Wrocław University of Science and Technology
Faculty of Information and Communication Technology

Field of study: **Artificial Intelligence**
Speciality: **---**

MASTER THESIS

A Method for Image Generation Using Conditional Multi-Views

Eryk Wójcik

Supervisor
dr Kamil Adamczewski

Machine Learning, Generative Models, Diffusion

WROCŁAW 2025

Streszczenie

Dodaj streszczenie pracy w języku polskim. Staraj się uwzględnić wymienione na stronie tytułowej słowa kluczowe. Uwaga przedstawiony rekomendowany szablon dotyczy pracy dyplomowej pisanej w języku angielskim. W przeciwnym wypadku, student powinien samodzielnie zmienić nazwy „Chapter" na „Rodział" itp stosując odpowiednie pakiety systemu LaTeXoraz ustawienia w pliku *latex-settings.tex*.

Abstract

Streszczenie w języku angielskim.

Contents

1 Introduction TODO **1**
- 1.1 Problem Statement . . . 1
- 1.2 Thesis Objectives . . . 1

2 Related Work **3**
- 2.1 Traditional 3D Reconstruction Approaches . . . 3
- 2.2 Deep Generative Models for Image Synthesis . . . 7
- 2.3 Conditioning Diffusion Models for Enhanced Control and New Tasks . . . 11
- 2.4 Diffusion-based Multi-View Image Generation . . . 16

3 Proposed Method **19**
- 3.1 Overview . . . 19
- 3.2 Limitations of Previous Methods . . . 19
- 3.3 Architectural Framework . . . 20
- 3.4 Model Training . . . 23
- 3.5 Inference Process . . . 24

4 Data Preparation **27**
- 4.1 Dataset Overview . . . 27
- 4.2 Processing Pipeline for ObjaverseXL Training Data . . . 28
- 4.3 Processing of Google Scanned Objects (GSO) Validation Data . . . 32
- 4.4 Training Dataset Characteristics . . . 32

5 Experiments **39**
- 5.1 Experimental Setup . . . 39
- 5.2 Performed Experiments . . . 41
- 5.3 Comparison with State-of-the-Art Methods on GSO Dataset . . . 47
- 5.4 Chapter Summary . . . 48

6 Conclusions and Future Work TODO **51**

1. Introduction TODO

W pracy formułuje się cele o charakterze badawczym wymagające doboru i zastosowania metod badawczych, wykorzystując wiedzę teoretyczną oraz naukową. Wskazane jest przedstawienie, co nowego jest zaproponowane w pracy oraz podanie ograniczeń i słabych/mocnych stron opracowanego rozwiązania (jeżeli dotyczy). Rozdział wprowadzający powinien służyć czytelnikowi do zrozumienia celu pracy.

1.1. Problem Statement

W tej sekcji student powinien przedstawić bliżej problem, którym chce się zmierzyć. Jasno zdefiniuj problem badawczy. Podaj swoje cele, zadania i pytania badawcze. Wyjaśnij znaczenie badania. Określ ograniczenia badań.

1.2. Thesis Objectives

W tej sekcji powinny zostać przedstawione konkretne działania, które określają pracę studenta w celu rozwiązania problemu.

2. Related Work

In this chapter, I provide a comprehensive review of existing approaches relevant to multi-view image generation.

The chapter begins by introducing the fundamental task of Novel View Synthesis (NVS) and its significance. We then delve into traditional 3D reconstruction techniques (Section 2.1) and their inherent limitations, particularly for sparse-input scenarios, which motivates the exploration of generative methods. Subsequently, the discussion shifts to the foundational principles of Deep Generative Models for Image Synthesis, with a focus on diffusion models (Section 2.2).

Building on this, we explore various methods for Conditioning Diffusion Models for Enhanced Control and New Tasks (Section 2.3), including the crucial role of camera parameter encoding and the concept of lightweight adaptation through adapters.

The core of the chapter then examines state-of-the-art Diffusion-based Multi-View Image Generation techniques (Section 2.4), covering both single reference image novel view synthesis and architectures for coherent multi-view generation, including specialized multi-view adapters.

2.1. Traditional 3D Reconstruction Approaches

Simultaneous Localization and Mapping (SLAM) is a fundamental concept in the field of computer vision and robotics. It refers to the process of simultaneously estimating the camera's position and orientation in a 3D environment while mapping the environment itself. Originally, SLAM was used to track the position of a robot in a 3D space, but it has since been applied to a wide range of problems, including augmented reality, medical applications and novel view synthesis.

SLAM works by processing sensor data in real-time to create a map of the unknown environment while simultaneously tracking the position of the sensor within that map. This process typically involves several key steps:

1. **Feature Detection**: Identifying distinctive points or features in the environment from sensor data
2. **Data Association**: Matching observed features with previously mapped features
3. **State Estimation**: Updating the estimated pose of the camera and the map of the environment
4. **Loop Closure**: Recognizing when the sensor has returned to a previously visited location and adjusting the map accordingly to reduce accumulated errors

To address these requirements, researchers have concentrated on creating techniques for machines to independently build ever more precise scene representations. This integration of

robotics, computer vision, sensor technology, and recent advancements in artificial intelligence has shaped this field.

Typically, SLAM techniques use a combination of data sources, such as images, laser range scans, sonars and GPS to effectively map an environment. In this work I will focus on the use of images for 3D reconstruction.

2.1.1. Structure from Motion Pipeline

One of the most popular methods for 3D reconstruction from images is COLMAP [31]. COLMAP is a general-purpose structure-from-motion (SfM) [14] system that can automatically reconstruct 3D scenes from a collection of images. It is a popular choice for 3D reconstruction due to its accuracy, speed, and ease of use.

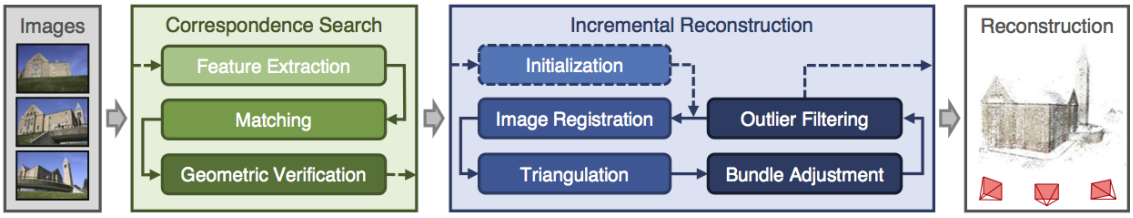

Figure 2.1: COLMAP pipeline

Structure from Motion works in two main steps [Figure 2.1]:

1. **Correspondence Search**: Identify the unique landmarks (features) in all of the images and match the same landmarks across images.

2. **Incremental Reconstruction**: Estimates the camera poses and triangulates 3D points through an iterative process.

First step of correspondence search is feature extraction. COLMAP uses SIFT [21] and ORB [30] methods to extract features from images. Scale-invariant feature transform (SIFT) is a method for detecting keypoints that contrast with their surroundings and describing the local image content around them. It is invariant to rotation and scale, making it robust for matching features across images taken from different viewpoints and distances. Oriented FAST and Rotated BRIEF (ORB) is a more computationally efficient alternative to SIFT, combining a high-speed FAST detector with optimized descriptors for rotation invariance, offering comparable matching performance with significantly reduced computational requirements.

Second step is matching features across images. Feature matching is a process of finding the best matches of features across the images. COLMAP uses a variant of the FLANN [25] library to find the best matches for each feature. FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. Feature matching is visualized in Figure 2.2.

Then the geometric verification step uses the prior knowledge about the camera model and motion to remove outliers, preparing the verified feature matches for the subsequent Incremental Reconstruction phase.

When it comes to Incremental Reconstruction, the process is as follows:

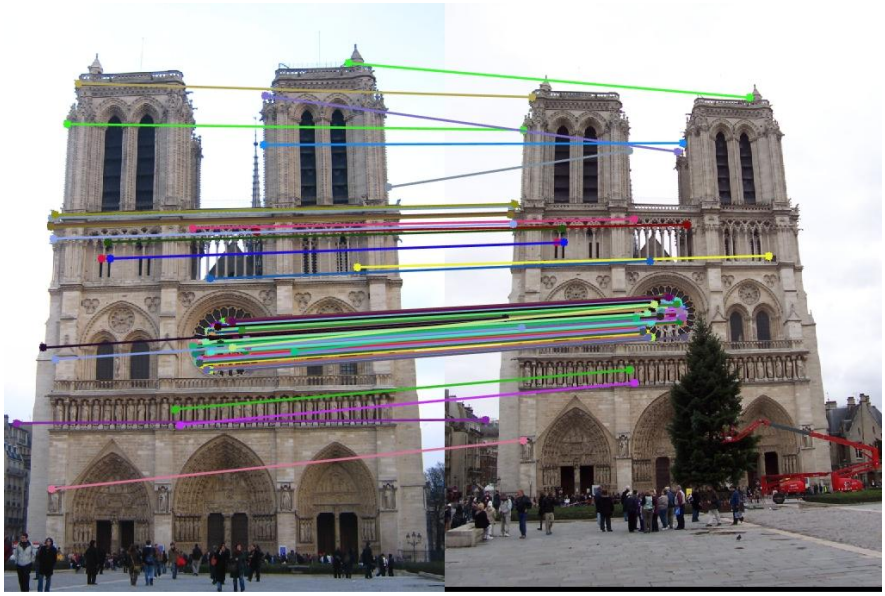

Figure 2.2: Feature matching result

1. **Camera Pose Estimation**: Estimate the camera location and direction in 3D space for each image.
2. **Triangulation**: Triangulate 3D points of the observed objects from the camera poses and matched features.
3. **Bundle Adjustment**: Refine the camera poses and 3D points to minimize the reprojection error.

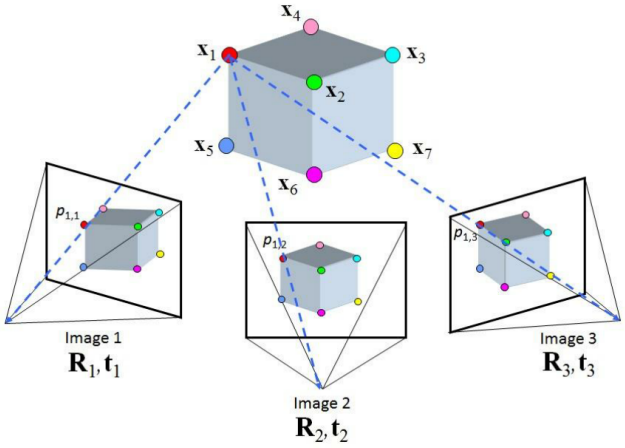

Figure 2.3: Camera pose estimation

The camera pose estimation (visualized in Figure 2.3) starts with an initialization step, where the initial pair of images and matched features between them are used to estimate the camera pose. Then, the algorithm proceeds to the loop (Figure 2.1), where the next image

is registered to the reconstruction and the camera pose is estimated again (triangulation step). After that the bundle adjustment step is performed to refine the camera poses and 3D points to be consistent with the entire dataset with images of a given scene. This optimization process typically uses the Levenberg-Marquardt algorithm to minimize reprojection error. This process is repeated for each subsequent batch of images, updating the camera poses and 3D points.

2.1.2. Methods for 3D Reconstruction

The field of 3D reconstruction encompasses various approaches beyond the basic Structure from Motion pipeline. These methods can be broadly categorized into two groups: sparse reconstruction (like SfM) and dense reconstruction methods.

Dense Reconstruction Methods

While SfM provides camera poses and a sparse point cloud, dense reconstruction methods aim to create a complete 3D model with detailed surface information. Notable methods include:

1. **Multi-View Stereo (MVS)**: After obtaining camera poses through SfM, MVS algorithms like PMVS [9] generate dense point clouds by matching pixels across multiple images.

2. **Depth Map Fusion**: Methods such as COLMAP's MVS pipeline estimate per-image depth maps and then fuse them into a consistent 3D model.

3. **Neural Radiance Fields (NeRF)** [24]: A more recent approach that represents scenes as continuous 5D functions (spatial location and viewing direction) encoded in neural networks. NeRF takes camera poses from SfM as input and uses ray tracing to synthesize novel views with remarkable detail and view consistency.

Learning-based Reconstruction

Recent approaches leverage deep learning for 3D reconstruction, often using SfM-derived data for supervision or initialization:

1. **Learned MVS**: Methods like MVSNet [37] use convolutional neural networks to learn the depth estimation process directly from images and camera parameters.

2. **Single-view Reconstruction**: Networks like Mesh R-CNN [11] can estimate 3D structure from a single image by leveraging prior knowledge learned from large datasets.

2.1.3. Limitations of 3D Reconstruction

Structure from Motion is a powerful tool for 3D reconstruction, demonstrating high effectiveness across a variety of scenarios. However, it encounters significant challenges. These include dealing with textureless surfaces, reflective materials, and the computational

complexity of processing high-resolution images. Most importantly in the context of this thesis, it struggles with sparse input scenarios, such as those involving a single image or only a few images.

Traditional 3D reconstruction methods like SfM and MVS typically require a dense collection of images with sufficient overlap to establish accurate feature correspondences and camera pose estimations. When faced with limited input views—particularly in the extreme case of a single image—these methods often fail to generate complete and accurate 3D representations. The quality of reconstruction degrades significantly due to:

1. **Geometric ambiguity**: A single image or sparse set of images provides incomplete information about occluded regions and depth, leading to ambiguous geometry.
2. **Feature matching limitations**: Fewer images means fewer opportunities to establish reliable feature correspondences across different viewpoints.
3. **Inability to triangulate**: Robust triangulation requires features to be visible from multiple viewpoints, which is not possible with very limited inputs.
4. **View-dependent effects**: Materials with specular reflections or varying appearance based on viewpoint cannot be accurately modeled without multiple observations.

These limitations have motivated the development of generative approaches to novel view synthesis, particularly using diffusion models trained on large datasets of rendered images of 3D models. Instead of explicitly reconstructing geometry, these methods leverage the power of deep learning to hallucinate plausible views from unseen perspectives.

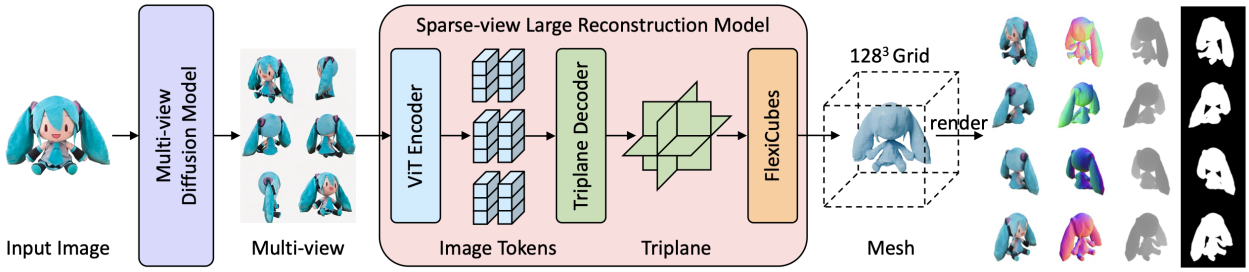

Figure 2.4: InstantMesh

Modern approaches of 3D object generation, such as InstantMesh [36], leverage these diffusion models to generate multiple consistent views of an object from minimal input (a single image or even a text prompt). This novel view synthesis is the first step in a pipeline 2.4 that can be used to create a 3D model of the object, and it is this particular step that forms the central focus of this thesis.

2.2. Deep Generative Models for Image Synthesis

The development of deep learning methods has led to remarkable progress in generative modeling, enabling the synthesis of highly realistic and diverse images. Among the various

approaches, Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models have emerged as the most prominent. While GANs and VAEs have laid significant groundwork and continue to be influential, this section will primarily focus on diffusion models, given their state-of-the-art performance in image quality and controllability, and their direct relevance to the multi-view synthesis tasks explored in this thesis.

2.2.1. Generative Adversarial Networks and Variational Autoencoders

Generative Adversarial Networks (GANs) [12] consist of two neural networks, a generator and a discriminator, trained in a competitive setting. The generator aims to produce realistic images, while the discriminator tries to distinguish between real images from the training dataset and fake images produced by the generator. Through this adversarial process, the generator learns to create increasingly plausible images. GANs are known for generating sharp images but often suffer from training instability.

Variational Autoencoders (VAEs) [19] are another class of generative models that learn a probabilistic mapping from a high-dimensional data space (e.g., images) to a lower-dimensional latent space, and then back to the data space. A VAE consists of an encoder that compresses the input data into a latent representation (typically a mean and variance defining a Gaussian distribution) and a decoder that reconstructs the data from samples drawn from this latent distribution. They are trained to maximize the evidence lower bound (ELBO), which involves a reconstruction loss and a regularization term (KL divergence) that encourages the latent space to be smooth and well-behaved. VAEs generally offer more stable training than GANs and can learn a meaningful latent space, but often produce slightly blurrier images. VAEs play a crucial role in the architecture of Latent Diffusion Models.

2.2.2. Diffusion Models

Diffusion models have recently become the dominant paradigm in high-fidelity image generation. They are inspired by non-equilibrium thermodynamics, specifically diffusion processes.

Core Concept: The Diffusion Process

The core idea behind diffusion models involves two processes: a forward (or diffusion) process and a reverse (or denoising) process. In the **forward process**, a known image x_0 from the dataset is gradually perturbed by adding small amounts of Gaussian noise over a sequence of T steps. This process progressively corrupts the image until, at step T, it becomes indistinguishable from pure isotropic Gaussian noise. The parameters of this noising process are fixed.

The **reverse process** aims to learn to reverse this noising. Starting from pure noise (equivalent to x_T), a neural network is trained to gradually denoise the signal, step-by-step, eventually producing a realistic image (an approximation of x_0). This learned denoising process is what allows the model to generate new images. Figure 2.5 illustrates this concept.

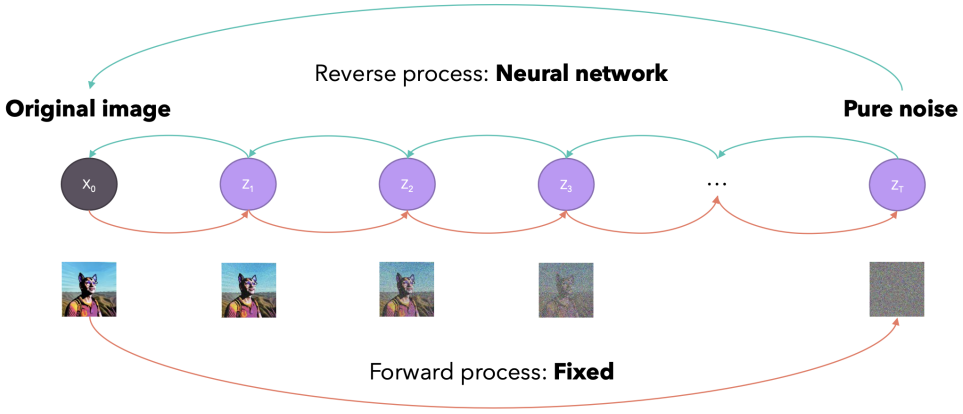

Figure 2.5: The forward (noising) and reverse (denoising/generation) stages of a diffusion model. The forward process gradually adds noise to an image until it becomes pure noise. The reverse process learned by a neural net to denoise, step-by-step, to generate an image from noise.

The U-Net Architecture

The neural network responsible for predicting the noise (or the denoised image) at each step in the reverse process is typically a U-Net architecture [29]. The U-Net, originally developed for biomedical image segmentation, features an encoder-decoder structure with skip (residual) connections. The encoder path progressively downsamples the input, capturing contextual information, while the decoder path progressively upsamples, localizing information. The skip connections concatenate features from the encoder to corresponding layers in the decoder, allowing the network to combine high-level semantic information with low-level detail, which is crucial for accurately predicting the noise and preserving image fidelity during the denoising steps.

Training Objective and Loss Function

The training objective of the diffusion model is to learn the conditional probability distribution $p_\theta(x_{t-1}|x_t)$, which represents the probability of the previous, less noisy state x_{t-1} given the current noisy state x_t. In practice, this is often simplified to training the U-Net to predict the noise ϵ that was added to an image x_0 to produce x_t at a given timestep t. The loss function is typically the Mean Squared Error (MSE) between the true added noise ϵ and the noise $\epsilon_\theta(x_t, t)$ predicted by the U-Net:

$$L = \mathbb{E}_{t\sim[1,T],x_0\sim q(x_0),\epsilon\sim\mathcal{N}(0,I)}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$$

where $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$, and $\bar{\alpha}_t$ are parameters from the noise schedule.

Conditional Generation: Text-to-Image Synthesis

To guide the image generation process, diffusion models can be conditioned on various inputs, most notably text prompts. This is the foundation of text-to-image synthesis. A

crucial component for text conditioning is a powerful text encoder that can convert textual descriptions into rich numerical representations (embeddings). Contrastive Language-Image Pre-training (CLIP) [27] is widely used for this purpose. CLIP is trained on a massive dataset of image-text pairs to learn a shared embedding space where semantically similar images and texts are close together.

The text embeddings from CLIP are then integrated into the U-Net, typically using cross-attention mechanisms. In these layers, the image representation at an intermediate layer of the U-Net queries the text embedding, allowing the model to align parts of the image with relevant words or phrases in the prompt. This enables fine-grained control over the generated image content based on the textual input.

2.2.3. Latent Diffusion Models (LDMs)

While standard diffusion models operate directly in the pixel space of images, this can be computationally very expensive, especially for high-resolution images, as the U-Net needs to process large tensors. Latent Diffusion Models (LDMs) [28], such as Stable Diffusion, address this challenge by performing the diffusion and denoising process in a lower-dimensional latent space.

LDMs employ a pre-trained autoencoder, typically a VAE. The VAE's encoder first compresses a high-resolution image from pixel space into a compact latent representation. The diffusion process (both forward and reverse) then occurs entirely within this latent space. Once the reverse denoising process generates a target latent representation from noise, the VAE's decoder maps this latent representation back into the high-resolution pixel space to produce the final image. Figure 2.6 depicts the architecture of an LDM.

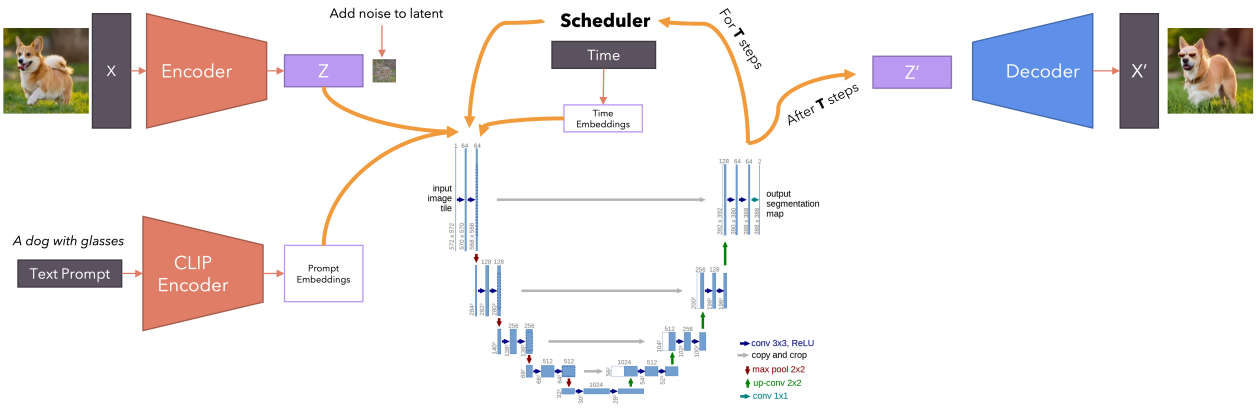

Figure 2.6: Architecture of a Latent Diffusion Model (LDM). An image is first encoded into a latent space by a VAE encoder. The diffusion process (noising and denoising via U-Net) occurs in this latent space. The generated latent is then decoded back to pixel space by the VAE decoder. Conditioning, such as text embeddings from CLIP, is incorporated into the U-Net.

By working in a compressed latent space, LDMs significantly reduce the computational burden during training and inference compared to pixel-space diffusion models. This makes it feasible to train powerful models on massive datasets and generate high-resolution images

more efficiently, without a substantial loss in quality. The VAE ensures that the latent space is perceptually equivalent to the pixel space, and the diffusion model learns to generate high-quality latents within this space.

2.3. Conditioning Diffusion Models for Enhanced Control and New Tasks

While textual prompts provide a powerful and intuitive way to guide diffusion models, the generation process can be conditioned on a much wider array of signals. This opens up possibilities for more fine-grained control over the output and enables new applications beyond text-to-image synthesis. This section explores several advanced conditioning mechanisms, focusing on image-based conditioning for detailed content and style control, and specialized techniques for incorporating geometric information, such as camera parameters, to enrich 2D diffusion models with 3D awareness.

2.3.1. Image-based Conditioning for Fine-Grained Control

Conditioning on reference images allows for precise control over spatial layout, object appearance, color consistency, or artistic style, going beyond what is often achievable with text alone.

ControlNet

ControlNet [40] introduces a method to add diverse spatial conditioning to pre-trained text-to-image diffusion models. Instead of fine-tuning the original large model, ControlNet involves training smaller, task-specific auxiliary networks that are attached to the frozen U-Net backbone of the diffusion model, typically to its encoder blocks. These auxiliary networks take an input conditioning image (e.g., an edge map, human pose skeleton, depth map, or segmentation map) and produce feature maps that are then added to the corresponding features of the main U-Net. This approach allows the pre-trained model to be guided by the spatial information in the conditioning image while retaining its vast generative capabilities learned from large datasets. The original U-Net weights are locked, making ControlNet modules efficient to train and portable. Figure 2.7 provides a conceptual overview.

IP-Adapter

IP-Adapter (Image Prompt Adapter) [38] offers an effective way to condition diffusion models using an image prompt, allowing the model to generate images that align with the content or style of the reference image. It achieves this by introducing a lightweight adapter module that can be plugged into a pre-trained text-to-image model. The core idea is to decouple the cross-attention mechanisms used for text and image features. Image features, extracted by an image encoder (like CLIP [27] image encoder), are fed into new cross-attention layers that work in parallel with the original text cross-attention layers. This allows the model to draw information from both text and image prompts simultaneously or prioritize one over

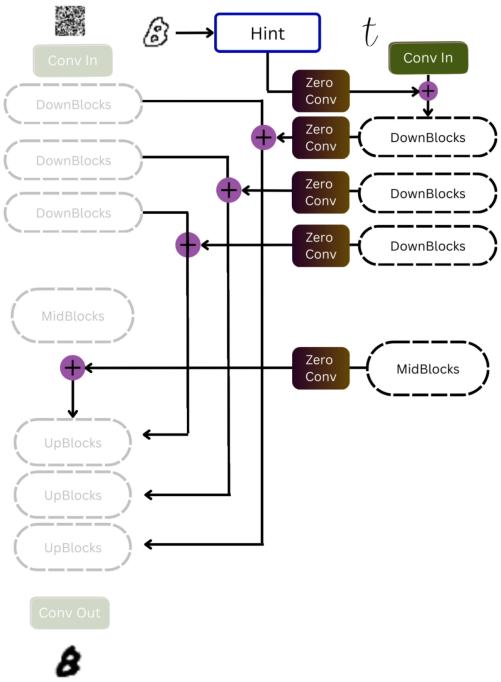

Figure 2.7: Conceptual overview of ControlNet, showing how an additional control module processes an input condition (e.g., edges, pose) and injects guidance into a pre-trained diffusion model. (Adapted from Zhang et al. [40])

the other. IP-Adapter is efficient as it avoids fine-tuning the large base model and only trains the adapter parameters, making it a versatile tool for tasks like style transfer or subject-driven generation. Figure 2.8 illustrates this concept.

CatVTON and UNet Feature Conditioning

A particularly relevant approach for image conditioning, especially for tasks requiring detailed transfer of appearance or garment characteristics, is found in methods like CatVTON [4]. The key insight in such methods is to leverage the rich, hierarchical features learned by the denoising U-Net itself. When provided with a reference image (e.g., an image of a specific garment), features can be extracted from various layers of the U-Net as it processes this reference. These extracted features, which encode information about texture, shape, and style at multiple scales, are then used to condition the generation of a new image (e.g., the same garment, but worn by a person). This conditioning can be achieved by injecting these features into the corresponding layers of the U-Net during the denoising process of the target image, often using attention mechanisms or direct feature fusion. This technique allows for a powerful form of image-based control that is highly attuned to the diffusion model's internal representations, facilitating tasks like virtual try-on with impressive fidelity by directly utilizing the U-Net's understanding of visual elements.

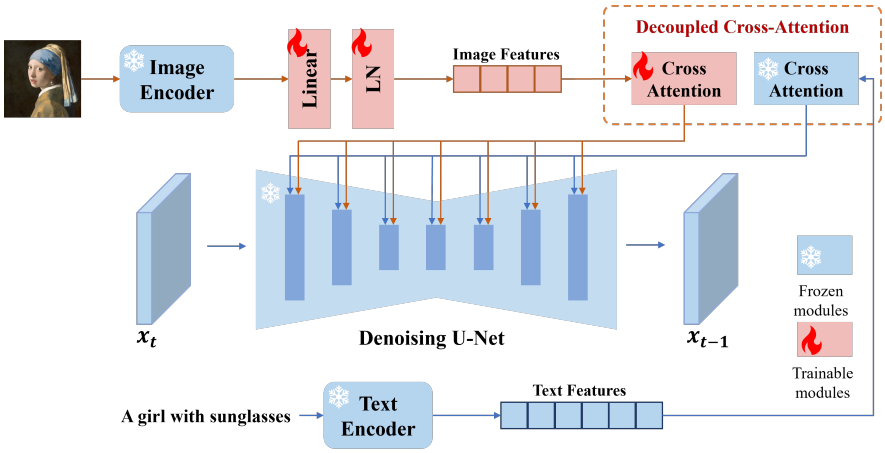

Figure 2.8: Conceptual illustration of the IP-Adapter architecture, showing how image features extracted by an image encoder are integrated into the U-Net via dedicated cross-attention layers to guide the generation process. (Image based on the IP-Adapter concept)

2.3.2. Modulating Network Activations

Feature-wise Linear Modulation (FiLM) [26] is a general and effective technique for conditioning neural network activations. Instead of directly concatenating conditioning information or using complex gating mechanisms, FiLM applies a simple affine transformation to feature maps based on a conditioning input. Given a feature map h from a layer in a neural network, and a conditioning vector c, a FiLM generator (typically a small neural network) produces a scale parameter γ and a shift parameter β from c. These parameters then modulate h as follows: $FiLM(h) = \gamma \cdot h + \beta$. This allows the conditioning information c to dynamically influence the behavior of the network layer by layer. FiLM is lightweight and can be integrated into various architectures to allow secondary inputs to control the processing of a primary input. Figure 2.9 illustrates the internals of a FiLM layer.

This approach enabled image-based models to effectively integrate and act upon conditioning information from outside the image domain. For instance, by modulating visual features based on encoded text tokens, it empowered (at the time) state-of-the-art image question answering models to ground their visual processing in textual context.

FiLMed-UNet

The FiLM technique has also been effectively applied to U-Net architectures in the medical imaging domain, notably in the FiLMed-UNet paper by Lemay et al. [20]. Their work focused on enhancing medical image segmentation by incorporating various forms of metadata as conditioning signals. FiLMed-UNet utilized patient-specific information, such as tumor type or the target organ for segmentation.

The core idea was to make the U-Net's segmentation process adaptable and more informed by this external metadata. The metadata (e.g., one-hot encoded tumor type) was passed through a FiLM generator network, which produced scale (γ) and shift (β) parameters. These parameters then modulated the feature maps at different layers within the U-Net. This allowed the network to tailor its feature extraction and segmentation logic based on the

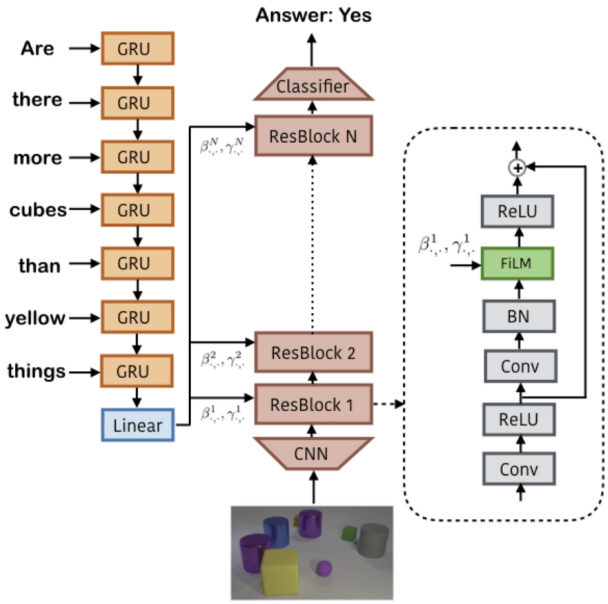

Figure 2.9: The Feature-wise Linear Modulation (FiLM) layer. A conditioning input is processed by a FiLM generator to produce scale (γ) and shift (β) parameters, which then modulate the feature maps (h) of a main network. (Adapted from Perez et al. [26])

provided metadata. For example, knowing the tumor type allowed the model to leverage type-specific visual characteristics, leading to improved segmentation accuracy. Furthermore, by conditioning on the desired output class (e.g., "segment kidney"), FiLMed-UNet demonstrated robustness in multi-task learning scenarios, even with missing labels for some tasks in the training data, and showed improved performance with limited annotations. While this application of FiLM is distinct from encoding camera parameters for 3D-aware synthesis, it highlights the versatility of FiLM in allowing neural networks to integrate and utilize diverse conditioning information to adapt their behavior for specialized tasks. Figure 2.10 shows a conceptual diagram.

2.3.3. Camera Parameter Encoding for 3D Awareness

Making 2D diffusion models inherently 3D-aware is crucial for tasks like multi-view image generation. A key aspect of this is encoding camera parameters (Rotation R, Translation t) to guide the image generation process. Various methods focus on integrating this geometric information directly into the model's input or intermediate representations.

Pixel-Level Geometric Conditioning: Ray-based Approaches

This category of methods aims to provide detailed, per-pixel geometric information derived from camera parameters directly to the diffusion model. A prominent technique is the use of a *raymap*, as demonstrated by Watson et al. [35] in 3DiM and also utilized in CAT3D [10]. The raymap is a tensor with the same spatial dimensions as the image or latent representation.

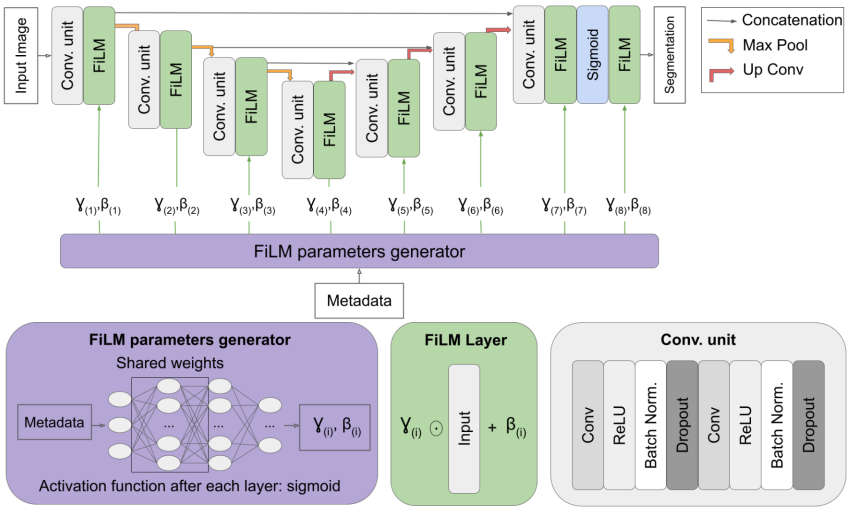

Figure 2.10: Conceptual diagram of a FiLMed-UNet. The metadata about patients are processed by a FiLM generator network, which outputs scale and shift parameters to modulate activations at various layers within the U-Net architecture, making the denoising process viewpoint-aware.

At each (u,v) coordinate, it stores the 3D origin and 3D direction of the camera ray passing through that pixel. In CAT3D, these rays are computed relative to the camera pose of the first conditional image. To help the network learn high-frequency details associated with viewpoint changes, these ray origins and directions are often further processed using positional encodings. The resulting positionally encoded raymap can be concatenated as an additional input channel to the U-Net, alongside the noisy image or latent representation [10, 35]. This allows the model to learn a direct, spatially varying mapping from ray geometry to the target pixel output.

Global Camera Pose Conditioning

Alongside per-pixel data, global camera pose conditioning methods provide a more compact, holistic representation of the camera's viewpoint or transformation. One approach involves *vectorized transformation embeddings*. For instance, Zero-1-to-3 [22] encodes the relative camera viewpoint transformation $T = [R|t]$ between an input view and a target view as a 4x4 matrix. This matrix is flattened into a 16-dimensional vector, which is then projected by a small Multi-Layer Perceptron (MLP) and added to the timestep embedding. This globally influences the diffusion process by providing context about the desired viewpoint change. Another technique employs *decomposed pose vectors for attention mechanisms*. MV-Adapter [17] represents each camera view using a 12-dimensional vector, which includes 3D camera coordinates and a 9D flattened rotation matrix. These vectors are processed by an MLP and then fed into multi-view attention layers as key and value components. This allows the model to explicitly attend to pose information when correlating features across different views, aiding in the generation of coherent multi-view imagery.

2.4. Diffusion-based Multi-View Image Generation

Building upon the camera parameter encoding techniques discussed in Section 2.3.3, this section delves into specific diffusion-based models and architectures designed for generating single or multiple consistent views of a 3D scene or objects. These methods aim to leverage the generative power of diffusion models to synthesize novel perspectives, often with the goal of creating coherent 3D representations.

2.4.1. Single Reference Image Novel View Synthesis

Novel View Synthesis (NVS) from a single reference image is a challenging task that several diffusion-based approaches have tackled.

Zero-1-to-3 [22], an early pioneer in this area, conditions a latent diffusion model on a single reference image and a relative camera pose transformation $T = [R|t]$ to generate a novel view. This model is fine-tuned from a pre-trained text-to-image model (Stable Diffusion). While demonstrating the potential for NVS without explicit 3D reconstruction, approaches like Zero-1-to-3 can sometimes face challenges in maintaining strong geometric consistency across widely varying generated views, a point often highlighted by subsequent multi-view generation methods [10, 33].

Concurrently, Watson et al. [35] proposed 3DiM, a diffusion model designed for NVS that operates in pixel space. It conditions on a single source RGB-D image (though it can function with RGB only) and an absolute target camera pose. 3DiM utilizes the raymap technique for camera conditioning, as detailed in Section 2.3.3, by constructing a raymap containing per-pixel 3D origin and direction, which is then positionally encoded and concatenated with the input image to the U-Net [35]. The model is trained to predict the clean target image from a noised source image.

2.4.2. Coherent Multi-View Generation Architectures

To improve consistency and enable robust 3D asset creation, several architectures focus on jointly generating or reasoning about multiple views simultaneously.

MVDream, often cited as a foundational multi-view model [10, 17, 33], extends diffusion models to generate multiple (e.g., four orthogonal) views from a shared text prompt by incorporating multi-view attention mechanisms. It learns from both 2D images and 3D assets to enhance 3D consistency [17].

Building on such concepts, ImageDream [33] introduces an image-prompt multi-view diffusion model. It takes a single image prompt and aims to generate multiple consistent views of an object. ImageDream leverages the canonical camera coordination (discussed in Section 2.3.3) established by MVDream, where a default camera view corresponds to the object's front view. Its key architectural contribution is a multi-level image-prompt controller that integrates image features at global (CLIP global embedding), local (CLIP hidden features), and pixel (VAE encoded image latent) levels into the MVDream-like diffusion U-Net [33]. The pixel controller, for instance, concatenates the input image prompt's latent as an additional frame to the four target views, enabling 3D self-attention across all five frames. ImageDream

employs Multi-View Score Distillation Sampling (MV-SDS) for reconstructing a NeRF from the generated views.

CAT3D [10] presents a multi-view diffusion model designed to simulate a real-world capture process by generating a large set of consistent novel views from one or more input images and specified target camera poses. Its architecture is similar to video latent diffusion models [2], employing 3D self-attention (2D in space and 1D corresponding to time across images) and is initialized from a pre-trained latent diffusion model. For camera conditioning, CAT3D uses the raymap approach (see Section 2.3.3), where ray origins and directions are computed relative to the first conditional image's camera pose and concatenated channel-wise to the latents [10]. To generate a large number of views (e.g., 80 for single-image input, up to 960 for few-view), CAT3D can group target viewpoints and, for single-image inputs, may first autoregressively sample a set of anchor views before generating the remaining views. These generated views are then used as input to a robust 3D reconstruction pipeline (e.g., Zip-NeRF).

ViewCrafter [39], as noted by CAT3D [10], combines video latent diffusion models with 3D point cloud priors. This approach aims to achieve high-fidelity and consistent novel views by leveraging explicit 3D information alongside the generative capabilities of video diffusion models, enabling precise camera control.

While these architectures show impressive results, methods requiring full fine-tuning of large pre-trained models are expensive and may risk performance degradation if high-quality, large-scale 3D data for fine-tuning is scarce.

2.4.3. Specialized Multi-View Adapters

To mitigate the challenges of full fine-tuning, adapter-based methods offer a lightweight alternative for enabling pre-trained models with multi-view generation capabilities.

MV-Adapter [17] is a notable example, introducing a plug-and-play adapter for multi-view image generation that enhances pre-trained text-to-image diffusion models while preserving their original network structure and feature space. It employs a decoupled attention mechanism: the original spatial self-attention layers of the pre-trained model are kept frozen, and new multi-view attention layers are created by duplicating the structure and weights of these original layers. These new layers, organized in a parallel architecture, process features from multiple views (typically packed along the batch dimension) and learn geometric knowledge efficiently [17].

For encoding geometric information, MV-Adapter [17] integrates camera parameters using raymaps, an approach also seen in methods like 3DiM [35] and CAT3D [10]. These raymaps can be positionally encoded and are then typically supplied as additional input channels to the U-Net, processed by the adapter's components. This method of providing direct geometric cues, combined with other conditioning signals like text or image embeddings processed through its attention mechanisms, allows for flexible control in tasks such as text-to-3D and image-to-3D generation. By updating significantly fewer parameters than full fine-tuning—primarily those of the adapter module—MV-Adapter enables efficient training, preserves the rich priors of the base model, and reduces the risk of overfitting.

3. Proposed Method

In this chapter, I describe the proposed method for adapting a pre-trained 2D diffusion model to understand 3D geometry and synthesize novel views of objects. The approach focuses on conditioning the generative process on both a reference image and relative camera transformations.

3.1. Overview

The core challenge in adapting 2D diffusion models for novel view synthesis lies in equipping them with an understanding of 3D space and object geometry. This work proposes a method that augments a pre-trained text-to-image diffusion model, specifically Stable Diffusion 2.1, with two primary conditioning mechanisms: one for visual appearance derived from a source reference image, and another for geometric understanding derived from camera pose information. By integrating these conditioning signals, the model learns to generate novel views of an object from a given source image and a target camera pose. The overall architecture is designed to be efficient by primarily training lightweight adapter modules and specialized encoders, while keeping the majority of the large pre-trained model frozen to preserve its generative priors and ensure computational efficiency during training.

3.2. Limitations of Previous Methods

Despite the significant progress in multi-view image generation and novel view synthesis, several limitations and research gaps remain:

1. **Computational Efficiency**: Full fine-tuning of diffusion models for multi-view generation is computationally expensive, especially when working with large base models and high-resolution images. While first adapter-based method MV-Adapter has improved efficiency of training, there is still room for improvement.

2. **Geometric Consistency**: Maintaining geometric consistency across generated views remains a challenge, particularly when generating views from significantly different perspectives. Current methods often struggle with complex occlusions, reflective surfaces and fine geometric details.

3. **Lighting issues**: The lighting plays a critical role in the visual appearance of rendered objects. The example rendering scripts provided by the ObjaverseXL authors [7] employed a randomized lighting setup. While intended to introduce variability, this approach often resulted in images with harsh lighting, strong shadows, or scenes where the object was underexposed or even completely obscured because the light source was

positioned unfavorably. This can negatively impact the 3D reconstruction from images since the Novel View Synthesis should provide meaningful information about texture and geometry.

4. **Adaptation to unseen objects**: The current methods are not able to generalize to unseen objects, which is a major limitation.

5. **Adaptation to different viewpoints**: The current methods are not able to adapt to different viewpoints, which is a major limitation. If a given method was trained on specific set of perspectives it cannot generalize to the new viewpoints.

My work aims to adress the **Computational Efficiency** by using only trainable adapter layers, **Lighting issues** by ensuring consistent lighting in datasets, and **Adaptation to different viewpoints** by conditioning the model on many camera parameter configurations. The problem of **Adaptation to unseen objects** can be adressed by large-scale training on a diverse set of objects.

3.3. Architectural Framework

The proposed method leverages the strong generative capabilities of a pre-trained 2D diffusion model and extends it for 3D-aware novel view synthesis. This is achieved through the introduction of specialized conditioning modules that inject visual and geometric information into the denoising U-Net.

3.3.1. Introduction and System Diagram

The primary goal is to adapt the Stable Diffusion 2.1 model, an inherently 2D generative framework, to comprehend 3D spatial relationships and generate images from novel viewpoints. The system achieves this by integrating two distinct conditioning streams into the diffusion model's U-Net architecture. The first stream provides visual context from a source (reference) image, while the second stream imparts geometric awareness through encoded camera parameters. Figure 3.1 illustrates the overall architecture of the proposed system, highlighting the flow of information and the interplay between the original diffusion model components and the newly introduced conditioning modules.

3.3.2. Backbone: Stable Diffusion 2.1

The foundation of proposed method is the Stable Diffusion 2.1 model. This choice is motivated by its robust text-to-image generation capabilities and its publicly available pre-trained weights, which encapsulate a vast understanding of visual concepts. The core component relevant to my modifications is its U-Net architecture [29], which performs the iterative denoising process. In my approach, several key components of the original Stable Diffusion 2.1 model are kept frozen to preserve their learned knowledge and ensure computational efficiency during training. These include the Variational Autoencoder (VAE) [19] used for encoding images into and decoding latents from the latent space, the text encoder

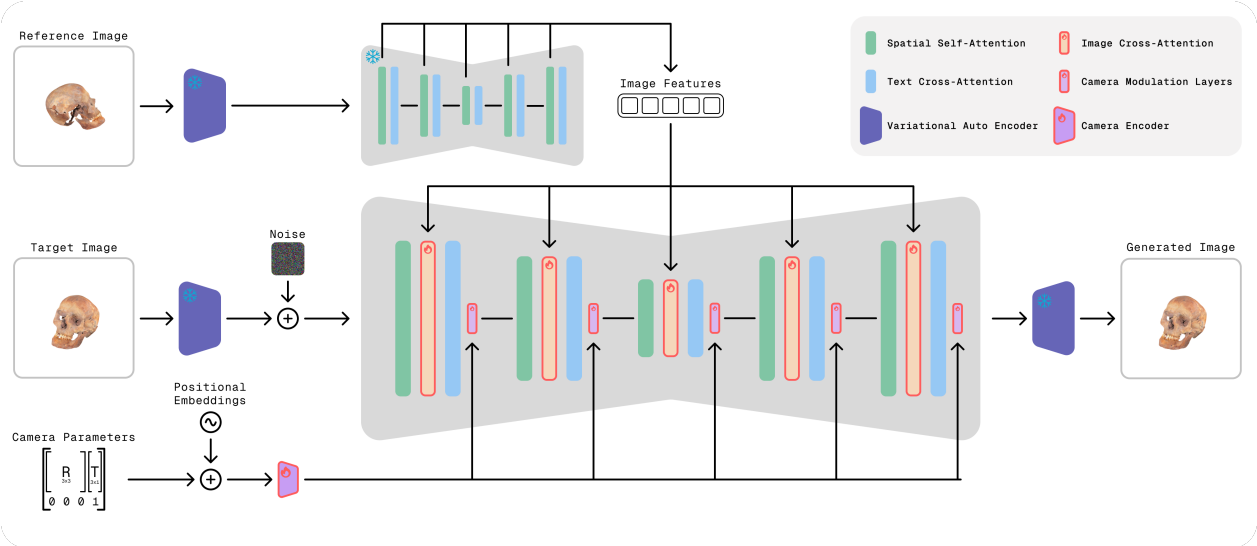

Figure 3.1: System diagram of the proposed multi-view diffusion model. It showcases the reference image encoding path, the camera parameter encoding path, and their integration into the main denoising U-Net.

CLIP [27] responsible for processing textual prompts, and the weights of the original U-Net when used within the Reference Image Encoder. The new conditioning modules and adapters are specifically designed to be trainable.

3.3.3. Reference Image Conditioning Stream

To enable the model to generate views that are visually consistent with a given input, a reference image conditioning stream is introduced. This stream aims to capture and transfer the appearance, texture, color, and identity of the object depicted in the source image to the novel view generation process.

Source Image Feature Extraction

The process of extracting visual features from the source image is handled by a dedicated *ImageEncoder* module. This module operates as follows:

1. The source image is first transformed into a latent representation using the frozen VAE of the backbone Stable Diffusion model. This compresses the image into a lower-dimensional space where the diffusion process typically operates.

2. This latent representation is then processed by the original, frozen U-Net of Stable Diffusion 2.1. Critically, this forward pass through the U-Net is performed with the timestep parameter set to zero ($t = 0$). This effectively instructs the U-Net to perform a ”denoising” pass on the clean latent, attending to the image and extracting rich, multi-scale visual features.

3. During this pass, the *ImageEncoder* captures the output activations from the self-attention layers present within each block of the frozen U-Net. These extracted features, which encode comprehensive visual information about the source image at different semantic levels, are then made available for conditioning the main denoising process.

Image Cross-Attention Adapters

The visual features extracted by the *ImageEncoder* are injected into the main denoising U-Net using newly introduced, trainable adapter modules that serve as cross-attention layers. These processors are designed to be lightweight and are integrated into each corresponding block of the denoising U-Net. A key architectural choice is the parallel integration of these image cross-attention adapters. Instead of serially passing information through the original attention layers and then the new adapters, the adapters operate in parallel to the U-Net's existing self-attention and text-cross-attention mechanisms. The extracted image features serve as the key and value for these new cross-attention layers, while the U-Net's intermediate hidden states act as the query. This allows the denoising U-Net to directly attend to relevant visual details from the source image at multiple stages of the generation process, with the influence of these reference features controlled by a scaling factor, which was tested in the range of 0.1 to 1.0 during training.

3.3.4. Camera Parameter Conditioning Stream

To ensure geometric consistency and enable precise control over the viewpoint of the generated image, a camera parameter conditioning stream is employed. This stream encodes the relative transformation between the source and target camera poses and uses this information to modulate the behavior of the denoising U-Net.

Camera Pose Encoding

The *CameraEncoder* module is responsible for processing the geometric information. The input to this module is the relative camera transformation from the source view to the target view, represented by a 3×3 rotation matrix R and a 3×1 translation vector T. These 12 parameters (9 for R and 3 for T) define the desired viewpoint change. To enrich this relatively low-dimensional signal, rotary positional embeddings are applied to the camera parameters. This augmentation aims to provide a more distinctive and structured representation for the subsequent neural network layers. The enriched camera parameters are then processed by a Multi-Layer Perceptron (MLP). This MLP typically consists of several layers with SiLU activation functions, transforming the camera pose information into a high-dimensional embedding suitable for conditioning.

FiLM-based Network Modulation

The high-dimensional camera embedding produced by the *CameraEncoder* is used to modulate the activations within the main denoising U-Net via Feature-wise Linear Modulation (FiLM) layers [26]. For a given feature map h in the U-Net, the FiLM layer applies an affine

transformation:

$$FiLM(h) = \gamma \cdot h + \beta$$

where γ (scale) and β (shift) are parameters generated by passing the camera embedding through dedicated linear layers within the $CameraEncoder$. These FiLM modulations are applied at each UNET block to the outputs of the feature maps (down-sampling, middle, and up-sampling blocks).

This allows the camera pose information to dynamically influence the feature representations throughout the network, guiding the geometric aspects of the image generation. The strength of this modulation can be adjusted by a scalar factor, which was tested in the range of 0.1 to 1.0 during training.

3.3.5. The Integrated Conditioned U-Net

The $MultiViewUNet$ module serves as the central component of the proposed architecture. It integrates the backbone Stable Diffusion U-Net with the two conditioning streams described above: the reference image conditioning via parallelly connected Image Cross-Attention Adapters and the camera parameter conditioning via Feature-wise Linear Modulation (FiLM) layers.

During each step of the reverse diffusion process, the $MultiViewUNet$ takes the noisy latent representation of the target image, the current timestep, text embeddings, the extracted reference image features, and the encoded target camera parameters. It then predicts the noise present in the noisy latent. The system diagram in Figure 3.1 provides a visual summary of this integrated data flow.

3.4. Model Training

The training process is designed to teach the $MultiViewUNet$ to effectively utilize the visual and geometric conditioning information to predict the noise required to generate a target view from a source view and camera transformation.

3.4.1. Training Objective and Loss Functions

The primary training objective is to minimize the difference between the noise predicted by the $MultiViewUNet$ and the actual noise that was added to the target image's latent representation. This is typically formulated as a Mean Squared Error (MSE) loss:

$$L_{noise} = \mathbb{E}_{x_0,\epsilon,t}[\|\epsilon - \epsilon_\theta(x_t, t, c_{img}, c_{cam}, c_{text})\|^2]$$

where x_0 is the clean target image latent, ϵ is the sampled Gaussian noise, t is the timestep, x_t is the noisy latent at timestep t, and ϵ_θ is the noise predicted by the network conditioned on image features c_{img}, camera embedding c_{cam}, and text embedding c_{text}. The noise scheduler is configured to predict the noise term ϵ.

To improve training stability and sample quality, especially at varying noise levels, an SNR-aware weighting strategy is employed for the loss function, specifically the Min-SNR-γ

approach [13]. The loss for each training instance is weighted by $min(SNR_t, \gamma)/SNR_t$, where SNR_t is the signal-to-noise ratio at timestep t, and γ is a hyperparameter (e.g., set to 5.0) as the method authors recommend [13]. This weighting scheme effectively gives more importance to timesteps with lower SNR values, preventing the model from focusing excessively on high-SNR (low noise) timesteps. The noise schedule itself is a DDPMScheduler, further modified using an interpolated shift based on the SNR, with a shift scale factor (e.g., 6.0), which adapts the noise levels experienced during training.

While the primary loss focuses on noise prediction, a set of auxiliary metrics are monitored during validation steps to provide a comprehensive assessment of image quality and consistency. For rapid validation during hyperparameter tuning and intermediate training checks, a Perceptual Loss is used alongside SSIM, CLIP score, and FID. For final model evaluation and reporting, more rigorous metrics such as LPIPS [41], CLIP score [15], and Fréchet Inception Distance (FID [16, 32]) are employed, in addition to PSNR and SSIM.

3.4.2. Training Data and Iteration

Each training iteration involves a batch of data samples. A single sample consists of:

- A source image.
- A target image (representing a different view of the same object).
- The relative camera transformation (rotation R and translation T) from the source camera pose to the target camera pose.
- A textual prompt describing the object (leveraging the underlying text-to-image capabilities of Stable Diffusion).

The model is trained to predict the noise in the target image's latent, conditioned on the source image, the camera transformation, and the text prompt.

3.4.3. Optimization and Implementation Details

The trainable parameters of the model include the weights of the modules mentioned in Section 3.3. The core U-Net of Stable Diffusion 2.1, the VAE, and the text encoder remain frozen during this fine-tuning phase. Optimization is performed using the AdamW optimizer [23] with a learning rate of 1×10^{-5}. A cosine learning rate schedule with a warm-up period is employed to manage the learning rate dynamics over the training duration, which is set for a total of 10 epochs. Gradient clipping is applied with a maximum norm of 1.0 to prevent exploding gradients. Training is performed using a batch size of 6 per GPU. Model was trained on 4 GPUs (NVIDIA A100 40GB). The model is implemented using PyTorch and PyTorch Lightning, and leverages mixed-precision training (e.g., $float32$) for efficiency.

3.5. Inference Process

Once trained, the model can be used to synthesize a novel view of an object from a given source image and a specified target camera pose.

3.5.1. Novel View Synthesis Pipeline

The inference process, managed by the $MVDPipeline$, proceeds as follows:

1. **Inputs**: The pipeline takes a single source image, the desired target camera parameters, and an optional text prompt.
2. **Source Image Encoding**: The source image is processed by the $ImageEncoder$ to extract its multi-scale visual features.
3. **Camera Parameter Encoding**: The target camera transformation is encoded by the $CameraEncoder$ to produce a camera embedding.
4. **Iterative Denoising**: Starting from a randomly sampled Gaussian noise tensor in the latent space (of the same dimensions as the VAE's output latents), the $MultiViewUNet$ iteratively denoises this latent over a predefined number of steps (in my case, 20). In each step, the U-Net receives the current noisy latent, the timestep t, the text prompt embedding, the extracted source image features, and the target camera embedding (via FiLM layers). Classifier-Free Guidance (CFG) is typically used, where the model makes both a conditional and an unconditional prediction, and the final noise estimate is a weighted combination, controlled by a guidance scale (e.g., 1.0).
5. **VAE Decoding**: After the final denoising step, the resulting clean latent representation is decoded back into pixel space using the frozen VAE's decoder.

3.5.2. Output

The final output of the pipeline is the synthesized image, which represents the object from the specified target viewpoint, conditioned by the appearance of the source image and guided by the geometric transformation. The output images are generated at a resolution consistent with the training data, maximum supported by the model is 768×768 pixels.

4. Data Preparation

In this chapter, I will discuss the datasets used in training the method presented in this thesis and the data augmentation techniques applied to them. The quality and nature of the data are paramount for training robust deep learning models, particularly for complex tasks such as novel view synthesis. This chapter details the meticulous process of curating and preparing the datasets used for training and validating the proposed method, with a primary focus on the extensive ObjaverseXL dataset used for training and the complementary Google Scanned Objects dataset used for validation.

4.1. Dataset Overview

Two primary datasets form the backbone of this work: ObjaverseXL, a large-scale repository of 3D models utilized for training the generative model, and Google Scanned Objects (GSO), a collection of high-fidelity 3D scans employed in my work for validation purposes.

4.1.1. ObjaverseXL: A Large-Scale 3D Model Repository

ObjaverseXL [6] is a vast, publicly accessible dataset comprising approximately 10 million 3D models. In order to use these models, one has to download them from their respective sources. The models are stored in diverse online sources, including platforms like GitHub, Thingiverse, and Sketchfab, as shown in 4.1, resulting in a rich and varied collection that reflects a wide array of object categories, complexities, and artistic styles. Given its scale and diversity, ObjaverseXL serves as the primary source for the training data in this thesis.

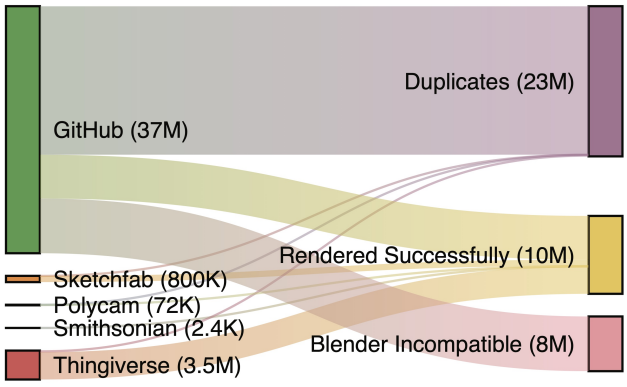

Figure 4.1: Illustrative overview of ObjaverseXL model sources and diversity.

4.1.2. Google Scanned Objects (GSO): High-Quality 3D Scans

The Google Scanned Objects (GSO) dataset [8] consists of high-fidelity 3D scans of real-world objects. In this work, the GSO dataset is utilized for the final validation of the trained multi-view image generation model. Its distinct origin and capture methodology compared to ObjaverseXL provide a benchmark for evaluating the generalization capabilities and output quality of the proposed method on unseen, high-quality data.

4.2. Processing Pipeline for ObjaverseXL Training Data

The transformation of raw 3D models from ObjaverseXL into a usable training dataset involved a comprehensive processing pipeline. This pipeline encompassed several critical stages: model acquisition, normalization, view rendering with specific lighting considerations, quality control filtering, and finally, textual annotation. Each step was carefully designed to address potential issues and ensure the final dataset's suitability for training an effective multi-view diffusion model.

4.2.1. 3D Model Acquisition and Initial Handling

The first step involved acquiring the 3D models from their respective sources as listed in the ObjaverseXL dataset. This task required scripting the download process due to the varied hosting platforms. The pipeline was designed to robustly handle common 3D model formats that typically support textures, including Object files (`.obj`), GLTF/GLB files (`.glb`, `.gltf`), Filmbox files (`.fbx`), and Collada files (`.dae`).

A critical part of initial handling was standardizing the coordinate system. 3D models from diverse origins often have different conventions for which axis represents "up". To ensure consistency, all imported mesh objects underwent a corrective rotation of -90 degrees around the X-axis. This transformation was immediately applied to the objects, establishing a common "Z-up" orientation for all models before subsequent processing steps. A preliminary inspection of model metadata was also performed upon successful download, which sometimes provided hints for orientation if the automated correction was insufficient, though the Z-up convention was paramount.

4.2.2. 3D Model Normalization and Scene Setup

To ensure consistency across all rendered images, a standardized scene setup was imperative. Each downloaded 3D model underwent a normalization process:

1. **Global Bounding Box Calculation**: The process began by calculating a single bounding box encompassing all mesh components of the loaded 3D model.

2. **Scaling to Unit Cube**: Based on the maximum dimension of this global bounding box, a uniform $scale_factor$ was determined. The model was then scaled by this factor to ensure it fit precisely within a 1x1x1 unit cube.

3. **Centering at Origin**: After scaling, the model was translated so that the center of its global bounding box was positioned at the world origin (0,0,0).

This normalization was critical for several reasons. Firstly, it ensured that objects, regardless of their original size, would fit within the camera's viewpoint. Secondly, it provided a consistent reference point for defining camera positions and distances, simplifying the multi-view rendering setup.

4.2.3. Multi-View Image Rendering

The core of the data generation process was rendering 2D images of each 3D model from multiple viewpoints.

Rendering Environment

All rendering tasks were performed using Blender [5], a powerful open-source 3D creation suite. To automate and scale the process, Blender was operated in windowless (headless) mode. The $BLENDER\_EEVEE$ rendering engine was chosen for its balance of speed and quality, making it suitable for generating a large volume of images. Render output was configured to $1024x1024$ pixels. Images were initially rendered with an alpha channel (e.g., to PNG format) to handle transparent backgrounds.

Specific EEVEE settings were configured to balance quality and performance while ensuring clear depiction of the objects:

- **Temporal Anti-Aliasing (TAA)**: Render samples set to 32 for smoother edges.
- **Ambient Occlusion (GTAO)**: Enabled to provide contact shadows and enhance perceived depth.
- **Bloom**: Disabled to prevent overly bright, glowing effects that could obscure object details.

Camera Viewpoint Configuration

Camera setup was designed for consistency and to provide a comprehensive view of the objects. Key camera parameters included:

- **Focal Length**: 35mm.
- **Sensor Width**: 32mm.
- **Camera Distance**: Cameras were positioned at a fixed distance of 1.8 Blender units from the world origin. Given the 1x1x1 object normalization, this distance ensured the object was well-framed.
- **Targeting**: A $TRACK\_TO$ constraint was applied to the camera, compelling it to always point towards the world origin (0,0,0), keeping the object centered.

To train a model capable of understanding objects from various perspectives and generalizing to different numbers of input views, images were rendered from a randomly selected set of predefined camera angles for each model. Three configurations were used for 6, 8 and 12 views, with specific azimuth (horizontal rotation) and elevation alternating between small positive and negative values to cover the object from all sides. The camera is focused on the origin. The placement is shown in 4.2 and 4.3.

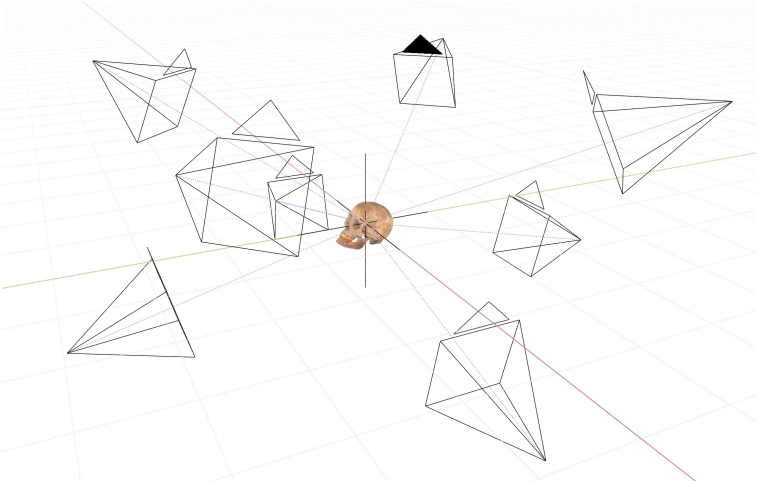

Figure 4.2: Example camera configurations for 8 views around an object.

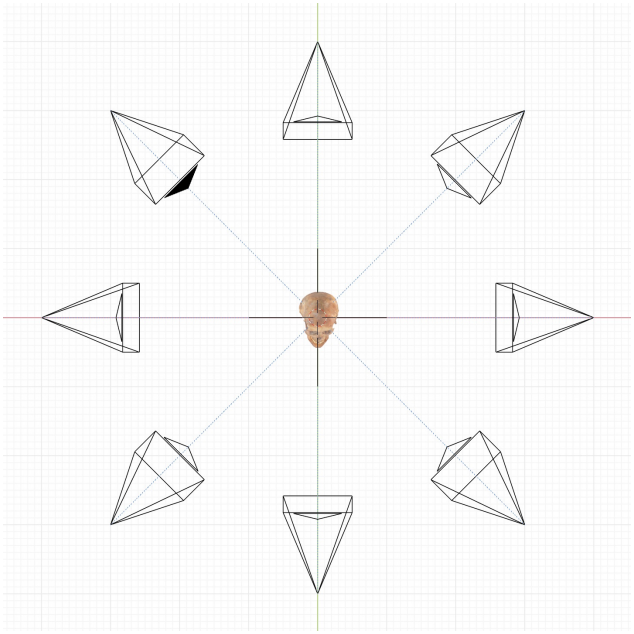

Figure 4.3: Camera configurations for 8 views around an object, visible from the top.

The choice of 6, 8, or 12 views for any given model was made randomly, with the dataset designed to have an approximately equal distribution of samples across these three groups. This strategy aimed to enhance the model's robustness to varying input view counts during

inference. Azimuth angles were defined to be counter-clockwise around the object to ensure compatibility with methods of 3D reconstruction like InstantMesh [36].

Lighting Strategy for Consistent Illumination

Lighting plays a critical role in the visual appearance of rendered objects. The example rendering scripts provided by the ObjaverseXL authors [7] employed a randomized lighting setup. While intended to introduce variability, this approach often resulted in images with harsh lighting, strong shadows, or scenes where the object was underexposed or even completely obscured because the light source was positioned unfavorably (e.g., behind the object relative to the camera). Such inconsistencies could negatively impact model training, potentially teaching the model to generate overly dark or inconsistently lit views.

To mitigate this, a revised, deterministic multi-point lighting strategy using four *SUN* type lights was adopted.The primary goal was to achieve soft, even illumination across the object's surface, ensuring that its features were clearly visible from all rendered viewpoints. This deterministic lighting strategy was specifically designed to create a dataset with consistent illumination, which helps address the 'Lighting issues' (Section 3.2 in Chapter 3) that can hinder the performance of multi-view generation models.

The resulting images featured more consistently well-lit objects, reducing the likelihood of the diffusion model learning to arbitrarily darken parts of an object from perspectives different from the main light source direction in the input. This consistency is vital for learning accurate geometry and appearance.

4.2.4. Post-Rendering Quality Control and Filtering

Despite the controlled rendering setup, not all rendering attempts yielded high-quality images. Some 3D models were inherently problematic (e.g., incompatible formats, missing textures, non-manifold geometry), while others, due to their material properties or fine details, resulted in renders that provided little useful visual information (e.g., images consisting mainly of reflections, transparency artifacts, or indiscernible silhouettes). It was observed that approximately one-third of the initially rendered samples suffered from such quality issues.

To ensure the training data was of sufficient quality, a rigorous filtering process was implemented. This was based on a contrast score, calculated as the standard deviation of pixel intensities in the grayscale version of each rendered image. A minimum contrast threshold was established. . If any of the rendered views for a particular 3D model fell below this threshold, the entire sample (the 3D model and all its associated renders) was discarded from the dataset. While this contrast metric doesn't capture all facets of 3D model quality, it proved to be a scalable and effective heuristic for discarding renders with insufficient visual information or severe artifacts. After this filtering stage, approximately $26,000$ high-quality 3D model samples, each with its set of 6, 8, or 12 views, remained.

4.2.5. Textual Prompt Generation via Multimodal LLM

To fine-tune a text-to-image diffusion model, rich textual descriptions (prompts) corresponding to the visual content are essential. Since the ObjaverseXL models do not inherently

come with such descriptive prompts, they needed to be generated. For this task, a state-of-the-art multimodal Large Language Model (LLM), Qwen2.5VL [1], was employed.

The prompt generation process was as follows:

1. For each 3D model, three views were randomly selected from its set of successfully rendered and filtered images.

2. These three views were provided as parallel input to the Qwen2.5VL model.

3. The LLM was tasked with analyzing these views and generating a single, comprehensive, and descriptive text prompt that accurately captured the essence of the 3D object depicted.

This approach aimed to distill visual information from multiple perspectives into a rich textual description, providing effective conditioning for the diffusion model during training. The use of multiple views was intended to help the LLM form a more holistic understanding of the object compared to relying on a single view.

4.3. Processing of Google Scanned Objects (GSO) Validation Data

The Google Scanned Objects (GSO) dataset was processed with a similar pipeline to ensure consistency in evaluation. The primary steps involved:

1. Downloading the GSO 3D models.

2. Passing them through the same normalization, multi-view rendering (using the same camera configurations and lighting strategy), quality control filtering pipeline and textual prompt generation described in Section 4.2.

4.4. Training Dataset Characteristics

Following the comprehensive processing pipeline, the final datasets were prepared for model training and validation. The ObjaverseXL training dataset consists of approximately 26,000 unique 3D models, each associated with a set of high-quality rendered views (6, 8, or 12 per model) and a descriptive textual prompt generated by Qwen2.5VL. This curated dataset provides a rich and diverse source of multi-view image-text pairs for training the diffusion model.

The GSO validation dataset comprises a set of consistently rendered views from the high-fidelity GSO models (approximately 1000 models), ready to be used for evaluating the performance of the trained model, particularly its ability to generalize to unseen objects and maintain multi-view consistency. With these meticulously prepared datasets, the subsequent stages of model training and experimentation, as detailed in the following chapters, could proceed on a solid foundation.

4.4.1. Dataset Statistics

This subsection presents a visual overview of key statistics derived from the processed ObjaverseXL dataset. These visualizations offer insights into the distribution of various data attributes, which are helpful in understanding the characteristics of the data used for training.

First, the distribution of the number of rendered views per 3D model is presented. As described in Section 4.2.3, models were rendered with 6, 8, or 12 views, with an approximately equal distribution. The exact frequencies (after filtering) are shown in Table 4.1.

Table 4.1: Distribution of Render Counts per Model.

Render Count	Frequency
6	9081
8	8261
12	8660

Figure 4.4 illustrates the distribution of file sizes (in Megabytes) of the original 3D models in the dataset.

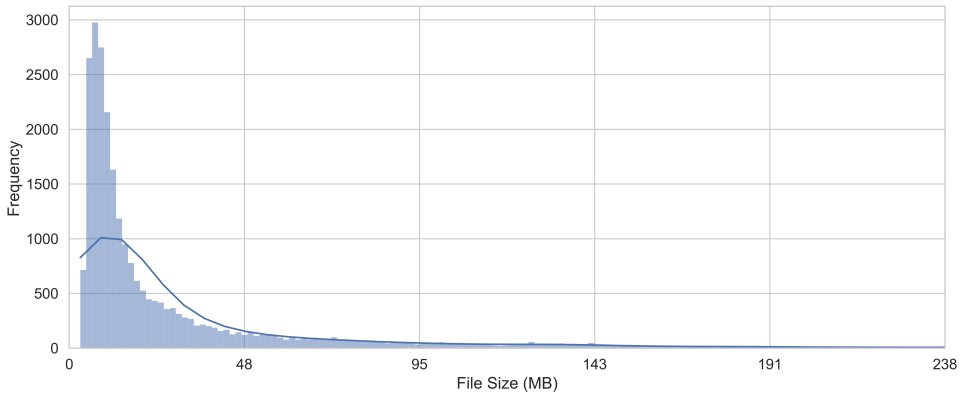

Figure 4.4: Distribution of sample sizes in the ObjaverseXL dataset. Statistics: $\mu = 37.25$ MB and $\sigma = 117.93$ MB.

The distribution of average image contrast across the rendered views is shown in Figure 4.5. This metric was crucial for the quality control filtering process, as detailed in Section 4.2.4.

Figure 4.6 displays the distribution of the lengths of textual prompts generated by the Qwen2.5VL model. These prompts are vital for conditioning the text-to-image diffusion model.

A word cloud visualization of the most frequent terms appearing in the generated prompts is presented in Figure 4.7. This offers a qualitative glimpse into the common themes and object characteristics described in the dataset.

Finally, Table 4.2 presents the top 20 topics identified from the textual prompts using Latent Dirichlet Allocation (LDA) [3]. Each topic is represented by its most characteristic words, providing insight into the semantic clusters present in the dataset's descriptions.

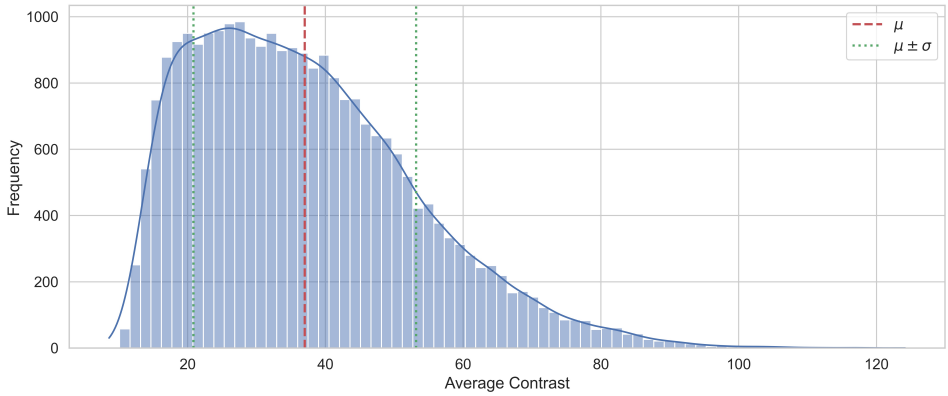

Figure 4.5: Distribution of average contrast scores for rendered images. Statistics: $\mu = 36.97$ and $\sigma = 53.15$.

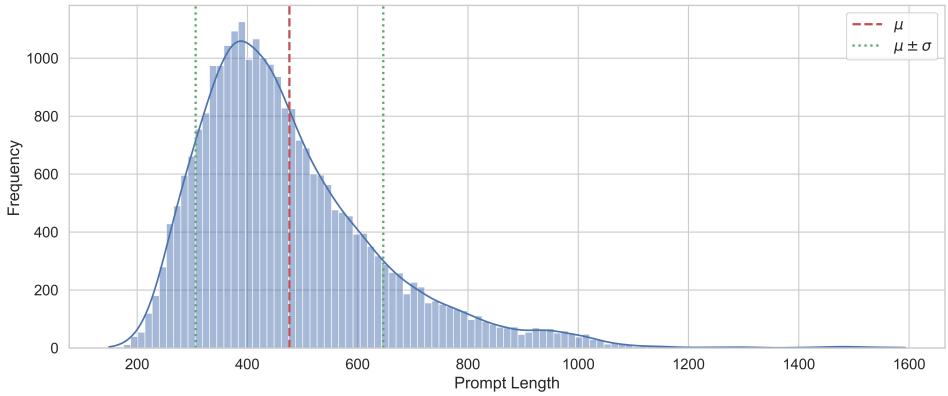

Figure 4.6: Distribution of generated prompt lengths. Statistics: $\mu = 476.06$ and $\sigma = 170.18$.

The identified topics highlight a diverse range of subjects, including common objects (e.g., 'box', 'chair', 'car'), structural elements (e.g., 'roof', 'legs', 'handle'), and descriptive attributes (e.g., colors like 'blue', 'red', 'green'; materials like 'metallic', 'wooden'; and characteristics like 'smooth', 'small'). This underscores the breadth of semantic content captured in the training data prompts.

These visualizations and statistics collectively provide a comprehensive overview of the dataset's characteristics, underscoring its suitability for the research presented in this thesis.

4.4.2. Dataset Samples

This section presents a selection of samples from the ObjaverseXL dataset. The samples are chosen to showcase the diversity of the dataset and the quality of the rendered views.

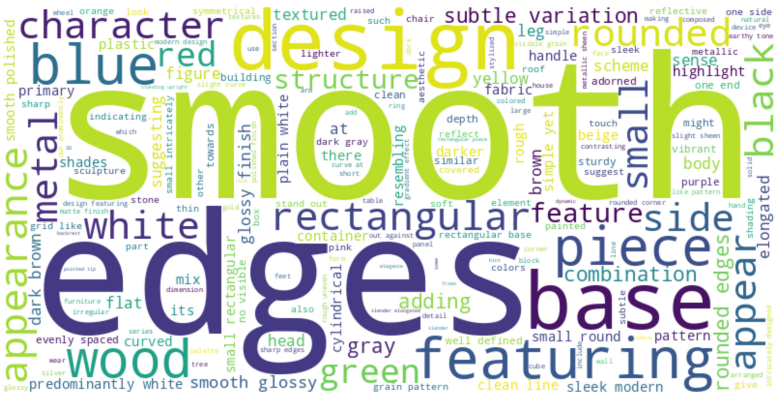

Figure 4.7: Word cloud of terms from generated prompts.

Table 4.2: Top 20 LDA Topics from Prompts with most characteristic words.

Topic	**Top Words**
#1	evenly, spaced, along, tall, easy, each, crate, rectangular, vertical
#2	device, rectangular, edges, rounded, black, small, industrial, panel
#3	character, small, body, figure, head, its, eyes, legs, black
#4	blue, red, glossy, smooth, vibrant, yellow, white, at, against
#5	body, sleek, streamlined, black, design, car, aerodynamic, accents, red
#6	metallic, metal, reflective, silver, polished, design, combination, gold, steel
#7	base, sculpture, intricately, ceramic, wear, statue, cylindrical, stone, signs
#8	box, ring, band, near, rectangular, black, pattern, either, sides
#9	green, block, small, tree, leaves, paper, delicate, plant, white
#10	well, defined, smooth, sharp, plain, skin, its, white, highlight
#11	roof, structure, small, rectangular, wooden, building, house, wood, painted
#12	legs, design, frame, chair, wood, backrest, sturdy, dark, seat
#13	rectangular, edges, modern, design, smooth, clean, base, white, lines
#14	smooth, subtle, brown, wood, rounded, edges, appearance, piece, variations
#15	section, part, upper, stick, base, lower, wider, than, more
#16	container, fabric, soft, rounded, lid, edges, small, pastel, suitable
#17	elongated, end, tapering, towards, rough, one, pointed, at, predominantly
#18	which, suggesting, might, indicating, similar, appears, plastic, could, like
#19	brown, shades, green, some, small, natural, rough, mix, areas
#20	handle, black, sleek, design, cylindrical, finish, modern, smooth, body

Figure 4.8: Sample image pairs from the ObjaverseXL dataset.

Figure 4.9: Sample views of Object 1 (6 views). Generated prompt: Create an image of a miniature lighthouse with a rustic vintage design. The base is made of dark wood with a polished finish, supporting a cylindrical tower made from a lighter material. The upper part features a lantern-like structure painted in a light color, adorned with small decorative elements like round windows or panels. The top is capped with a small, pointed roof made from the same light-colored material, slightly tilted.

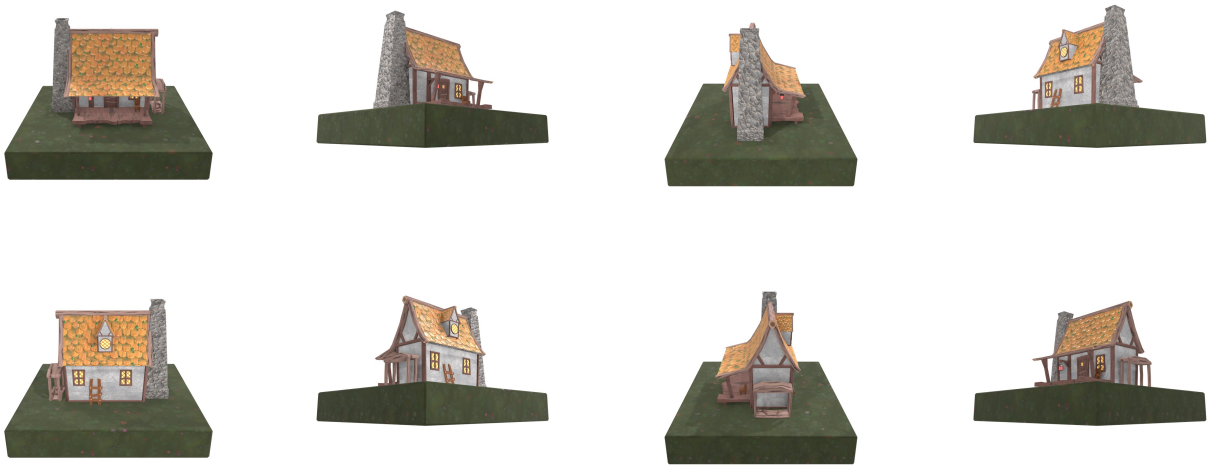

Figure 4.10: Sample views of Object 2 (8 views). Generated prompt: Create a detailed 3D model of a rustic cabin with a rectangular shape and slightly rounded corners. The cabin should have a warm, golden hue roof covered with shingles, wooden walls painted in a light color, small rectangular windows and door, and a raised foundation on stilts. The base should be a flat, green surface representing grass or another natural material. The overall aesthetic should be charming and rustic, with attention to detail in the textures and colors of the materials used.

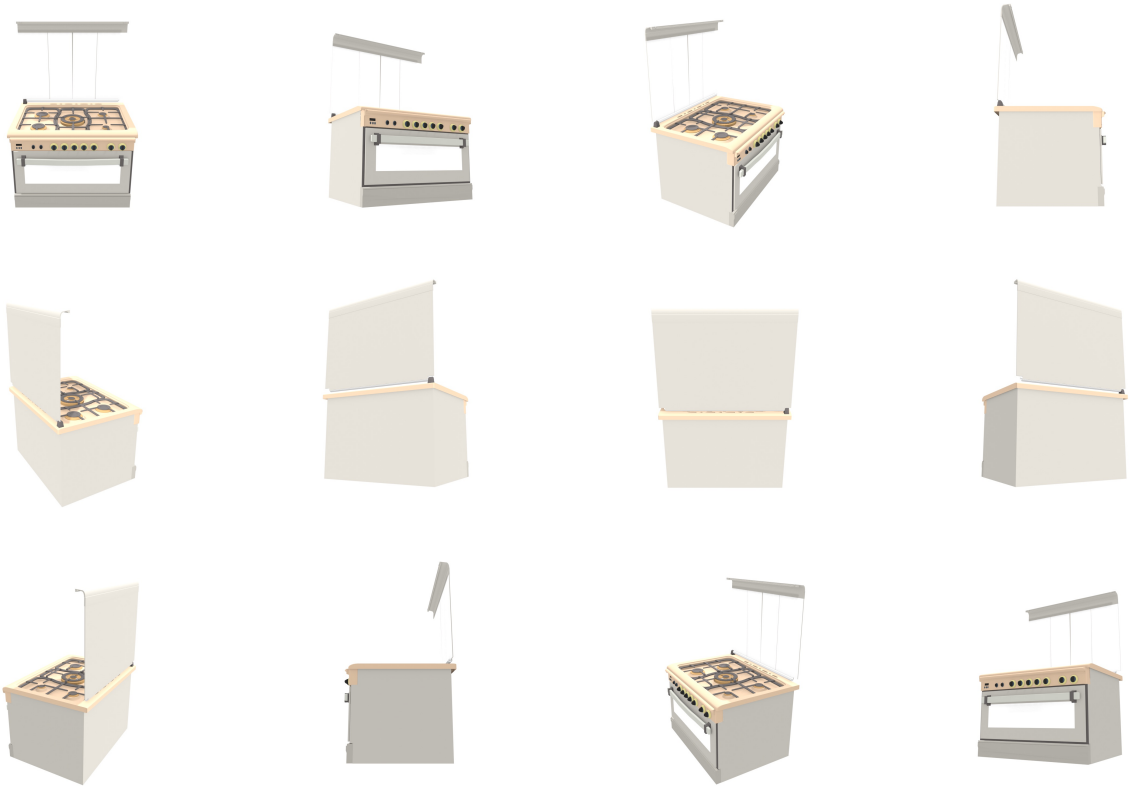

Figure 4.11: Sample views of Object 3 (12 views). Generated prompt: Create an image of a gas cooktop with an integrated oven. The design is sleek and contemporary. The cooktop section is made from a light-colored material, likely stainless steel, which reflects light and gives it a polished appearance. The hood extends outward, suggesting it is designed to cover the entire cooktop area. The edges of the hood are rounded, adding to the modern aesthetic. The color scheme is primarily metallic silver and black, with the black elements providing contrast against the silver surfaces.

5. Experiments

In this chapter, I describe the experimental validation of the proposed novel view synthesis method. The chapter begins by detailing the experimental setup, including data loading, evaluation metrics, and baseline training configurations. Subsequently, a series of targeted experiments are presented, designed to analyze the impact of different architectural components, conditioning strategies and training parameters. Each experiment's rationale, methodology, and results are discussed. The chapter also includes a qualitative analysis of generated images, a comparison with state-of-the-art methods where applicable, and an analysis of the model's inference efficiency.

5.1. Experimental Setup

This section outlines the common foundation for all experiments, including the data loading setup, the metrics for evaluation, and the standard training configuration of the proposed model.

5.1.1. Datasets

The experiments leverage two primary datasets, as detailed in Chapter 4:

- **Training Data**: The core training dataset is derived from ObjaverseXL [6], consisting of approximately 20,000 processed 3D models. For each model, a set of 6, 8, or 12 views were rendered at 1024×1024 resolution and subsequently resized to 768×768 for training. Textual prompts corresponding to these views were generated using the Qwen2.5VL multimodal LLM, as described in Section 4.2.5.
- **Validation and Test Data**: The Google Scanned Objects (GSO) dataset [8] serves as the testing set for evaluating generalization capabilities and for quantitative comparisons against state-of-the-art methods. Additionally, a small fraction of the processed ObjaverseXL dataset was reserved as an internal validation set to monitor training progress.

5.1.2. Data Loading and Preprocessing

The training and evaluation data, derived from the ObjaverseXL and GSO datasets, are processed and loaded as follows:

- **Dataset Splitting**: The collection of object files is deterministically split into training, validation, and test sets. This division is based on a specified ratio (e.g., 80% train, 10% validation, 10% test) applied after shuffling the list of all available object archives using

a fixed random seed. This ensures reproducibility of the splits across different runs and does not mix up the same object in different splits.

- **View Pair Generation**: For each object within a given split, pairs of source and target view pairs are constructed. They are randomly selected between each other, with a limit of 8 view pairs per object.

- **Individual Item Preprocessing**: When a specific view pair is constructed, the source and target images are loaded in a RGBA format, and then alpha-composited onto a white background before being converted to RGB. Images are resized to the specified $target_size$ (e.g., 768×768 pixels) using Lanczos resampling. The pixel values of the images are then normalized to the range [-1.0, 1.0] by scaling from [0, 255], since this is an expected format by VAE encoder.

This data loading pipeline ensures that the model receives consistently processed and structured input, suitable for training a Latent Diffusion Model.

5.1.3. Evaluation Metrics

To quantitatively test the performance of the novel view synthesis, the following metrics were employed:

- **Peak Signal-to-Noise Ratio (PSNR)**: Measures pixel-wise reconstruction accuracy. Higher is better.

- **Structural Similarity Index Measure (SSIM)** [34]: Assesses perceptual similarity based on luminance, contrast, and structure. Values range from -1 to 1, with 1 indicating perfect similarity.

- **Perceptual Loss**: Calculates the distance between feature activations in VGG16 model of generated and reference images, aligning better with human perception of image similarity [18]. Lower is better.

- **Learned Perceptual Image Patch Similarity (LPIPS)** [41]: Calculates the distance between deep features of generated and reference images, aligning better with human perception of image similarity. Lower is better.

- **Fréchet Inception Distance (FID)** [16, 32]: Compares the statistical similarity between distributions of generated images and real target views using InceptionV3 embeddings. Lower is better.

- **CLIP Score** [15]: Measures the cosine similarity between CLIP embeddings of the generated image and the target view. Higher is better.

5.1.4. Baseline Training Configuration

The proposed model, unless specified differently for a particular experiment, is trained using the following configuration:

- **Base Model Architecture**: Stable Diffusion 2.1, with the VAE and text encoder frozen. The U-Net is modified with the proposed image cross-attention adapters and FiLM layers for camera conditioning.
- **Trainable Parameters**: Only the newly introduced adapter layers, the $ImageEncoder$'s non-U-Net parts (if any beyond feature extraction), and the $CameraEncoder$ MLP and FiLM generator layers are trained. The original Stable Diffusion U-Net weights (when used for feature extraction in $ImageEncoder$ or as the backbone for the $MultiViewUNet$) remain frozen unless part of a full fine-tuning experiment.
- **Optimizer**: AdamW [23].
- **Learning Rate**: 1×10^{-5}.
- **Learning Rate Schedule**: Cosine annealing with a warm-up period of 500 steps.
- **Batch Size**: 6 per GPU, with gradient accumulation steps set to 1 (defined to change the effective batch size if needed).
- **Hardware**: 4 x NVIDIA A100 40GB GPUs.
- **Training Duration**: Typically 10 epochs for the 20k ObjaverseXL dataset, or a proportionally scaled number of steps for smaller datasets/experiments.
- **Image Resolution**: Input and output images are 768×768 pixels.
- **Default Conditioning Strengths**: img_ref_scale, $cam\_mod\_strength$.
- **Loss Function**: MSE loss with Min-SNR weighting strategy ($\gamma = 5.0$).
- **Classifier-Free Guidance (CFG) Scale (Inference)**: 1.0.
- **Inference Steps**: 20 steps using a DDPMScheduler.

5.2. Performed Experiments

This section presents a series of experiments designed to dissect the proposed architecture and study the impact of key design choices and training methodologies.

5.2.1. E1: Impact of Conditioning Strengths (img-ref-scale and cam-mod-strength)

Objective: To determine the optimal operating range and sensitivity of the model to the *img_ref_scale* (controlling influence of reference image features) and *cam_mod_strength* (controlling influence of camera FiLM modulation) hyperparameters.

Methodology: A hiperparameter tuning was performed by training separate models with various combinations of conditioning strength parameters. These experiments were conducted on a 5,000-sample subset of the ObjaverseXL dataset. The *img_ref_scale* values tested were {0.1, 0.25, 0.5, 0.75, 1.0} and *cam_mod_strength* values tested were {0.1, 0.25, 0.5, 0.75, 1.0}. Performance during these validation runs was evaluated using PSNR, SSIM, a Perceptual Loss (shown in Figure 5.2a), FID, and CLIP score. The training loss was also monitored.

Results and Discussion: The impact of varying conditioning strengths on different evaluation metrics is presented below. The Fréchet Inception Distance (FID) is a key metric for assessing the quality of generated images. Figure 5.2 shows the effect of conditioning strengths on Perceptual Loss, SSIM, CLIP score, and FID.

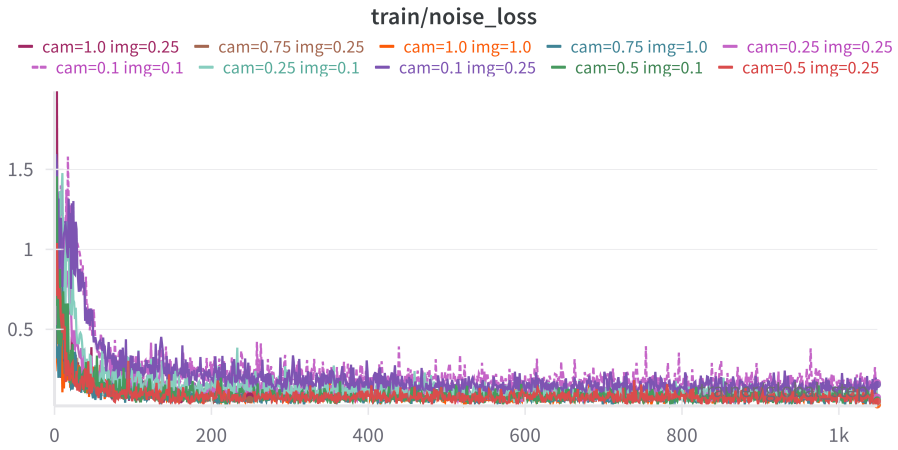

Figure 5.1: E1: Training MSE noise loss.

Impact of Camera Modulation Strength:

- Low Camera Strength: Consistently, low *cam_mod_strength* values, particularly 0.1, resulted in markedly inferior performance. These configurations exhibited the highest (worst) FID scores, generally above 170, and the lowest (worst) CLIP scores, around $0.68 - 0.69$, as well as the lowest SSIM values, around 0.88. This indicates that with weak camera conditioning, the model struggles to accurately interpret and apply the target view geometry, leading to poor synthesis quality and geometric inaccuracies.
- A *cam_mod_strength* of 0.25 showed an improvement over 0.1 but still significantly underperformed compared to stronger camera conditioning.
- High Camera Strength: Increasing *cam_mod_strength* to 0.5, 0.75, or 1.0 generally led to substantial improvements across FID, CLIP, and SSIM.

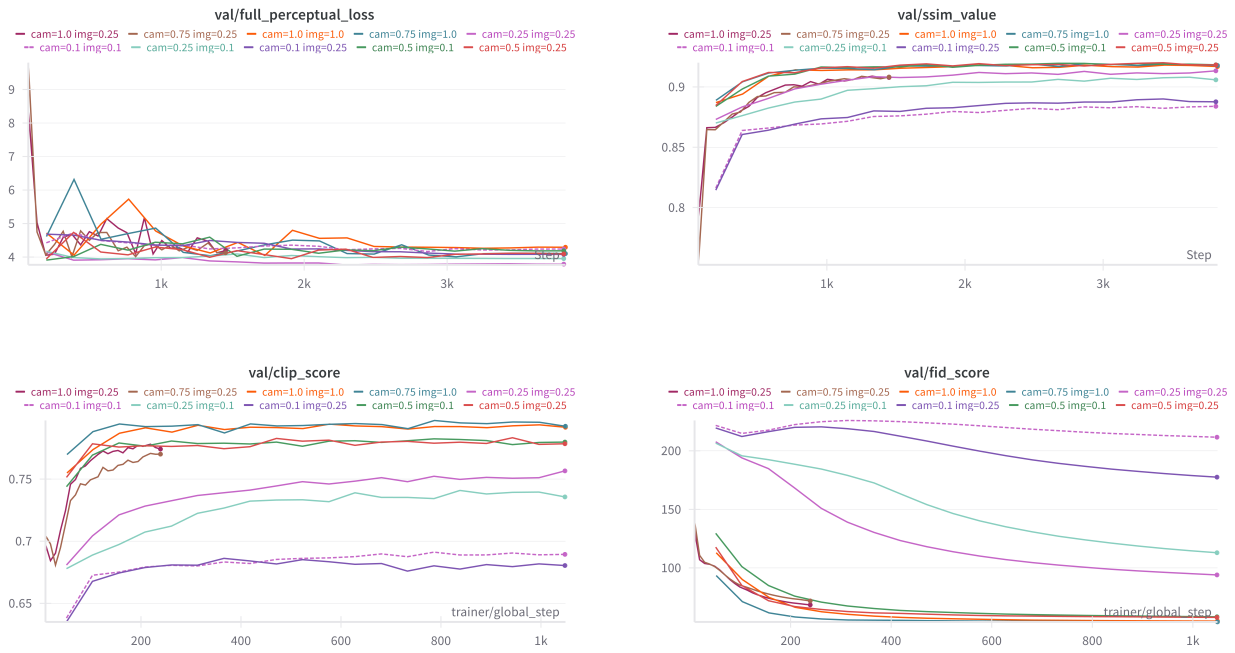

Figure 5.2: E1: Impact of conditioning strengths on various metrics: (a) Perceptual Loss, (b) SSIM, (c) CLIP Score, (d) FID Score.

Impact of Reference Image Conditioning Strength:

- Low Image Reference Strength: An img_ref_scale of 0.1, when paired with low to moderate $cam\_mod\_strength$, generally resulted in poorer FID, CLIP, and SSIM scores compared to higher img_ref_scale values. This suggests that some degree of visual information from the source view is beneficial for guiding the synthesis process.

- Moderate Image Reference Strength: An img_ref_scale of 0.25 emerged as a particularly effective value, especially for optimizing FID. When combined with adequate $cam\_mod\_strength$ (0.5 or higher), configurations such as $cam = 0.75img = 0.25$, $cam = 1.0img = 0.25$ and $cam = 0.5img = 0.25$ consistently achieved among the lowest FID scores (around 50-60). These combinations also yielded strong CLIP scores (approx. 0.76-0.78) and SSIM values (approx. 0.91-0.925).

- High Image Reference Strength: An img_ref_scale of 1.0, when paired with high $cam\_mod\_strength$, also demonstrated top-tier performance, particularly excelling in CLIP score (reaching up to ≈ 0.79) and SSIM (reaching up to ≈ 0.93). However, these configurations sometimes resulted in slightly higher (worse) FID scores compared to their $img\_ref\_scale = 0.25$ counterparts with similar high $cam\_mod\_strength$. This might suggest that while a strong reference image signal can enhance perceptual similarity and structural details (benefiting CLIP and SSIM), it might slightly constrain the model's ability to generate the most statistically faithful novel view if it adheres too closely to the source image features.

This hyperparameter sweep demonstrates that the model is sensitive to both $cam\_mod\_strength$ and img_ref_scale. For the most robust novel view synthesis, a $cam\_mod\_strength$ of at least 0.5 is recommended, with 0.75 or 1.0 often yielding the best results. For img_ref_scale, a value of 0.25 provides an excellent balance, particularly for FID, while $img\_ref\_scale = 1.0$ can also be highly effective, especially for maximizing CLIP and SSIM scores when paired with strong camera conditioning. In the future experiments, the parameters were set to $img\_ref\_scale = 0.75$ and $cam\_mod\_strength = 1.0$.

5.2.2. E2: Training Strategy: Adapters vs. Full Fine-tuning

Objective: To compare the performance, training efficiency (time and computational resources), and parameter efficiency of the proposed adapter-based training approach against full fine-tuning of the U-Net. This addresses the "Training adapter only vs full fine-tuning" point.

Methodology: Two main training strategies were compared. The first approach involved training only the lightweight adapter modules and camera conditioning encoder (adapter only training), keeping the backbone Stable Diffusion U-Net frozen. For a direct comparison on a smaller scale, this was performed on a 1,000-sample subset of ObjaverseXL for 10 epochs.

The second approach involved fine-tuning the entire U-Net along with the adapters and conditioning encoders (full U-Net fine-tuning). This was performed on a 1,000-sample subset of ObjaverseXL for 10 epochs for comparison with the adapter-only training on the same dataset size.

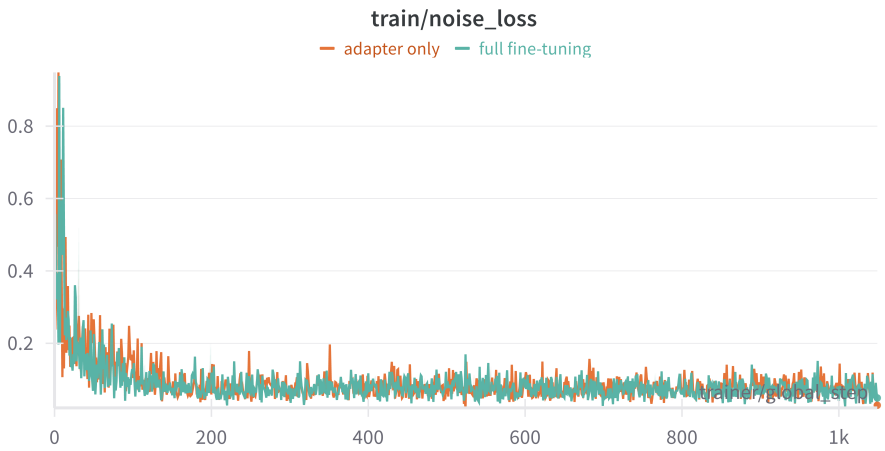

Figure 5.3: E2: Training MSE noise loss.

Results and Discussion: Training the adapter only was possible with batch size of 6 (per GPU) and full fine-tuning was possible with batch size of 2 (per GPU) with gradient accumulation steps set to 3 to match the effective batch size between the two approaches.

Out of the $2.9B$ parameters of the full model (U-Net + VAE + Text Encoder + Adapters), $585M$ are trainable in the adapter only method and $2.3B$ are trainable in the full fine-tuning method.

Figure 5.4: E2: Training adapters only vs. full fine-tuning of U-Net on various metrics: (a) Perceptual Loss, (b) SSIM, (c) CLIP Score, (d) FID Score.

The adapter only method, trained for 10 epochs took 9 hours on 4 x NVIDIA A100 40GB GPUs. The full fine-tuning method, trained for 10 epochs took 35 hours on the same hardware. This is a significant difference in training time, but the adapter only method is still able to achieve a comparable performance to the full fine-tuning method.

The quantitative results on auxiliary metrics are shown in Figure 5.4. The full fine-tuning method is able to achieve a slightly better performance on the Perceptual Loss metric, but the adapter only method achieves a comparable performance based on the SSIM, CLIP Score, and FID metrics.

5.2.3. E3: Camera Encoder Architecture Details

Objective: To investigate the impact of the *CameraEncoder*'s architectural design on novel view synthesis performance. Specifically, this experiment explores two main factors: (1) the depth of the Multi-Layer Perceptrons (MLPs) used to process relative rotation and translation camera parameters, and (2) the dimensionality of the internal hidden representations and the final camera embedding used for FiLM modulation.

Methodology: The *CameraEncoder* module computes a relative transformation (rotation matrix R and translation vector T) between source and target camera poses. The translation vector undergoes positional encoding. Both R (flattened) and the positionally encoded T are then processed by separate MLP streams before their outputs are concatenated and passed through a final projection MLP to produce the camera embedding. Four configurations of the *CameraEncoder* were evaluated, varying MLP depth and embedding dimensionality:

- **Shallower MLPs, Lower Dimensionality**: Rotation and translation encoder MLPs each consist of two linear transformation stages. The internal hidden dimension within these MLPs is 512, and the final camera embedding dimension is 1024.

- **Shallower MLPs, Higher Dimensionality**: Rotation and translation encoder MLPs each consist of two linear transformation stages. The internal hidden dimension is 1024, and the final camera embedding dimension is 2048.

- **Deeper MLPs, Lower Dimensionality**: Rotation and translation encoder MLPs each consist of three linear transformation stages. The internal hidden dimension is 512, and the final camera embedding dimension is 1024.

- **Deeper MLPs, Higher Dimensionality**: Rotation and translation encoder MLPs each consist of three linear transformation stages. The internal hidden dimension is 1024, and the final camera embedding dimension is 2048.

All four model variants, differing only in their *CameraEncoder* configuration as described, were trained using the adapter-only approach on a 1,000-sample subset of the ObjaverseXL dataset for 10 epochs.

Results and Discussion: The results are shown in Figure **??**. The deeper *CameraEncoder* is able to achieve a slightly better performance on the FID metric, but the standard *CameraEncoder* is able to achieve a comparable performance.

5.2.4. E4: Impact of Training Data Scale

Objective: To investigate how the size of the training dataset (number of unique 3D objects from ObjaverseXL) influences the modelś performance, particularly its generalization ability to unseen objects (GSO dataset) and overall robustness. This is especially relevant for addressing the "Adaptation to unseen objects" limitation of current state-of-the-art methods.

Methodology: The proposed model, using the adapter-only training approach and consistent conditioning strengths where possible, was evaluated on increasingly larger subsets of the processed ObjaverseXL dataset:

- **1,000 samples**.

- **5,000 samples**.

- **20,000 samples**.

All models were trained for a comparable number of effective updates relative to their dataset size where possible (e.g., by adjusting epoch counts or comparing at similar total iterations/epochs across all scales). Performance was evaluated on the GSO test set using all key metrics.

Results and Discussion: The results are shown in Figure **??**.

5.3. Comparison with State-of-the-Art Methods on GSO Dataset

This section presents the quantitative performance of the final proposed model on the Google Scanned Objects (GSO) dataset and compares it against relevant state-of-the-art methods.

5.3.1. Performance of the Proposed Method

The primary configuration of the proposed method for SOTA comparison was trained on 1,000 ObjaverseXL pairs for 25 epochs. Additionally, our best overall model, identified through the preceding experiments, was trained using the adapter-based approach on the 20,000-sample ObjaverseXL dataset with $img\_ref\_scale = 1.0$ and $cam\_mod\_strength = 1.0$. Both configurations were evaluated on a held-out test set of 100 samples from the GSO dataset. The evaluation protocol involved providing a source view and camera transformation to generate a target novel view, which was then compared against the ground truth.

5.3.2. Comparison with State-of-the-Art Methods

The performance of the proposed method is compared against two contemporary novel view synthesis techniques: Zero123++ and MVAdapter. The results, evaluated on 100 samples from the GSO dataset with aligned camera angles and consistent conditioning (reference image and text prompt), are presented in Table 5.1.

Table 5.1: Quantitative comparison with state-of-the-art methods on 100 samples from the GSO dataset.

Method	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**
Zero123++ [22]	19.27	0.84	0.12
MVAdapter [17]	11.48	0.78	0.16
Proposed Method (1k samples, 25 epochs)	13.12	0.81	0.20
Proposed Method (20k samples, 10 epochs)	[PSNR]	[SSIM]	[LPIPS]

5.3.3. Inference Time Analysis

Objective: To quantify the computational efficiency of the proposed model during the inference phase (novel view generation).

Methodology: The average inference time was measured for generating a single novel view at the target resolution of 768×768 pixels.

- **Hardware Used**: Single NVIDIA A100 40GB GPU.
- **Model Configuration**: The best performing adapter-based model (trained on 20,000 ObjaverseXL samples).

- **Inference Steps**: 20 diffusion steps.
- **Batch Size for Inference**: 1 (generating one view at a time).

The results will be compared to the inference time of other state-of-the-art methods Zero123++ [22] and MVAdapter [17] (with their default settings).

Table 5.2: Inference time comparison with state-of-the-art methods.

Method	**Inference Time (seconds)** ↓
Zero123++ [22]	6.51
MVAdapter [17]	19.42
Proposed Method	16.04

Results and Discussion: The proposed adapter-based model achieves an average inference time of 16.04 seconds per 768×768 image on an NVIDIA A100 GPU, using 20 denoising steps. This suggests that the proposed model is able to generate novel views in comparable time to the MVAdapter method. The Zero123++ method is still significantly faster, with an average inference time of 6.51 seconds.

5.4. Chapter Summary

This chapter detailed the comprehensive experimental evaluation of the proposed multi-view novel view synthesis method. The experimental setup, including datasets (ObjaverseXL for training, GSO for testing) and evaluation metrics (PSNR, SSIM, LPIPS, FID, CLIP score), was established.

Key findings from the experiments include:

- **Optimal Conditioning Strengths**: The investigation into img_ref_scale and $cam\_mod\_strength$ (Section 5.2.1) identified optimal ranges for these hyperparameters, demonstrating a balance is needed to effectively incorporate conditioning signals without introducing artifacts. The combination of $img\_ref\_scale = 1.0$ and $cam\_mod\_strength = 1.0$ was found to be effective.
- **Efficiency of Adapter-Based Training**: The comparison between adapter-only training and full U-Net fine-tuning (Section 5.2.2) highlighted the significant advantages of the adapter approach in terms of parameter efficiency and training resource reduction, while achieving competitive performance.
- **Camera Encoder Design**: Experiments with the *CameraEncoder* architecture (Section 5.2.3) suggested that the baseline moderately complex encoder offered a good balance of performance and efficiency.
- **Impact of Data Scale**: Increasing the training data size (Section 5.2.4) generally led to improved performance and generalization on the GSO dataset, underscoring the importance of large-scale diverse data for training robust NVS models.

Quantitative evaluation on the GSO dataset (Section 5.3) positioned the proposed method relative to SOTA approaches like Zero123++ and MVAdapter, showing that the proposed method is able to achieve a comparable performance to the state-of-the-art methods.

Overall, the experiments validate the architectural design choices and training strategies employed in this thesis, demonstrating a capable and efficient approach to diffusion-based novel view synthesis. Identified limitations and observations from these experiments also pave the way for potential future work, such as further exploration of more sophisticated camera encoders, or training on even larger and more diverse datasets to push generalization boundaries.

6. Conclusions and Future Work TODO

Zakończenie, podsumowuje najważniejsze wnioski, podaje możliwości dalszego rozwinięcia wykonanych prac i wskazuje obszar potencjalnego zastosowania pracy. Rezultaty pracy mają charakter poznawczy, mogą mieć charakter użytkowy. Należy dokonać analizy uzyskanych wyników. Rezultaty powinny charakteryzować się oryginalnością, a nawet w pewnym stopniu nowatorstwem. Praca zawiera (…). Zostało pokazane (…). Eksperymenty wykazały (…). Tu piszemy wnioski i obserwacje.

Widzimy, że (…). Z tego powodu przyszła praca powinna obejmować (…).

Bibliography

[1] S. Bai, K. Chen, X. Liu, J. Wang, W. Ge, S. Song, K. Dang, P. Wang, S. Wang, J. Tang, H. Zhong, Y. Zhu, M. Yang, Z. Li, J. Wan, P. Wang, W. Ding, Z. Fu, Y. Xu, J. Ye, X. Zhang, T. Xie, Z. Cheng, H. Zhang, Z. Yang, H. Xu, and J. Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[2] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, V. Jampani, and R. Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, Mar. 2003.

[4] Z. Chong, X. Dong, H. Li, S. Zhang, W. Zhang, X. Zhang, H. Zhao, and X. Liang. Catvton: Concatenation is all you need for virtual try-on with diffusion models, 2024.

[5] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[6] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi. Objaverse-xl: A universe of 10m+ 3d objects, 2023.

[7] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[8] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022.

[9] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[10] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv preprint arXiv:2405.10314*, 2024.

[11] G. Gkioxari, J. Malik, and J. Johnson. Mesh r-cnn, 2020.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[13] T. Hang, S. Gu, C. Li, J. Bao, D. Chen, H. Hu, X. Geng, and B. Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7441–7451, October 2023.

[14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[15] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi. Clipscore: A reference-free evaluation metric for image captioning, 2022.

[16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc.

[17] Z. Huang, Y. Guo, H. Wang, R. Yi, L. Ma, Y.-P. Cao, and L. Sheng. Mv-adapter: Multi-view consistent image generation made easy. *arXiv preprint arXiv:2412.03632*, 2024.

[18] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022.

[20] A. Lemay, C. Gros, O. Vincent, Y. Liu, J. Cohen, and J. Cohen-Adad. Benefits of linear conditioning for segmentation using metadata. 02 2021.

[21] T. Lindeberg. Scale invariant feature transform. *Scholarpedia*, 7, 05 2012.

[22] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023.

[23] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

[24] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

[25] M. Muja and D. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATIONs*, volume 1, pages 331–340, 01 2009.

[26] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer, 2017.

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.

[28] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[29] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571, 11 2011.

[31] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. 06 2016.

[33] P. Wang and Y. Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023.

[34] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[35] D. Watson, W. Chan, R. Martin-Brualla, J. Ho, A. Tagliasacchi, and M. Norouzi. Novel view synthesis with diffusion models, 2022.

[36] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[37] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[38] H. Ye, J. Zhang, S. Liu, X. Han, and W. Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. 2023.

[39] W. Yu, J. Xing, L. Yuan, W. Hu, X. Li, Z. Huang, X. Gao, T.-T. Wong, Y. Shan, and Y. Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis, 2024.

[40] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[41] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.