domain, especially for InstructPix2Pix. Many building training samples have limited identity-preserving level or geometric deformation scope as shown in Figure 1.2. It makes it more difficult to generate or edit specific photorealistic building images as the desired detailed instructions with existing pre-trained models, especially for complex or connected buildings. Pre-trained models have biases towards building types and objects with more samples in the training dataset, such as landmark buildings. They may gain better generation effects. Furthermore, pre-trained models tend to generate images viewing from the front side (even with prompts like back or top views) [Armandpour et al., 2023]. It adds to the challenges of dealing with buildings in the 3D space, which usually require information from all other directions (e.g. back, side and top) to gain the complete model.

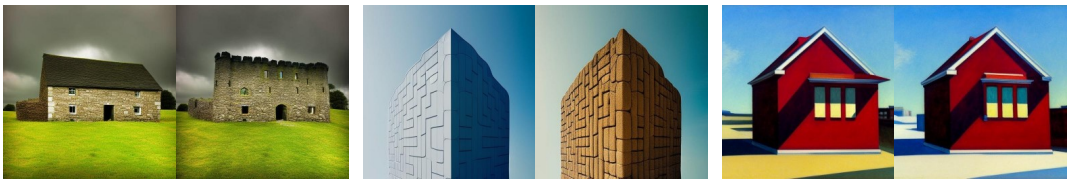


Figure 1.2.: Building samples in InstructPix2Pix dataset [Brooks et al., 2022]: The first one does not preserve part of the original features, the second one only changes the building material and the last one only changes the surroundings instead of the building.

Therefore, the thesis aims to contribute to applying 3D edit pipelines with generative AI methods in the building domain to attain higher quality models. The editing process can include creating the texture and changing the geometry. Existing state-of-the-art 3D edit pipelines are evaluated with diverse building samples and prompts and innovative modifications based on them are experimented with. Open-source editable building model datasets captured from the real world, which usually do not contain too many details, are chosen as the major experimental targets and editing automatically with generative AI methods is expected to widen their usage scopes. Proper user scenarios for different pipelines based on the features of the input models are also discussed for practical solutions in this domain.

1.2. Research objective

1.2.1. Research question

The goal of the thesis is to evaluate the existing generated AI based 3D edit methods for the application in the general building model domain and develop practical 3D edit pipelines for this domain. To achieve it, the following sub-questions need to be answered:

- What existing pipelines are promising in the building model edit field?
- How do these chosen pipelines perform in different building cases?
- How to develop a new pipeline or modify existing ones to make the edit results better comply with user guidance and have higher fidelity?
- What are the user scenarios and limits of the existing and newly proposed edit pipelines?

1.2.2. Scope

The thesis focuses on exploring the potential of generative AI methods in building model edits. The experimental pipelines include the chosen promising existing ones and the modified versions based on them. The pipelines are based on pre-trained 2D image generative models. Instead of fundamental structural changes, the pipeline development is mainly done by modifying the existing pipelines and limited to better fit the building cases and user instructions, as well as increasing the quality of the output. Both qualitative and quantitative evaluations are conducted on the pipelines to figure out their advantages and limitations.

The input for pipelines is different types of untextured building mesh from open-source datasets. The major 3D representation type is mesh with small triangles as faces. The types of user control include text and image prompts. The output is building mesh with more realistic details according to the control condition. To gain the result, part of the pipelines only create texture while others create texture and edit geometry at the same time.

1.3. Thesis outline

Chapter 2 reviews studies on methods of 3D representation, generative AI for images and 3D models, which are mainly divided into implicit and explicit 3D representation based methods. Chapter 3 introduces the methodology. The failure cases for implicit representation based methods are first summarized (details can be found in the Appendix B). Then the chosen representative 3D edit pipelines are introduced and possible modifications are discussed. Chapter 4 displays the results. Firstly, the implementation details and guidance engineering are documented. Secondly, qualitative and quantitative evaluations of the existing pipelines and modified versions are conducted. Chapter 5 concludes the thesis. The application scenarios for outstanding pipelines are proposed based on their advantages and limitations. Contributions, limitations and suggestions for future work are pointed out.

2. Related work

In this chapter, multiple types of 3D representation are first discussed in Section 2.1. Then a general introduction to generative AI in images is given in Section 2.2, which serves as a base for a majority of generative AI in 3D. At last methods for both explicit and implicit representation-based 3D generative AI methods are studied in Section 2.3.

2.1. 3D representation

2.1.1. Implicit representation

Implicit ways can represent 3D models with various types of functions or neural networks [Li et al., 2023a]. Neural networks show a promising future for their increasing quality and ability to directly reconstruct 3D models from multi-view images, and it is also easier to preserve additional information like lighting conditions [Mildenhall et al., 2020; Yariv et al., 2020; Wang et al., 2021]. However, in implicit methods, both the geometry and texture information of 3D models are deduced from images. Thus they are closely connected and influenced mutually [Wang et al., 2021], which adds to the difficulty of editing such models. Furthermore, the complex nature of building models, such as self-occlusion, tiny details and glass surface, also makes it harder to gain accurate geometry.

There are two major ideas for implicit methods, namely surface representation and volumetric rendering. Surface representation only records the exterior layer of the shape while volumetric rendering treats the target as a filled shape with different features like density and colour in each region. The Signed Distance Function (SDF) that can define the signed distances between points and the surface is a base for the former one. Surface representation is good at extracting high-quality surfaces. It focuses on representing objects and masks are needed as input to separate the target object and the background. One successful example following this idea is the Implicit Differentiable Renderer (IDR) [Yariv et al., 2020]. The volumetric idea performs better at dealing with abrupt depth changes. Masks are not required and it can also deal with scenes. NeRF is a representative [Mildenhall et al., 2020].

Neural Implicit Surfaces (NeuS) applies the volume rendering approach to learn SDF for surface representation and is a successful attempt to combine the advantages of surface and volume representations [Wang et al., 2021]. Similar to NeRF, a neural network is constructed to represent each individual 3D model and the colour of each pixel of the multi-view images is used as ground truth to train the network. The rendered pixel colour from the network is calculated by accumulating the colour of sampled points along the ray of the corresponding camera view at that pixel (see Figure 2.1). The adaptation is that instead of merely using the radiance density as weight, the signed distance and an unbiased and occlusion-aware weight function are combined as the new weight. In this way, the SDF is integrated into the neural network [Wang et al., 2021]. NeuS achieves better geometric results compared to

the original NeRF (also called Vanilla NeRF) [Mildenhall et al., 2020] with more detailed and less noisy surfaces. To improve the efficiency, multi-resolution hash encoding, fully fused networks, and occupancy grid pruning and rendering techniques are used for acceleration [Müller et al., 2022].

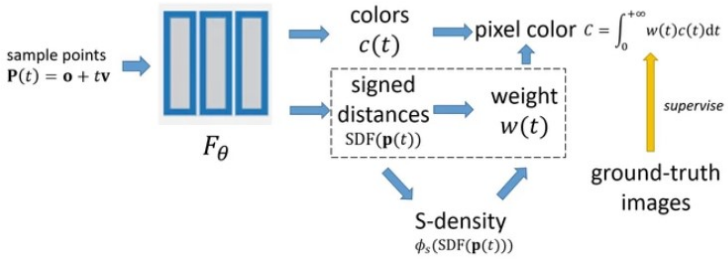


Figure 2.1.: NeuS model [Wang et al., 2021]: utilize both volume rendering and surface representation to gain better geometric results

2.1.2. Explicit representation

Explicit representation is still a popular way to represent 3D models, especially in the field of the built environment. They can naturally represent both the geometry and texture details of target objects faithfully. Besides, multiple manipulations, such as combinations of several models, and modifications of size and direction, are relatively easy, which extends the usage scenarios. However, explicit representation also suffers from the limit of memory size and detailed level, as the file size usually grows rapidly when the fidelity of 3D models increases.

The traditional explicit methods include point cloud, voxel and mesh. Point cloud uses a disordered set of discrete samples to represent surfaces of objects, containing rich information like position, colour and normal, to represent 3D objects. It can be directly obtained from sensors or sampled from other forms of representations, but its irregular nature limits its usage to some extent. Voxel represents 3D models with a regularly spaced three-dimensional grid and multiple features can be stored. The data is well-structured but not efficient, as both empty and occupied parts are recorded. Therefore, it is hard to store high-fidelity models with voxels in limited memory space [Li et al., 2023a]. Attempts are also made to increase the storage efficiency and flexibility. A successful example is the DMTet method, which represents 3D models with the deformable tetrahedral grid and differentiable Marching Tetrahedral layer [Shen et al., 2021].

Mesh represents 3D objects using a set of faces with the connectivity relationship and corresponding vertices. As only the surface information is stored, it is more compact than the voxel representation. It also contains topological information on surfaces and is organized in a structured way, making it easier to process compared to the point cloud. Additionally, mesh is a popular choice of explicit representation in 3D content generation and edit [Michel et al., 2022; Mohammad Khalid et al., 2022; Metzer et al., 2023; Chen et al., 2023a; Ma et al., 2023] and also a widely applied representation in the architectural field [Li et al., 2023a]. Therefore, mesh is chosen as the major mode of 3D explicit representation in the thesis.

2.2. Generative AI in image

The Diffusion Model is a typical generative AI method used in 2D and 3D content generation and edit [Chao and Gingold, 2023]. It is easier to scale and train and can generate diverse and high-quality results [Dhariwal and Nichol, 2021]. Thus it is further discussed in Section 2.2.2 and chosen as the major methodology. CLIP is also explained in detail in Section 2.2.1 as it is the backbone for some popular 2D Diffusion Models and can be utilized independently in generative content.

2.2.1. CLIP

Trained on a large-scale text-image paired dataset, CLIP (see Figure 2.2) is a popular pre-trained model in image-based deep learning. A text encoder and an image encoder are jointly trained to match the correct text description and image pair. CLIP can predict the closest text label from the given image and generate the possibly matching image from the text instruction. In addition, it can encode the input text or image to a compact vector while preserving the most important details, which serves as a useful backbone for loss computation for other pipelines [Radford et al., 2021].

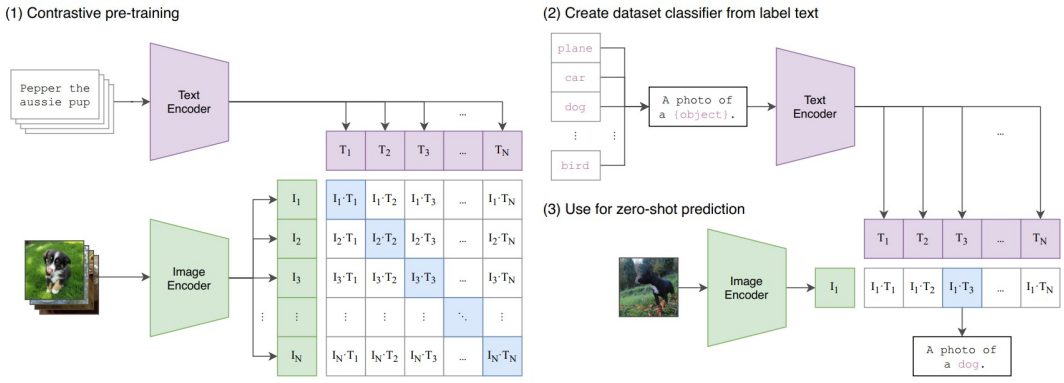


Figure 2.2.: Training and inference process of the CLIP Model [Radford et al., 2021]: connect text and image through encoders.

2.2.2. Diffusion model

The general idea of the Diffusion Model (see Figure 2.3) is to reverse the process of transforming an image to noise by adding Gaussian noise in each time step, and conditions can be added in the reversion process to generate target results [Po et al., 2023]. In the conditioned Diffusion Model, a neural network is trained to minimise the difference between the predicted noise and the actual noise added (sampled noise). The training datasets are images with conditions (e.g. text descriptions of images). During the inference stage, conditions, time step and noise (or noisy image) are inputs, and the model subtracts the noise added and outputs the desired image.

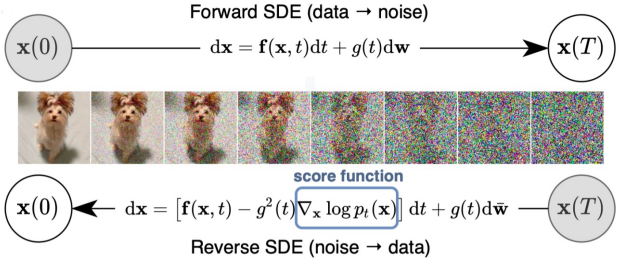


Figure 2.3.: General process of the Diffusion Model [Po et al., 2023]: add Gaussian noise at each time step in the training stage and reverse the process in the inference stage

There are some outstanding image generation and edit models that utilize the Diffusion Model idea. They can be used as bases for 3D generative methods and below is a brief introduction to them. The classical Stable Diffusion Model is a popular text-to-image generation model with the text prompt as the condition. To reduce the trainable parameter size, it uses the latent space to represent the original pixels, which are encoded to and decoded from the latent space at the beginning and the end respectively [Rombach et al., 2021]. The text encoder of the pre-trained CLIP is also used to bridge the text to the target image [Radford et al., 2021]. The modified DeepFloyd IF is another popular text-to-image generation model, which uses pixel-based and triple-cascaded techniques [Saharia et al., 2022]. It can generate high-quality images at the cost of higher computational resources.

There are also variations on the basic text-to-image Diffusion Model. Depth-to-image Stable Diffusion Model infers the depth map from the input image using MiDaS and generates images following the depth information [Rombach et al., 2021]. DreamBooth put forward the potential to finetune the text-to-image Diffusion Model efficiently with a small set of images of a subject. Leveraging the learned semantic embedding of the original model, DreamBooth can then generate personalized images with desired postures and backgrounds [Ruiz et al., 2023]. Zero-1-to-3 uses a large-scale view-aware 3D dataset to train upon a pre-trained Diffusion Model to learn to deduce novel views of the object from one input image. Using the object image, text prompt and camera parameters as input, the model can generate a consistent image from any specified view [Liu et al., 2023]. Stable Zero123 is an improved version of Zero-1-to-3 with advanced datasets of high-quality 3D objects [Stabilityai, 2023].

As for the image edit, InstructPix2Pix is a popular text-guided image editing model. It is based on Stable Diffusion and uses both the original image and the text prompt as conditions. Its loss function consists of three parts, the classifier-free part to maintain diversity, the image-conditioned part and the text-conditioned part. In inference time, the weights of the two instructions (image CFG and text CFG) can be adjusted to control the similarity with the original image and the consistency with the text [Brooks et al., 2022]. InstructEdit is another text-guided image editing model and claims to perform well in preserving the original details in the unchanged part. It utilizes a language processor to pre-process the prompt, a segmenter to identify the target editing part and a mask-based image editor from DiffEdit (Diffusion-based Semantic Image Editing). Its main idea is to segment the image according to the prompt and only edit the region that needs to be changed to preserve the original information of the other parts [Wang et al., 2023a]. SDEdit (Stochastic Differential Editing) enables stroke-based image editing and image composition. It adds noise partially to the original image and then denoises it to get the output that strikes a balance between

realism and faithfulness to the input [Meng et al., 2021]. Later on, this trick is also combined with Stable Diffusion for creating realistic images from sketches or improving the quality of blurry images toward the text-instructed direction [Rombach et al., 2021].

2.3. Generative AI in 3D

To train a high-quality generative AI model, large datasets are needed. However, compared to 2D images, 3D model datasets are relatively scarce [Poole et al., 2022]. Additionally, the 3D generative model contains more parameters, which greatly increases the computation cost. Therefore, it is a natural idea to utilize 2D text-guided image generative models to guide 3D content generation and editing. In this section, the focus is mainly on methods using text as guidance due to its easy-to-use nature and the high possibility of adaptation of such pipelines. Both implicit and explicit representation based methods are discussed. The former usually changes the texture and geometry jointly, while the latter has the freedom to modify only one part.

2.3.1. Implicit representation based methods

In the field of editing 3D models in implicit representation, a straightforward idea is to edit the base multi-view images dataset and then train the 3D model based on the updated dataset. Instruct-nerf2nerf (see Figure 2.4) is a representative one in this direction, which applies the iterative dataset update strategy. It represents the 3D model with NeRF and chooses InstructPix2Pix to edit the rendered images with the text prompt and original images as conditions at each iteration [Haque et al., 2023].

Another more elegant way is to directly backpropagate the image loss to the 3D implicit neural model to guide 3D content generation or editing. Different methods of calculating image loss are proposed by researchers. Dreamfusion (see Figure 2.5) which focuses on the text-guided generation of 3D content, puts forward the concept of SDS loss. Dreamfusion proves that the image loss from the frozen 2D Diffusion Model can be backpropagated to the 3D model to modify the geometry and colour towards the desired direction. SDS loss is calculated by matching denoising scores between the sampled noise added to the rendered image and the predicted noise by the pre-trained 2D Diffusion Model. Using SDS loss requires maintaining the gradient flow, so both the 2D Diffusion Model and the 3D representation should be differentiable [Poole et al., 2022]. Instruct 3D-to-3D also uses the SDS loss to guide 3D model editing with the pre-trained InstructPix2Pix model [Kamata et al., 2023]. ProlificDreamer argues that Variational Score Distillation (VSD) loss, which has a similar format as SDS loss but adds additional camera parameters to the condition embeddings of the network, is a more general version of SDS loss and has better performance [Wang et al., 2023b]. However, introducing camera parameters also poses more requirements to the 2D pre-trained model used for loss calculation. VSD loss is also more suitable for 3D generation or editing of the complete object, as the same set of pre-defined camera poses covering 360 degrees can be used.

To better preserve the feature of original objects while changing towards the desired direction, an optimized loss term Posterior Distillation Sampling (PDS) is put forward. It matches the stochastic latents of the source and the target so that the samples are moved towards

the boundary of the marginal of the source to be distributed more evenly [Huberman-Spiegelglas et al., 2023; Wu and De la Torre, 2023]. Utilizing the PDS loss can gain better source-aligned results and extend the scope of effective text instructions [Koo et al., 2023].

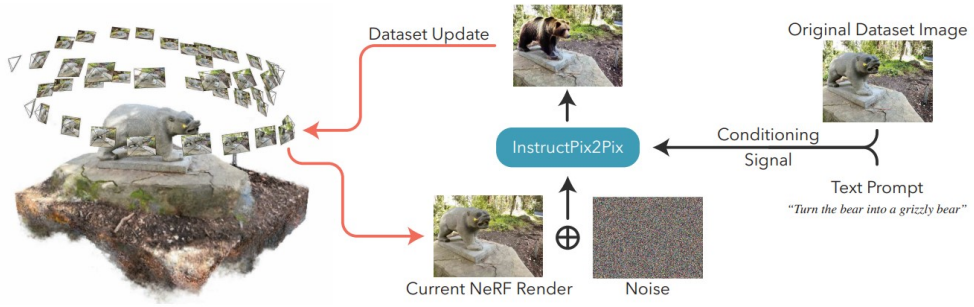


Figure 2.4.: Overview of Instruct-nerf2nerf [Haque et al., 2023]: edit NeRF with the iterative dataset update strategy based on InstructPix2Pix.

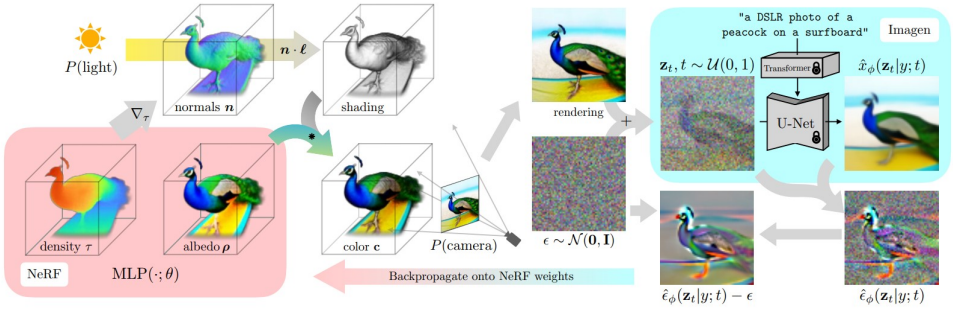


Figure 2.5.: Overview of Dreamfusion [Poole et al., 2022]: backpropagate 2D SDS loss to 3D model.

How to ensure multi-view consistency is an important topic in 3D edit as most of the pipelines are based on editing 2D images which can not guarantee multi-view consistency. Possible techniques include changing the loss function to regularize different views [Hong et al., 2023; Armandpour et al., 2023], adding additional pre-trained modules to incorporate camera parameters [Li et al., 2023b], directly modifying 2D Diffusion model to add camera parameters [Shi et al., 2023] or additional loss terms to gain multi-view consistent images [He et al., 2023], and generating a more consistent edited images dataset with multiple pre-processing steps [Dong and Wang, 2023; Fang et al., 2023].

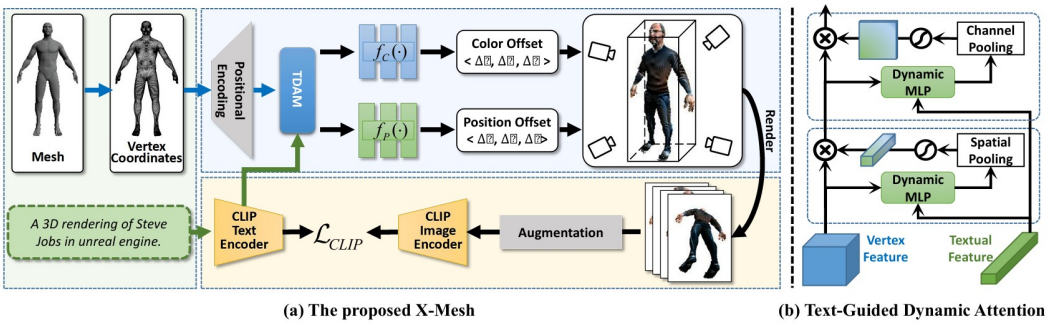
How to improve computation efficiency is another one as many methods require large GPU memory size and long computational time. Researches show that updating the whole multi-view images dataset for 3D neural representation in a single pass with the adjusted 2D Diffusion Model [Song et al., 2023], editing 3D model in latent space [Chen et al., 2023b], and using first low-fidelity and then high-fidelity 3D representations in training stages [Wang et al., 2023b] are possible techniques.

2.3.2. Explicit representation based methods

There are two major types of mesh-based 3D content edit methods. The first type treats the input mesh as an editable object and modifies the geometry and texture jointly. The second type keeps the original geometry of the mesh unchanged and focuses only on creating the proper texture.

In the category of joint edit, Text2Mesh is a representative early work. A neural style field network is trained using CLIP-based semantic loss with text prompt as the condition. The network takes vertices coordinates as input and outputs colour and positional displacement. The CLIP loss is computed by the similarity of the encoded averaging image rendered and augmented results from different views and the encoded text prompt. It can add geometric and textural details to the low-fidelity mesh [Michel et al., 2022]. CLIP-Mesh, which is initially developed for generation tasks but can also be used for 3D joint edit, trains a differentiable renderer instead. The module uses vertices, the texture and the normal map as input and generates rendered images. The loss includes the CLIP loss, Diffusion Model loss and the Laplacian regularised terms for intact geometry. It has the potential to generate more consistent texture but suffers from the limit of resolution as the differentiable renderer has higher memory demand [Mohammad Khalid et al., 2022].

X-Mesh (see Figure 2.6) can be seen as an improved version of Text2Mesh. It introduces a Text-guided Dynamic Attention Module (TDAM) to fully utilize the textual features, which helps generate higher-quality meshes and speed up convergence. The module incorporates spatial and channel attention layers to dynamically activate the vertices corresponding with the text instruction [Ma et al., 2023].



(a) The proposed X-Mesh (b) Text-Guided Dynamic Attention

Figure 2.6.: Overview of X-Mesh [Ma et al., 2023]: use CLIP loss make geometric changes and add texture to meshes with text prompts.

As for papers about texturing, TANGO is an early attempt. It aims to generate photorealistic texture for bare mesh. Similar to Text2Mesh, the loss is also computed by the similarity between the encoded text prompt and the encoded rendered images with CLIP. The difference lies in the training networks used. Here spatially varying Bidirectional Reflectance Distribution Function (BRDF) and Normal network are constructed for training. The modules predict BRDF parameters and normal variation with the given ray and the position and normal of surface points as input [Chen et al., 2022].

Latent-paint (see Figure 2.7) deals with this task in a different approach. SDS loss is calculated based on Stable Diffusion. The training process happens in the latent space to improve

efficiency and at last, the result is decoded for higher resolution. In each iteration, the texture map is attached to the input mesh through UV parameterization and then images are rendered for loss calculation [Metzer et al., 2023]. The appearance modelling stage of Fantasia3D can also be used for texture creation. It uses Stable Diffusion based SDS loss to guide the training of the BRDF, which takes in the diffuse, specular and tangent space normal to predict colour [Chen et al., 2023c].

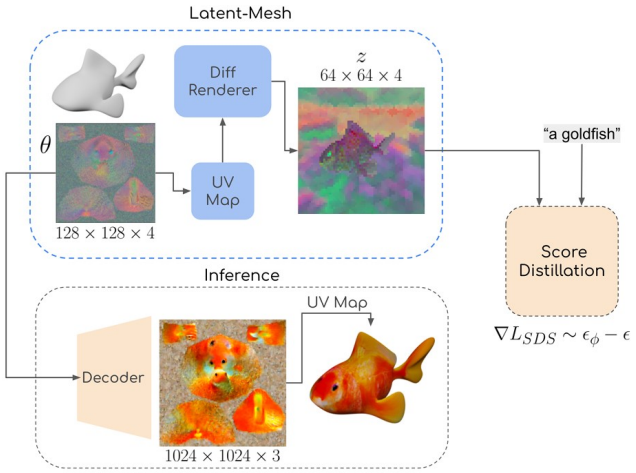


Figure 2.7.: Overview of Latent-Paint [Metzer et al., 2023]: use text-to-image Stable Diffusion based SDS loss to generate texture for meshes.

TEXTure and Text2Tex (see Figure 2.8) are two similar pipelines that utilize the text guided depth-to-image Diffusion Model to generate the texture map for mesh. In each iteration, both pipelines render the image from the mesh to provide depth and existing colour information from the chosen viewpoint. A mask highlighting the regions that require texture generation and update is also created. They are passed to the pre-trained model to get the texture and then projected to the input mesh. Text2Tex puts forward an additional refinement stage that updates texture from multiple new viewpoints to reduce the artifacts in the texture [Chen et al., 2023a]. TEXTure allows texture transfer with a given coloured mesh or a set of images of the target object [Richardson et al., 2023].

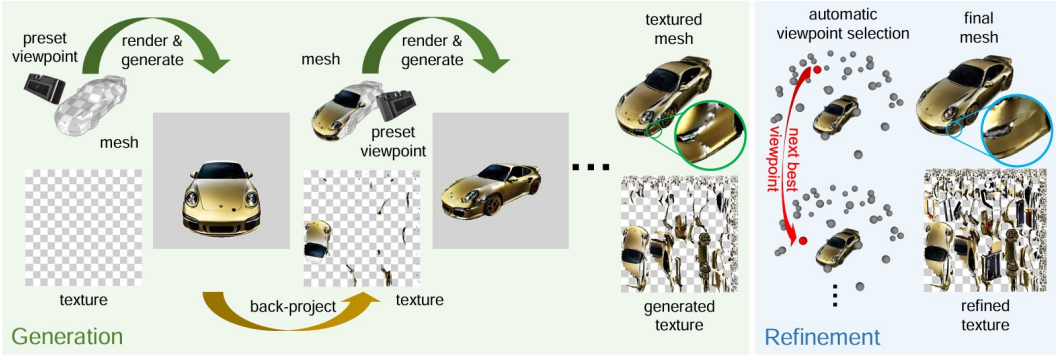


Figure 2.8.: Overview of Text2Tex [Chen et al., 2023a]: generate and refine proper texture with the region defining mask and depth-to-image Stable Diffusion

Paint-it is the latest related research for mesh based texture generation and claims to produce the most realistic and detailed texture compared to existing works. It trains the Deep Convolutional Physically-Based Rendering (DC-PBR) Map by the text-to-image Diffusion Model based SDS loss calculated from the text prompt and rendered multi-view images of the input mesh [Youwang et al., 2023]. Similar to TANGO and Fantasia3D, Paint-it also uses renderer-based training techniques and has high computational demands.

3. Methodology

3.1. Overview

To attain the desired building model edit results and propose the proper edit method for different buildings, several steps are followed. The first one is to choose the proper input and 3D representation method. The second one is to explore the corresponding edit pipelines and conduct experiments on different samples to figure out their advantages and limitations. The third one is to develop practical solutions by modifying the pipelines to alleviate the limitations. Finally, quantitative evaluations are made.

An intuitive idea is to input multiple-view images of buildings (see Figure 3.1), as they are easy to obtain for both buildings constructed in the real world and manually modelled ones and contain abundant details. The implicit 3D representation NeuS based on neural networks is a reasonable choice. It can potentially be connected directly with the deep-learning-based generative AI models. It can also be directly attained from 3D reconstruction of the images, and preserve implicit geometric details regardless of the resolution limit.

The chosen and modified 3D edit pipelines for implicit 3D representation use text prompt as control and can be divided into two categories, namely updating image datasets and incorporating 2D image loss to 3D model. Pre-trained high-performance 2D image models are used as foundations in these methods due to the lack of 3D generative models for implicit representations and the limitation of computational resources. For the first type, images in the dataset are updated by the newly edited (and denoised) ones and used to train NeuS in each iteration. Pre-trained 2D models including InstructPix2Pix, InstructEdit and Dreambooth (a personalized finetuned model with a small set of target images based on text-to-image Stable Diffusion) are tested. The view selection procedure and the denoising module with SDEdit are also experimented with. For the second type, the 2D image loss is calculated and directly backpropagated to NeuS to guide its training. The tested loss terms are SDS loss based on InstructPix2Pix or Stable-Zero123 and PDS loss based on text-to-image Stable Diffusion.

However, though successful cases can be obtained from simple objects like the chair and skull, the implicit representation based direction fails when attempting to edit high-fidelity building models (see Figure 3.2). In the training process, severe view inconsistency problems exist and noise is incorporated into the NeuS model. The final model can not preserve the original features and generate blurry results. One reason is that the pre-trained 2D models have difficulties in dealing with high-resolution complex buildings and generate inconsistent results for different views. Another reason is that the two key features of 3D models, colour and geometry, are interconnected in implicit representation and thus it is challenging to control. As the edited results of different views are sometimes conflicting, 3D models are guided to change towards different directions, resulting in blurry colour, and rough and incomplete geometry. The detailed methodology, results and analysis can be found in Appendix B.

As the implicit representation-based 3D edit pipelines can not meet expectations, the thesis turns to focus on explicit representation based methods for building model edits. Mesh is chosen as the 3D representation method for its compactness, well-structured nature and wide application. The input mesh is uncoloured and generally contains limited details as shown in Figure 3.1, which has the shared feature of most open-source building model data captured from the real world. Thus, the edit pipelines are required to add texture and if possible, make geometric changes.

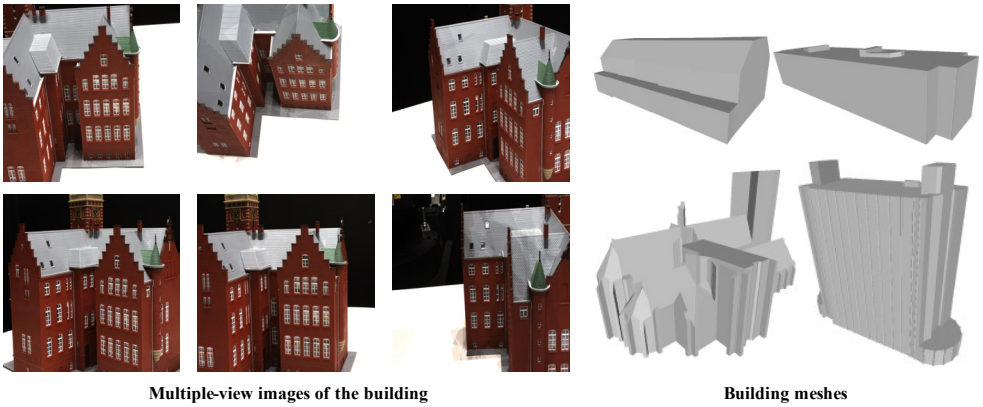


Figure 3.1.: Input building data samples: The left building sample is used for implicit representation based methods and is high in fidelity. The right group is for explicit methods. The meshes are uncoloured and contain limited details.

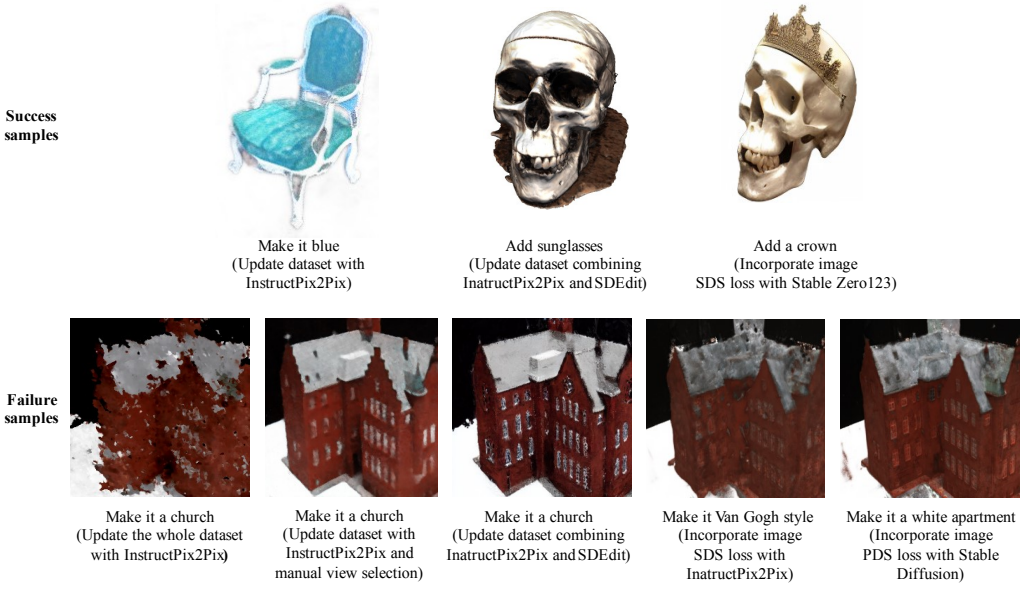


Figure 3.2.: Representative results of implicit representation based methods: Pipelines succeed in simple samples of common objects and small-scale edits. But they fail in buildings and have blurry results.

A chronological study is then conducted on the existing mesh edit papers. Pipelines that adopt different ideas and outperform others in their specific directions, as well as having reasonable computational requirements, are chosen as representatives for further experiments:

- Latent-Paint: The pipeline only creates the texture of mesh and uses text-to-image Stable Diffusion as the pre-trained model. The trainable object is the latent texture image. UV parameterization is used to project the texture on the mesh and the rendered feature map is attained by a differential renderer for loss calculation. The chosen loss term is SDS. After training, the latent texture image is decoded to a high-resolution texture image and mapped to the mesh [Metzer et al., 2023].
- Text2Tex: It also only creates texture and uses the adapted depth-to-image Stable Diffusion guided by the text prompt as the pre-trained model. There is no explicit trainable object. In each iteration of the generation stage, the image of the mesh from the chosen viewpoint is rendered. A mask that identifies the regions for new, updating, keeping and ignoring is also created by viewpoint information and previously rendered results, which is back-projected onto the mesh. The Diffusion model creates texture for the new region and updates the texture with a moderate denoising strength in the updating region. In each iteration of the refinement stage, the new view with a high updating region percentage is chosen and the texture is mildly modified to reduce artifacts [Chen et al., 2023a].
- X-Mesh: The pipeline both edits the geometry and creates the texture of the mesh. The CLIP model is used with text as instruction for loss calculation. Three trainable networks are constructed, including the TDAM module, the Multilayer Perceptron (MLP) for colour offset and the one for position offset. The vertex coordinates are positionally encoded as the input for TDAM. TDAM processes vertices with text control using dynamic linear layers, as well as channel and spatial attention to obtain the features that are closely correlated with the text prompt. The output is then passed through the colour and position MLP to get the corresponding offsets compared to the original mesh. The resulting mesh is rendered from different views by a differentiable render. The rendered images are augmented and used to compute the average CLIP loss across views. The above process is repeated using a grey mesh (regardless of the colour offset) to enhance geometry and the two losses are added together and backpropagated to the networks [Ma et al., 2023].

Name	Latent-Paint	Text2Tex	X-Mesh
Texture creation	yes	yes	yes
Geometry edit	no	no	yes
Trainable object	latent texture map	none (directly back-project to mesh)	TDAM, colour and geometry MLP
Loss term	SDS loss	none	CLIP loss
Pre-trained model	text-to-image Stable Diffusion	depth-to-image Stable Diffusion	CLIP
Special techniques	trained in latent space	generation and refinement stages, regions division mask	TDAM module, geometry enhancement

Table 3.1.: Comparison of chosen pipelines

Building samples with representative features are chosen for experiments. The samples include various building types and different combinations of sizes and fidelity levels. The results of the existing pipelines are then evaluated qualitatively from the following angles.

- Reality: how natural the output is.
- Consistency: how consistent the different views of the output are.
- Fidelity: which detail level the output has.
- Loyalty to the original model: to what extent the output keeps the identity of the input.
- Similarity to the control: to what extent the output is aligned with the control (the text or image prompt).
- Generalizability: how robust the pipeline is towards different samples.

Based on the potential and problems summarized, modifications are made to better fit the user control and increase the quality of the output mesh. Modifications (see Figure 3.3) include changing the user control module (see Section 3.2) and the edit modules (see Section 3.3). The results are also evaluated qualitatively from various angles.

Quantitative evaluations are conducted finally on the results of both existing and modified pipelines. Comparing the average score calculated by the selected evaluation method of each 3D edit pipeline can get a relatively objective result of the quality of outputs. The user study is an intuitive solution. Users are usually asked to give scores for the outputs based on the realism, the features preserving level compared to the original object and the matching level of the instruction of the result [Michel et al., 2022; Chen et al., 2022].

Other than the widely used user scores, there are also studies to develop metrics to compare the generated results. One idea is to compare the similarity of the rendered images of the output 3D model and the ground truth object [Chen et al., 2023a], and Frechet Inception Distance (FID), which is regarded as an effective metric in this field, can be used [Heusel et al., 2017]. CLIPscore is another method to evaluate the realism of the generated image by

computing the similarity of the target image and the corresponding text caption using the CLIP model [Hessel et al., 2021].

However, they both have limitations in the thesis scenario. For the FID score calculated by comparing the generated result with the ground truth, difficulties lay in collecting such images. For buildings, which are more complex than common objects, it is difficult to get the proper text prompt to generate the same building as the ground truth, which adds uncertainties to the FID score. Besides, it is difficult to get high-quality ground truth images for building mesh samples used in the thesis, which are mostly buildings in the real world. The attained images may have problems like barriers in front of the building, unsatisfying lighting conditions and viewing angles. FID score can also be calculated by comparing the generated results with realistic image datasets of the same categories. However, attaining a representative building domain dataset that closely fits the types and features of the experimental cases is challenging.

As for CLIPscore, it suffers from bias caused by the loss calculation method of pipelines. If the pipeline already uses CLIP-related loss terms (X-Mesh related methods here), the CLIP-score would naturally become higher as the method already tries to reduce the difference between the result and the prompt (both encoded by the CLIP model) in the training process.

Therefore, due to the limitations of the above-mentioned metrics, the user score is chosen as the quantitative evaluation metric in the thesis for its wide range of applicable scenarios and lower system error.

3.2. Control module

3.2.1. Text prompt engineering

To fully exploit the power of the pipelines, the first step is to figure out the proper text prompts for the chosen 2D pre-trained models. This process involves studying the mechanism of models, referring to the related papers and conducting experiments. The experiments should be first made with the 2D models to figure out the proper text prompt. It can help improve efficiency as editing a 3D model is time-consuming and a text prompt that performs well in 2D has a higher possibility to generate good results in 3D. CLIP model uses a text encoder to transform text prompts into tokens and words from the whole prompt contribute together to the final matched result. As the Stable Diffusion Model also uses the text encoders trained with CLIP, similar basic rules for text prompts can be summarized.

Apart from that, the camera angle can also be specified in the text prompt to make generated results better fit the corresponding rendered views. The Text2Tex and Latent-Paint pipelines already implement this idea. The view specification (e.g. front view, side view) is attached to the input text prompt based on the viewing direction (elevation and azimuth). Then for each view, the loss calculation or the image rendering follows the adapted prompt. Such a trick can be added to the X-Mesh pipeline as well. In each iteration, images from multiple views are rendered and corresponding text prompts with view descriptions are created for individual loss calculation. To introduce more novel views in the training process, random numbers smaller than the set interval across defined views are added to the chosen elevation and azimuth. The modification aims to increase the accuracy of the final average loss to better guide the training of networks.

3.2.2. Image as control

For buildings, sometimes it is hard to directly describe the desired result with text precisely due to the complexity. Therefore, it is natural to turn to images that contain abundant information for help. It is relatively easy to find a suitable building image as guidance. For text-to-image or depth-to-image Stable Diffusion models, a text prompt is needed. CLIP-interrogator offers a way to get the optimized text prompt for a given image by searching for the best matching combination of words [Pharmapsychotic, 2023]. Thus, this tool can be used to find better-performing text prompts with the help of images. To get better control over the result, the applied images should also follow basic principles summarized by experiments.

Another modification attempt is to directly use the image as the control for the CLIP based X-Mesh pipeline. Instead of encoding the text prompt with the CLIP text encoder to compare the similarity with the rendered images, the image can be encoded directly to the same dimension with the CLIP image encoder and used for loss calculation. Therefore, the complete information from the image can be incorporated into training.

The input image for X-Mesh inevitably introduces view bias and the loss is likely to be lower when the rendered image shares similar viewing angles with the image and vice versa. To alleviate this problem, higher attention can be given to views from neighbouring angles of the input image in CLIP loss calculation. It can be done by ensuring that in each iteration a certain percentage of views, which is higher than the random selection possibility, are neighbouring ones. Assigning a higher weight for the loss of neighbouring views when computing the final average loss is another possible solution.

To allow more flexible control and generate higher quality results, another attempt is to edit key parts of the building separately. Semantically, the building can be separated into the facade and roof. The facade can be further divided into smaller segments like windows, doors and columns. Different image prompts, which ideally only contain the targets, can be provided for separate parts. Each semantic part can be trained separately so that the TDAM module and MLPs can represent their distinct features. The loss is also calculated only using the rendered images of the corresponding parts of the building.

3.3. Edit modules

3.3.1. Freeze samples in X-Mesh

It is a common practice in machine learning to freeze part of the layers or parameters in the networks to switch focus on different parts, which may allow the specific patterns to be better recognized and improve the training efficiency. Inspired by it, the modification attempt is made on the X-Mesh pipeline to enable more reasonable and larger-scale geometric edits. In each iteration, a certain percentage of vertices are selected randomly and their positions are frozen. In the loss calculation process, the geometry of the selected vertices remains the same as the original input and only the colour offset is applied. It may enable the networks to focus on different parts at each time to attain more information for the networks. An advanced idea is to freeze connected regions of vertices instead of mere random choice to make the switching attention better organized.

3.3.2. Combine X-Mesh and Text2Tex

The Text2Tex pipeline features in generating texture based on depth information while X-Mesh features in editing the geometry and colour jointly to get better stylization results and output mesh with more geometric details. For the depth-to-image Diffusion Model, it is likely that if more detailed depth variance information is provided, more realistic images can be generated following the text control and the consistency across views may increase.

Therefore, combining the two pipelines by using the geometry output of the X-Mesh pipeline as the input for the Text2Tex can be useful. The rendered depth map in Text2Tex thus contains richer information. It aims to generate textures that better fit in with building models and are more consistent across different views as more restrictions are provided. Before using Text2Tex for texture generation, the obvious geometric artifacts can be manually fixed.

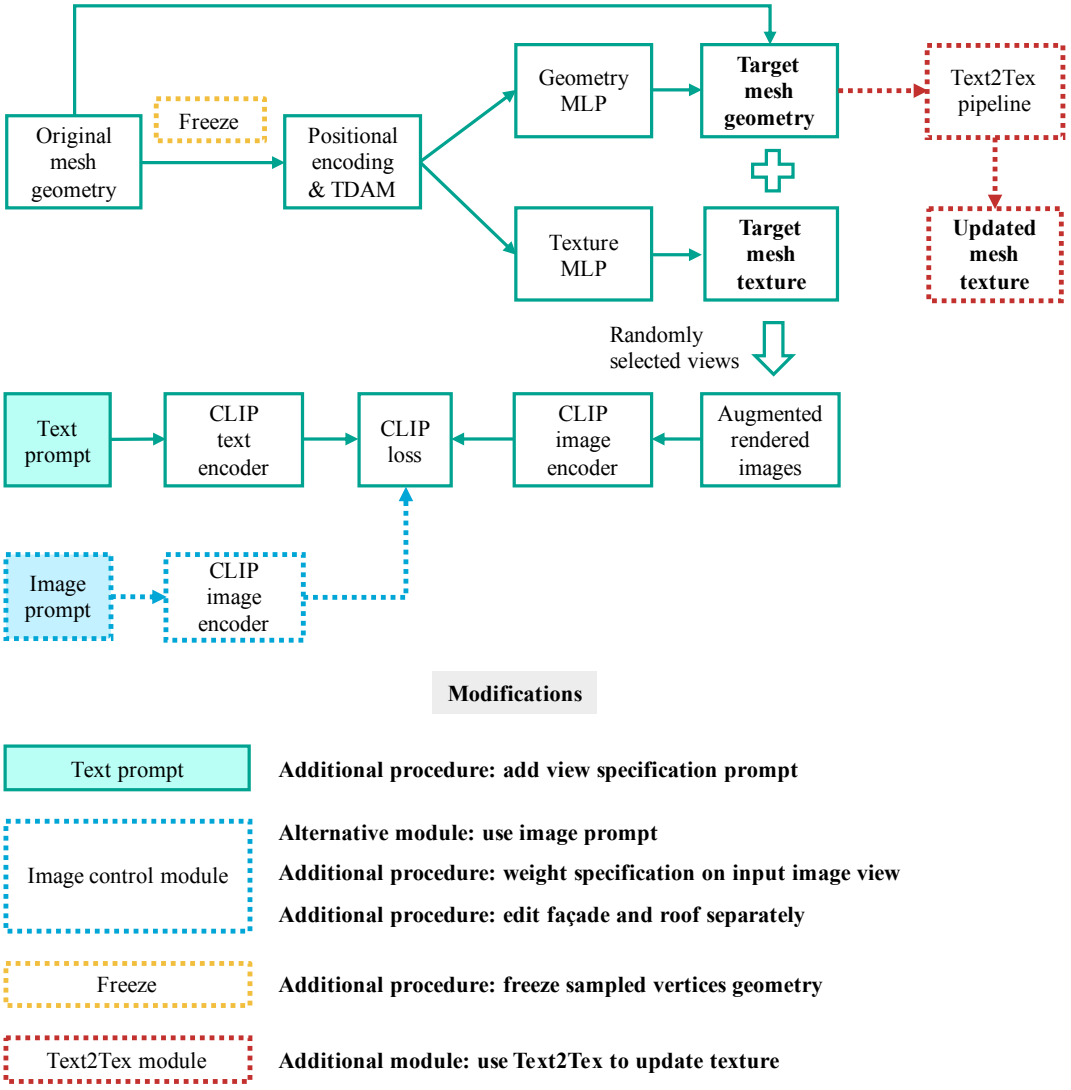


Figure 3.3.: Control module and pipeline modifications: X-Mesh serves as the basic and six modifications are made

4. Results

4.1. Implementation details

4.1.1. Data

The thesis focuses on adding details to building meshes. Therefore, the high-quality open-source 3D BAG data is a reasonable choice (*here*). It captures buildings from the whole Netherlands, covering various sizes and types of architecture, which provides rich examples for experiments. LoD 2.2 level mesh is chosen, which is the highest LoD version provided by 3D BAG. At this level, buildings are modelled as simple structures containing standard and simplified roof structures, and smaller building parts and extensions and roof superstructures are also acquired [Biljecki et al., 2016]. To cover more types of building models, 3D Warehouse (*here*) is used as an additional source. It offers more complex models than those found in 3D BAG and the downloaded Sketchup (.skp) file can be easily transformed to mesh (.obj) format. Lastly, one building mesh is obtained from the sample provided by the X-Mesh paper as a supplementary [Ma et al., 2023]. Building meshes used in the experiments and their characteristics are shown in Figure 4.1.

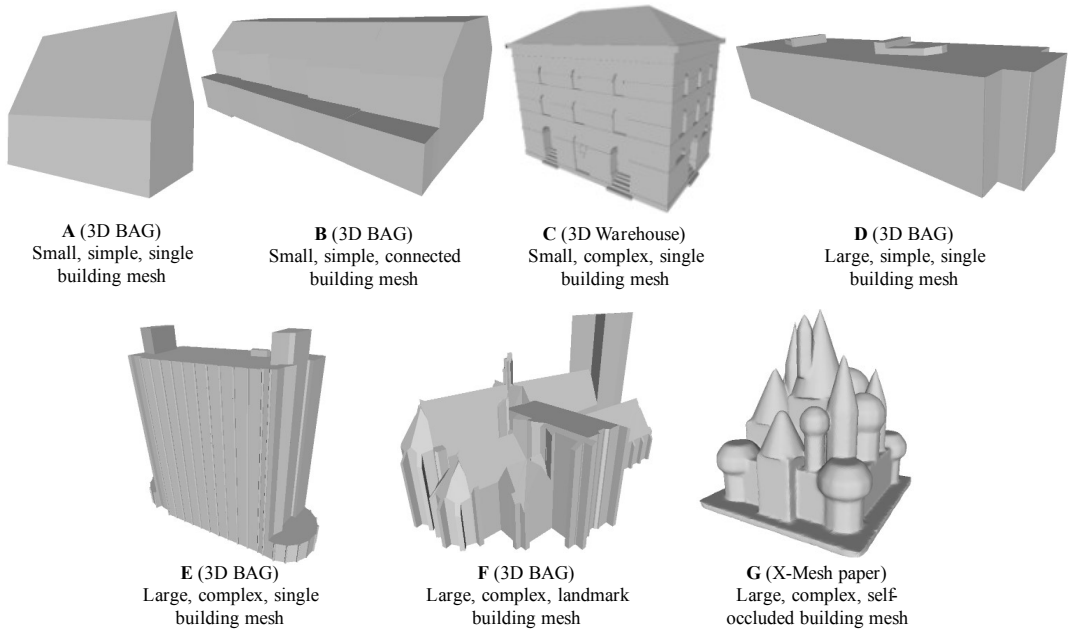


Figure 4.1.: Building mesh samples (with data source and characteristics): cover various sizes, complexity and building types

4.1.2. Tools

The programming language for this thesis is Python as it is one of the most popular languages in machine learning and many related open-source packages are available. The packages for the creation and training of neural networks include PyTorch and related ones like PyTorch Lightning and PyTorch3D. [Paszke et al., 2017]. As for the usage of pre-trained models, CLIP from Openai provides simple APIs for different versions of CLIP [Radford et al., 2021]. Accelerate, Diffusers and Transformers from Hugging Face jointly offer easy-to-use functions for applying original and modified Stable Diffusion Models [Wolf et al., 2019]. For automatic UV parameterization, the Xatlas library is used [Jpcy, 2022]. Additionally, Threestudio provides well-structured modules to apply and modify 3D generation and edit pipelines, which is especially useful for exploring implicit representation based methods [Guo et al., 2023].

The open-source software Meshlab is a useful tool to deal with meshes. In the pre-processing stage, Meshlab can repair the major artifacts of meshes, like non-manifold edges, isolating small pieces, duplicate vertices and faces, self-intersecting faces and wrongly oriented faces. It can also be used for surface subdivision to create meshes with smaller faces as the input for X-Mesh to allow higher-quality deformation. In the final stage, it can be used for mesh visualization [Cignoni et al., 2008].

4.1.3. Experiment setup

For representative existing pipelines, the default parameters and network structure are used. In the fine-tuning of X-Mesh, the positional offset ratio is tuned between 0.02 and 0.2 (the default one being 0.1). The number of vertices of input mesh ranges between 80,000 to 400,000. The number of iterations is set between 1000 and 1500, depending on the complexity of the input mesh. The frozen vertices ratio ranges between 0.05 and 0.5. As for the choice of pre-trained model, the Latent-Paint uses the text-to-image Stable Diffusion Model, the Text2Tex uses the depth-to-image Stable Diffusion Model, and the X-Mesh series uses ViT-B/32 CLIP Model.

The experiments are done on an NVIDIA GPU with 24GB memory (RTX 3090 or A10). The experiments take around 30 to 60 minutes for one building sample in every single pipeline (without combination).

4.2. Results and evaluation

4.2.1. Control engineering in 2D space

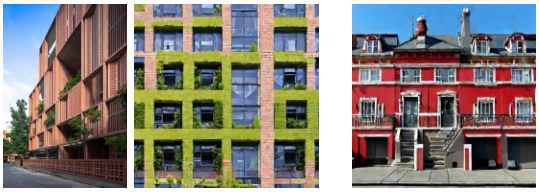
Experiments on the text-to-image Stable Diffusion Model are done in 2D space before 3D edits and some basic principles can be summarized. The text prompts should be clear and specific so that the uncertainty can be reduced and the results can better meet the demand. For example, instead of only mentioning the type of the building, size, colour and material of key parts, additional features and the overall style can be specified. Certain keywords, like 'unreal engine' or 'DSLR photo', can also be added to generate realistic style images

[Ma et al., 2023]. Additionally, adding conditions like 'zoom-out' or 'exterior' helps to get the complete architecture from the outside.

As shown in Figure 4.2, compared to simply specifying the building type, adding keywords for realistic style and view conditions specifications enables the model to generate more complete and realistic buildings from exterior angles. Adding detailed descriptions to the building is useful in case of brief and common conditions but the pre-trained model sometimes fails to fully recognize and follow more advanced ones as shown in the first example in Figure 4.3. Additionally, the 2D experiments also show that the pre-trained model performs worse in dealing with connected buildings compared to single ones, as experiments show that it has difficulty in generating the correct number of connected elements (see the second example in Figure 4.3).

Apartment | An exterior apartment | An exterior four-storey red apartment with grey roof

Figure 4.2.: Comparison of generated images by different text prompts from text-to-image Stable Diffusion Model: the third sample with specific keywords and detailed descriptions outperforms the previous two



A four-storey office building with perforated brickwork and plant decorated façade
Left: photo of desired building in the real world
Right: image generated by Stable Diffusion

A zoomed out DSLR photo of a two-Storey red townhouse with small windows and grey roof, five connected

Figure 4.3.: Failure examples of text-to-image Stable Diffusion Model: the first sample fails to generate the fully desired building according to the complex prompt and the second one shows its limitation in generating connected buildings

CLIP Interrogator is also tested (see Figure 4.4) to find better text prompts with the help of images. However, it can not perform perfectly [Pharmapsychotic, 2023]. Using the text result from CLIP Interrogator to generate images with the text-to-image Stable Diffusion Model, the output images differ a lot from the original input images in some cases. The possible reason is that one-to-one image-to-text matching is impossible in CLIP. Features are extracted from the image and the CLIP Interrogator finds the combinations of words with features that have the highest similarity with the image features. The text prompt that has the highest possibility of describing the image is returned but it is usually unlikely to find

the word combination that has exactly the same features as the input image. Additionally, the diffusion process in the generation model involves uncontrollable randomness.

The CLIP Interrogator can still help to choose better images as control and some basic rules can be attained. Firstly the building should be complete and clear. The building should take up the majority of the image and the fewer barriers in front of the building the better the image is. Thus, fewer irrelevant features are encoded and more complete information about the building is conveyed. The image should also be from representative angles (e.g. the front view is better than the back view) to give more useful control. As shown in Figure 4.4, images with many objects unrelated to the building like plants or the building being too small or incomplete are not ideal choices.

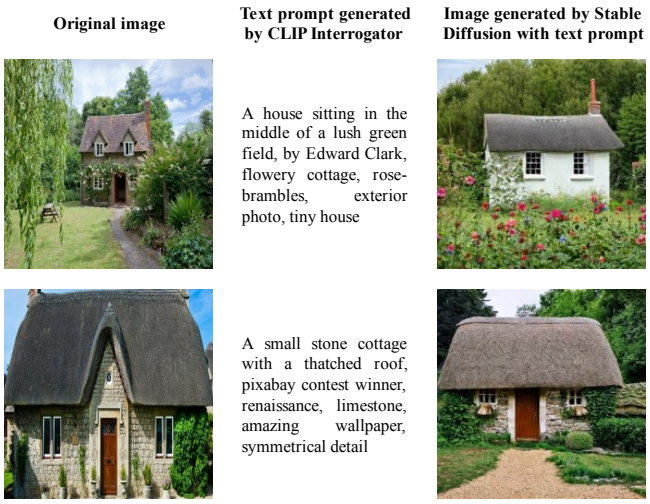


Figure 4.4.: Experimental results of CLIP Interrogator: the image with closer-up and unobstructed buildings has the better result though the deduced text prompt can not generate the perfectly matching image.

4.2.2. Representative existing pipelines

Experiments are done with different building samples on the chosen representative existing 3D edit pipelines (see Figure 4.6). Important parameters are also fine-tuned and discussed below (mainly for X-Mesh). A qualitative analysis is conducted and both advantages and limitations are summarized as a starting point for further modifications.

Among the three pipelines, Latent-paint performs the worst in the majority of cases. The large surfaces of the output mesh are assigned the correct colour and large structural features can be barely recognized in some cases. For example, the textures of models F and G can differentiate the roof and facades and the glass facade in model E is painted with the correct colour. However, finer details and smaller structures like windows and doors are missing or unclear. For example, in models B, C and D, the pipeline completely fails to generate the brick texture and openings. The small building model A is an exception, which attains relatively consistent and detailed texture, and correctly located windows and doors. The main reason behind the lack of details and blurry texture of the larger building samples is the view inconsistency problem that the pre-trained Diffusion Model based SDS loss training

method suffers. Similar issues are also observed in the exploration of implicit representation based methods.

Text2Tex generates outstanding texture for those building samples with relatively more details. Building materials and small structural elements are generated. For example, the thatched roof texture is generated for model A and in model E the glass facade and openings on the ground floor can be observed. Especially for the landmark architecture recorded in the Diffusion Model (model F), and high-fidelity building mesh (model C), the output mesh is very realistic with all necessary textural details properly generated. However, for building samples with fewer details, the output mesh fails to gain fine structural features and the whole facades and roofs can not be correctly recognized. In models B and D, the pipeline fails to assign the correct structural and building material information and the facades and roofs have the wrong textures. The possible reason is that the depth-to-image Diffusion Model requires higher-quality depth maps to deduce the proper corresponding images. Additionally, it also faces the self-occlusion problem. As the pipeline relies on back-projected texture to the original mesh in limited iteration, only limited views are considered. Thus for buildings (model G) with adjacent structures that block each other, a small part of surfaces can only be observed from acute views and may have limited chances of updating, which results in worse texture.

X-Mesh can generate relatively reasonable results for most samples, though results may suffer from noise problems in both geometry and texture. For instance, the results of models B, C, D and G all have instructed brick materials and openings, but the roofs and facades are not fully differentiated and the textures are not smooth enough. In model A the geometric deformation is not fully aligned with the texture. In model F too many unwanted features related to the church are created on the texture. The noise problem may originate from the multi-view inconsistency problem in the CLIP model and the uncertainty in the image features related to the input text features. An exception to the generalizability of X-Mesh is model E, where the generated texture for the front facade of the office is not aligned with the facade but is more like a combination of two glass walls.

Another limitation of this pipeline is that significant geometry editing is impossible. The geometric deformation is limited to remaining close to the original surface, like the added windows structure in the geometry result of models A, B and D, and large structural changes, like adding a balcony, can not be achieved. The position offset ratio is also tuned in experiments, which show that increasing the ratio results in uncontrollable deformation of meshes (see Figure 4.5). The potential of the pipeline is also limited by computational resources. The more vertices are sampled on the building models (within a reasonable extent), the higher quality the output mesh can be (also see Figure 4.5), but the more GPU memory and the longer execution time are needed. Thus for complex or very large buildings that already require many vertices to represent the basic structure of the input mesh, limited additional vertices on each surface can be used, which results in relatively low-fidelity edits, especially in terms of deformation (e.g. complex models C, E and F have no obvious geometric deformation).

To further explore the potential of X-Mesh, two reduced versions are tested, which only calculate the photometric loss (i.e. freeze the geometry of the original mesh) and the geometry loss (i.e. freeze the colour of the original mesh) respectively. As shown in Figure 4.7, both the texture and geometry of the original X-Mesh pipeline generally outperform the reduced versions. The texture generated by the original X-Mesh is more realistic. In the group only considering the photometric loss, for instance, model C misplaces the arched window and model F assigns the wrong colour to the roof. The geometry edited by the original pipeline

also contains more reasonable details and fewer artifacts. In the group only calculating the geometry loss, for example, model A has a window on the roof and model D loses part of the original identity in the roof part. The reason for the relatively better performance of the original X-Mesh is that combining the geometry loss and the photometric loss can incorporate more information in the networks. The original photometric loss is also calculated on the mesh with edited geometry, which improves the quality of the rendered images and leads to more accurate loss calculation.

As found in the experiments of the implicit representation based methods (see Appendix B), view inconsistency problems always exist to some extent in 3D edit pipelines. Considering the successful cases, the Text2Tex pipeline outperforms the others. It may take advantage of the relatively independent texture generation and back-projection process, as well as the design of the region-defining mask and the refinement stage. However, it is not robust and has failing cases (models B and D). On the contrary, there are no complete failures in Latent-Paint, because the generated texture has fewer details, and some flaws are hidden by the over-smoothing texture. The X-Mesh seems to suffer mostly from the view inconsistency problem and unreasonable elements can be observed on part of the surfaces, among which the side facades and roof are worse than others (e.g. windows can be found on the roof in models B and D). The reason may lie in the special features of building models, especially for the mesh with lower fidelity. Part of the input surfaces share similar geometric features (e.g. the flat roof is similar to the vertical facade) and the neural network has trouble differentiating them even with positional encoding, thus resulting in wrongly assigned edits. In general, the Text2Tex and X-Mesh pipelines have more application potential in building model edits and further modifications can be based on them.

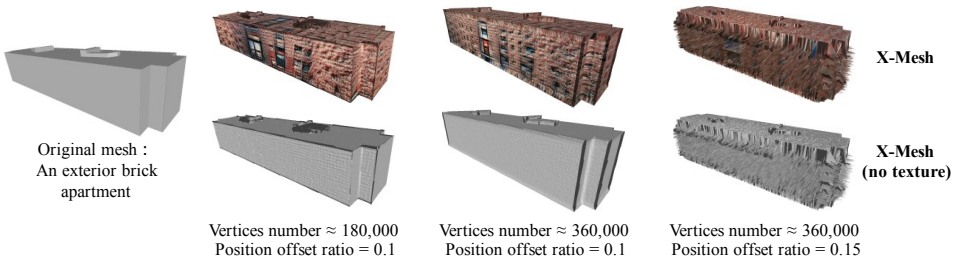


Figure 4.5.: Results of X-Mesh vertices number and position offset ratio tuning: using more vertices generates better results and larger position offset ratio may generate excessive deformation

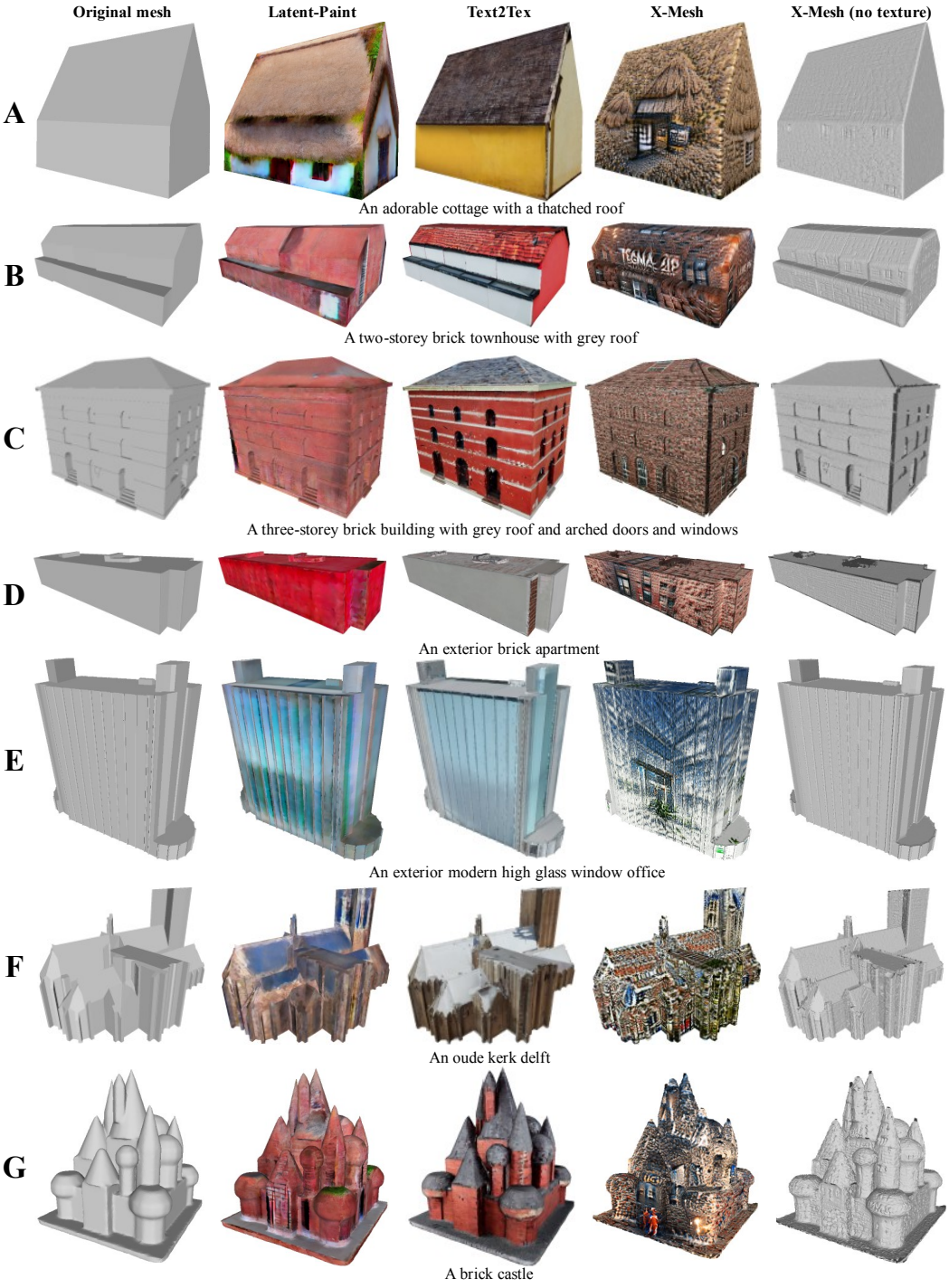


Figure 4.6.: Results of representative existing pipelines: Text2Tex (generates realistic and smooth texture in part of cases) and X-Mesh (edit the geometry and creates the texture matching the prompt) generally perform better than Latent-Paint (only generates textures with lower detail level in most cases) are chosen for further modifications.

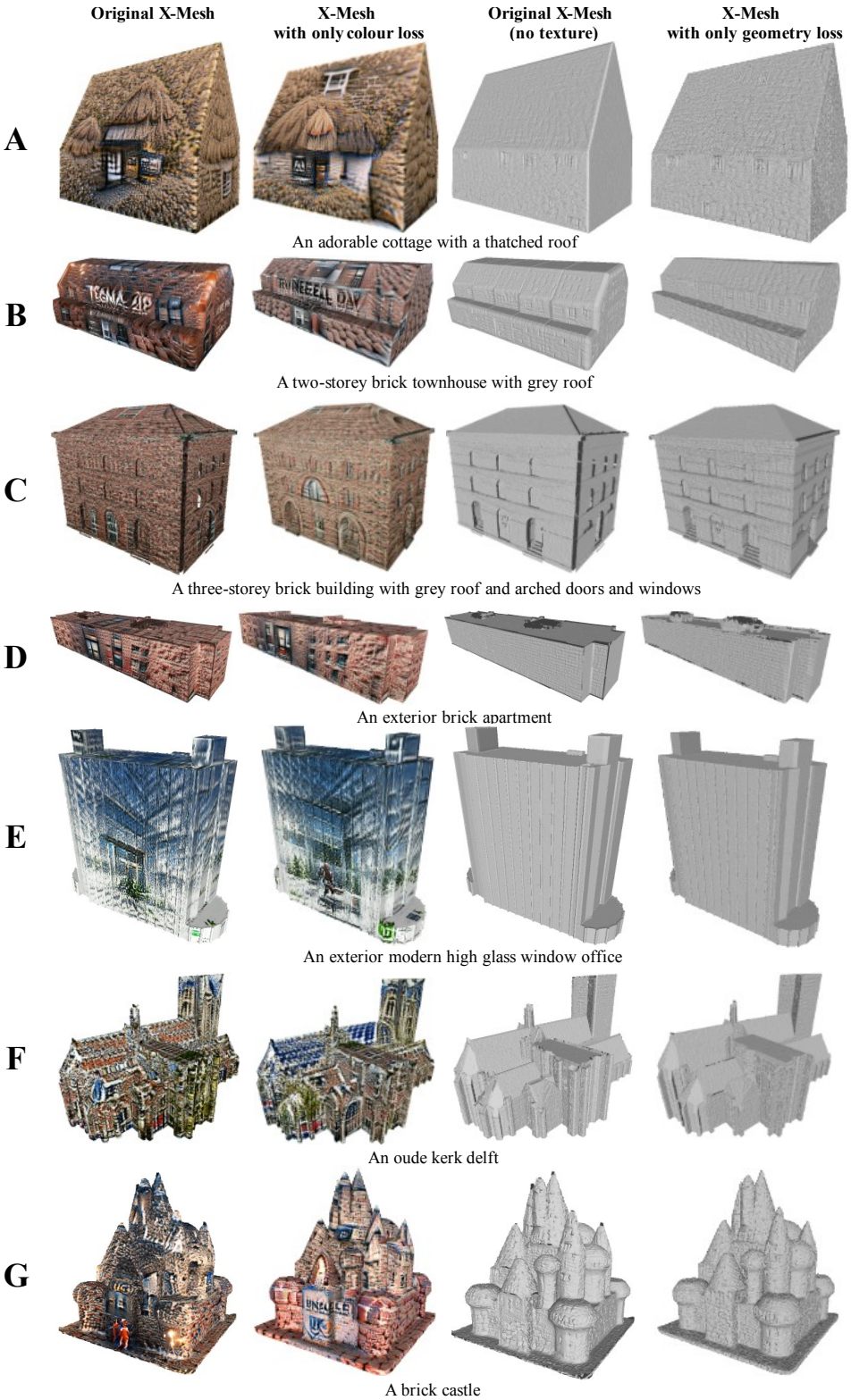


Figure 4.7.: Comparison of original X-Mesh, X-Mesh with only photometric loss and X-Mesh with only geometry loss: original X-Mesh outperforms the reduced versions in both texture creation and geometry edit.

4.2.3. Modifications in X-Mesh

The first and intuitive modification attempt is the view specification in X-Mesh (see Figure 4.8). 36 pre-defined views are created and named with direct viewing keywords (e.g. top, front, right). Later another version of viewing keywords that fit in the building scenario better is also tested (e.g. replacing top with roof, right with side). Small improvements can be observed and the second version of the view specification performs slightly better than the first one. A possible reason is that the CLIP model itself does not perform very well in differentiating different views while remaining consistent.

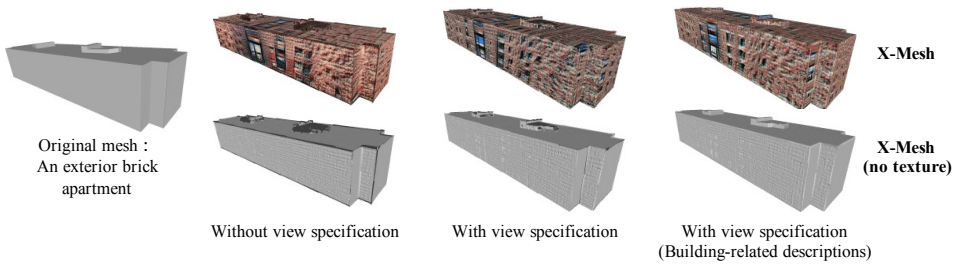


Figure 4.8.: Comparison of text control X-Mesh without and with view specification: slightly improve the texture and geometry and using building-related descriptions is better

The second attempt, using images instead of texts as the control in X-Mesh, successfully improves the quality of the generated mesh (see Figure 4.9) and shares the robustness of the original method. The texture of the output mesh is more realistic and less noisy, and fits in with the shape of the building, especially in the views similar to the input image angle. The geometry of the output mesh is also more satisfying. High-level details like windows are added more coherently and the size and location are closer to reality. Outstanding examples are models D and G, which have clear windows and doors of the correct size and location, and facade decorations sharing the style of the input image. Other samples also have significant improvements compared to the original X-Mesh pipeline. It proves that the rich and contextual information provided by images can better guide the CLIP model based 3D model edit. Though the edited mesh can not follow the image control completely, style similarity can be observed from the image and the output mesh.

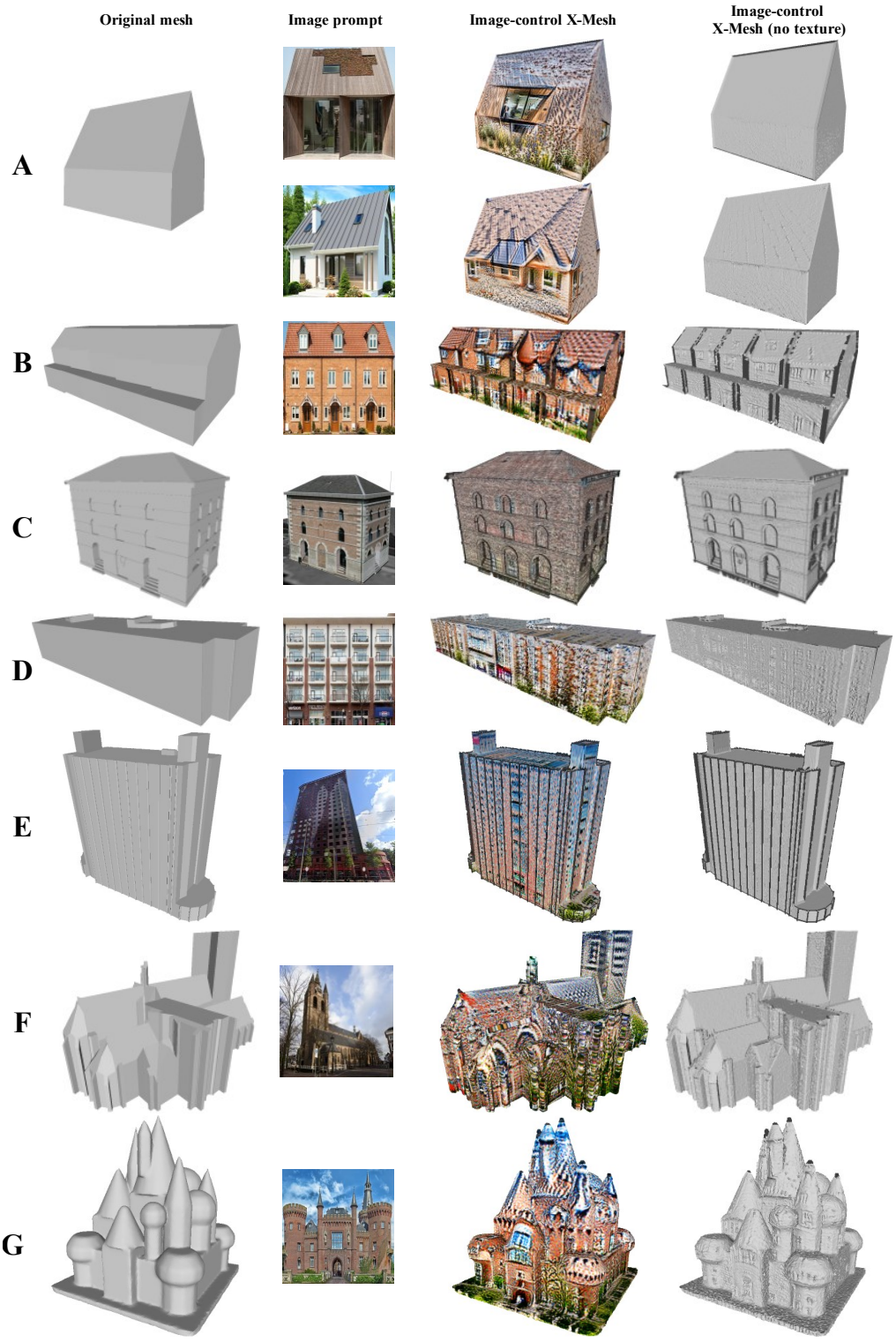


Figure 4.9.: Results of Image control X-Mesh: higher-quality texture and geometry compared to text control X-Mesh

Minor problems can still be observed in the image-guided results. In some cases, the texture of building parts unseen from the input image still suffers from unrealistic (e.g. windows on the roof in model B) and noisy (e.g. the windows on the back side are now as clear as those on the front as shown in the first sample in Figure 4.10) problems. As for geometry, changes in unseen building parts (like the back of the building) are not obvious enough and the added structures like windows are not complete (also see the first sample in Figure 4.10). Therefore, a new attempt that gives more attention to views from neighbouring angles of the input image is made (see the second sample in Figure 4.10). Small enhancements can be observed in colour but not in geometry. More realistic and suitable textures are created for some other views (e.g. back and side). A possible reason is that the given image lacks complete information about the whole building. Therefore, the quality of the generated mesh varies in different views and the trick can not completely solve the problem.

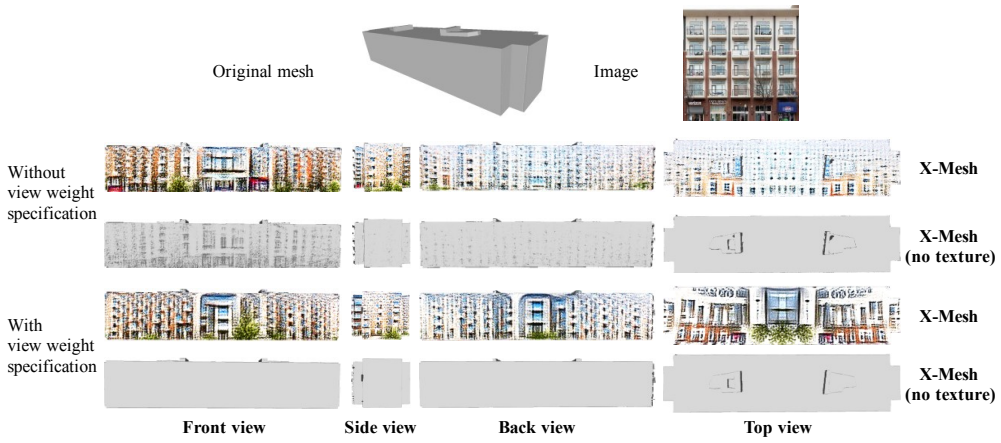


Figure 4.10.: Comparison of image control X-Mesh without and with view weight specification: the modification improves the texture of the generated mesh, especially for the side and back views.

To further improve the quality of the output mesh, experiments on freezing part of the geometry of vertices in each iteration are made (see Figure 4.11). The vertices are chosen randomly and different percentages of freezing points are tested. However, the geometry nearly remains unchanged in these attempts. A possible reason is that the position offset of vertices is of small scale and influenced by the neighbouring ones. The random frozen points are distributed across the whole model and can affect all other vertices.

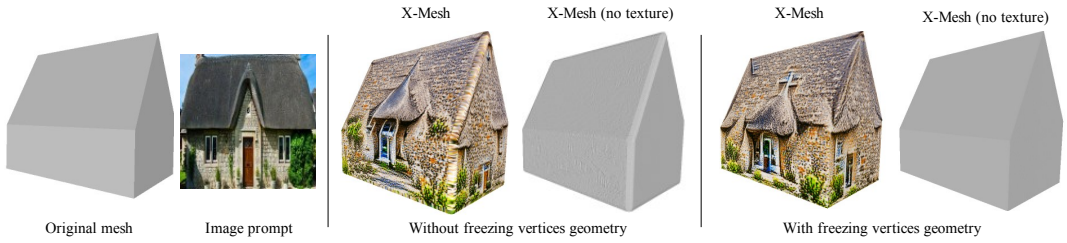


Figure 4.11.: Comparison of image control X-Mesh without and with freezing vertices geometry: geometry remains unchanged when freezing randomly chosen vertices geometry

Confusion of roof and facade can be observed in part of the results from Figure 4.9. Model B suffers most severely from this problem, where walls and windows can be seen clearly on the roof. To address this issue, an attempt to edit the facade and roof part of the building separately is made, but it fails to meet expectations. As shown in Figure 4.12, the image prompt only includes the facade but the edited facade result still has obvious roof elements. A possible reason is that the CLIP image encoder treats the object as a whole. Thus, even though the input image only contains a segment of the building, the CLIP encoder still deduces the represented complete building and then guides the mesh to include unwanted elements.

Figure 4.12.: Facade result of separate edit with image control X-Mesh: CLIP encoder deduces the complete house from the facade image and roof elements are depicted on the facade.

4.2.4. Combine X-Mesh and Text2Tex

Based on the observation that the Text2Tex pipeline performs better in general when the input mesh has more geometric details and the X-Mesh pipeline sometimes generates noisy texture but adds proper geometric details to the mesh, it is a natural idea to consider combining the two pipelines to take the advantage of both.

The experiments combine text control X-Mesh and Text2Tex to enable better comparisons (see Figure 4.13). It shows that using the geometrically edited mesh of X-Mesh as the input for Text2Tex enables the generated texture to depict more structural and material details in simple building models. For example, in model B, the combined pipeline can generate textures with clear windows and doors. For input mesh with a higher level of detail, the combination makes little improvement compared to the original Text2Tex result (model E), and in some cases, the texture is even slightly noisier (models C and F). One possible reason is that the geometrically edited mesh has noisy geometric information and hinders better image generation. Inevitably, there are still failing cases where the geometric changes from X-Mesh can not be recognized and the generated textures do not contain structural details like windows and doors, like models A and D. It is possibly due to the limited generalizability of the depth-to-image Stable Diffusion model.

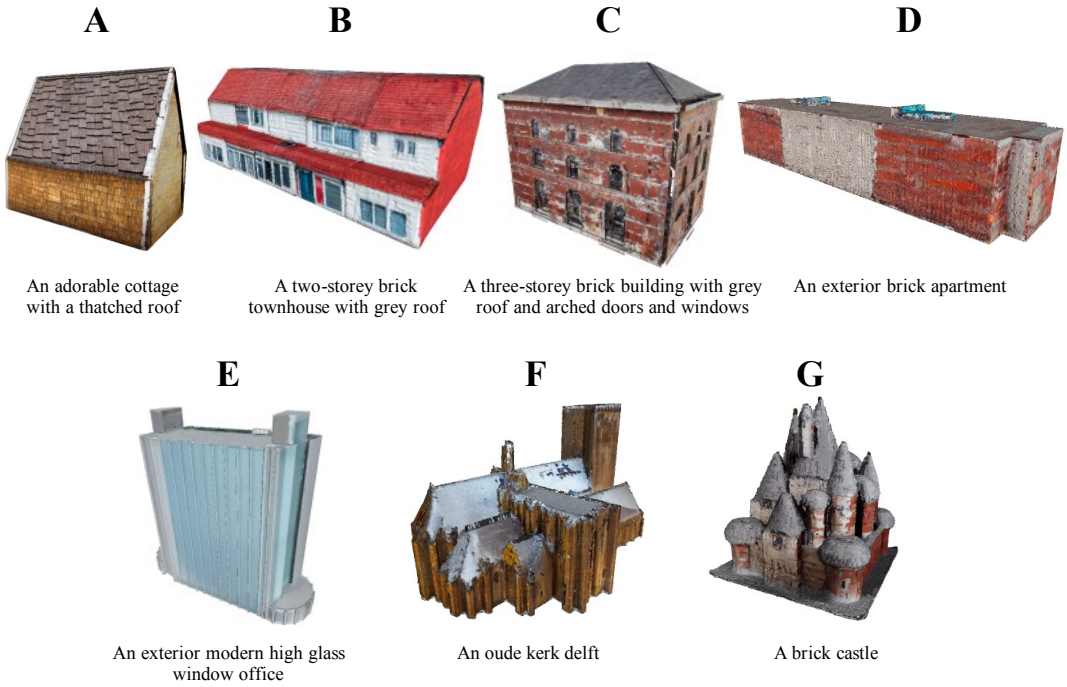


Figure 4.13.: Results of combining X-Mesh and Text2Tex: the combined pipeline generates more realistic and detailed textures in some cases.

4.2.5. Quantitative results

User study is conducted to quantitatively evaluate the results. The results of three existing pipelines (Latent-Paint, Text2Tex, X-Mesh) and two successful modifications (Image control X-Mesh and Combination of X-Mesh and Text2Tex) are compared. Seven representative building model samples of different sizes and fidelity levels are shown. Users are asked to score the 35 edited result images on how realistic they are from 1-5 with 1 being the lowest and 5 being the highest score. To reduce bias, outputs edited by different pipelines are ordered randomly in each set of the building model. The text or image prompt used for editing is also provided. The user scores are collected by online forms and there are 50 respondents in all. The average user score of the five experimented methods is calculated both for the overall result and for seven separate building models.

As for the overall result shown in Table 4.1, the Combination of X-Mesh and Text2Tex receives the highest score while Image control X-Mesh gets the second highest, which is obviously higher than the original X-Mesh result. It proves that the two modifications can produce user-favored results. Text2Tex pipeline receives the third highest score (close to the previous two), showing its potential in the building model domain. Latent-Paint attains the lowest score and it is aligned with the qualitative evaluation.

Method	Latent-Paint	Text2Tex	X-Mesh	Image control X-Mesh	Combination of X-Mesh and Text2Tex
Average score	1.98	2.73	2.31	2.79	2.86
Standard deviation	1.15	1.22	1.36	1.18	1.26

Table 4.1.: Overall user scores of five methods

When it comes to the separate results (see Figure 4.14), Latent-Paint receives low scores in almost every model. Image control X-Mesh receives higher scores than X-Mesh in five cases while having similar scores in the other two, which are the same as the overall results. However, different performances of surveyed pipelines can be observed across each building model. For input mesh with lower fidelity (models A, B and D), Image control X-Mesh performs well and the Combination of X-Mesh and Text2Tex is also a potential choice. On the contrary, for more complex models (models C, E, F and G), Text2Tex performs better in general and the Combination of X-Mesh and Text2Tex also attains high scores. Such results can serve as support for proposing user scenarios for different methods.

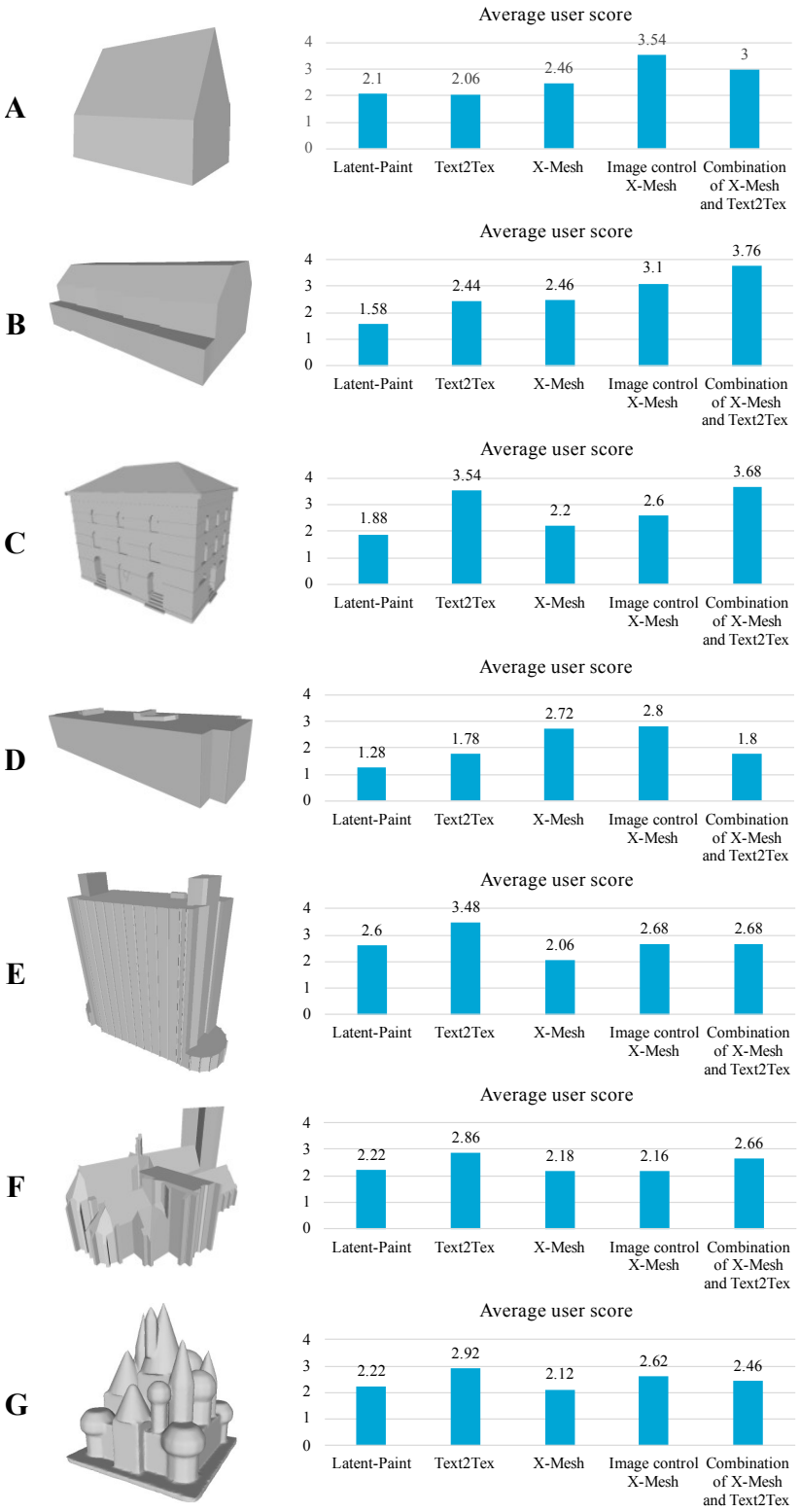


Figure 4.14.: Average user scores of each building sample with five edit pipelines

4.2.6. Application cases

The proposed 3D building model edit pipelines can be used in various cases. They mainly edit one building at a time and the edited buildings can be combined to create larger scenes. The methods can efficiently create buildings or scenes with pre-defined structures. Different styles can be assigned and these pipelines have application potential for concept comparison in the early design stage, and general scene creation for urban planning, gaming and animation. Figure 4.15 and Figure 4.16 are two application cases using the image control X-Mesh method.

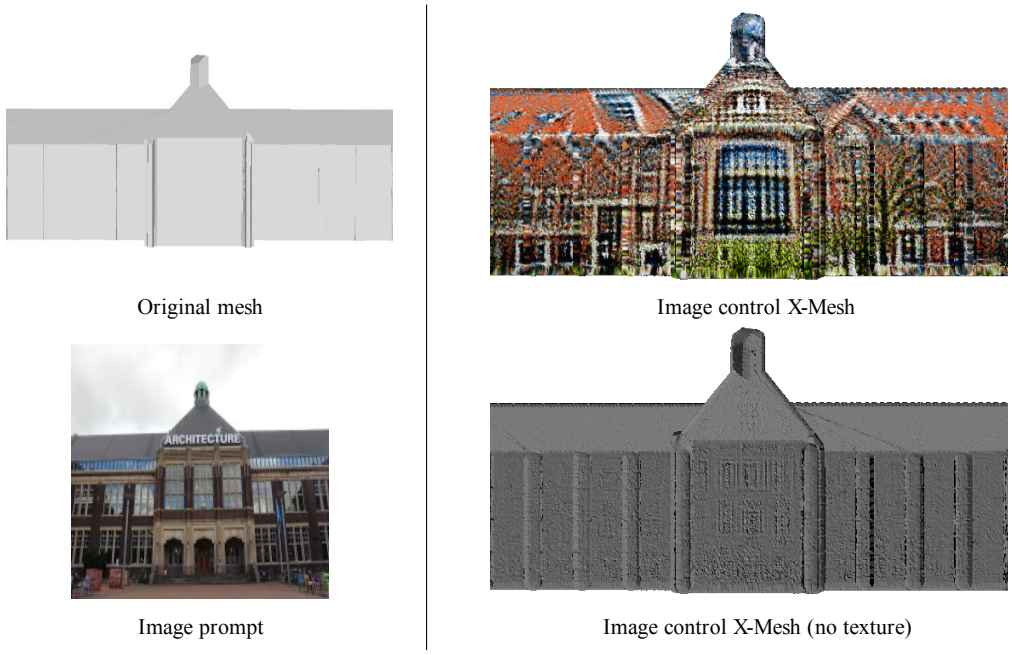


Figure 4.15.: Single building edit: faculty building of Architecture and the Built Environment in TU Delft (with Image control X-Mesh)

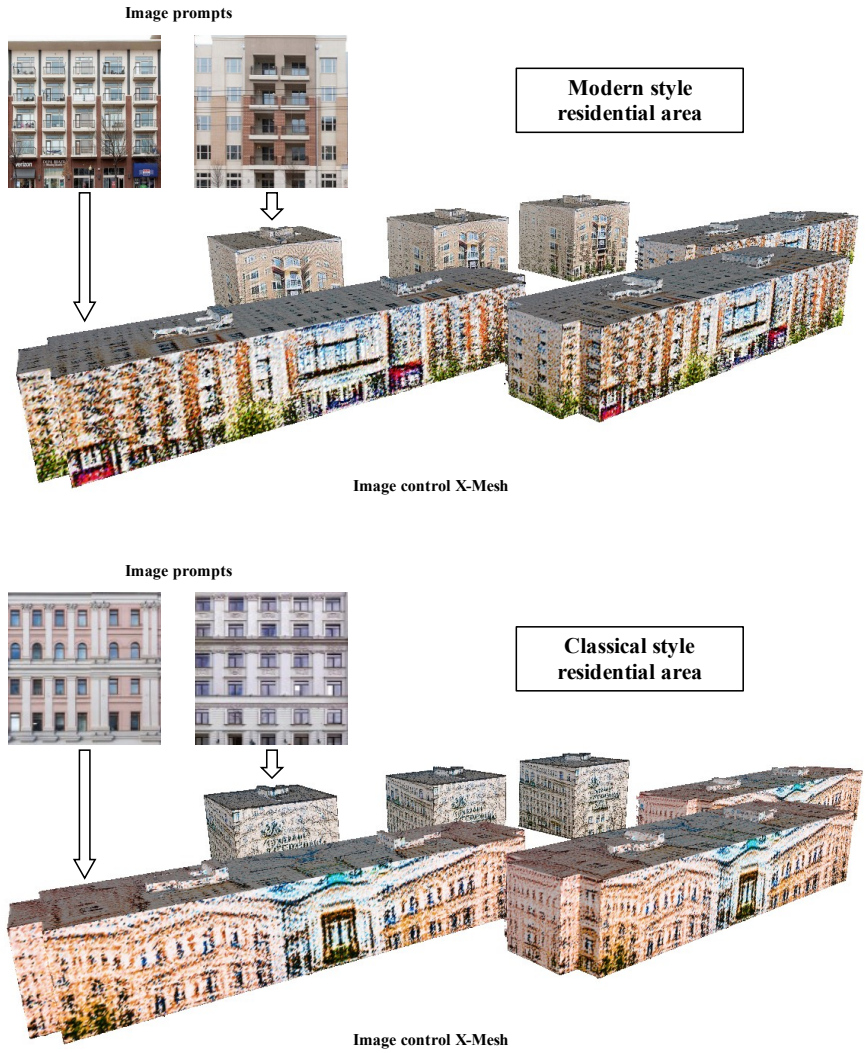


Figure 4.16.: Multiple buildings edit: residential area of modern and classical styles (with Image control X-Mesh)

5. Conclusion

The thesis focuses on exploring the effective pipelines for building model edits. To achieve this, the proper 3D representation type is chosen and existing representative pipelines are evaluated to figure out their advantages and limits. To achieve more desirable edit results, multiple attempts are made and the proposed pipelines are obtained by modifying the current methods. The application scenarios (Section 5.1), contribution (Section 5.2), limitations (Section 5.3) and future work (Section 5.4) are discussed below.

Experiments are first conducted on implicit representation based methods but the results are failing and unpromising. Explicit representation based methods are then chosen as the major direction and mesh is selected as the representation type. Latent-Paint, Text2Tex and X-Mesh, which apply three different ideas and perform well in their sub-domains, are then chosen as representative edit pipelines. Latent-Paint uses text-to-image Stable Diffusion based SDS loss to guide texture generation. Text2Tex uses the depth-to-image Stable Diffusion model to generate and refine texture with masks defining new and updating regions. X-Mesh applies CLIP loss to edit geometry and create texture.

Their performances are compared and results show that Latent-Paint can only generate textures with low-level details in most cases with possible exceptions in small and simple building models. Text2Tex has the potential to generate high-quality textures with high-level details for buildings, but it may encounter failures when dealing with low-fidelity buildings. X-Mesh can both edit geometry and generate texture fitting different samples but suffers from noisy problems. Its combination of geometry edit and texture creation jointly improves the accuracy of loss calculation and the effectiveness of network training, leading to higher-quality geometry and texture results compared to single-item changes.

Practical solutions are gained from various modifications based on X-Mesh and Text2Tex. The two most successful attempts are using the image as the control in X-Mesh (the basic version) and combining X-Mesh and Text2Tex pipelines. Image control X-Mesh outperforms the original version in all cases with the ability to produce more realistic texture and geometry and the main reason is that the image contains more abundant information with context. The combined pipeline takes advantage of geometry edits by X-Mesh and smoother texture creation by Text2Tex. It is highly useful for lower-fidelity samples. Two other modifications, adding view specification text prompt to the original X-Mesh and placing higher attention on input image view in Image control X-Mesh, only make small improvements to the results. The other modifications fail to meet expectations. Editing the facade and roof of the building separately can not avoid unwanted elements on each building segment and freezing sampled vertices geometry results in nearly unchanged model geometry.

In general, the proper user scenarios can be categorized by the size and fidelity of input models. Based on the qualitative and quantitative studies, Latent-Paint, Text2Tex, Image control X-Mesh, and the Combination of X-Mesh are chosen for their relatively good performance in at least one type of building model. Details can be found in Section 5.1. Drawbacks of the current methods include limited achievable quality, generalizability and computational cost, and are further summarized in Section 5.3.

5.1. Application

For building models with different features, the proper edit pipelines vary. The major classification criteria are size and level of fidelity and four major pipelines are considered for application, namely Latent-Paint, Text2Tex, Image control X-Mesh, and Combination of X-Mesh and Text2Tex. General principles of decision are summarized based on qualitative and quantitative studies as shown in Figure 5.1.

For small and low-fidelity buildings, Latent-Paint can be first experimented with. If the result is not satisfying, Image control X-Mesh can be used. It generally has the ability to generate reasonable geometric edited and textured meshes with details for small buildings but suffers from noisy problems. Then if more smoothed and coherent texture is desired, the Text2Tex can also be combined. For larger low-fidelity buildings, image control X-Mesh can also create well-textured meshes (but with more limited geometric deformation) and then combining Text2Tex is also the suggested choice. As for more complex input meshes, Text2Tex can be the first choice. The Combination of X-Mesh and Text2Tex can still be an alternative. Lastly, for cases with severe self-occluded problems, Image control X-Mesh can be the proper choice.

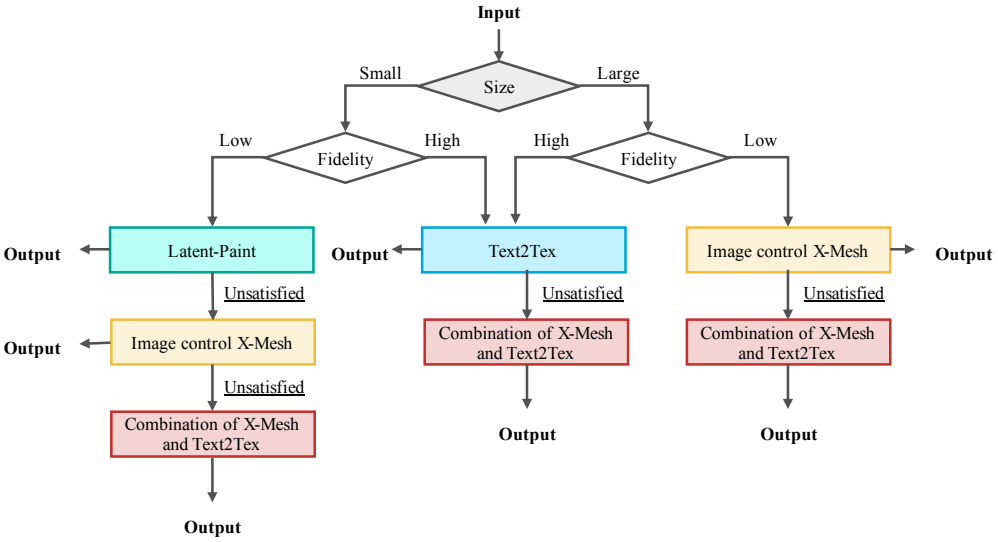


Figure 5.1.: Proposed pipelines selection diagram

Editing 3D building models with generative AI with the proposed pipelines offers an interesting and easy-to-use opportunity for the general public to explore architecture stylization and an efficient alternative for professionals to deal with buildings in large scenes or the early design stage. One application example is that the general public can use these tools to freely redesign the appearance of their own houses with text or image prompts, which can serve as preparation for renovation before they consult professionals. Another example is that urban planners can use such methods to quickly compare different design concepts of a certain region in the draft stage. It is also possible for game designers to utilize these pipelines to edit real-world buildings from open-source datasets to the target style and use them as the background of scenes.

5.2. Contribution

The thesis contributes to the exploration of generative 3D edit pipelines in the building domain. Representative pipelines in the 3D edit field are chosen. Their performances on different building models are tested and the advantages and limitations are discussed. Further modifications are made to enable the methods to better fit the demand of users in the building domain and generate high-quality results. Application scenarios of the displayed pipelines are also proposed, which helps users choose the proper method based on the features of input models.

5.3. Limitation

Limitations exist in the proposed 3D edit pipelines for the building domain. The edited buildings can not achieve very high fidelity due to the limited consistency and realism of textures. They also have limited capabilities in terms of geometric edit scope and significant deformation is difficult to achieve while preserving the identity of the original models. Thus, they can be applied in simple cases with lower requirements like comparing different ideas in the concept design stage and editing models in large-scale scenes.

As the discussed 3D edit pipelines are based on the 2D pre-trained models, they inevitably inherit the limits. The generalizability of such methods is limited. Certain engineered prompts related to training datasets of the 2D models achieve superior results than others and some very specific and professional control is not possible in the building domain. Additionally, like the implicit representation based methods, the current methods still suffer from view consistency problems to some extent.

The computational demand is another barrier to the application of generative 3D edit methods in the building domain. It is more time-consuming and difficult to edit large and complex building samples. The thesis also only explores and modifies limited types of 3D representations (only NeuS and mesh) and 3D edit pipelines due to the limited computational resources and time. Some pipelines, like Fantasia3D and Paint-it, are temporarily neglected due to their high demand in GPU memory.

5.4. Future work

To extend the potential of generative AI based 3D edit methods in the building domain, further attempts can be made. Image Diffusion Model with high capacity and less bias in the architectural domain can be trained based on the state-of-art current models if high-quality building datasets can be collected and computational needs met. A wider exploration can be done on the existing studies related to 3D edits. The network structure can also be further modified to produce significant and realistic geometric deformation and consistent and detailed texture and improve training efficiency. Finally, more user-friendly tools that combine traditional and generative AI based methods can also be developed to allow wider application.

A. Reproducibility self-assessment

A.1. Marks for each of the criteria

1. Input data: 3
2. Preprocessing: 2
3. Methods: 2
4. Computational environment: 2
5. Results: 3

A.2. Self-reflection

The results of the thesis can be relatively easily reproduced if computational resources are available. All input data are open-source and available for everyone without cost. Preprocessing can be done easily with open-source Python libraries and open-source software. For methods, the existing pipelines by other researchers are already available on GitHub and the modifications are made available. They are all easy to use. Hardware and software requirements are documented and can be easily set up. The only problem is that a good NVIDIA GPU is needed to run the scripts, which may pose restrictions on some users. All results and the parameters used are organized in an ordered way. The codes of successful modified pipelines and results are available on https://github.com/fengyingxin/MSc-Thesis.git.

B. Implicit 3D representation based edit

B.1. Methodology

B.1.1. Overview

Two directions of implicit 3D representation based edit can be summarized from the literature review. The first and intuitive one is to update the image dataset (see Section B.1.3). Instruct-nerf2nerf, which uses NeRF as the 3D representation, is a base for it due to its relatively outstanding performance. It uses the iterative dataset update technique and InstructPix2Pix as the pre-trained model. It applies a trick to maintain identity to the original model. In the image edit process, both the text prompt and the corresponding images from the original dataset are used as conditions [Haque et al., 2023]. Modifications can then be made to it.

The second one is to incorporate 2D image loss to 3D model (see Section B.1.4). SDS and PDS are two popular loss terms, with the former using image edit pre-trained models and the latter using image generation pre-trained models.

For both directions, other pre-trained models that have potential in image edit and specified generation, like InstructEdit, Dreambooth, SDEdit and Stable Zero123, can also be experimented with.

B.1.2. Preparation

A proper 3D representation needs to be selected. NeuS is chosen for its potential to represent high-fidelity 3D models and preserve geometric detials and its differentiable nature [Wang et al., 2021]. Thus, the initial input is multi-view images with camera parameters. To fit the 2D generative model to increase efficiency and quality, the images from the dataset also need to be resized accordingly. Then an initial NeuS model is trained to represent the original 3D model.

To fully exploit the power of the pre-trained model, proper text prompts need to be created. InstructPix2Pix is an important model here and follows different rules from the CLIP based principles mentioned in Section 3.2.1, and thus the discussion is focused on it (see Section B.2.3).

B.1.3. Update image dataset

Four different modifications are made and summarized in Figure B.1. The modifications of the chosen base pipeline start with module replacement. The first attempt is to change the 3D representation to NeuS, aiming to gain geometry with finer details and fewer artifacts. The second attempt is to experiment with different pre-trained models for image edit. InstructEdit and Dreambooth are possible options. Though Dreambooth is designed for personalized image generation, it can also do text-guided view synthesis, property modification and accessorization for the object in the small training dataset [Ruiz et al., 2023]. Such properties can be used for image edit. To do so, a set of images of obviously different viewpoints can be selected from the original multi-view dataset as the training samples for Dreambooth and then the tuned model is used for object-specified image generation (e.g. use the prompt a red <object> to change the colour).

View consistency has been recognized as an important problem for 3D edit pipelines and modifications can be focused on it. A direct solution is to add a procedure to the pipeline by updating the whole multi-view images dataset with both the edited image and the rendered images of all the other views from the current NeuS model. It aims to relieve the possible divergence problem caused by inconsistent edited images from different views as with such a method the images in the current dataset always come from the same 3D model and tend to be more consistent.

Pre-trained 2D models inevitably have limitations and one significant one is that it has difficulty in dealing with images from unusual angles (e.g. tilted side view). Unrealistic and inconsistent edited images may be generated from such views and undesired information may be incorporated into the NeuS model. To alleviate this problem, an additional module for view selection according to camera parameters can be added to the pipeline. The original multi-view images dataset, which usually contains more images than needed for 3D reconstruction, can be filtered beforehand and a small percentage of the images from acute views can be ignored.

SDEdit can denoise blurry images towards the desired direction and combining it into the pipelines is also a possible idea to relieve the view-consistency problem and generate higher-fidelity results. The modified Diffusion Model for denoising can be based on Stable Diffusion. In each iteration, the dataset is updated with a newly edited image, which contains different and to some extent even conflicting information. Thus, noise may be added to NeuS. When the noise accumulates, the training may go towards the wrong direction and it is hard for the model to converge. To deal with the problem in time, SDEdit can be used to denoise the images from other unedited views with the text prompt containing the similar instruction as the prompt for the image edit model. Furthermore, this trick can also be combined with the view selection technique. Key views from representative normal angles can be selected for image edit while other views can only be denoised with SDEdit. Additionally, the parameter strength (noise adding steps percentage) for SDEdit also needs to be finetuned. Higher strength means adding more noise to the input image in the forward stage to make the generated result more realistic [Meng et al., 2021].

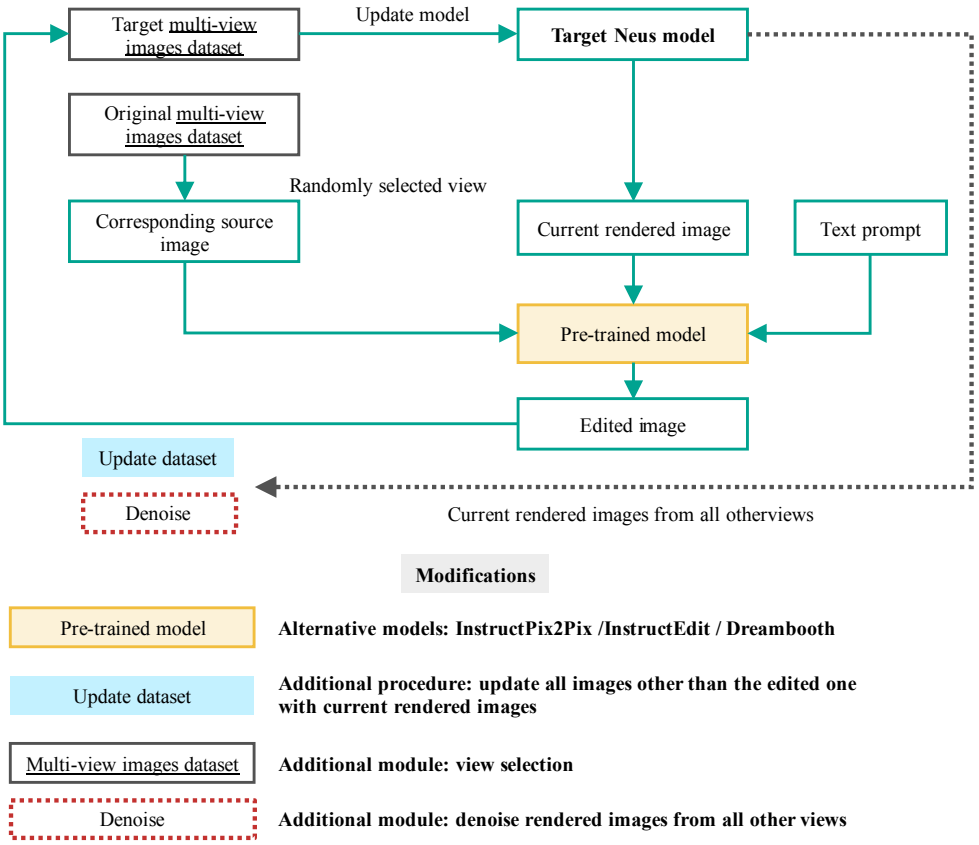


Figure B.1.: Basic pipeline of updating image dataset and four possible modifications

B.1.4. Incorporate 2D image loss to 3D model

Incorporating 2D image loss to 3D model is widely used in 3D generation and edit tasks and studies claim that it has high potential. Four modifications are made and summarized in Figure B.2. The intuitive idea is to directly use the InstructPix2Pix to calculate the SDS loss. In each iteration, the SDS loss is calculated using the rendered image of the randomly chosen view. Similar to the Instruct-Nerf2nerf pipeline, the text prompt and the original image of the same view are used as conditions in the InstructPix2Pix model to preserve the input identity. It is then added with the Eikonal loss, which is used to regularize the SDF [Wang et al., 2021]. The total loss is finally backpropagated to the NeuS model.

Using SDS loss may lose part of the identity of the original model in the training process. Therefore, further modifications can be experimented with. The first one is to try adding the loss between the rendered image and the corresponding original image with a smaller weight to the final loss, which may help to prevent the NeuS model from forgetting the original information. To avoid using the original image information twice, the InstructPix2Pix model here only uses the text prompt as the condition. The second one is to replace SDS loss with PDS loss, which is based on the text-to-image Stable Diffusion model instead and claimed to perform better in preserving identity.

Ensuring view consistency is also an important problem that needs to be considered. The edited images generated by the InstructPix2Pix may not be consistent across different views and noisy information may be backpropagated to the NeuS. The view selection techniques mentioned in Section B.1.3 may also be applied here. Another idea is to utilize the Stable Zero123 pre-trained model, which claims to generate view-consistent images of the input object. An original image from a major view, like the front view which contains more information, can be chosen and edited with the InstructPix2Pix model. Then in each iteration, camera parameters of a randomly chosen view together with the originally edited image are passed to the Stable Zero123 model to gain the newly edited image. SDS loss is calculated with the currently rendered image from the chosen view and backpropagated to the NeuS model together with the Eikonal loss.

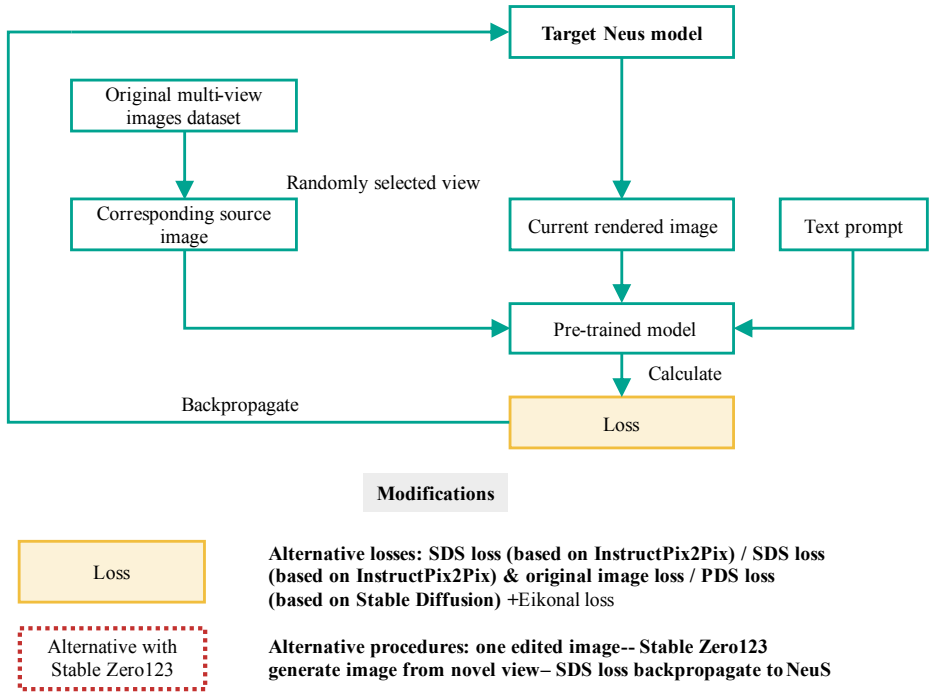
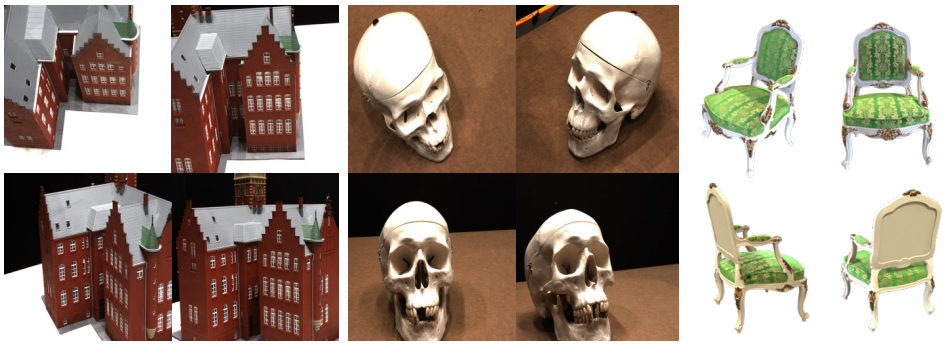


Figure B.2.: Basic pipeline of incorporate 2D image loss to 3D pipeline and four possible modifications

B.2. Results and analysis

B.2.1. Data

The data required for implicit 3D representation edit experiments is 3D models represented by multi-view images with camera parameters. The major datasets selected are the DTU MVS Dataset and NeRF-Synthetic Dataset. The first one is available *here* and a prepossessed version for 3D model reconstruction can be found *here*. The second one is available *here*. The major samples used in the experiments are shown in Figure B.3.



Building
(DTU MVS Dataset: scan24)

Skull
(DTU MVS Dataset: scan65)

Chair
(NeRF-Synthetic Dataset : chair)

Figure B.3.: Major multiple view images samples used in the thesis

B.2.2. Experiment setup

For the NeuS model, the network structure and parameters are set as default. The input images are resized to 512×512. In the InstructPix2Pix, the text CFG ranges from 1.5 to 2, the image CFG ranges from 7.5 to 12, and the diffusion step ranges from 25 to 50, which are tuned to get better results for different cases. In the SDEdit, the noise adding steps percentage ranges from 0.3 to 0.6 as suggested by the paper [Meng et al., 2021]. The weight of the Eikonal loss is set to be 10% of the other image-guided loss. As for the training times, the initial NeuS model is trained for 20000 iterations. In the dataset update methods, the number of training iterations ranges from 100 to 1000. And in each iteration, the NeuS model is trained for 10 to 100 iterations. In the 2D loss incorporation methods, the number of training iterations ranges from 2000 to 15000.

The experiments are done on a NVIDIA GPU with 24GB memory (RTX 3090 or A10). The experiments take up to one day for one building sample for dataset update pipelines and up to half a day for 2D image loss incorporation pipelines.

B.2.3. Success cases

To attain successful 3D edit results, experiments are first conducted with InstructPix2Pix in 2D space to find the general principles of prompts and CFGs. The prompt should be short and precise, and focus on the major feature of the input image. A certain format (e.g. make it something) is needed to hint the model. InstructPix2Pix also requires tuning the image and text CFG weight. Higher image CFG weight poses more restrictions of the original image while higher text CFG allows the result to change more towards the instruction [Brooks et al., 2022]. The two weights should be finetuned to strike a balance between preserving original details and making enough changes.

Before conducting experiments on buildings, tests are made on simple and common objects to figure out the potential of pipelines. The initial idea that replaces the NeRF representation with NeuS in the Instruct-nerf2nerf pipeline, is tested on a chair sample with the text prompt that only changes the colour (see the first example in Figure B.4). The experiment shows that it is edited successfully in general. The result preserves the identity of the original object,

changes consistently following the prompt across different views and still has relatively high-quality geometry.

A slightly difficult case also succeeds. It is a skull sample with adding accessories as the text prompt. In the dataset update pipeline that combines the InstructPix2Pix and SDEdit, the accessory is added to the correct location and integrated well with the initial model (see the second example in Figure B.4). The result is also consistent and the high-fidelity geometry is preserved. Another similarly successful case is experimented with incorporating 2D image loss into the 3D model pipeline that uses SDS loss with Stable Zero123 (see the third example in Figure B.4). Though a small part of geometric details loss and tiny view-inconsistent problems exist, the result is satisfying.

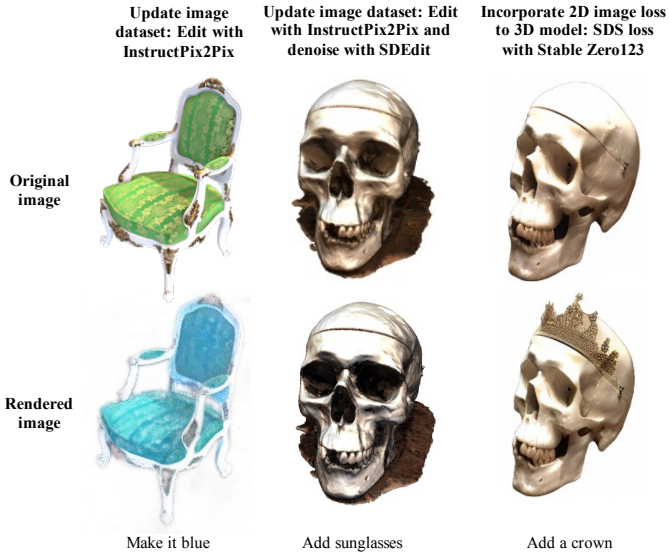


Figure B.4.: Results of successful cases: the edited models follow text prompts and the identity of the original model is preserved

B.2.4. Failure cases

The implicit 3D representation based pipelines do not perform well on the building sample with the text prompt that modifies its style (aiming for both texture and geometry changes) in general.

Update image dataset

The first experiment is also conducted on the simplest modification that uses NeuS instead of NeRF. The result fails to meet the expectation and the 3D model can not reach convergence as shown in Figure B.5. There are several obvious problems that can be observed in this experiment. The most severe one is that the edited images vary a lot across different views. Such inconsistency is so severe in the tested example that it is nearly impossible for the target NeuS model to converge and the newly rendered images become blurred. This further leads

to undesired edited images, which differ a lot from the original image. Such a trend seems to show little possibility for the NeuS model to converge as the updated dataset becomes further away from the desired direction. The second one is that it inevitably inherits the limits of InstructPix2Pix, meaning that it is impossible to make big geometrical changes on complex objects while preserving the identity. Besides, the normal views (like the parallel front view) editing can preserve more original details and in general performs much better in the tested examples than the acute angles, which may generate strange results at the very beginning (see Figure B.6). The last one is that the pipeline is not very efficient and one iteration costs more than half a minute on a single RTX 3090 GPU.

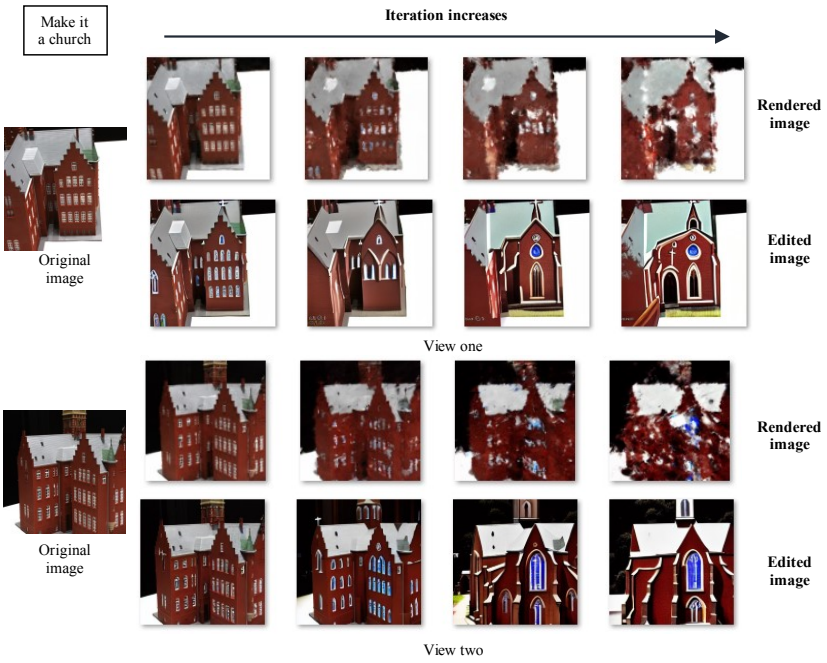


Figure B.5.: Represented rendered images and edited images during the training process: rendered images become blurry and edited images differ more and more from the original ones and the desired direction.

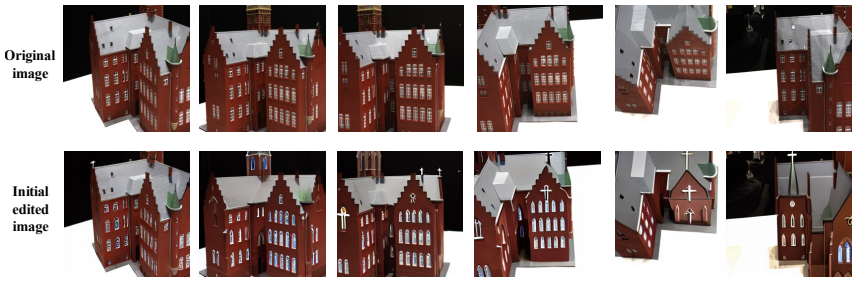


Figure B.6.: Original images and edited images during the first round of editing from represented views: the multi-view consistency problem exists, and normal views editing outperforms acute ones.

Experiments are then conducted on the alternative pre-trained 2D models. For the InstructEdit, the edited results are not stable and generally perform worse than the InsturctPix2Pix model. The possible reason is that the segmenter sometimes fails to identify the correct part, which causes failure in the following edit. Additionally, more computation resources are needed, adding to the existing inefficient problem. For the Dreambooth, the finetuned model also fails to generate edited images that preserve the identity of the input object. Furthermore, it shares similar problems as the InsturctPix2Pix that image quality is not stable across views and the edited results change towards different directions. Besides, the training set of Dreambooth usually requires the object images to have different backgrounds and postures to better incorporate the new object with the learned semantic information. But the images from the multi-view dataset have the same background and posture and the quality of the tuned model may be lower.

The third modification is updating the whole dataset at each iteration and it also fails for the building sample (see the first example in Figure B.7). The target NeuS model can not converge and the rendered images become blurry. The cause is still that the relatively highly inconsistent edited multi-view images guide the target dataset and the 3D model to change to conflicting directions and thus result in an apparent divergence trend.

The next modified attempt is view selection. However, this idea is difficult to implement automatically. For different input objects, views that perform well in the pre-trained image edit model differ and certain randomness and uncertainty are involved. Thus it is difficult to define a universal rule to select the desired view. An attempt is made to manually select views with relatively desired and consistent edit results initially (see the second example in Figure B.7). To ensure enough views are provided for the training of NeuS model, the neglected acute view takes up less than 15% of the original views. In general, the model performs better after selection and a convergence trend can be observed in the training process. However, the final rendered images are still a bit blurry and the depth and normal estimate is noisy.

The last modification is to use the InsturctPix2Pix and SDEdit models together. Though the SDEdit has the risk of adding additional variation to images, this attempt seems to perform better than others (see the third example in Figure B.7). In the final result, the rendered images successfully follow the text prompt (i.e. the window shapes of the building change). But noise can be seen around the building. Besides, the depth estimate is also not precise, resulting in holes and rough surfaces in the generated mesh.

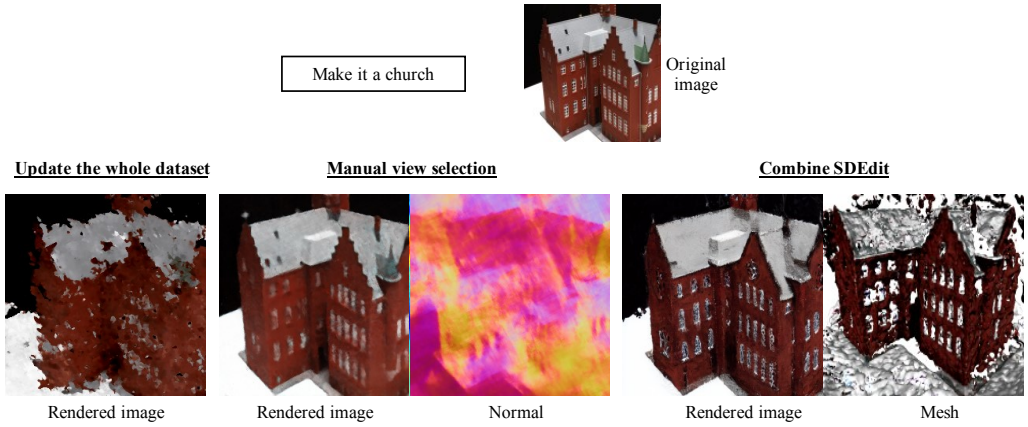


Figure B.7.: Results of updating image dataset pipeline modifications: all fail to generate desired texture and geometry. The first one diverges, the second one is blurry and the third one suffers from the noise problem.

Incorporate 2D image loss to 3D model

In this direction, the first attempt is to directly use SDS loss term with InsturctPix2Pix (see the first example in Figure B.8). However, the detailed geometric and textural features are forgotten in the training process though the overall structure is preserved. Experiments on both the chair and the building cases fail. The chair case converges towards an undesired direction, losing photometric and geometric details. The building cases with different text prompts show a divergence trend as it is more complex and thus the loss of details in the training process leads to the loss of identity. The final rendered result becomes totally blurry. Adding the loss of the original image also failed. After testing on different combinations of weights, the rendered images still remain nearly unchanged in the training process. A possible reason is that such a loss term poses too strict restrictions on the model and thus the model remains identical to the original one.

Then the PDS loss term is tested [Koo et al., 2023] and the result is similar to the SDS loss experiment (see the second example in Figure B.8). Identity and geometric details are not preserved and the rendered images become blurry. It seems that the new loss term still can not overcome the limitations of view inconsistency in 2D pre-trained models.

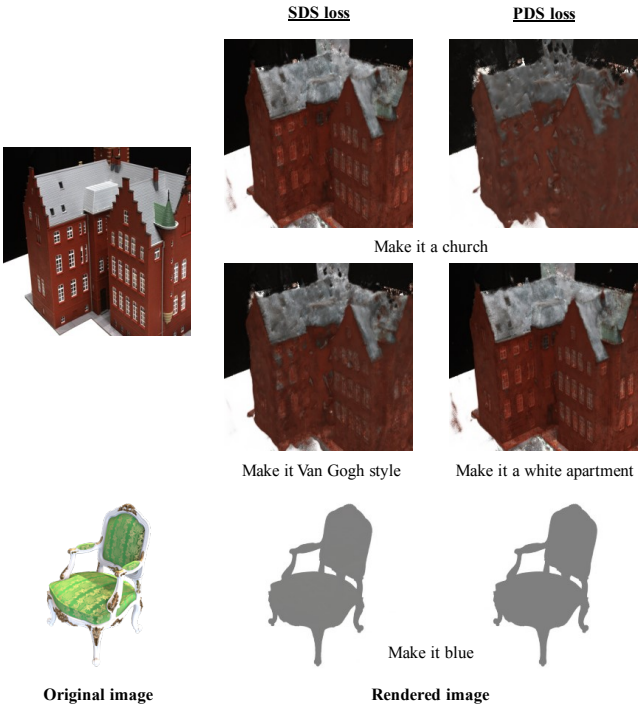


Figure B.8.: Results of incorporating 2D image loss to 3D pipeline modifications: different loss terms fail to guide the NeuS model towards the text instructed direction.

The final pipeline is using the SDS loss term with Stable Zero123. For the building case, obvious problems already arise when testing in 2D space as shown in Figure B.9. The Stable Zero123 seems to have difficulty in inferring consistent novel views for complex objects like buildings. The building already deforms wrongly in shape and loses some details when inferring an angle slightly different from the original one. In other novel views, the overall structure of the building is changed.

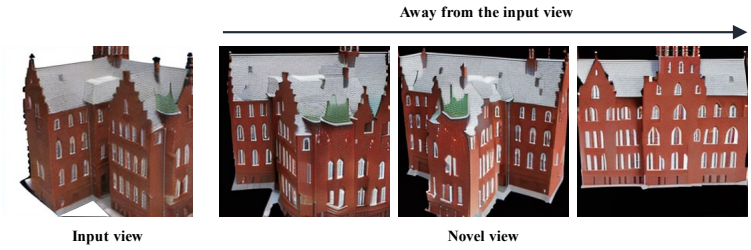


Figure B.9.: Results of the building case of Stable Zero123: images from novel views lose details and fail to preserve identity.

B.2.5. Analysis

Certain experimented implicit 3D representation based pipelines (mentioned in Section B.2.3) can be used in very simple cases. The successful cases use common objects without complex structures as inputs and text prompts only require mild edits without significant geometric changes. It shares the advantages of the implicit representation, which can render high-resolution images. It can be applied in cases that require editing high-fidelity common 3D objects slightly. Another limitation is that such pipelines are not very efficient.

However, in more complex cases especially for buildings, such pipelines fail to generate satisfying results. The possible reason can be summarized into two parts. The first one is that these pipelines inherit the limitations of the selected pre-trained models. These pre-trained models are designed for general cases and the training datasets have limited cases from the architectural domain (especially for the InstructPix2Pix model as shown in Figure 1.2) and thus have limited ability to generate desired building image edit results. Furthermore, except for the Stable Zero123 model, pre-trained models are not designed for generating view-consistent images and significant variation can usually be observed across different views. The noisy information in the 2D space accumulates and brings about more severe problems in 3D models.

The second reason is the nature of the NeuS model. The geometry and colour of the object are represented by the networks and interconnected. It may lead to the instability of the model, meaning that changing one part also possibly influences the other, and small noisy information may quickly lead the model towards the wrong direction. Trained by the conflicting information from the 2D space, the model may finally suffer from a divergent trend or convergent towards a blurry stage (i.e. strike a balance between the conflicting views and thus lose the detailed features).

Based on the observation and the deduced reasons, the thesis turns to explicit 3D representation based pipelines, which may be more promising. The explicit representation may be more stable, as the geometry and colour are usually represented separately and explicitly. Therefore, the training of such representation may be more robust when dealing with the inevitable noise.

Bibliography

Armandpour, M., Zheng, H., Sadeghian, A., Sadeghian, A., and Zhou, M. (2023). Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *arXiv preprint arXiv:2304.04968*.

Bandi, A., Adapa, P. V. S. R., and Kuchi, Y. E. V. P. K. (2023). The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8):260.

Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved lod specification for 3d building models. *Computers, Environment and Urban Systems*, 59:25–37.

Brooks, T., Holynski, A., and Efros, A. A. (2022). Instructpix2pix: Learning to follow image editing instructions. *arXiv e-prints*, pages arXiv–2211.

Chao, C.-K. T. and Gingold, Y. (2023). Text-guided image-and-shape editing and generation: A short survey. *arXiv preprint arXiv:2304.09244*.

Chen, D. Z., Siddiqui, Y., Lee, H.-Y., Tulyakov, S., and Nießner, M. (2023a). Text2tex: Text-driven texture synthesis via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18558–18568.

Chen, M., Xie, J., Laina, I., and Vedaldi, A. (2023b). Shap-editor: Instruction-guided latent 3d editing in seconds. *arXiv preprint arXiv:2312.09246*.

Chen, R., Chen, Y., Jiao, N., and Jia, K. (2023c). Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*.

Chen, Y., Chen, R., Lei, J., Zhang, Y., and Jia, K. (2022). Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *Advances in Neural Information Processing Systems*, 35:30923–30936.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., et al. (2008). Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy.

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794.

Dong, J. and Wang, Y.-X. (2023). Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Fang, S., Wang, Y., Yang, Y., Tsai, Y.-H., Ding, W., Zhou, S., and Yang, M.-H. (2023). Editing 3d scenes via text prompts without retraining. *arXiv e-prints*, pages arXiv–2309.

Gozalo-Brizuela, R. and Garrido-Merchán, E. C. (2023). A survey of generative ai applications. *arXiv preprint arXiv:2306.02781*.

Guo, Y.-C., Liu, Y.-T., Shao, R., Laforte, C., Voleti, V., Luo, G., Chen, C.-H., Zou, Z.-X., Wang, C., Cao, Y.-P., and Zhang, S.-H. (2023). threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio.

Haque, A., Tancik, M., Efros, A. A., Holynski, A., and Kanazawa, A. (2023). Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*.

He, Y., Bai, Y., Lin, M., Sheng, J., Hu, Y., Wang, Q., Wen, Y.-H., and Liu, Y.-J. (2023). Text-image conditioned diffusion for consistent text-to-3d generation. *arXiv preprint arXiv:2312.11774*.

Hessel, J., Holtzman, A., Forbes, M., Bras, R. L., and Choi, Y. (2021). Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hong, S., Ahn, D., and Kim, S. (2023). Debiasing scores and prompts of 2d diffusion for view-consistent text-to-3d generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Huberman-Spiegelglas, I., Kulikov, V., and Michaeli, T. (2023). An edit friendly ddpm noise space: Inversion and manipulations. *arXiv preprint arXiv:2304.06140*.

Jpcy (2022). Xatlas. https://github.com/jpcy/xatlas.

Kamata, H., Sakuma, Y., Hayakawa, A., Ishii, M., and Narihira, T. (2023). Instruct 3d-to-3d: Text instruction guided 3d-to-3d conversion. *arXiv preprint arXiv:2303.15780*.

Koo, J., Park, C., and Sung, M. (2023). Posterior distillation sampling. *arXiv preprint arXiv:2311.13831*.

Li, C., Zhang, C., Waghwase, A., Lee, L.-H., Rameau, F., Yang, Y., Bae, S.-H., and Hong, C. S. (2023a). Generative ai meets 3d: A survey on text-to-3d in aigc era. *arXiv preprint arXiv:2305.06131*.

Li, W., Chen, R., Chen, X., and Tan, P. (2023b). Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *arXiv preprint arXiv:2310.02596*.

Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., and Vondrick, C. (2023). Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309.

Ma, Y., Zhang, X., Sun, X., Ji, J., Wang, H., Jiang, G., Zhuang, W., and Ji, R. (2023). X-mesh: Towards fast and accurate text-driven 3d stylization via dynamic textual guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2749–2760.

Meng, C., He, Y., Song, Y., Song, J., Wu, J., Zhu, J.-Y., and Ermon, S. (2021). Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*.

Metzer, G., Richardson, E., Patashnik, O., Giryes, R., and Cohen-Or, D. (2023). Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673.

Michel, O., Bar-On, R., Liu, R., Benaim, S., and Hanocka, R. (2022). Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*.

Mohammad Khalid, N., Xie, T., Belilovsky, E., and Popa, T. (2022). Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8.

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Pharmapsychotic (2023). clip-interrogator. https://github.com/pharmapsychotic/clip-interrogator.

Po, R., Yifan, W., Golyanik, V., Aberman, K., Barron, J. T., Bermano, A. H., Chan, E. R., Dekel, T., Holynski, A., Kanazawa, A., et al. (2023). State of the art on diffusion models for visual computing. *arXiv preprint arXiv:2310.07204*.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Richardson, E., Metzer, G., Alaluf, Y., Giryes, R., and Cohen-Or, D. (2023). Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*.

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510. IEEE.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Shen, T., Gao, J., Yin, K., Liu, M.-Y., and Fidler, S. (2021). Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101.

Shi, Y., Wang, P., Ye, J., Long, M., Li, K., and Yang, X. (2023). Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*.

Song, L., Cao, L., Gu, J., Jiang, Y., Yuan, J., and Tang, H. (2023). Efficient-nerf2nerf: Streamlining text-driven 3d editing with multiview correspondence-enhanced diffusion models. *arXiv preprint arXiv:2312.08563*.

Stabilityai (2023). Stable zero123. https://huggingface.co/stabilityai/stable-zero123.

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*.

Wang, Q., Zhang, B., Birsak, M., and Wonka, P. (2023a). Instructedit: Improving automatic masks for diffusion-based image editing with user instructions. *arXiv preprint arXiv:2305.18047*.

Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., and Zhu, J. (2023b). Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Wu, C. H. and De la Torre, F. (2023). A latent space of stochastic diffusion models for zero-shot image editing and guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7378–7387.

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., and Lipman, Y. (2020). Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33:2492–2502.

Youwang, K., Oh, T.-H., and Pons-Moll, G. (2023). Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. *arXiv preprint arXiv:2312.11360*.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.