Rheinische
Friedrich-Wilhelms-Universität Bonn

Master Thesis

Context-aware Camera Controllable Video Generation

Author:
Luis Denninger

First Examiner:
Prof. Dr. Jürgen Gall

Second Examiner:
Prof. Dr. Maren Bennewitz

Supervisor:
Sina Mokhtarzadeh Azar

Date: November 10, 2025

Abstract

Generating controllable, high-fidelity videos remains challenging for image-to-video (I2V) diffusion models, which typically rely on a single reference frame. This limited context hampers temporal and 3D consistency under large camera motions. We introduce context-to-video (C2V) generation, which augments the diffusion process with additional posed context views, supplying missing scene information. Building on DynamiCrafter with camera control via CamI2V, we develop two variants: (i) an *Implicit Model* integrating the context views into the diffusion latent space through a 3D-aware encoder employing epipolar constrained attention, and (ii) an *Explicit Model* that leverages a 3D Gaussian scene representation that fuses context in world space before conditioning the diffusion model. We train on the RealEstate10K, and evaluate visual quality and camera trajectory alignment. Our approach improves visual quality by 28.89% (FVD ↓) and reduces the trajectory error by 18.35% (CamMC ↓) over strong camera-controlled I2V baselines. The *Explicit Model* variant yields the best fidelity and consistency, highlighting the benefit of explicit 3D supervision for video diffusion models.

Contents

1 Introduction

Over the last several decades, synthesizing novel data has been a key research goal across the graphics and vision community. While early approaches focus on knowledge-driven or physics-based methods [1], [2], more recently data-driven deep learning approaches trained on internet-scale data have shown impressive world understanding allowing them to generate novel data nearly indistinguishable from real data [3]–[5].

Such approaches are particularly interesting for the visual domain, generating images and videos, opening up new opportunities for content creation and editing. They have the potential to substantially reduce effort in complex design workflows and, in some cases, substituting years of specialized expertise with prompt engineering. However, these gains come with their own set of challenges. The high dimensionality of visual data, their inherent diversity and redundancy drive extensive computational demands and introduce critical failure cases, yet to be solved.

While many architectures such as Generative Adversial Networks (GANs) [6] and Variational Autoencoders (VAEs) [7] have been proposed to generate images, diffusion models [5], [8] have recently shown superior performance. Given a textual description of a desired image, text-to-image diffusion models are capable of generating high-fidelity images with impressive details [4], [9], [10].

To extend these models to the video domain, it has become common practice to distill video models from pre-trained image models [11], [12]. These text-to-video models are capable of generating short video clips from text prompts, but often lack temporal consistency, 3D consistency and motion dynamics [13]. To solve some of these issues, more recent models add an image condition that conveys the visual context of the target scene. While these models are faithful to the scene, they often lack significant motion, diverging to an image animation task instead of full video generation [14]–[16].

Moreover, their freedom in generation and the weak conditioning signals make it difficult to achieve a desired output. With the ultimate objective of building world generation models that can compete with traditional rendering engines, it is crucial to have fine-grained control over the generation process. Adding more control signals such as depth maps [16], 2D keypoints [14] or motion trajectories [15] has proven to

be effective for more generative control. One crucial control, the camera trajectory, has shifted into focus over the past year [13], [17], [18], allowing fine-grained camera control. This forces the models to not only represent the semantic and visual conditions conveyed through image and text inputs faithfully, but also generate new content along the camera trajectory beyond the scope of the image reference. This effectively takes image animation models to full video generation models that produce novel content with visible motion.

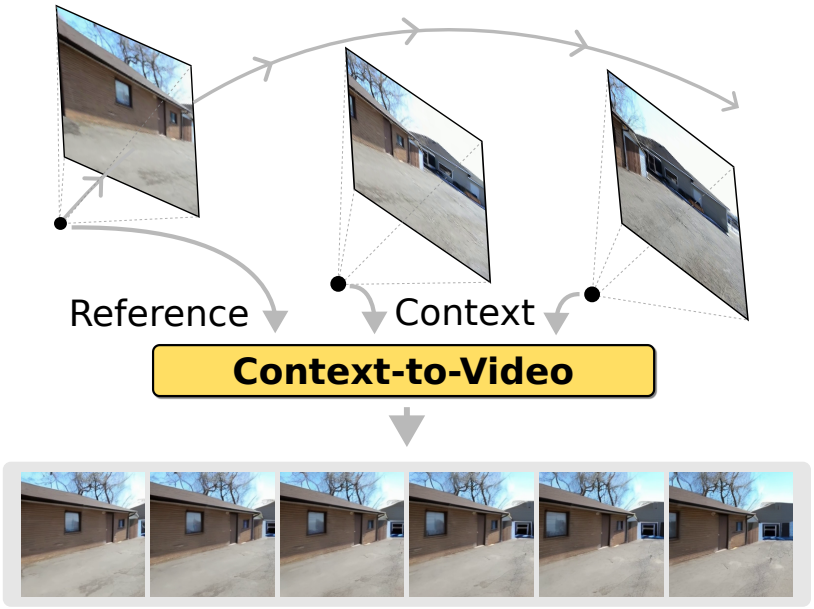


Figure 1.1: **Context-to-Video generation.** Conventional image-to-video generation employs a single reference image which only provides limited scene context. To provide additional scene context, e.g., after the camera pans to the right, context-to-video generation leverages additional context views providing crucial scene context unobserved in the reference image.

Problem Statement. Image-to-video (I2V) models leverage a single reference image which typically marks the initial frame of the video as shown in Figure 1.1. This provides a general visual context to the diffusion model. Optionally, models typically take a text prompt which primarily induces temporal dynamics, such as scene or camera motion, not visible in a still image. In the context of camera-controllable video generation, a camera trajectory additionally defines the camera position and orientation for each frame within the video. This directly enforces camera motion onto the diffusion process, e.g., panning the camera to the right as can be seen in Figure 1.1. While the reference image provides sufficient context for the initial frames, later frames reveal an unobserved portion of the scene. The diffusion model is required

to extrapolate from the given context, which often leads to impaired visual quality and an unfaithful scene representation. To mitigate this, we propose to add additional posed context views to provide a complete scene context to the generative process. As this conditioning significantly alters the visual grounding of the generative process and adds additional information, we coin this new setting context-to-video (C2V) generation.

Contributions. This thesis explores how existing I2V diffusion models can be augmented with the additional context conditioning to adjust them to the C2V generation task. Specifically, we build on top of DynamiCrafter [19], an open-source I2V diffusion model, and add fine-grained camera control using the CamI2V's approach [18]. For this, we explore two different architectures, one that implicitely merges context in the latent space of the diffusion model without significant 3D supervision, and one that builds an explicit 3D Gaussian representation in world space to condition the denoising U-Net on. To evaluate our models, we curate a new dataset based on RealEstate10K [20] to obtain high-quality captions and aligned depth maps and camera poses. We assess each model's performance on the new context-to-video generation task and compare them to state-of-the-art camera-controlled I2V models, showing the effectiveness of additional context in the generative process. In summary, our contributions are as follows:

- We propose an *Implicit Model* which fuses context implicitely in the latent space with weak geometric supervision using a context- and 3D-aware encoder.
- We present an *Explicit Model* that represents the context in world space using 3D Gaussians. To incorporate a strong geometric grounding, the view-specific renderings from the 3D Gaussians are used to condition the model.
- We extensively evaluate and compare our approaches in different settings on the RealEstate10K dataset achieving a 28.89% improvement in the FVD score and a 18.35% improvement in trajectory alignment (CamMC), demonstrating the effectiveness of additional context and explicit geometric grounding.

2 Related Work

In this chapter, we review existing literature and models related to our work on context-aware camera controllable video generation. First, in Section 2.1, we provide a comprehensive overview of existing diffusion models. As most video diffusion models are distilled from image diffusion models, and inherit their generative capabilities from those, this section mainly discusses approaches in the image domain. In the next section, Section 2.2, we discuss existing video diffusion models in detail and how they are distilled from image models. To get a better understanding of conditioning mechanisms used in concurrent diffusion models, we review theoretical conditioning approaches in Section 2.3. Next, in Section 2.4, we discuss existing approaches for camera control in video generation, highlighting their strengths and weaknesses. In Section 2.5, we finally discuss different scene representations suitable for our *Explicit Model*.

2.1 Diffusion Models

Diffusion models, originally proposed for the image domain by Sohl-Dickstein *et al.* [8] and later refined by Ho *et al.* [5], have emerged as a powerful class of generative models, capable of producing high-quality images that rival or surpass those generated by Generative Adversarial Networks (GANs) [6] and Variational Autoencoders (VAEs) [7]. Under the hood, they resemble a latent variable model that gradually transforms a simple distribution (i.e., Gaussian noise) into a complex data distribution (e.g., images) through a series of denoising steps. Please refer to Section 3.2 for a detailed explanation of the theoretical background of diffusion models.

Initial text-to-image (T2V) diffusion models directly operate in pixel space, posing a significant computational burden due to the high dimensionality of images. Previous works by Kingma and Welling [7] have demonstrated that operating in a lower-dimensional latent space can significantly reduce computational requirements while maintaining high-quality generation. Thus, Rombach *et al.* [4] propose a novel family of diffusion models, coined *Latent Diffusion Models* (LDMs), that operate in the latent space of a pre-trained VAE. Not only does this significantly reduce required

resources during training, but also significantly speeds up the generation process. As a compromise between computational efficiency and generation quality, the authors choose a VAE with a downsampling factor of 8×.

Ever since, LDMs have become the de-facto standard for T2V diffusion models with many concurrent works building upon this architecture [9], [10]. Subsequent works such as Imagen [10], DALL-E 2 [9] and Stable Diffusion [4] maintain the overall structure with small changes to the architecture. One major focus of these works is improving the text embeddings used to condition the generative process. Ramesh *et al.* [9] propose a two-stage model, with a prior model mapping captions to CLIP [21] image embeddings which a decoder is conditioned on to produce high-quality images. Saharia *et al.* [10] instead use a frozen T5 [22] text encoder, which in contrast to CLIP, was not trained on textual-visual alignment, and argue that increased complexity of the text encoder improves the generative quality more than increasing the diffusion model's size itself, indicating that the text encoder is the bottleneck in current T2V models. Further improvements were achieved through scaling up the model size, training data and further tuning of the VAE and text embedding [23].

Up until recently, the convolutional U-Nets [24] proposed by Ho *et al.* [5] have been the golden standard architecture for diffusion models. Contrary to the consensus that this architectural choice is critical to the diffusion model's performance, Peebles and Xie [25] demonstrated that a fully transformer-based architecture can achieve competitive results and benefits from modern training techniques and scaling properties of transformers. This architecture has since been adopted by several models such as Hunyuan-DiT [26] or PixArt-α [27] and show an improved visual quality compared against U-Net-based models such as SDXL [23].

In contrast to the common U-Net structure that disentangles local and global reasoning among convolutional and transformer layers and directly implements feature compression through the encoder-decoder architecture, *Diffusion Transformers* (DiTs) exhibit a free reasoning structure implemented through cascaded transformer layers, which inherently makes them less interpretable. Moreover, this architecture further allows simple extension to the video domain using 3D attention to model an additional time axis.

2.2 Video Diffusion Models

The next step in the realm of diffusion models is extending these approaches to the video domain. Stable Video Diffusion (SVD) [28] builds on top of the Stable Diffusion

2.1 (SD2.1) [4] to extend its approach to videos. The text-to-video model variant augments the diffusion U-Net's architecture with block-wise temporal attention layers to model the time dimension. It is trained on a curated dataset of $\sim$ 1M samples, in an iterative training scheme that initially trains on low-resolution videos of size 256×384 and gradually increases the resolution up to 576×1024. By replacing the text embedding with the CLIP image embedding of a reference frame and concatenating it to the noisy input latents, the model is further distilled to an image-to-video (I2V) model. The iterative training scheme to obtain a high-resolution model and the distillation process to an I2V model have since become a common practice in video domain [11], [29].

VideoCrafter1 [29] builds on the same idea extending Stable Diffusion 2.1 [4] through additional temporal attention layers to build a T2V model. In the further distillation process cross-attention layers are inserted into the U-Net blocks to aggregate features from a CLIP embedded image condition. While this provides a visual context, CLIP images features only express high-level semantic features and fail to convey crucial fine-grained visual details. The extension, DynamiCrafter [19], resolves this issue through its proposed *Dual-stream Image Injection* module which encodes the reference image into the latent space and concatenates it with the noisy latents along the channel dimension. Further, CLIP image and text embeddings are processed through a query transformer to achieve a better visual-language alignment, achieving videos closely following the provided visual context.

Similarly, I2VGen-XL [30] choose to directly inject the encoded VAE features of the reference image into the noisy latents and add a global encoder which processes the high-level CLIP features. In contrast to previous methods, instead of iteratively distilling higher resolution models, the authors propose to train a second LDM in a subsequent refinement stage which is trained on a video super-resolution task to further improve the generative quality and render videos of resolution 720×1280.

Recently, DiT-based methods such as VDT [31], GenTron [32] or CogVideoX [33] have become more prominent. While these models continue working in the latent space spanned by the SDXL [23] VAE, or the SD2.1 [4] VAE for lower resolution, they employ a fully transformer-based architecture. VDT and GenTron adopt the discussed distillation process pretraining the spatial self-attention on images, followed by injecting temporal self-attention layers to further distill it to a video diffusion model. In contrast, CogVideoX [33] employs full 3D self-attention layers mitigating any image pretraining.

2.3 Conditioning Mechanisms

Diffusion models are typically trained on a large corpus of data enabling it to learn the underlying data distribution generalizing well to novel scenes. Adding new conditions such as human poses [34], [35], motion [13], [36] or camera trajectories [17], [18], [37] to guide the generative process requires fine-tuning the model to incorporate them into the denoising process. However, fine-tuning the entire models is computationally expensive due to the large model size, making it nearly infeasible for most people. Moreover, such a step may significantly deteriorate the model's rendering quality due to overfitting on single datasets, *forgetting* its deep understanding learned during extensive training and distillation processes. Thus, recent works have focused on efficient techniques to incorporate new conditions into pre-trained diffusion models, while keeping the original model frozen.

A prominent approach, Low-Rank Adaptation (LoRA) [38], was introduced in the NLP community. Specifically, instead of adjusting the complete weight matrix $W_0 \in \mathbb{R}^{d\times k}$ of a given layer, an offset δW to the pre-trained weights is estimated through two matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ of significantly smaller rank $r << \min(d, k)$, yielding the update formula:

$$h = W_0 x + \delta W x = W_0 x + BAx. \tag{2.1}$$

This not only reduces the number of trainable parameters but also prevents large changes to the networks parameters, mitigating the *forgetting* problem.

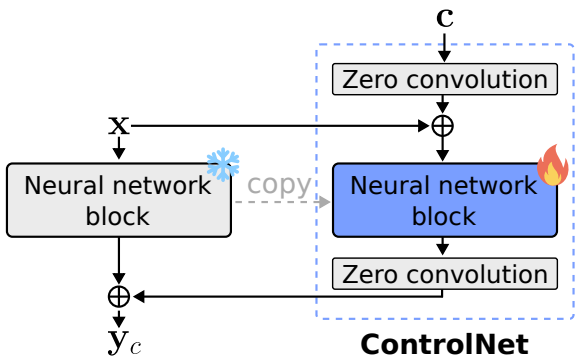


Figure 2.1: **ControlNet architecture.** The ControlNet architecture adds arbitrary conditioning signals by adding a control branch to a pre-trained diffusion model. The control branch is fine-tuned while the original model's weights are kept frozen, preventing the *forgetting* problem. The zero-convolution allows the model to gradually learn to incorporate the new condition without disrupting the original model's reasoning.

A more prominent approach for diffusion models is the ControlNet architecture [39], which introduces arbitrary conditions without fine-tuning of generative diffusion models. To prevent the common *forgetting* problem, the authors propose to introduce a copy of the original diffusion model, or parts of it, as seen in Figure 2.1. While the original network's weights are kept frozen, the copied network is then fine-tuned to incorporate the new condition. This ensures that the original model's capabilities are preserved, while the copied network learns to adapt the generative process based on the new condition. The added zero-convolution, a convolution with weight and bias parameters initialized to zero, allows the model to gradually insert the new condition without disrupting the model's reasoning and prevents the complete re-learning of the weights. Their experiment and follow-up work have shown that this approach can be widely applied across different modalities such as sketches, normal maps, human poses and many more [27], [40].

Following, ControlNet++ [41] focusses on improving the controllability through a pre-trained descriminative model which estimates the specific condition from the generated image, and compares it with the provided condition to define a cycle-consistency loss. As the copied network poses a significant computational overhead, Peng *et al.* [42] hypothesize that the main contribution of the control networks is to embed control features in the embedding space of the diffusion model. Instead, they propose a cross-normalization mechanism which normalizes denoising and control features to the same embedding space, which allows the definition of a light-weight conditioning module which can be trained from scratch, achieving a reduction of parameters of up to 90%.

Another line of work focusses on injecting such conditions into the existing conditional paths through small adapter networks that align the new condition to the existing conditioning space. This significantly reduces the number of trainable parameters and computational overhead but is known to struggle when inserting multiple different modalities as conditioning signal [43]–[45].

2.4 Camera Control in Video Generation

With the objective of building flexible video rendering engines from diffusion-based models, fine-grained camera control is a crucial component. Early works such as AnimateDiff [46] or Direct-a-Video [47] model camera movements through camera-motion primitives. In contrast, MotionCtrl [13] directly condition on the camera poses represented as rigid body transformations $T \in SE(3)$. These poses are processed independently for each block of the denoising U-Net through a shallow MLP

and injected into the temporal attention modules. While this supplies coarse cues about camera motion, it fails to provide fine-grained camera control. This stems from the fact that mapping a low-dimensional transformation to a dense pixel-wise representation is inherently ill-posed, especially since the method is also not provided with camera intrinsics.

More recently, models such as CamCo [37],CamI2V [18] or CameraCtrl [17] instead choose to directly condition on a pixel-level representation including camera extrinsics and intrinsics. Similar to Sitzmann *et al.* [48], they propose to embed it into a per-pixel 6D representation, encoding the ray direction $d \in \mathbb{R}^3$ from the camera center $p \in \mathbb{R}^3$ as Plücker coordinates $r \in \mathbb{R}^6$. Please see Section 3.4 for a detailed derivation of such representation. Using this representation, CameraCtrl [17] proposes a camera encoder of similar architecture as the denoising U-Net's encoder path and add shallow adaption layers [43] to insert it into the temporal attention blocks. This dense supervisory signal, allows for fine-grained camera control. CamI2V [18] extends their approach through injected epipolar attention layers that constrain the feature aggregation to geomtrically plausible regions, further improving the 3D consistency of the generated videos. CamCo [37] propose a camera conditioning mechanism inserted through adaption layers [43] into the temporal attention layers to guide the information aggregation between different frames.

2.5 3D Representations and Explicit Mapping

Building 3D scene representations is a well-studied problem in computer vision and robotics, and comes with many different requirements dependending on the application. In our case, we identify the following requirements with decreasing priority:

1. **Rendering Quality.** Our objective, rendering views along the given camera trajectory to condition the diffusion model on, requires high-quality renderings with fine details and minimal artifacts.
2. **Fast inference.** We intend to integrate and render new views on-the-fly during training, which requires fast integration and rendering times to prevent bottlenecks during training.
3. **Memory Efficiency.** As the rendering pipeline is only a marginal part of the overall model, we want to keep its memory footprint small to mitigate limitations during training.

In the following, we discuss different 3D representation and their high-level characteristics with respect to our requirements. For a detailed study comparing different

approaches we refer the reader to Section 6.7.2.

Natively each image can be back-projected to a colored point cloud as described in Section 3.4. While this representation maintains the maximum amount of information, thus allowing for high rendering quality, the memory size and rendering time grows linearly with the amount of context images. Moreover, the rendering quality heavily depends on the prior reconstruction and registration step producing the depth maps or point clouds.

Voxel grids. A common practice to reduce the computational overhead of raw point clouds is building voxel-based maps [49], [50]. A voxel grid is described through a set of voxels $\mathbf{V} = \{V_i\}_{i=0}^{M}$ of size m that are placed in a three-dimensional regular-spaced grid. In its simplest form each voxel describes the probability $p \in [0, 1]$ of a point being located in it and an additional color $\mathbf{c} \in \mathbb{R}^3$, called an *Occupancy Voxel Grid*. To integrate new information into the voxel grid, the ray of each pixel is casted through the voxel grid with intersecting voxels being updated according to a predefined sensor model as either *free* or *occupied*. Because the voxel size is fixed, image quality is bounded by that size, and both memory usage and runtime scale cubically with the grid resolution. The aggregation of points in voxels can also further correct small errors in the prior reconstruction step.

Instead of modeling the occupancy probability, *Truncated Signed Distance Field* (TSDF) voxel grids model the distance of each voxel to the closest surface using a truncated signed distance function with a truncation band η [50]. The truncation band limits the number of voxels being updated for a single ray, thus reducing the complexity of the integration step compared to *Occupancy Voxel Grids*. Moreover, the distance function allows for surface reconstruction with sub-voxel accuracy and intuitively allows for hole-filling [51].

In contrast, *Euclidean Signed Distance Field* (ESDF) voxel grids represent the global euclidean distance to the closest surface. Starting from an TSDF voxel grid, the ESDF voxel grid is typically built through a waveform propagation starting at each voxel to determine the distance of the closest surface. While this is advantageous for tasks like motion planning or collision checking, the additional integration step increases the complexity and does not provide any benefit for our application.

Neural reconstruction. Recently, neural representations of 3D scenes have gained popularity due to their high rendering quality and memory efficiency. Neural Radiance Fields (NeRF) [52] represent a scene through a neural network that maps a 3D location and viewing direction to a color and density value. While NeRFs can produce high-

quality renderings, they employ a test-time optimization strategy posing a significant overhead in both memory and runtime to learn the 3D representation. More recently, Gaussian Splatting [53] has been proposed, which represents a scene through a set of anistropic 3D Gaussians that can be directly optimized from posed images. Each Gaussian represents a position, covariance, opacity and view-dependent color, forming an explicit volumetric scene representation. The Gaussians can then be splatted onto the image plane through a simple alpha compositing approach producing high-quality images without expensive ray marching.

Follow-up works, such as MVSplat [54] or PixelSplat [55], have shown that these 3D Gaussians can be efficiently predicted in a feed-forward manner using a pre-trained neural network. Paired with an efficient GPU-based rendering approach, this allows for near-instant rendering of images, but requires more GPU memory than voxel-based approaches.

3 Preliminaries

Generative models approximate a hidden generative distribution $p(x)$ given a dataset $\{x_i\}_{i=1}^N$ sampled i.i.d. with some parameterized function $p_\theta(x)$ that we can effectively sample from. In modern era, the function approximator is defined through a neural network with parameters θ. The models can roughly be categorized into the following:

- Autoregressive generative models,
- Flow-based models,
- Latent variable models,
- Energy-based models,

where latent variable models such as VAEs, GANs and diffusion models have been the predominant choice in the visual domain.

The following first introduces the theoretical foundations of latent variable models and their training strategy in Section 3.1. Section 3.2 connects these insights to diffusion models and derives the training and sampling procedure. Further, we discuss the attention mechanism and its variants as they are the prevailing architecture of choice in diffusion models. Finally, in Section 3.4, we make a detour into projective geometry to establish the theoretical foundation of the employed camera model and epipolar geometry used to model the geometrical relation between distinct camera view points.

3.1 Latent Variable Models

To generate novel data, generative models typically approximate the unknown generative distribution function $p(x)$ using an internal tractable model $p_\theta(x) \sim p(x)$ given a set of samples $\{x_i\}_{i=1}^N$ sampled i.i.d. from the inaccessible generative distribution. Modern approaches modeling visual data, such as GANs, VAEs or diffusion models, simplify the generative process through a latent variable model that introduces a set of latent variables z such that the joint probability distribution of data x and latent variables z is learned:

$$p_\theta(x, z) = p_\theta(x|z)p(z). \tag{3.1}$$

The generative distribution is described by the marginal distribution:

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p(z)dz. \tag{3.2}$$

The parameters θ are then learned by maximizing the log-likelihood, i.e., minimizing the negative log-likelihood:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p_\theta(x_i) = \sum_{i=1}^{N} \log \int p_\theta(x_i|z)p(z)dz. \tag{3.3}$$

Since integrating over the unobservable latent space is infeasible, thus $\log p_\theta(x)$ being intractable, an approximate posterior $q_\theta(z|x)$ of simpler form is introduced. This allows us to reformulate the log-likelihood:

$$\begin{aligned}
\log p_\theta(x) &= \log \int p_\theta(x, z)\, dz && (3.4)\\
&= \mathbb{E}_{q_\theta(z|x)}\big[\log p_\theta(x)\big] && (3.5)\\
&= \mathbb{E}_{q_\theta(z|x)}\Big[\log \frac{p_\theta(x, z)}{q_\theta(z|x)}\Big] && (3.6)\\
&= \underbrace{\mathbb{E}_{q_\theta}\big[\log p_\theta(x, z) - \log q_\theta(z|x)\big]}_{\text{ELBO}} + \underbrace{\mathbb{E}_{q_\theta}\big[\log q_\theta(z|x) - \log p_\theta(z|x)\big]}_{\text{KL divergence}}. && (3.7)
\end{aligned}$$

The KL divergence serves as a similarity measure between two probability distributions and is by definition non-negative:

$$D_{KL}(q_\theta(z|x)||\log p_\theta(z|x)) = \int_Z q_\theta(z|x) \log \frac{\log p_\theta(z|x)}{q_\theta(z|x)} \geq 0. \tag{3.8}$$

The evidence lower bound (ELBO) in Equation (3.7) then postulates a lower bound

on our intractable log-likelihood:

$$\log p_\theta(x) = \log \int p_\theta(x,z)dz \tag{3.9}$$

$$= \log \int p_\theta(x,z)\frac{q_\theta(z|x)}{q_\theta(z|x)}dz \tag{3.10}$$

$$= \log \mathbb{E}_{q_\theta(z|x)}\Big[\frac{p_\theta(x,z)}{q(z|x)}\Big] \tag{3.11}$$

$$\geq \mathbb{E}_{q_\theta(z|x)}\Big[\log \frac{p_\theta(x,z)}{q_\theta(z|x)}\Big] \tag{3.12}$$

$$= \mathcal{L}_{ELBO}(x;\theta). \tag{3.13}$$

Finally, we can reformulate the ELBO term to:

$$\mathcal{L}_{ELBO}(x;\theta) = \mathbb{E}_{q_\theta}\big[\log p_\theta(x|z)\big] + \mathbb{E}_{q_\theta}\big[\log p_\theta(z)\big] - \mathbb{E}_{q_\theta}\big[\log q_\theta(z|x)\big] \tag{3.14}$$

$$= \underbrace{\mathbb{E}_{q_\theta}\big[\log p_\theta(x|z)\big]}_{\text{reconstruction term}} - \underbrace{D_{KL}(q_\theta(z|x)||p_\theta(z))}_{\text{prior-matching term}}, \tag{3.15}$$

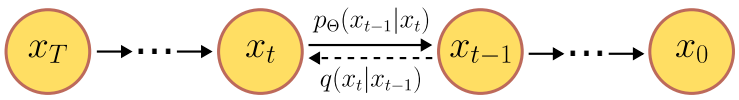
providing us with a more intuitive interpretation of the ELBO. Since the ELBO serves as a lower bound to the log-likelihood, maximizing the ELBO drives $p_\theta(x|z)$ towards the true posterior. Latent variable models use this insight to approximate the inaccessible generative distribution by maximizing the evidence lower bound.

3.2 Diffusion Models

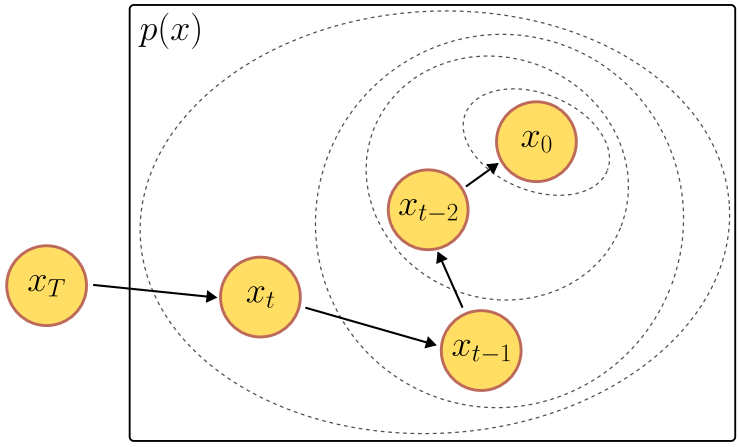
Diffusion Denoising Probabilistic Models [56] (DDPM) are latent variable models of form $p_\theta(x_0) = \int p_\theta(x_{0:T})dx_{1:T}$ defined through a parameterized Markov chain as depicted in Figure 3.1 (a). The latent space is spanned by the intermediate steps $x_1, \ldots, x_T$ which have the same dimensionality as the original data $x_0 \sim q(x_0)$

Forward process. The *forward process* or *diffusion process* $q(x_t|x_{t-1})$ gradually adds noise to the original image x_0 until all information is lost. Specifically, the Markov chain adds Gaussian noise according to a variance schedule $\beta_1 \ldots \beta_T$:

$$q(x_{1:T}|x_0) \coloneqq \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}). \tag{3.16}$$



(a) Diffusion markov chain.



(b) Diffusion probabilistic view.

Figure 3.1: **Diffusion process.** (a) Markov chain showing the forward process $q(x_t|x_{t-1})$ transforming data x_0 to pure noise x_T and the learned reverse process $p_\theta(x_{t-1}|x_t)$ gradually reconstructing the data. (b) Probabilistic view in data space $p(x)$: starting from x_T the reverse process moves samples toward high-density regions of the data distribution, yielding a sample x_0.

While the variance schedule may be learned through reparameterization [57], $\beta_1 \dots \beta_T$ are usually defined as constant hyperparameters. Through reformulation of the forward process with $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, we can sample from it at an arbitrary timestep t in closed form:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)\mathbf{I}). \tag{3.17}$$

Reverse process. The *reverse process* or *denoising process* then subtracts the noise to step-wise recover the original information starting from $p(x_T) = \mathcal{N}(x_T; \mathbf{0}, \mathbf{I})$. It is defined through a Markov chain with learned Gaussian transitions:

$$p_\theta(x_{0:T}) \coloneqq p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \quad p_\theta(x_{t-1}|x_t) \coloneqq \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \boldsymbol{\Sigma}_\theta(x_t, t)), \tag{3.18}$$

where $\boldsymbol{\mu}_\theta(x_t, t)$ and $\boldsymbol{\Sigma}_\theta(x_t, t)$ describe the output of some learnable score predictor network. In practice, the later is typically replaced by parameterization through fixed variances $\boldsymbol{\Sigma}_\theta(x_t, t) = \sigma_t^2\mathbf{I}$.

Training. Learning the parameters of the score predictor network $\boldsymbol{\mu}_\theta(x_t, t)$ follows the idea derived in Section 3.1 minimizing the variational bound on the negative log-likelihood:

$$\mathbb{E}\left[-\log p_\theta(x_0)\right] \leq \mathbb{E}_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \tag{3.19}$$

$$= \mathbb{E}_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] =: L. \tag{3.20}$$

Further reformulation allows us to decompose the log-likelihood in the timstep-wise terms:

$$L = \mathbb{E}_q\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right] \tag{3.21}$$

$$= \mathbb{E}_q\left[-\log p(x_T) - \sum_{t\geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] \tag{3.22}$$

$$= \mathbb{E}_q\left[-\log p(x_T) - \sum_{t>1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} - \log \frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right] \tag{3.23}$$

$$= \mathbb{E}_q\left[-\log p(x_T) - \sum_{t>1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \cdot \frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} - \log \frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right] \tag{3.24}$$

$$= \mathbb{E}_q\left[-\log \frac{p(x_T)}{q(x_T|x_0)} - \sum_{t>1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} - \underbrace{\log p_\theta(x_0|x_1)}_{\mathcal{L}_0}\right] \tag{3.25}$$

$$= \mathbb{E}_q\Big[\underbrace{D_{\mathrm{KL}}(q(x_T|x_0) \parallel p(x_T))}_{\mathcal{L}_T} + \underbrace{\sum_{t>1} D_{\mathrm{KL}}(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t))}_{\mathcal{L}_{t-1}} - \mathcal{L}_0\Big]. \tag{3.26}$$

Thus, learning the hidden generative distribution boils down to fitting the distribution $p_\theta(x_{t-1}|x_t)$ to the approximate posterior $q_\theta(x_{t-1}|x_t)$ at each discrete timestep t by minimizing the Kullback-Leibler divergence. Since $q_\theta(x_{t-1}|x_t)$ is not tractable, x_0 is added to the evidence such that the forward step is well-defined and we can reduce the posterior to a tractable Gaussian:

$$\begin{gathered} q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}; \tilde{\boldsymbol{\mu}}_t(x_t, x_0), \tilde{\beta}_t \mathbf{I}), \\ \text{where} \quad \tilde{\boldsymbol{\mu}}_t(x_t, x_0) \coloneqq \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t \quad \text{and} \quad \tilde{\beta}_t \coloneqq \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \end{gathered} \tag{3.27}$$

As the forward process does not contain any learnable parameters, the $\mathcal{L}_T$ term is constant and thus can be ignored during training.

Further, the terms $\mathcal{L}_{t-1}$ can be simplified to a re-weighted squared error term since both, the forward process $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \boldsymbol{\mu}_\theta(x_t, t), \sigma_t^2\mathbf{I})$ and the reverse process $q(x_t|x_{t-1}) \coloneqq \mathcal{N}(x_t; \tilde{\boldsymbol{\mu}}_t(x_t, x_0), \beta_t\mathbf{I})$, describe Gaussian distribution with small variances such that $\sigma_t^2 \approx \beta_t$:

$$\mathcal{L}_{t-1} = \mathbb{E}_q\left[\frac{1}{2\sigma_t^2}\|\tilde{\boldsymbol{\mu}}_t(x_t, x_0) - \boldsymbol{\mu}_\theta(x_t, t)\|^2\right] + C. \tag{3.28}$$

It is evident that the objective is to approximate the posterior mean $\tilde{\boldsymbol{\mu}}_t$ of the forward process. To further simplify the objective, we drop constant C and use the reparameterization trick on Equation (3.17) to obtain $x_t(x_0, \boldsymbol{\epsilon}) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$ for $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and insert it into Equation (3.27):

$$\begin{aligned} L_{t-1} &= \mathbb{E}_{x_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\tilde{\boldsymbol{\mu}}_t\left(x_t(x_0,\boldsymbol{\epsilon}), \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t(x_0,\boldsymbol{\epsilon}) - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon})\right) - \boldsymbol{\mu}_\theta(x_t(x_0,\boldsymbol{\epsilon}), t)\right\|^2\right] \\ &= \mathbb{E}_{x_0,\boldsymbol{\epsilon}}\left[\frac{1}{2\sigma_t^2}\left\|\frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0,\boldsymbol{\epsilon}) - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}\right) - \boldsymbol{\mu}_\theta(x_t(x_0,\boldsymbol{\epsilon}), t)\right\|^2\right] \end{aligned} \tag{3.29}$$

Thus, $\boldsymbol{\mu}_\theta$ is trained to predict $\frac{1}{\sqrt{\alpha_t}}(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon})$. Through the specifically choosen parameterization:

$$\begin{aligned} \boldsymbol{\mu}_\theta(x_t, t) &= \tilde{\boldsymbol{\mu}}_t\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(x_t))\right) \\ &= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(x_t, t)\right), \end{aligned} \tag{3.30}$$

we can further simplify the objective to:

$$\mathbb{E}_{x_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]. \tag{3.31}$$

This objective now resembles a weighted squared error term close to the one used in denoising score matching. Further, we drop the variance weighting of the error term:

$$L(\theta) \coloneqq \mathbb{E}_{t,x_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right], \tag{3.32}$$

to obtain the final simplified loss function used to train the parameters θ of the score predictor $\boldsymbol{\mu}_\theta$.

Sampling. To sample from the learned generative function (Equation (3.18)), the initial state of the reverse process $x_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is sampled from a normal distribution. Then, reparameterizing Equation (3.18), we can auto-regressively sample x_{t_1} for timesteps $T, \dots, 1$ using:

$$x_{t-1} = \frac{1}{\sqrt{\hat{\alpha}_t}}(x_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\boldsymbol{\epsilon}_\theta(x_t, t)), \tag{3.33}$$

such that we obtain the fully denoised variable x_0 with $p_\theta(x_0) \approx p(x_0)$.

3.3 Attention

The transformer architecture and its attention mechanism, proposed by Vaswani *et al.* [58], has revolutionized neural network architectures. The implemented scaled-dot product attention excels in global feature aggregation and has ever since been widely adopted in most generative models, especially to model global temporal and spatial relations.

Specifically, the attention mechanism computes the similarity between a set of queries $q \in \mathbb{R}^{N \times D_k}$ and a set of keys $k \in \mathbb{R}^{M \times D_k}$ to retrieve information from a set of values $v \in \mathbb{R}^{M \times D_v}$, using:

$$\text{Attn}(q, k, v) = \text{SoftMax}(\frac{qk^\intercal}{\sqrt{D_k}})v. \tag{3.34}$$

Since each token is processed independently to its position in the sequence the default attention mechanism is position-invariant. To retain positional information, a sinusoidal frequency encoding is added to each token:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(\frac{pos}{10000^{2i/D_k}}), \\ PE_{(pos,2i+1)} &= \cos(\frac{pos}{10000^{2i/D_k}}), \end{aligned} \tag{3.35}$$

where *pos* describes the position and i the dimension.

Through an additional attention mask $\mathbf{M}$, the attention can be restricted to specific tokens, e.g. to ensure causal reasoning:

$$\text{Attn}(q, k, v, \mathbf{M}) = \text{SoftMax}(\frac{qk^\intercal}{\sqrt{D_k}} \odot \mathbf{M})v. \tag{3.36}$$

Different heads head_i then project q, k and v into separate feature spaces defined through the projection matrices $W_i^Q \in \mathbb{R}^{D_{model} \times D_k}$, $W_i^K \in \mathbb{R}^{D_{model} \times D_k}$ and $W_i^V \in$

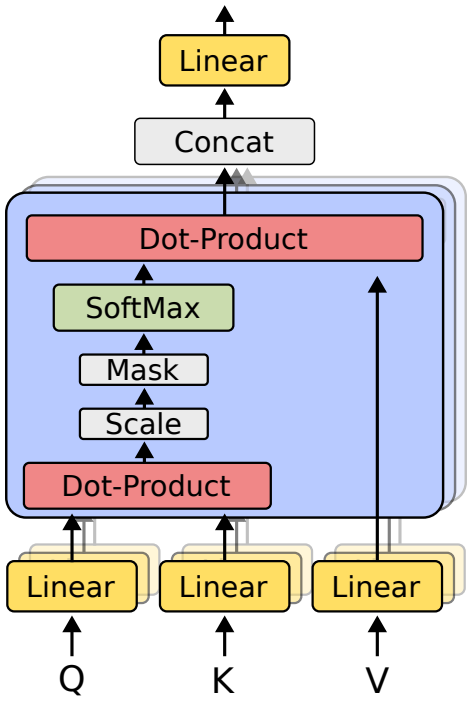


Figure 3.2: **Scaled dot-product attention.** The queries $Q \in \mathbb{R}^{N \times d_q}$, keys $K \in \mathbb{R}^{N \times d_k}$ and values $V \in \mathbb{R}^{N \times d_k}$ are linearly encoded into a embedding space of dimension d. The attention weights $A = \text{SoftMax}(\frac{QK^\intercal}{\sqrt{d}})$ score each key for every query in order to selectively retrieve from the values: $out = AQ$. Multi-head attention repeats this in parallel over independent embedding spaces and concatenates the per-head outputs.

$\mathbb{R}^{D_{model} \times D_v}$ and independently apply the attention mechanism:

$$\text{MultiHead}(q, k, v) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O, \tag{3.37}$$

where $\text{head}_i = \text{Attn}(QW_i^Q, KW_i^K, VW_i^V)$. This defines the multi-head attention mechanism as displayed in Figure 3.2.

Natively, key k, query q and value v are the same input sequence, coined *Self-Attention*. Alternatively, exchanging k and v with another sequence allows to attend to different sources, coined *Cross-Attention*.

3.4 Projective Geometry

To guide the diffusion process along a provided camera trajectory, we first need to define the camera model and establish the geometric relation between the different

viewpoints. We define a camera viewpoint through its extrinsics $E \in SE(3)$ and the intrinsics $K \in \mathbb{R}^{3\times3}$. Dependent on the application and lense type various camera models exists, ranging from simple pinhole cameras to models compensating for complex fish-eye lenses [59], [60]. For many use-cases the simple pinhole camera model provides a good trade-off between accuracy and complexity.

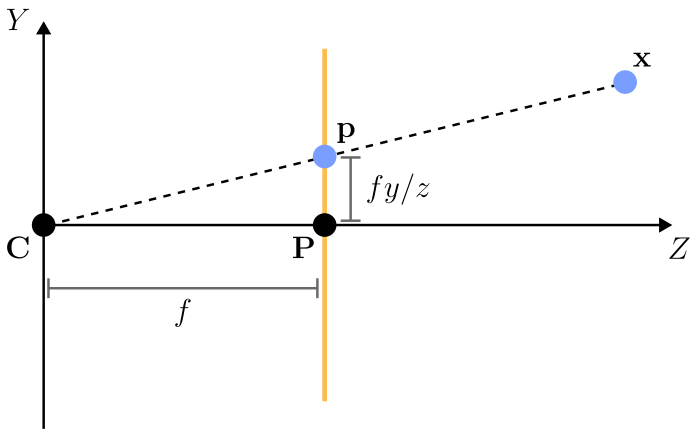


Figure 3.3: **Pinhole camera model.** A 3D point $\mathbf{x}$ is projected onto the image plane at pixel coordinates $\mathbf{p}$ through the camera center C.

Pinhole camera model. The pinhole camera model, illustrated in Figure 3.3, describes the projective transformation of a 3D point $\mathbf{x} = [x, y, z, 1]^\intercal$ in homogeneous coordinates to its corresponding pixel coordinates $\mathbf{p} = [u, v, 1]^\intercal$ on the image plane through the camera center $\mathbf{C}$. The model is uniquely described through the focal length parameters f_x, f_y defining the distance of the image plane to the camera center, and the principal point c_x, c_y which are summarized in the camera intrinsic matrix K:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.38}$$

The camera extrinsics $E = [R|t]$ describe the rigid body transformation from world coordinates to camera coordinates through a rotation $R \in SO(3)$ and a translation $t \in \mathbb{R}^3$. Combining both, we can describe a point in world coordinates as pixel coordinates on the image plane through the projective function $\pi(\cdot)$:

$$\mathbf{p} = \pi(\mathbf{x}) = K[R|t]\mathbf{x}. \tag{3.39}$$

To obtain the final pixel coordinates $[u, v]^\intercal$ in Cartesian coordinates, we dehomogenize $\mathbf{p} = [p_1, p_2, p_3]^\intercal$:

$$u = \frac{p_1}{p_3}, \quad v = \frac{p_2}{p_3}. \tag{3.40}$$

The inverse projection $\pi^{-1}(\cdot)$, which maps pixel coordinates back to 3D world coordinates, is not uniquely defined since depth information is lost during the projection. Thus, the inverse projection can only be determined up to a scale factor s, leading to a ray in 3D space rather than a unique point:

$$s\mathbf{x} = \pi^{-1}(\mathbf{p}) = [R^\intercal | -R^\intercal t]K^{-1}\mathbf{p} \tag{3.41}$$

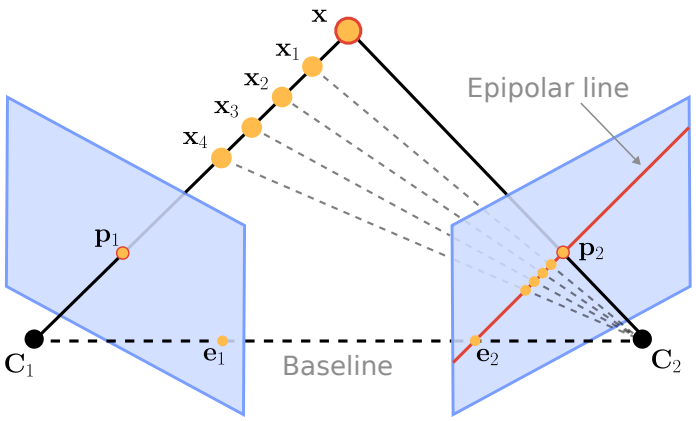


Figure 3.4: **Epipolar geometry.** Given a pixel $\mathbf{p}_1$ in the first image, the inverse projection defines a ray in 3D space along which the corresponding 3D point $\mathbf{x}$ lies on. Projecting this ray into the second image, we obtain the epipolar line on which the corresponding pixel $\mathbf{p}_2$ lies.

Epipolar geometry. Employing the pinhole camera model, the epipolar geometry describes the geometric relation between two distinct camera viewpoints. Specifically, it describes the relation of a 3D point and its projections onto the respective image planes. Assuming a static scene, cameras C_1 and C_2 are described through their extrinsics extrinsic E_1, E_2 and intrinsic values K_1, K_2. Given a pixel $\mathbf{p}_1$ in the first image, the objective is to define the corresponding pixel $\mathbf{p}_2$ in the second image.

Without additional information the backprojection, as illustrated in Figure 3.4, defines a ray in 3D space along which the 3D point $\mathbf{x}$ may lie. Projecting this ray into the second image, we obtain the epipolar line on which the corresponding pixel $\mathbf{p}_2$ lies. All epipolar lines intersect in the epipole, which describes the projection of the camera center C_1 into the second image. Thus, the epipolar line can be described

through the epipole $\mathbf{e}_2$ and an arbitrary point $\mathbf{p}_i$ on the casted ray, projected into the second image plane:

$$\mathbf{p}_i = K_2 E_2 E_1^{-1} K_1 \mathbf{p}_1. \tag{3.42}$$

The fundamental matrix F then describes the mapping between the image planes with:

$$\begin{aligned} F &= K_2 E_2 E_1^{-1} K_1^{-1} \\ &= K_2 E K_1^{-1}, \end{aligned} \tag{3.43}$$

where E is the essential matrix describing the geometrical relation between two calibrated cameras.

Finally, we can define the *Epipolar Constraint* that constraints the corresponding pixel $\mathbf{p}_2$ to lie on the epipolar line:

$$\mathbf{p}_2^\intercal F \mathbf{p}_1 = 0. \tag{3.44}$$

3D ray representation. Instead of parameterizing a camera pose through its extrinsics $E = [R|t]$ and intrinsics K, baseline models such as CameraCtrl [17] or CamI2V [18] opt for a pixel-wise ray embedding which provides the neural network with a dense supervisory signal. Each ray $r_i \in \mathbb{R}^6$ casted through pixel $\mathbf{p}_i$ is parameterized by its direction $d \in \mathbb{R}^3$ from the camera center $\mathbf{C}$ in Plücker coordinates:

$$r = \langle m, d \rangle \in \mathbb{R}^6, \tag{3.45}$$

with $m = p \times d \in \mathbb{R}$ describing the moment vector, where its norm represents the distance to the world origin. This allows the definition of a dense ray representation $M_{ray} \in \mathbb{R}^{h \times w \times 6}$ for pixels $\{(u, v)_i\}_{i=0}^{hw}$ computed through the previously introduced inverse projection function:

$$d = R^{-1} K^{-1} \cdot [u, v, 1]^\intercal, \quad m = (-R^{-1} t) \times d. \tag{3.46}$$

4 Dataset

High-quality video datasets are generally scarce. Often such datasets are produced for specific domains such as temporal action segmentation or action recognition which show diverse scenes in terms of actions but fail to deliver visually diverse scenes as required by deep learning-based video generation models [19], [28]. Moreover, our application not only requires visually diverse scenes but also annotated camera parameters and aligned depth maps. One commonly used dataset is RealEstate10K [20] which contains diverse camera trajectories and scenes in a limited domain of indoor and outdoor house tours.

4.1 RealEstate10K

RealEstate10K	**Original**		**Ours**	
	Train	Test	Train	Test
# Videos	71,556	7,711	62,596	7,218
Video length (s)	19.54 ± 11.44	19.60 ± 11.41	19.44 ± 11.39	19.61 ± 11.39
Displacement (m)	2.52 ± 5.52	2.53 ± 2.74	0.61 ± 0.53	0.61 ± 0.52
Rotation (Deg)	28.07 ± 28.07	27.50 ± 28.65	29.52 ± 29.66	29.55 ± 29.93

Table 4.1: **RealEstate10K statistics.** We compare the statistics of the ground-truth annotations provided with the dataset to our pseudo ground-truth annotations obtained through our annotation pipeline. The displacement and rotation statistics are reported without metric scale correction.

The RealEstate10K dataset comprises clips of indoor and outdoor house tours annotated with camera extrinsics and intrinsics. The dataset is gathered from about $10,000$ YouTube videos depicting house tours of Sotheby's real estate agency. These videos are further split into about $71,556$ short clips in the training set and $7,711$ in the test set lasting 20 s on average, comprising a total of $\sim$ 10 million frames. The authors further employ the ORB-SLAM2 [61] pipeline with a subsequent bundle adjustment step to generate camera extrinsic and intrinsic annotations. As a subset

of the YouTube videos have been deleted since or are region-locked, only 62,596 clips in the training set and 7,218 in the test set are available to us. In-depth statistics of the dataset are presented in Table 4.1.

4.2 Annotation Pipeline

Our data annotation pipeline consists of a captioning model further explained in Section 4.2.1 and a 3D reconstruction model detailed in Section 4.2.2. The complete pipeline is depicted in Figure 4.1.

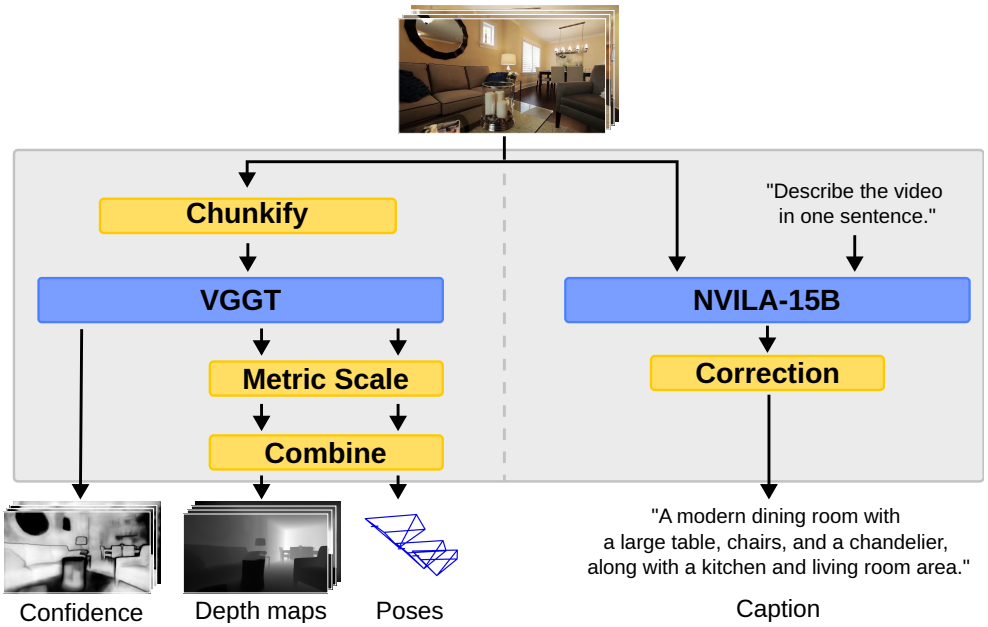


Figure 4.1: **Data annotation pipeline.** An overview of our data annotation pipeline. NVILA-15B [62] generates a caption for each video and VGGT [63] produces camera extrinsics, intrinsics, depth and confidence maps. To fit memory limits, videos are split into 100-frame chunks and later recombined. Using the ground-truth camera poses, we estimate a metric scale correcting the depth maps and estimated poses.

4.2.1 Captioning

Captions are not natively provided for the RealEstate10K dataset. The authors of CameraCtrl [17] provide captions which were consecutively used by the following works of Zheng *et al.* [18]. Unfortunately, our training sets mismatch due to the

above discussed problems, resulting in missing captions for many videos. Thus, we decide to produce captions for the complete dataset from scratch.

Our captioning pipeline builds on top of the NVILA [62] model family. Specifically, we use the NVILA-15B model. We have tested out different captioning prompts with the objective of producing a short and concise description. For instance, the default command *"Please describe the video"* produces long descriptions, which significantly mismatch in form to the captions used to train the baseline DynamiCrafter model. Thus we decided for the short prompt of:

"Describe the video in one sentence.",

which proved to give the best results.

An issue observed with all prompts, is the prepended descriptive phrase *"The video showcases"*. Explicitely mitigating this phrase through the prompt lead to impaired caption quality. Thus, we decided to simply remove the beginning phrase post-captioning, resulting in our final captions.

We ran our captioning pipline on one NVIDIA A100 with a throughput of 1.87 seconds per video, resulting in a total compute time of 36.26 hours. The captions are made publicly available[1].

4.2.2 Geometry Annotation

Further, for our ablation studies on different scene representations used in the *Explicit Model*, we require a dense 3D reconstruction of the scene. Recent advances in neural structure-from-motion approaches such as VGGSfm [64], DeepSfm [65] or VGGT [63] have shown impressive performance in producing dense 3D reconstructions and aligned camera annotations, which we require for our approach. Thus, we decided to build our geometry annotation pipeline on the state-of-the-art model VGGT.

VGGT consists of a feature encoder producing geometry-aware frame-wise features which are then processed by different heads producing camera parameters, comprised of the camera poses $P_{vggt} = ([\hat{\mathbf{R}}_i|\hat{\mathbf{t}}_i])_{i=1}^{T}$ and intrinsic values K_{vggt}, as well as depth maps $\mathbf{d}_{vggt}$. Additional confidence maps $\mathbf{c}_{vggt}$ are produced describing the pixel-wise confidence in the produced dense reconstruction. While the memory footprint and compute time is low for a small batch of images, it scales exponentially with the number of images leading to out-of-memory errors beyond 100 images. To minimize compute time, while staying in our resource boundaries, we choose to split each video

[1] https://github.com/LDenninger/RealEstate10K-Captions

in batches of 100 images with an overlap of 4 images. The chunks are then aligned to the reference of the initial batch.

Since VGGT outputs geometry up to an unknown global scale, as seen in the large difference in displacement in Table 4.1, we estimate a metric scaling factor m by comparing translation magnitudes of the estimated and ground-truth trajectories. Let $P = ([\mathbf{R}_i|\mathbf{t}_i])_{i=1}^{T}$, with $\mathbf{R}_i \in SO(3)$ and $\mathbf{t}_i \in \mathbb{R}^3$, be a camera trajectory. Then,

$$d(P) = \sum_{i=1}^{T-1} \left\| \mathbf{t}_{i+1} - \mathbf{t}_i \right\|_2, \tag{4.1}$$

defines the arc length of said camera trajectory. The metric scale can be recovered as:

$$m = \frac{d(P_{gt})}{d(P_{vggt})}, \tag{4.2}$$

where P_{gt} describes the ground-truth camera trajectory. Finally, the scale-corrected poses and depths can be obtained as $P_{vggt} = ([\hat{\mathbf{R}}_i|m \cdot \hat{\mathbf{t}}_i])_{i=1}^{T}$ and $\mathbf{d}^c_{vggt} = m \cdot \mathbf{d}_{vggt}$, respecitvely.

Since we do not want to lose the accuracy of the produced depth data, we decided against compressing it with common video or image compression pipelines. Video compression algorithms such as H.264 make use of visual redundancies and especially remove high-frequency details blurring out edges and fine-grained details in depth maps. Storing such large-scale data in raw float16 format, would require about 22.2 Tb of disk space. Instead, we have implemented our custom depth compression

Method	Accuracy	Disk space	MSE	Loading time (ms)	Saving time (ms)
Raw	FLOAT16	$\sim$ 22.2Tb	0.0	21	54
H.265	INT8	$\sim$ 214Gb	$9.3 \cdot 10^{-4}$	37	124
TRVL	INT16	$\sim$ 690Gb	$7.57 \cdot 10^{-6}$	161	267

Table 4.2: **Depth compression statistics.** We compare memory footprint, round-trip reconstruction error (MSE), and I/O times for our depth compression pipeline versus raw storage and H.265. Our method yields a $\sim 20\times$ reduction in disk usage with near-lossless accuracy.

library FDCOMP[2], which implements the depth compression algorithm TRVL [66] and provides it with fast python bindings. This allows us to store the depth maps and confidence maps near lossless with a compression rate of $\sim 20\times$. Additionally, the reduced file size and fast C++ bindings allow quick loading on saving of such

[2]https://github.com/LDenninger/fast-depth-compression

files during training. The detailed statistics of the depth compression are shown in Table 4.2.

Our geometry pipeline was run on 8× NVIDIA A100 GPUs in parallel with an average time of 13.4 s per video, amounting to a total of 256.86 GPU-hours. Due to the large size, depth and confidence maps are not made public but can be provided upon request.

4.3 Verification

Finally, we verify the newly produced ground-truth annotations against the existing ground-truth annotations.

Algorithm 1 CLIP Embedding Similarity

Require: Our CLIP embeddings F_1, original CLIP embeddings F_2 ▷ shape: $N \times D$
Ensure: Similarity logits
$F_1 \leftarrow \text{L2\_NORMALIZE}(F_1)$
$F_2 \leftarrow \text{L2\_NORMALIZE}(F_2)$
logits $\leftarrow \mathit{logit\_scale} \cdot (F_1\, F_2^\top)$ ▷ Cosine similarity
similarity $\leftarrow \text{SOFTMAX}(\mathit{logits}, \text{dim} = -1)$ ▷ Normalization
similarity $\leftarrow \text{DIAGONALIZE}(\mathit{logits})$

Captions. The baseline method CamI2V [18] employs the captions provided by the authors of CameraCTRL [17] produced with LAVIS [67]. To assess the quality of our newly created captions we compare our captions against the original ones in two evaluation setups, shown in Table 4.3.

Similarity	Video	Original	Ours
Video	1.0		
Original	0.69	1.0	
Ours	0.83	0.76	1.0

Table 4.3: **Caption comparison.** We compare the CLIP embeddings of our captions against the original captions of CameraCTRL and the embeddings of the videos themeselves. To compute the similarity score we compare each caption against 10 random candidate captions.

Specifically, we evaluate the similarity of the CLIP embeddings using the image and text encoder of the OpenCLIP [68] ViT-L-14 model trained on LAION-2B. First, we

compare our embedded captions directly with the original ones using the similarity score described in Algorithm 1. Each similarity score is computed over a set of 20 randomly chosen captions. The results in Table 4.3 show that our produced captions are close to the original ones with a similarity score of 0.76.

Next, we embed the images of each video using the CLIP image encoder. A prior study has shown that the initial image embeddings of each video are closest to the caption embeddings. Thus, we compare the embedding of the first image of each video against the captions. The first column of Algorithm 1 show that our generated captions with a score of 0.83 are more descriptive of the videos than the original ones.

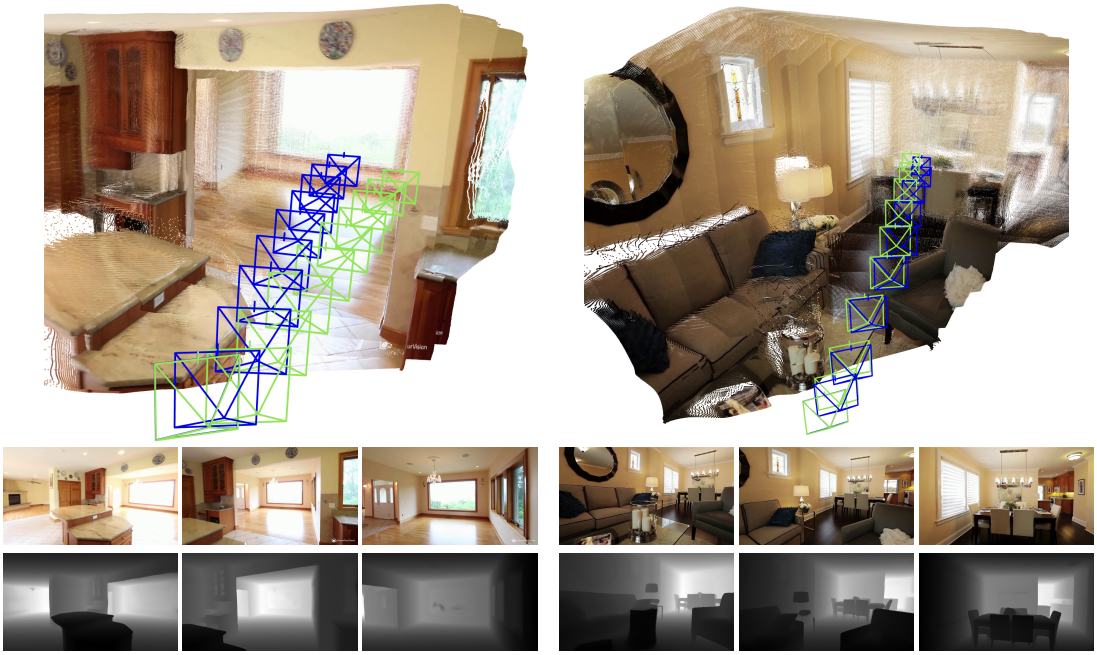


Figure 4.2: **VGGT camera trajectories.** We visualize the 3D reconstruction produced from our geometry annotation pipeline using VGGT. The ground-truth poses are visualized in green and the predicted poses in blue. The reconstruction and translation is corrected through the manually estimated metric scale.

Camera parameters. To assess the camera parameters produced by VGGT, we separately evaluate the camera extrinsics and intrinsics. We measure the Relative Rotation Accuracy (RRA) and Relative Translation Accuracy (RTA) at angular thresholds of 5, 15 and 30 degrees. Given predicted relative rotation $\hat{\mathbf{R}}_{ij}$ and translations $\hat{\mathbf{t}}_{ij}$ between views i and j and the corresponding ground-truth annotations $\mathbf{R}_{ij}$ and $\mathbf{t}_{ij}$, we can compute the angular difference as:

$$\varepsilon_R(i,j) = \arccos\left(\frac{\mathrm{tr}\big(\mathbf{R}_{ij}^\top \hat{\mathbf{R}}_{ij}\big) - 1}{2}\right), \quad \varepsilon_t(i,j) = \arccos\left(\frac{\hat{\mathbf{t}}_{ij}^\top \mathbf{t}_{ij}}{\|\hat{\mathbf{t}}_{ij}\| \, \|\mathbf{t}_{ij}\|}\right). \tag{4.3}$$

The RRA and RTA at a given threshold λ are then defined as:

$$\mathrm{RRA}(\lambda) = \frac{1}{|P|} \sum_{(i,j)\in P} \mathbf{1}\{\varepsilon_R(i,j) \le \lambda\}, \quad \mathrm{RTA}(\lambda) = \frac{1}{|P|} \sum_{(i,j)\in P} \mathbf{1}\{\varepsilon_t(i,j) \le \lambda\}. \tag{4.4}$$

We follow the evaluation protocol as proposed by DUSt3R [69], computing the metric between all pairwise combinations across the views of a video.

Further, we assess the global trajectory alignment with the Aligned Trajectory Error (ATE), defined as the root-mean-square Euclidean distance between the corresponding camera centers after aligning the estimated trajectory to the ground truth.

Let $(\hat{\mathbf{t}}_i)_{i=1}^T$ be the estimated camera centers and $(\mathbf{t}_i)_{i=1}^T$ the ground-truth centers. using Horn's closed-form method [70], we compute the rigid alignment $S^\star = (\mathbf{R}_S, \mathbf{t}_S) \in SE(3)$ as

$$S^\star = \underset{\mathbf{R}_S, \mathbf{t}_S}{\arg\min} \sum_{i=1}^{T} ||\mathbf{t_i} - (\mathbf{R}_S \hat{\mathbf{t}}_i + \mathbf{t}_S)||^2. \tag{4.5}$$

The ATE is then defined as:

$$\mathrm{ATE} = \sqrt{\frac{1}{T} \sum_{i=1}^{T} \big\| \mathbf{t}_i - (\mathbf{R}_S \hat{\mathbf{t}}_i + \mathbf{t}_S) \big\|^2}. \tag{4.6}$$

RRA			RTA			ATE
@5	@15	@30	@5	@15	@30	
95.55	99.32	99.68	36.17	69.30	85.00	0.48

Table 4.4: **Camera pose verification.** The produced camera annotations of VGGT are compared against the ground-truth camera annotations of RealEstate10K using the Relative Rotation Accuracy (RRA) and the Relative Translation Accuracy (RTA).

Across our evaluation in Table 4.4, the RRA remains high ($> 90\%$) at all thresholds, indicating that predicted camera orientations closely match the ground truth. In contrast, the RTA is significantly lower, especially at low thresholds, reflecting a well-known sensitivity of the translation component to projective ambiguities and global scale drift in reconstruction tasks from uncalibrated cameras. This discrepancy is

also visible in Figure 4.2, where the predicted and ground-truth trajectories exhibit a small but systematic displacement.

This gradual shift is captured by an ATE of 0.48, which primarily reflects translational discrepancies after alignment. To keep the employed camera poses aligned with the baseline during training, we choose to only leverage the estimated camera trajectories in conjunction with the depth maps.

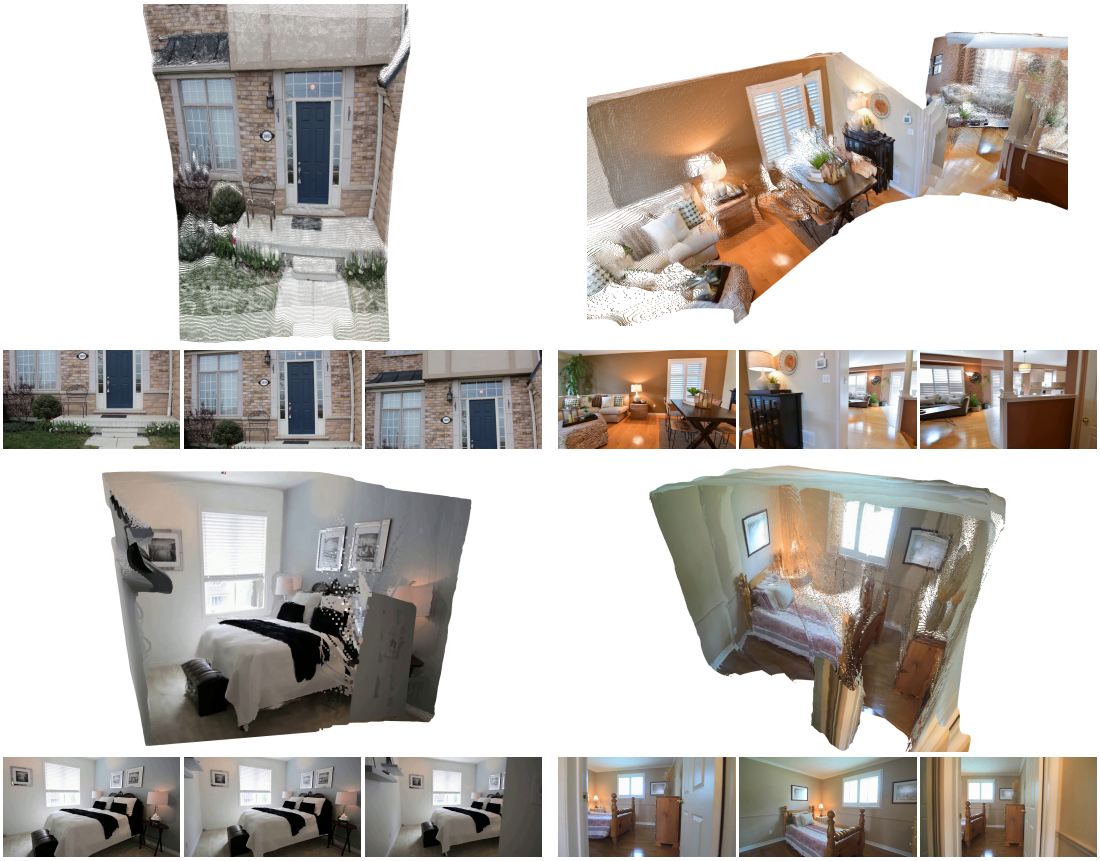


Figure 4.3: **VGGT reconstruction.** Examples of 3D point cloud reconstructions from uncalibrated, unknown-pose inputs. While global structures are captured accurately, frame-to-frame alignment can be imperfect due to the inherent projective ambiguity (up to a 15-DoF transform), which manifests as misaligned fine structures and object boundaries.

3D Geometry. In our last verification step, we aim to evaluate the reconstruction quality of our pipeline to produce high-quality point maps representing the 3D geometry of the scene. Since, the RealEstate10K natively does not provide any depth maps or 3D reconstruction of the scenes, we have to resort to a visual assessment of the qualitative results. Besides the examples provided in Figure 4.2, we add further

qualitative results in Figure 4.3. As the reconstruction from uncalibrated cameras at unknown poses suffers from an ambiguity up to a 15-DoF projective transformation, which is not explicitely considered in VGGT, the reconstruction quality suffers from misalignment between frames. The qualitative results show that fine structures as well as object edges are poorly aligned in some cases.

5 Methodology

This section details our proposed methods to inject additional context into a camera-controlled I2V diffusion model. In Section 5.1, we first formalize the task of context-to-video generation and introduce the notation used throughout this thesis. We introduce the baseline DynamiCrafter [19] and the camera control mechanism in Section 5.2. Finally, in Section 5.3, we introduce our *Implicit Model*, which learns a direct mapping from the context to timestep-wise latent features employed as conditioning in the diffusion model, and in Section 5.4 the *Explicit Model* employing additional 3D geometric information to build a scene representation which is queried to render trajectory aligned conditioning images.

5.1 Problem Statement

To formalize our problem, we start with introducing the notation used throughout our method description. I2V diffusion models, specifically DynamiCrafter [19], are provided with a reference image c_{ref} to ground the generation in and an optional caption c_{txt} to guide the generation semantically and induce motion dynamics. In addition, camera-controlled models, i.e., CamI2V [18], are provided with a camera trajectory defined as a sequence of camera poses $\mathcal{P}_{\text{cam}} = [P^1_{\text{cam}}, \dots, P^T_{\text{cam}}]$, where $P^\tau_{\text{cam}} \in SE(3)$ describes a relative rigid body transformation to the reference frame, and intrinsic matrices $\mathcal{K}_{\text{cam}} = [K^1_{cam}, \dots, K^T_{cam}]$. For clearity, t refers to the diffusion timestep and τ describes the timestep within the video sequence, i.e., the index of the frame.

The model is then tasked to generate a video $V \in \mathbb{R}^{T \times H \times W \times 3}$ of T frames, each of size $H \times W$, that is consistent with the provided camera trajectory and the provided visual context. Since the reference image c_{ref} only provides an insufficient amount of context, especially for videos depicting multiple distinct scenes, the video quality degrades significantly with increasing camera motion. To mitigate this issue, we propose to provide the model with additional context $\mathcal{C} = \{(c^i_{\text{ctx}}, d^i_{\text{ctx}}, P^i_{\text{ctx}}, K^i_{\text{ctx}})\}_{i=1}^N$ consisting of N RGB frames c^i_{ctx}, camera poses P^i_{ctx}, intrinsic matrices K^i_{ctx}, and optionally depth maps d^i_{ctx} used in our explicit approach.

5.2 Baseline Model

To set the stage, we first examine the baseline DynamiCrafter [19] with the injected camera control via CamI2V [18]. An overview of the architecture is shown in Figure 5.1.

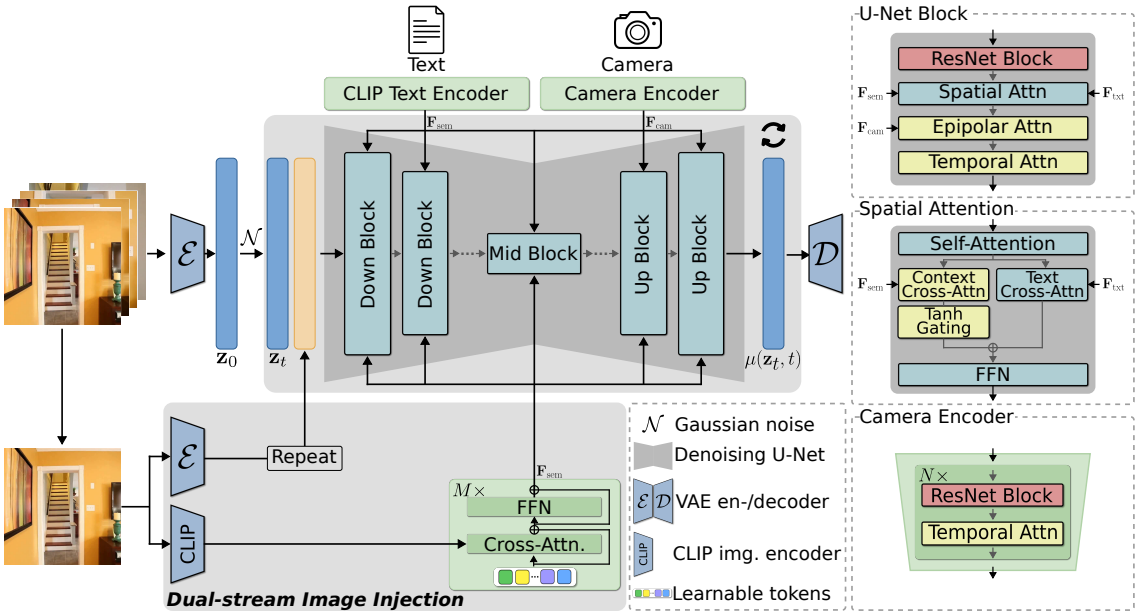


Figure 5.1: **Baseline architecture.** The model is conditioned on a reference image c_{ref}, an optional text prompt c_{txt} and a camera trajectory c_{cam} (Plücker-embedded). The *Dual-stream Image Injection* module provides fine-grained visual details and context $\mathbf{F}_{\text{sem}}$ from CLIP image/text tokens. Concatenated with the noisy latents z_t, the reference image is passed to the denoising U-Net, which consists of stacked downsampling, middle and upsampling blocks. Each block consists of a ResNet block, followed by spatial, epipolar and temporal attention blocks, where later two resemble closely the standard transformer architecture [58].

DynamiCrafter, shown in Figure 5.1 is an image-to-video (I2V) *Latent Video Diffusion Model* (LVDM) operating in the latent space of the frame-wise VAE of Stable Diffusion 2.1 [4]. It is conditioned on a reference image c_{ref} providing visual context and, optionally, a text prompt c_{txt}, primarily leveraged to induce temporal dynamics for animating the still reference. Building on VideoCrafter1 [29], it enhances fine-grained visual details through its *Dual-stream Image Injection* module consisting of a visual branch inducing fine-grained visual details and a semantic branch for high-level semantic information.

Visual branch. The visual branch encodes c_{ref} into the latent space $z_{\text{ref}} \in \mathbb{R}^{4\times1\times32\times32}$, repeats it along the time dimension (T frames) and concatenates it with the noisy latents $z_t \in \mathbb{R}^{4\times16\times32\times32}$ along the channel dimension, yielding the input $z_{\text{inp}} \in \mathbb{R}^{8\times T\times32\times32}$ to the denoising U-Net.

Semantic branch. The semantic branch is a query transformer with learnable query tokens $\mathbf{Q}_{\text{sem}} \in \mathbb{R}^{256\times1024}$, based on the Perceiver architecture [71]. The reference image c_{ref} and text prompt c_{txt} are encoded using OpenCLIP [68]. Using the visual tokens $\mathbf{F}_{\text{img}} \in \mathbb{R}^{256\times1024}$ from the last ViT block and the CLIP text tokens $\mathbf{F}_{\text{txt}}$, the query tokens retrieve a multi-modal semantic feature representation $\mathbf{F}_{\text{sem}}$ through cascaded cross-attention and feed-forward layers. This is finally injected into each block of the denoising U-Net.

Denoising U-Net. The U-Net comprises stacked downsampling blocks, a middle block, and a symmetric set of upsampling blocks, as depicted in Figure 5.1. The blocks follow an identical architecture except of initial down-/upsampling operations in the down- and upsampling blocks. Each block contains a ResNet block that processes features frame-wise with 2D convolutions, followed by a frame-wise spatial attention block conditioned on $\mathbf{F}_{\text{sem}}$ and, optionally, $\mathbf{F}_{\text{txt}}$.

The subsequent epipolar and temporal attention blocks adopt the standard transformer architecture following Vaswani *et al.* [58]. Employing an epipolar mask that masks out tokens beyond the epipolar line, as described in Section 3.4 and further detailed in Section 5.3.2, the epipolar attention block constraints the 3D geometry. Additionally, through a cross-attention layer the camera features $\mathbf{F}_{\text{cam}}$ from the camera encoder are aggregated into the U-Net.

Finally, the temporal attention blocks distribute the previously frame-wise aggregated features across timesteps by reshaping the features $h \in \mathbb{R}^{T\times h\times w\times d}$ as:

$$h' = \text{reshape}\big(h,\ \texttt{"T h w D -> (h w) T D"}\big) \tag{5.1}$$

and attending over the T tokens.

Camera pose conditioning. The camera encoder is queried with the Plücker embedding of the provided camera poses $c_{\text{cam}} \in \mathbb{R}^{6\times T\times h\times w}$, as detailed in Section 3.4. Its architecture mirrors the U-Net encoder without the spatial and epipolar attention blocks. Using the intermediate features after each block, the multi-resolutional features are injected into the epipolar attention blocks at given resolutions.

5.3 Implicit Model

The proposed context-to-video task and context conditioning of the Implicit Model has been accepted to the 13th International Conference on 3D Vision [72].

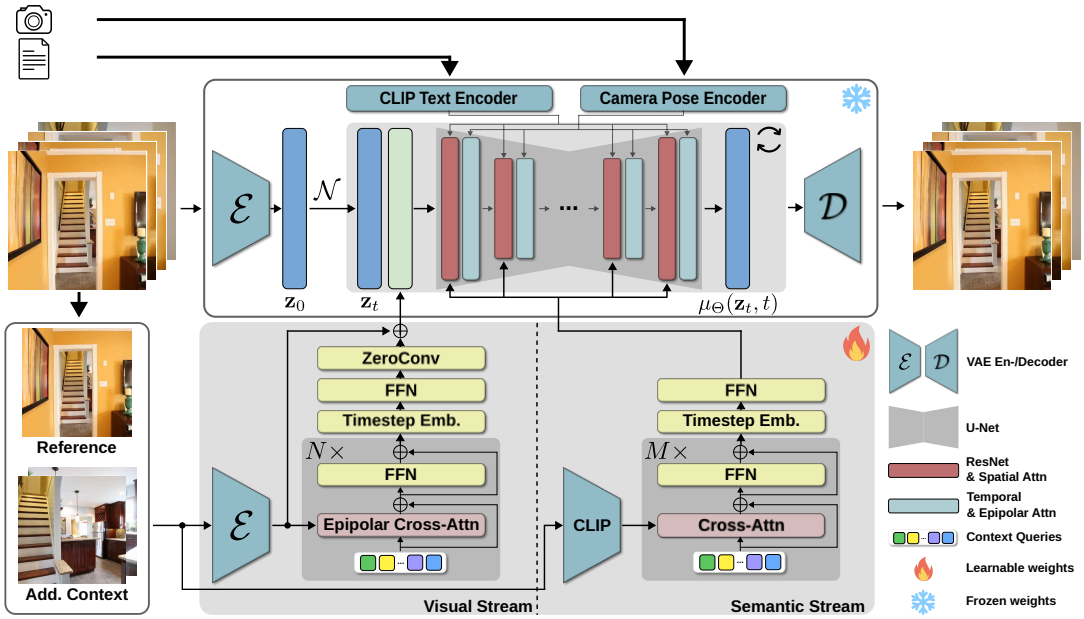


Figure 5.2: **Implicit model architecture.** Our pipeline generates videos conditioned on a reference image, an optional text description and a camera trajectory encoded through a camera pose encoder conditioning. Additionally, the *Implicit Context Encoder* processes frames in two parallel streams, one providing pixel-level visual cues and the other a global context. The pixel-level stream employs epipolar attention to enforce 3D consistent feature aggregation. Finally, both stream are augmented with a timestep embedding to ensure timestep-wise conditioning of the diffusion process.

To build a complementary conditioning mechanism with minimal re-training, our *Implicit Model* employs a 3D-aware *Implicit Context Encoder* which replaces the *Dual-stream Image Injection* module to not only condition the model on a single reference image c_{ref} but additionally on a set of context images c^i_{ctx}. An overview over our architecture is given Figure 5.2. The *Implicit Context Encoder* consists of a semantic stream, detailed in Section 5.3.1, which generates a high-level feature representation of the context $\mathcal{C}$ and a visual stream, described in Section 5.3.2, embedding the context into latent space of the diffusion model.

5.3.1 Semantic Stream

DynamiCrafter's query transformer $\mathcal{E}_{\text{sem}}$ integrates cross-modal information from the CLIP-embedded reference image $\mathbf{F}_{\text{img}}$ and the text condition $\mathbf{F}_{\text{txt}}$. It defines a set of $N = 256$ learnable latent query tokens $\mathbf{Q}_{\text{sem}} \in \mathbb{R}^{256\times d}$ which iteratively retrieve information from $\mathbf{F}_{\text{img}}$ and $\mathbf{F}_{\text{txt}}$. The final feature representation is then split into a timestep-wise embedding $\mathbf{F}_{\text{sem}} \in \mathbb{R}^{16\times 16\times d}$, modeling the semantic features for each timestep with 16 tokens. It employs no initial positional encoding that encodes the timesteps into the feature aggregation of the query transformer.

Through an extensive pre-training of the query transformer on a large corpus of data, it learned to effectively combine multi-modal features into a shared feature space usable by the diffusion model. Thus, we decide to adopt their model and the pre-trained weights to retain its semantic understanding. Concatenating the additional CLIP-embedded context tokens $\mathbf{F}_{\text{ctx}}$ with the reference image and text embedding $\mathbf{F}_{\text{img}}$ and $\mathbf{F}_{\text{txt}}$, the final semantic feature representation $\mathbf{F}_{\text{sem}}$ is defined as:

$$\mathbf{F}_{\text{sem}} = \mathcal{E}_{\text{sem}}([\mathbf{F}_{\text{img}}, \mathbf{F}_{\text{txt}}, \mathbf{F}_{\text{ctx}}], \mathbf{Q}_{\text{sem}}). \tag{5.2}$$

The semantic stream encoder is then further fine-tuned to effectively aggregate information from multiple image conditions.

5.3.2 Visual Stream

While the semantic stream provides a well-suited global context representation, it lacks fine-grained visual details due to CLIP's inherent training on visual-language alignment, which favors high-level representations of single entities over a pixel-accurate visual representation.

To enhance context-aware generation, we integrate our visual condition directly into DynamiCrafter's image conditioning. Specifically, we intend to replace the concatenated reference image latents z_{ref} with a combined timestep-wise feature embedding z_{cond} retrieved from the context $\mathcal{C}$. To do so, our visual stream implicitely realizes the context-to-latent mapping:

$$f_\theta : \mathcal{C} \to z_{\text{cond}} \in \mathbb{R}^{C\times T\times h\times w}. \tag{5.3}$$

Specifically, we first embed the context frames $c^0_{\text{ctx}}, \ldots, c^N_{\text{ctx}}$ into the latent space $\mathbf{Z}_{\text{ctx}} = [z^0_{\text{ctx}}, \ldots, z^N_{\text{ctx}}]$. We then introduce a set of pixel- and timestep-wise learnable context tokens $\mathbf{Q}_{\text{vis}} \in \mathbb{R}^{T\times h\times w\times D}$ which are then used to retrieve the information for each

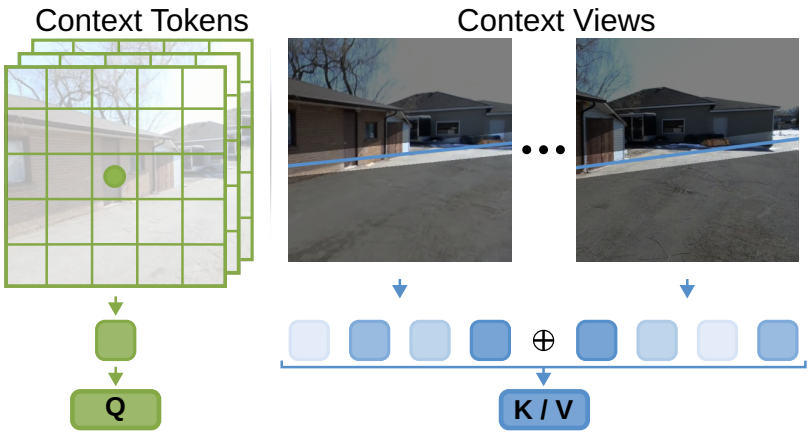


Figure 5.3: **Epipolar cross-attention.** Learnable context tokens act as queries to retrieve pixel-level features for each timestep from context views, masked according to epipolar lines to incorporate 3D geometric constraints.

pixel separately from the embedded context $\mathbf{Z}_{\text{ctx}}$, similar to the semantic stream.

3D awareness. Since the feature mapping into our conditioning latents z_{cond} is completely implicitely defined, we further introduce explicit geometry-grounded constraints to augment our encoder with 3D awareness and guide the overall feature aggregation. Specifically, we implement an epipolar cross-attention mechanism which employs the epipolar constraint described in Section 3.4 to restrict the feature aggregation to only consider potentially relevant features. Each token $q_i \in \mathbf{Q}_{\text{vis}}$, illustrated in Figure 5.3, describes a pixel (u, v) at timestep t. Employing the provided camera pose P^{τ}_{cam} at the given timestep, we can compute the epipolar line $l_{ij} = Ax + Bx + C$ in each context view c^{j}_{ctx}. Using the point-to-line distance:

$$d(u', v') = \frac{[A, B, C]^{\intercal} \cdot [u', v', 1]}{\sqrt{A^2 + B^2}}, \tag{5.4}$$

we produce the epipolar mask $\mathbf{m} \in \mathbb{R}^{Thw \times Nhw}$ masking out pixels (u', v') with a distance larger than a threshold δ, set to half of the diagonal of the latent feature space. Employing this mask in the cross-attention mechanism builds our epipolar attention mechanism:

$$\text{EpiCrossAttn}(q, k, v, \mathbf{m}) = \text{SoftMax}\left(\frac{qk^{\intercal}}{\sqrt{d}} \odot \mathbf{m}\right) v, \tag{5.5}$$

where $q \in \mathbb{R}^{Thw \times D}$ describes the learnable context queries, $k, v \in \mathbb{R}^{Nhw \times D}$ the latent embedded context frames and $\odot$ the Hadamard product.

Temporal awareness. The native pixel-level embedding of DynamiCrafter is agnostic to the timestep within the video as each timestep is provided with the same condition. Thus, to further enforce the diffusion model to attend to context provided at specific timesteps, we found it advantageous to employ a learnable sinusoidal timestep embedding ϕ:

$$\phi_{2i}(\tau) = \sin(\tau \cdot 10000^{-\frac{2i}{d}}), \quad \phi_{2i+1}(\tau) = \cos(\tau \cdot 10000^{-\frac{2i}{d}}), \tag{5.6}$$

where τ describes the specific timestep and i the dimension. In practice, we set the timestep embedding's dimension d to 32. The timestep embedding is then further encoded through a shallow two-layered MLP mapping into a four-dimensional embedding that is added to the output of the query transformer.

Finally, the visual stream of our *Implicit Context Encoder* maps a spatially distributed embedding represented through the latent embedding of posed views to a timestep-wise embedding:

$$\mathbf{F}_{vis} = \mathcal{E}_{vis}(Z_{\text{ctx}}, \mathbf{Q}_{\text{vis}}, \mathbf{m}). \tag{5.7}$$

To retain the reference image as a strong anchor to the generation and smoothly insert the new condition, we employ a 3D zero-convolution which weighs the usage of DynamiCrafter's native condition z_{ref} and ours $\mathbf{F}_{vis}$ before adding them together.

5.3.3 Timestep-weighted Loss

To better leverage the scene context, we reweight the standard noise-prediction loss. Let $\varepsilon_{\theta,k}(\mathbf{x_t}, \mathbf{c}, t)$ be the predicted noise at frame index k and ε_k the ground-truth noise. We apply a logarithmic weighting emphasizing the loss contribution of later frames which depend more on the added context:

$$\mathcal{L} = \frac{\sum_{k=0}^{15} \log_{10}(k+1) \cdot ||\varepsilon_k - \varepsilon_{\theta,k}(\mathbf{x_t}, \mathbf{c}, t)||_2^2}{\sum_{k=0}^{15} \log_{10}(k+1)}. \tag{5.8}$$

In practice, this weighted loss function stabilizes the training process in later stages, preventing divergence and improving video quality.

5.4 Explicit Model

In contrast to the *Implicit Model*, which implicitely learns the mapping f_θ of the context $\mathcal{C}$ to the conditioning latents z_{cond} through 3D-aware attention layers, our

Explicit Model performs this mapping explicitely in 3D world space by representing the context in a 3D scene representation which is then queried to render the condition frames $[c^1_{\text{cond}}, \ldots, c^T_{\text{cond}}]$ aligned with the camera trajectory $\mathcal{P}_{\text{cam}}$. In particular, we employ anistropic 3D Gaussians, as proposed by Kerbl *et al.* [53], to represent the scene context and employ their Gaussian splatting approach to render the condition frames. These condition images are then embedded into the latent space, producing the conditioning latents $\mathbf{z}_{\text{cond}}$ which are further encoded through a *Explicit Context Encoder* injecting the conditioning into the skip-connections of the denoising U-Net, similar to ControlNet's conditioning mechanism [39]. During training, the same reweighted loss function as in the *Implicit Model* is employed. An overview of our approach is depicted in Figure 5.4. In the following, we first explain the inner-working of the scene representation, we coin the *3D Cache*, managing the context in Section 5.4.1 and then discuss how the *Explicit Context Encoder* injects the produced condition images into DynamiCrafter in Section 5.4.2.

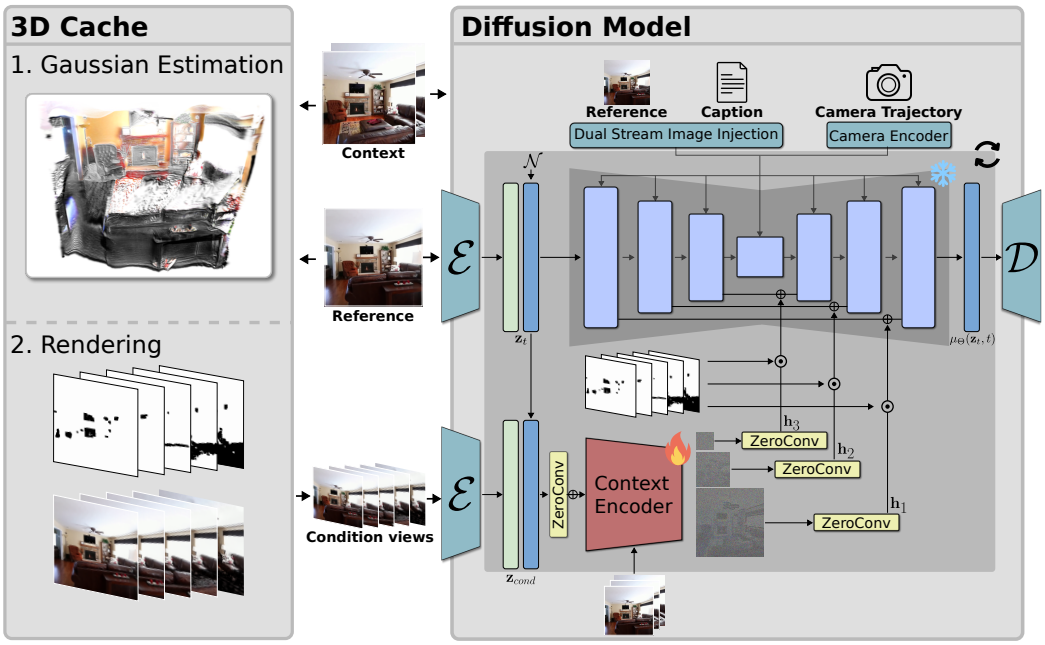


Figure 5.4: **Explicit model overview.** Our *Explicit Model* employs the *3D Cache* to produce the trajectory aligned condition frames $[c^1_{\text{cond}}, \ldots, c^T_{\text{cond}}]$ and rendering masks $[m^1_{\text{cond}}, \ldots, m^T_{\text{cond}}]$. The *Explicit Context Encoder* runs parallel to the denoising U-Net for every diffusion timestep t and encodes the latent condition frames $[z^1_{\text{cond}}, \ldots, z^T_{\text{cond}}]$ into the hierarchical features h_l. Masking out unobserved regions employing m^τ_{cond}, the hierarchical features are injected into the skip connections of the denoising U-Net to generate the final video.

5.4.1 Context Representation

To build an explicit context representation we propose the *3D Cache* which keeps track of context provided through the additional context images c^i_{ctx}. We choose to represent the context through a set of anistropic 3D Gaussians $\mathcal{G} = \{g_i\}_{i=1}^N$ [53]. Each Gaussian g_i is defined through its mean μ_i, its covariance Σ_i, an opacity value σ_i, and a view-dependent color modelled through 3-degree spherical harmonics defined through its coefficients $\beta_i \in \mathbb{R}^3$. These Gaussians are then predicted in a fully feed-forward manner using MVSplat [54].

Our *3D Cache* builds on top of this representation, implementing an *update* function which first estimates the 3D Gaussians representing the scene and a *rendering* function which renders a video along a defined camera trajectory using Gaussian splatting [53], producing our conditioning images $[c^1_{\text{cond}}, \ldots, c^T_{\text{cond}}]$, as well as rendering masks $[m^1_{\text{cond}}, \ldots, m^T_{\text{cond}}]$ masking out unobserved scene regions.

Updating the 3D Cache. To update the *3D Cache*, MVSplat is queried with the context images c^i_{ctx}, as well as corresponding extrinsics P^i_{ctx} and the intrinsics K^i_{ctx}. In a feed-forward manner, it produces a set of 3D Gaussians $\mathcal{G} = \{g_i\}_{i=1}^N$. Contrary to previous work [53], which relies on an initialization step producing a sparse or dense reconstruction, MVSplat proposes an internal depth estimator that is directly integrated into the Gaussian estimation process, making additional geometry annotations superfluous.

Rendering the 3D Cache. In the final step, given the camera trajectory of the video being generated $\mathcal{P}_{\text{cam}}$ and the corresponding intrinsic matrices $\mathcal{K}_{\text{cam}}$, we employ the Gaussian splatting approach proposed by Kerbl *et al.* [53].

First each 3D Gaussian g_i with mean $\mu_i \in \mathbb{R}^3$ and covariance $\Sigma \in \mathbb{R}^{3\times3}$ is splatted onto the 2D image plane through an affine mapping to obtain a 2D Gaussian on the image plane. Specifically, the camera projection $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ is first linearized around μ_i with:

$$\pi(x) \approx \pi(\mu_i) + \mathbf{J}(x - \mu_i), \tag{5.9}$$

where $\mathbf{J} \equiv \frac{\partial \pi}{\partial x}\big|_{x=\mu_i}$ is the Jacobian of the affine approximation of π. The covariance matrix in the image plane Σ'_i is then obtained by first transforming it into the camera coordinates:

$$\Sigma'_i = \mathbf{J}T\Sigma_i T^\intercal \mathbf{J}^\intercal, \tag{5.10}$$

with T describing the world-to-camera transform for the view to be rendered, followed

by dropping the depth dimension, i.e., the third row and column, of Σ_i' creating a 2×2 variance matrix in image space.

Finally, these primitives are passed to a rasterizer sub-dividng the image plane into 16×16 tiles, which the splatted Gaussians are binned in using $\mu_i^{'} = \pi(\mu_i)$ and Σ_i'. Within each tile, the Gaussians are sorted front-to-back using the projected depth to the Gaussians' centers. This allows, a simple alpha compositing approach to render the color $C(\mathbf{p})$ for each pixel $\mathbf{p}$ by iterating through the sorted splats:

$$C(\mathbf{p}) = \sum_{k \in sorted} T_k \alpha_k c_k, \tag{5.11}$$

where $\alpha_k = \sigma_k \mathbf{G}_k(\mathbf{p})$ describes pixel-specific contribution of the $k-$th Gaussian and T_k the accumulated transmittance across the Gaussians. It is computed recursively in the alpha compositing process by:

$$\begin{aligned} T_1 &= 1, \\ T_{k+1} &= T_k(1-\alpha_k)\,. \end{aligned} \tag{5.12}$$

Obtaining rendering masks. Additionally, we require rendering masks m_{cond}^{τ} that represent the pixel-wise confidence to mask out unseen or poorly reconstructed regions. In theory, such mask can be directly recovered from the accumulated transmittance T_k but unfortunately those are kept within the CUDA-kernels and can not be obtained without re-implementation of those. Instead, we adjust the spherical harmonics of all Gaussians to a white color in all viewing directions by setting the base coefficient to $\mathbf{c} = [1, 1, 1]$ and all higher-degree coefficients to zero. It is evident from Equation (5.11), that this resolves to a rendered color $C(\mathbf{p})$ representing the total transmittance reaching each pixel:

$$C(\mathbf{p}) = \sum_{k \in \text{sorted}} T_k \alpha_k = 1 - T_K\,, \tag{5.13}$$

where K indexes the last Gaussian along the ray.

Thresholding with $\lambda = 0.8$, then provides us with the masks m_{cond}^{τ} for each rendered frame. In a further post-processing step, we then apply a dilation followed by an erosion operation, to close small holes in the mask. Ultimately, our *3D Cache* provides us with a set of condition frames $[c_{\text{cond}}^1, \ldots, c_{\text{cond}}^T]$ and corresponding masks $[m_{\text{cond}}^1, \ldots, m_{\text{cond}}^T]$ along the defined camera trajectory, which are then further encoded and inserted into the diffusion model.

5.4.2 Context Encoding

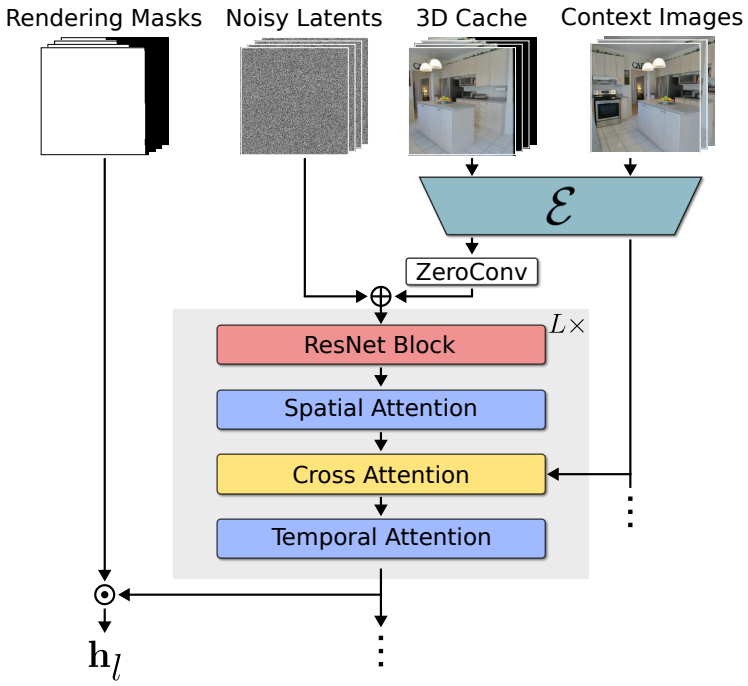


Figure 5.5: **Explicit context encoder architecture.** Adopting the architecture of the denoising U-Net, the context encoder consists of stacked encoder blocks with injected cross-attention layers that aggregate features from the context images. The output of each block l is then masked by the rendering masks to produce the hierarchical feature maps h_l.

Our *Explicit Context Encoder*, illustrated in Figure 5.5, embeds the rendered condition images c^{τ}_{cond} into hierarchical feature representations h_l at different resolutions l, which are then inserted into the skip-connections of the denoising U-Net, as shown in Figure 5.4. Adopting ControlNet's conditioning mechanism, the *Explicit Context Encoder* runs in parallel to the denoising U-Net for each diffusion step t. After embedding the condition images into the latent space, the conditioning latents z_{ctx} are concatenated with the noisy latents z_t, similar to DynamiCrafter's native input structure.

In order to retain the strong video understanding of DynamiCrafter, the *Explicit Context Encoder* resembles the U-Net's encoder architecture, where each block l is composed of a ResNet block for spatial reasoning and separated spatial and temporal attention layers to effectively model the spatio-temporal relations across the video. These blocks are then cascaded into three downsampling stages employing each with an initial downsampling operation, composing the encoder's architecture as shown in Figure 5.5. The hierarchical features h'_l are obtained after each block at downsampling

rates $1\times, 2\times$ and $4\times$ of the latent space's dimensionality of 32×32.

Context grounding. Since the *3D Cache* naturally produces its own artifacts, like color interpolation or blank spaces in unseen regions, the visual quality of the renderings are impaired and the fine-grained visual details of the context images are not perfectly propagated. Thus, we add two mechanism to better ground the encoding process in the context images and reject artifacts for an improved context conditioning.

First, to allow our encoder to aggregate features from the original context views, we inject additional transformer blocks between the spatial and temporal transformer blocks, which employ the cross-attention mechanism to attend into the latent-embedded context views z_{ctx}. This transformer block closely resembles the original spatial transformer block. This mechanism primarily injects fine-grained visual features and improves the encoder's ability to recover from poorly reconstructed scene regions. Secondly, we leverage the rendering masks $\mathbf{m}_{\text{cond}} \in \mathbb{R}^{T\times 1\times h\times w}$ to mask out unseen scene regions in the context views. Given the hierarchical features obtained after each encoding block $h'_l \in \mathbb{R}^{T\times d\times h\times w}$, the final feature representation is given as:

$$h_l = h'_l \odot \mathbf{m}_{\text{cond}}\,. \tag{5.14}$$

We apply masking only in the first two encoder stages, where features are still local. At larger downsampling rates, spatial masking may discard crucial context information.

Condition injection. Building on DynamiCrafter's strong video understanding, we initialize the *Explicit Context Encoder* from DynamiCrafter's pretrained encoder. Only the newly added cross-attention blocks are trained from scratch. The encoder input is the concatenation of the latent condition frames z^j_{cond} and the noisy latents z_t. To prevent drastic input changes and mitigate the *forgetting* issue discussed in Section 2.3, we fuse this input with DynamiCrafter's native input using a zero convolution operation (ZeroConv)

The produced hierarchical context features h_l are then injected through the encoder-decoder skip connections of the denoising U-Net. For each level l with original skip features h_l^{skip}, we gradually mix in the new context condition with:

$$h_l^{\text{skip}'} = h_l^{\text{skip}} + \text{ZeroConv}(h_l). \tag{5.15}$$

yielding context-conditioned skip features $h_l^{\text{skip}'}$ that are added to the corresponding decoder inputs.

6 Evaluation

We evaluate the *Implicit* and *Explicit Model* across diverse settings. In Section 6.1 we describe the evaluation protocol, the RealEstate10K splits, and provide further implementation details. Section 6.2 presents the evaluation metrics used the evaluate the video quality and adherence to the camera trajectory. Section 6.3 compares the two models with strong baselines, assessing both video quality and trajectory accuracy, and examines prominent failure cases of our models. Following, Section 6.4 quantifies the effect of the provided context under different context-sampling strategies. Because our task is closely related to novel view synthesis (NVS), Section 6.5 benchmarks against a state-of-the-art NVS method. In Section 6.6 we report the resource requirements of our method demonstrating its usability even on consumer-grade GPUs. Finally, Section 6.7 ablates key design choices underlining the effectiveness of our approaches.

6.1 Setup

In the following we define our evaluation protocol and model setup used throughout our experimental evaluation if not stated otherwise.

Dataset. The RealEstate10k [20] dataset available to us consists of $62,596$ training and $7,218$ test video clips. Since the base model DynamiCrafter [19] works on a resolution of 256×256 with a temporal length of 16 frames, we center-crop the videos and resize them to the given resolution. The video frames are sampled from a random starting point within the video and a random stride between 1 and 10. Following the baseline models, the stride is fixed to 8 during evaluation [17], [18]. Additionally, we sample 1 to 4 context frames that are used to define the scene context in our method. During training these frames are sampled from anywhere in the original video and might overlap with the video to be generated. For evaluation, we sample a custom split of $2,000$ videos which are longer than 20s to ensure sufficient context outside of the target video sequence. To prevent information leak and ensure a fair evaluation,

we only sample context frames from outside the video with an offset of at least 4 frames. For further details on the RealEstate10K dataset, please see Chapter 4.

Implementation details. Our models are implemented using the deep-learning framework PyTorch [73] and built on top of the baseline implementations of DynamiCrafter [19] and CamI2V [18]. Both methods employ the codebase of the Latent Video Diffusion Model (LVDM) framework proposed by He *et al.* [74]. Our training and evaluation pipelines are then implemented within the Lightning framework. We employ a FP16 mixed-precision training strategy through DeepSpeed ZeRo-1 [75] using the Adam [76] optimizer. The *Implicit Model* is trained for 50K iterations with a total batch size of 64. A training run takes approximately 7 days on 4 NVIDIA A100 GPUs. In contrast, our *Explicit Model* is only trained for 20K iterations with a smaller batch size of 32. Trained on 4 NVIDIA H100 GPUs, a single training run only takes about 1.5 days.

6.2 Evaluation Metrics

We compare our models on different metrics evaluating the video quality, the frame-wise image quality and the overall trajectory alignment of the generated video V_g and the target video V_t.

FVD. The Fréchet Video Distance (FVD) embeds the generated and target videos into a spatio-temporal feature space using an Inception3D [77] network $f(\cdot)$. Specifically, the mean and covariances for the target, μ_t and Σ_t, and generated videos, μ_g and Σ_g are computed for the features extract before the final SoftMax layer computing the class-logits:

$$\mu_t = \mathbb{E}_{v \in V_t}[f(v)], \quad \Sigma_t = \mathrm{Cov}_{v \in V_t}[f(v)], \tag{6.1}$$

$$\mu_g = \mathbb{E}_{v \in V_g}[f(v)], \quad \Sigma_g = \mathrm{Cov}_{v \in V_g}[f(v)]. \tag{6.2}$$

The FVD is then defined as the 2-Wasserstein distance between the gaussianized feature distributions:

$$\mathrm{FVD}(V_t, V_g) = \|\mu_t - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_t + \Sigma_g - 2\big(\Sigma_t^{1/2}\,\Sigma_g\,\Sigma_t^{1/2}\big)^{1/2}\Big). \tag{6.3}$$

Larger scores represent better video quality.

SSIM. The Structural Similarity Index (SSIM) [78] compares the color distribution around each pixel in an 11×11 window. Let $v_g \in V_g$ be a frame from the generated video and $v_t \in V_t$ the corresponding target frame. Using a normalized Gaussian kernel w with weights $\sum_i w_i = 1$, we can compute the local mean, variances and covariances at pixel p as:

$$\begin{aligned} \mu_{v_g}(p) &= \sum_{i \in \mathcal{N}(p)} w_i \, v_g^i, & \mu_{v_t}(p) &= \sum_{i \in \mathcal{N}(p)} w_i \, v_t^i, \\ \sigma^2_{v_g}(p) &= \sum_i w_i \big(v_g^i - \mu_{v_g}(p)\big)^2, & \sigma^2_{v_t}(p) &= \sum_i w_i \big(v_t^i - \mu_{v_t}(p)\big)^2, \\ & & \sigma_{v_g v_t}(p) &= \sum_i w_i \big(v_g^i - \mu_{v_g}(p)\big)\big(v_t^i - \mu_{v_t}(p)\big). \end{aligned} \tag{6.4}$$

The frame-wise SSIM can then be computed as:

$$\mathrm{SSIM}(v_g, v_t) = \frac{1}{|v_g|} \sum_{p \in v_g} \frac{(2\,\mu_{v_g}(p)\,\mu_{v_t}(p) + c_1)\,(2\,\sigma_{v_g v_t}(p) + c_2)}{\big(\mu^2_{v_g}(p) + \mu^2_{v_t}(p) + c_1\big)\,\big(\sigma^2_{v_g}(p) + \sigma_{v_t}(p)^2 + c_2\big)}, \tag{6.5}$$

where $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ and L describes the intensity range of the images. k_1 and k_2 are set to $k_1 = 0.01$ and $k_2 = 0.03$ by default. Higher values reflect a better performance.

LPIPS. The Learned Perceptual Image Patch Similarity (LPIPS) [79] metric compares the frame-wise feature embeddings of the generated frames ϕ_g^l and the target frames ϕ_t^l at different layers $l \in \mathcal{L}$ of a pre-trained CNN, i.e. AlexNet [80]. In practice the output of the initial five convolutional blocks is chosen. The LPIPS score then describes the weighted l_2 distance between the channel-wise normalized embeddings using weights w_l learned from human preferences:

$$\mathrm{LPIPS}(v_g, v_t) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\phi_g^l - \phi_t^l)||_2^2 \tag{6.6}$$

A lower value indicates a better result.

PSNR. The Peak Signal-to-Noise Ratio (PSNR) measures the fidelity of the generated frames. Let v_g and v_t be a generated and target frame, the PSNR can directly

be derived from the mean squared error (MSE):

$$\text{MSE}(v_g, v_t) = \frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} (v_g(i,j) - v_t(i,j))^2, \tag{6.7}$$

as:

$$\text{PSNR}(v_g, v_t) = 10 \log_{10}(\frac{L}{MSE(v_g, v_t)}), \tag{6.8}$$

with L describing the maximum intensity value. A higher value reflects a better performance.

Parameter	**Value**
ImageReader.single_camera	True
ImageReader.camera_model	SIMPLE_PINHOLE
ImageReader.camera_params	$\{f\}, \{c_x\}, \{c_y\}$
SiftExtraction.estimate_affine_shape	True
SiftExtraction.domain_size_pooling	True
SiftMatching.guided_matching	True
SiftMatching.max_num_matches	65,536
RelPoseEstimation.max_epipolar_error	4
BundleAdjustment.optimize_intrinsics	False

Table 6.1: **GLOMAP configuration.** We adopt custom parameters for the GLOMAP pipeline to achieve robust registration on low resolution images of size 256×256. Additionally, we inject the ground-truth camera intrinsics to improve the overall registration quality. All other parameters are kept to their default values.

Camera trajectory metrics. Finally, to examine the generated camera trajectory we follow the evaluation paradigm proposed by CameraCtrl and CamI2V. Using the structure-from-motion pipeline GLOMAP [81], we estimated the camera rotation $\hat{\mathbf{R}}_\tau$ and translation $\hat{\mathbf{t}}_\tau$ for each camera at timestep τ and compute the independent rotation and translation errors, *RotErr* and *TransErr*, as well as the combined element-wise error *CamMC*, compared against the ground-truth rotation $\mathbf{R}_\tau$ and translation

$\mathbf{t}_\tau$:

$$\text{RotErr} = \sum_{\tau=1}^{T} \cos^{-1}\left(\frac{\text{tr}\left(\hat{\mathbf{R}}_\tau \mathbf{R}_\tau^\intercal\right) - 1}{2}\right), \tag{6.9}$$

$$\text{TransErr} = \sum_{\tau=1}^{T} \left\|\hat{\mathbf{t}}_\tau - \mathbf{t}_\tau\right\|_2, \tag{6.10}$$

$$\text{CamMC} = \sum_{\tau=1}^{T} \left\|[\hat{\mathbf{R}}_\tau \mid \hat{\mathbf{t}}_\tau] - [\mathbf{R}_\tau \mid \mathbf{t}_\tau]\right\|_2. \tag{6.11}$$

To counteract the randomness of GLOMAP, we follow the scheme to average the metrics over successful runs out of 5 trials for each video.

6.3 Baseline Comparison

In the following, we compare our models against several camera-controllable video diffusion models. Specifically, we compare against the baseline diffusion model DynamiCrafter [19], and several fine-tuned models injecting additional camera control, such as MotionCtrl [13], CameraCtrl [17] and CamI2V [18]. We leverage the re-implementations provided by Zheng *et al.* [18] based on DynamiCrafter for comparison.

Method	FVD ↓	SSIM ↑	LPIPS ↓	PSNR ↑	MSE ↓
DynamiCrafter	321.523	0.534	0.373	17.866	0.0179
MotionCtrl	78.301	0.621	0.264	19.751	0.0119
CameraCtrl	71.223	0.640	0.243	20.311	0.0107
CamI2V	71.014	0.656	0.219	20.628	0.0103
Implicit Model	<u>53.907</u>	<u>0.695</u>	<u>0.190</u>	<u>21.196</u>	<u>0.0082</u>
Explicit Model	**50.498**	**0.721**	**0.172**	**22.559**	**0.0077**

Table 6.2: **Visual quality on RealEstate10K.** We compare our methods against state-of-the-art camera-controlled methods. Results were obtained using 25 DDIM steps with classifier-free guidance (CFG) scale set to 7.5, except for our method performing best with a CFG scale of 3.5.

Visual quality. To assess the produced visual quality, we compare the different methods based on the evaluation metrics described in Section 6.2. The quantitative

results of this study are shown in Table 6.2.

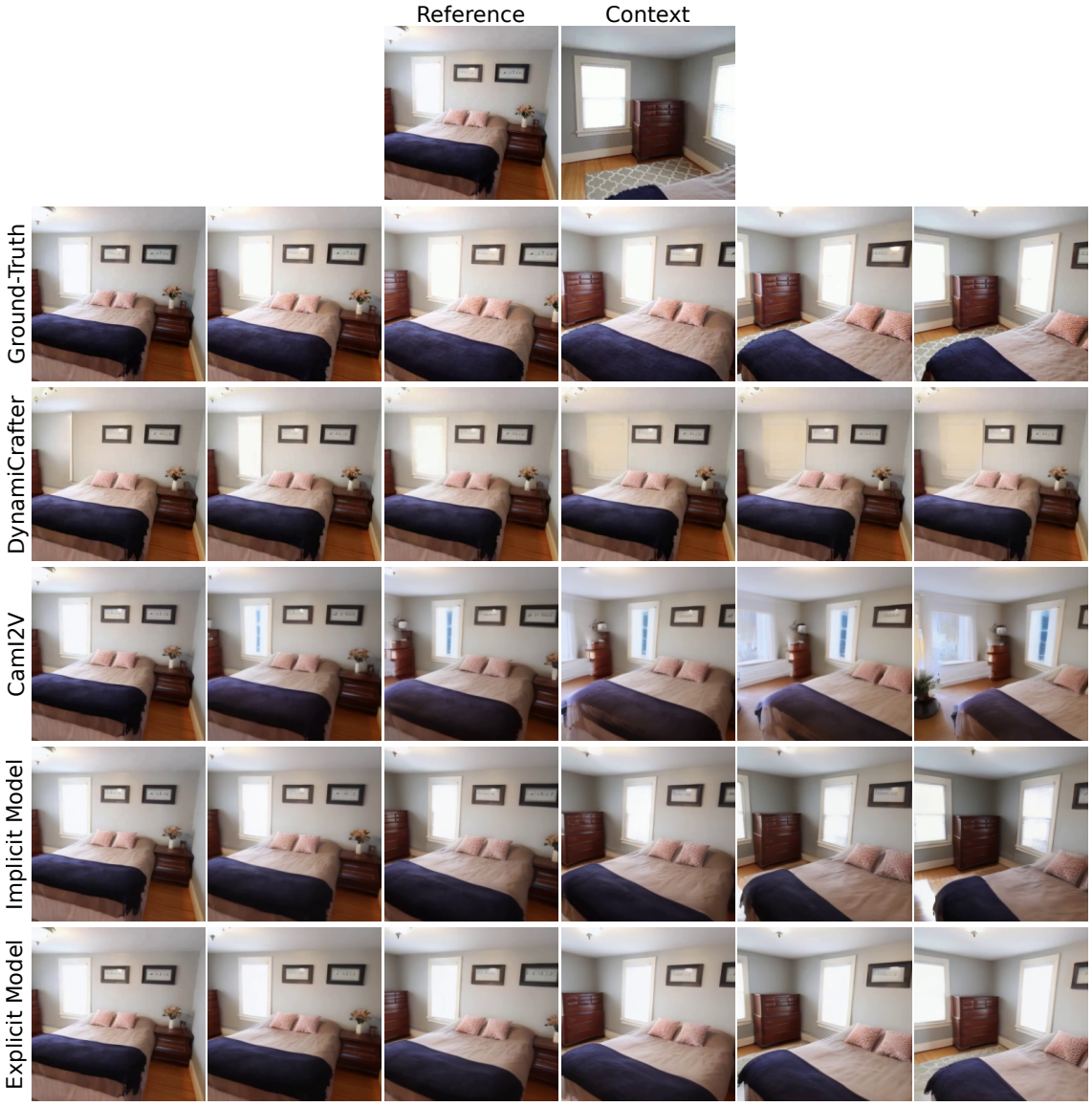


Figure 6.1: **Qualitative comparison.** Qualitative results of our models compared against DynamiCrafter and CamI2V. An additional context image after the sequence is added to provide scene context to the diffusion models after the camera pans.

DynamiCrafter fails to produce consistent videos on RealEstate10K, only achieving an FVD of 321.523. Looking at the example provided in Figure 6.1, it is evident that DynamiCrafter without injected camera control only generates a static scene. Furthermore, it is unable to correctly integrate the provided visual context of the reference image and places it randomly in the video sequence which is inherent from its training strategy placing the reference image at a random timestep. Another

reason for the poor performance of DynamiCrafter is a training artifact leaving a *shutterstock* watermark in some generated videos which is even visible in their own evaluation.

In contrast, the models with injected camera control, MotionCtrl, CameraCtrl and CamI2V, significantly improve on DynamiCrafter's results, as they are not only fine-tuned on the RealEstate10K but are also given additional information about the camera trajectory which is crucial for the RealEstate10K scenes which are heavily afflicted by camera motion. Since these models are only provided with a single reference frame which is placed at the beginning of the video sequence, the visual quality heavily degrades especially for larger camera movements. For instance, Figure 6.1 shows a scene where the camera pans to the left revealing a large scene portion unobserved in the reference frame. CamI2V is required to extrapolate beyond its provided context, a difficult task for models with limited world understanding, leading to a unrealistic scene generation and blurry visuals.

This missing context may be bridged by providing additional context images as done with our approach. Our *Implicit Model* and *Explicit Model* improve the visual quality of the generated videos across all employed metrics. The increased FVD indicates an improved temporal coherence of the scene which can be mainly attributed to the additional constraint implemented through the context. Especially in terms of the frame-wise metrics and the MSE our models achieve an improvement of $\sim 20\%$, indicating that the injected context allows more faithful generation adhering the scene constraints.

Our models in Figure 6.1 are provided with an additional context image that provides an additional view of the window and cupboard. After the camera pans, our models employ the additional context to correctly generate the partially seen cupboard and unseen window. While both models are able to faithfully represent the context in generated video sequence, our *Explicit Model* better represents the original color distribution which can be linked to the fact that, in contrast to the *Implicit Model* which injects the condition into the U-Net input, the former injects the condition directly into the decoding process of the U-Net placing a heavier constraint on the generation process. This shift in color is especially visible in the lower example of Figure 6.2.

Moreover, the carpet floor, clearly visible in the generated video sequence of the *Explicit Model*, fails to be generated by the *Implicit Model*. Similarly, the toilet in the middle sample of Figure 6.2 is missing in the *Implicit Model*. This is mainly linked to the explicit geometric grounding of the *Explicit Model* through its 3D scene representation, conveying complex compositional 3D structures, in contrast of the weak 3D supervision of the *Implicit Model*. This weak geometric supervision allows

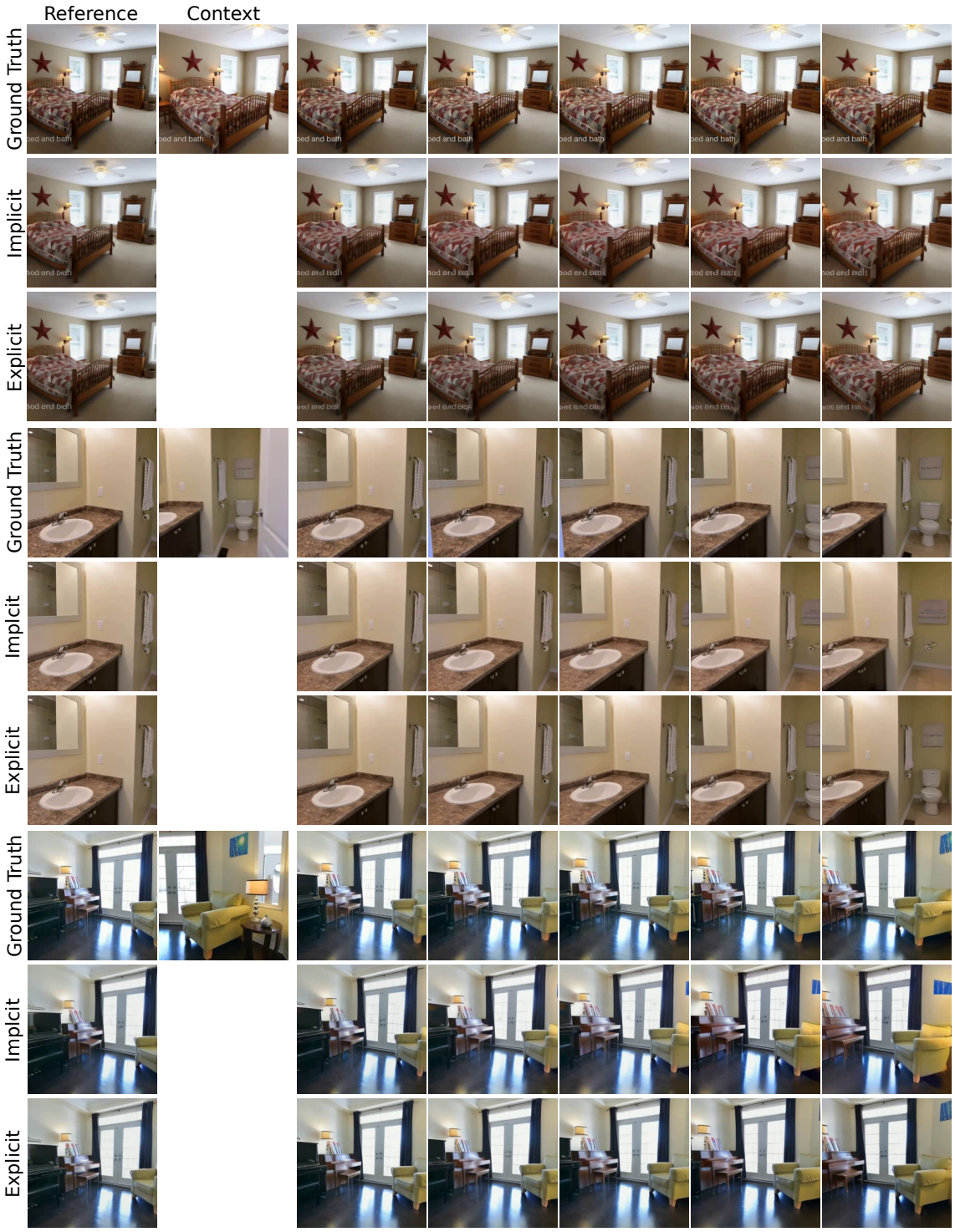


Figure 6.2: **Qualitative results.** Comparison of our *Implicit Model* and *Explicit Model*. Results were produce with 25 DDIM steps and a classifier-free guidance (CFG) scale of 3.5.

the model to freely aggregate as well as ignore information, making it less reliable.

Camera trajectory alignment. We further evaluate our methods in terms of the alignment to the provided camera trajectories. We employ the evaluation protocol as described in Section 6.2 to evaluate the camera trajectories. DynamiCrafter is excluded from this evaluation as it does not employ any camera conditioning and our evaluation pipeline failed to produce valid camera trajectories on a majority of the samples ($> 80\%$).

Method	TransErr ↓	RotErr ↓	CamMC ↓
MotionCtrl	2.89	2.04	4.34
CameraCtrl	2.54	1.84	3.85
CamI2V	1.79	1.16	2.58
Implicit Model	<u>1.53</u>	**1.09**	<u>2.29</u>
Explicit Model	**1.48**	<u>1.11</u>	**2.18**

Table 6.3: **Camera trajectories on RealEstate10K.** The camera trajectories are estimated through the GLOMAP [81] pipeline. Compared against the ground-truth trajectories, the evaluation metrics are averaged over successful runs out of 5 trials.

Since camera and scene motion in terms of their visual appearance are heavily correlated, it is evident that constraining not only the camera motion but the scene geometry as well, as done by our models, improves the alignment to the camera trajectory. The *Implicit* and *Explicit Model* show similar improvements in the camera trajectory over CamI2V. As both models keep the camera encoder and related modules injecting the camera condition frozen throughout the training, this improvement can mainly be attributed to an improved 3D consistency and scene representation steming from the injected scene context. The slight edge of the *Explicit Model* can be linked to its superior geometric grounding as discussed previously.

Time-wise comparison. To further investigate the faithfullness of the generated video sequence to the provided context, we report the image-based metrics independently for each timestep in Figure 6.3. As discussed, DynamiCrafter fails to produce plausible video sequences, yielding consistently poor performance across all timesteps. A common phenomenon that the performance degrades logarithmically with the distance to the reference frame is visible for the remaining models. This indicates a strong grounding of the generated visuals in the reference frame and lacking generative freedom of the baseline model as their is no inherent architectural bias, such

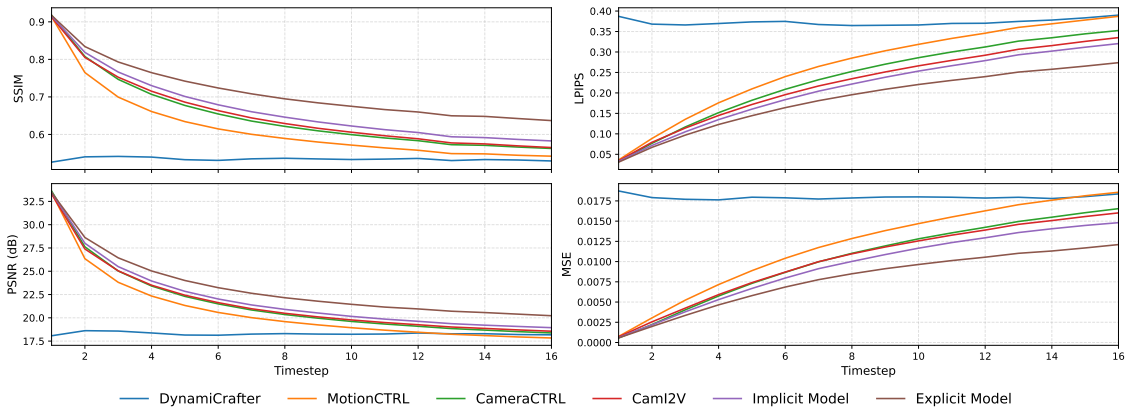


Figure 6.3: **Timewise comparison.** We compare our models in terms of frame-wise metrics SSIM, LPIPS, PSNR and MSE indepedently for each timestep. Our models especially for later timesteps show an improved visual quality through the integrated context.

as causal attention blocks, inducing such behavior. Our models achieve an improved image quality especially for frames later in the video sequence which can be attributed added context. Especially in terms of the MSE, which directly compares the frames pixel by pixel, our models excel over the baseline showing the effectiveness of additional scene context. This is most pronounced for our *Explicit Model* confirming its superior geometric grounding.

Failure cases. To better characterize our method's limitations, we summarize the two most frequent failures cases shown in Figure 6.4. In the upper example, a rightward pan reveals empty space next to a washing machine visible in the context view. The model fails to exploit this context and instead extrapolates from the reference image, filling the unobserved region with a washing-machine-like artifact. This mirrors the baseline models' behavior that originally motivated our work. Mitigating such artifacts, would require fine-tuning the denoising U-Net itself to fundamentally alter its reasoning and conditioning structure.

In the lower example, we present another common pitfall of the *Implicit Model*. The semantic branch provides a strong conditioning signal indicating the presence of a brick wall, while the visual branch fails to convey the exact structure and color of such, yielding an unrealistic brick-wall artifact. This likely arises because the semantic conditioning path is native to DynamiCrafter [19], whereas our visual branch introduces a new conditioning path, making the generation more reliant on the former.

Beyond our models' specific failures, we want to note a broader limitation inherent

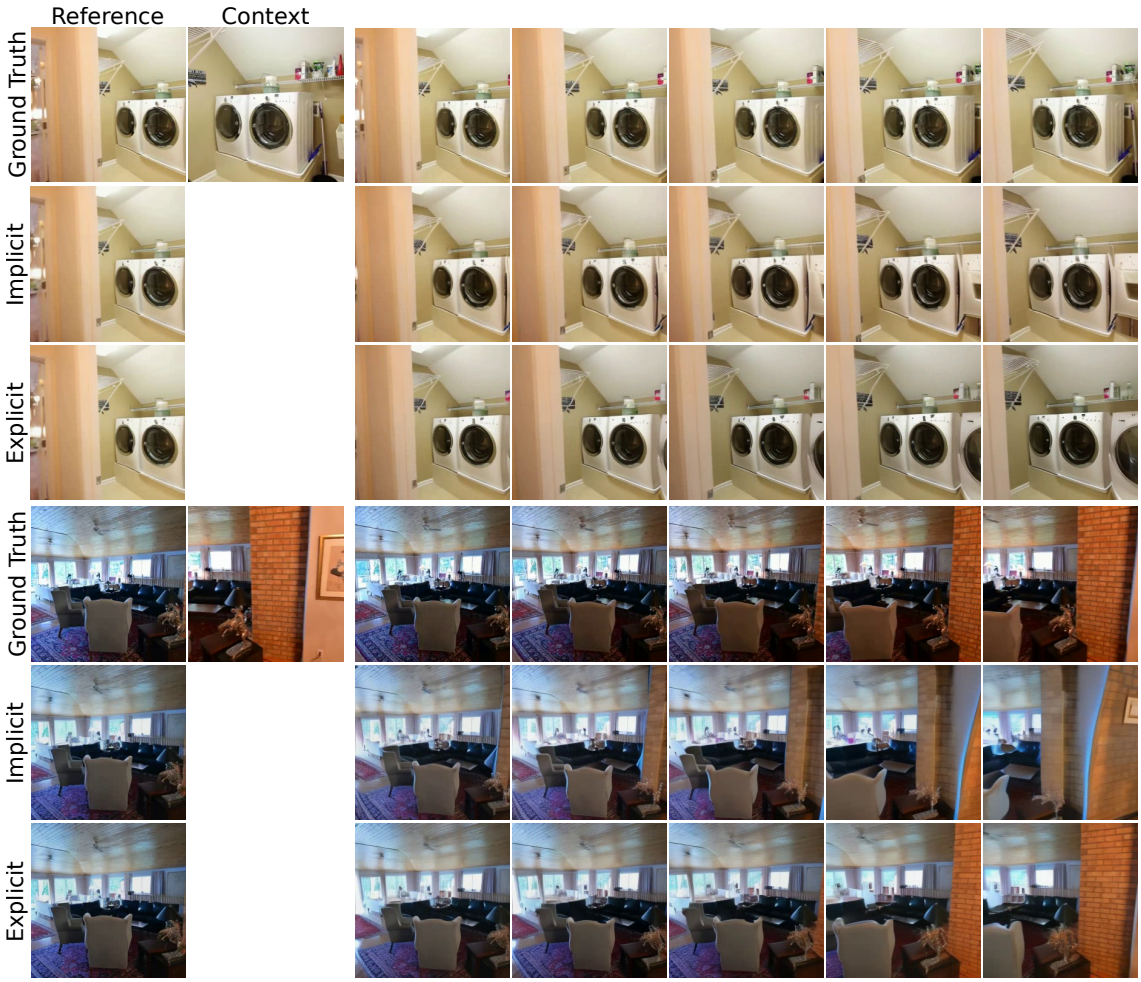


Figure 6.4: **Failure cases.** We present two distinct failure cases prominent in our methods. Upper: The model extrapolates from the reference image and generates semantically-aligned artifacts. Lower: The *Implicit Model* primarily propagates semantic instead of visual cues, leading to artifacts aligned to the semantic conditioning.

by all latent diffusion models. VAE compression into a latent space hampers the faithful representation of high-frequency details. Fine-grained objects are especially prone to errors, resulting in blurry details and temporal flickering. This is evident in the lower example of Figure 6.4, where the statue gradually loses detail while larger objects remain stable. Text and lettering, as in the upper example of Figure 6.2, are particularly difficut to preserve, which poses a challenge even for strong commercial video generators such as SORA-2 [3].

6.4 Context Sampling

We further study how the choice of context affects the generative process by evaluating three strategies for sampling context frames. All strategies choose frames disjoint from the video segment to be generated. *(end,-1]*, the strategy used throughout our evaluations, samples frames between the end of the target segment and the end of the full video. *end+1* uses the first frame after the last target frames, providing the maximum amount of scene information to the generation. *Furthest* selects the frame with the greatest temporal distance from the target segment, providing minimal to no additional context to the diffusion model. Results are reported in Table 6.4 and Figure 6.5 presents examples of the two former strategies.

Sample Strategy	Implicit model			Explicit model		
	FVD	SSIM	MSE	FVD	SSIM	MSE
$(\text{end}, -1]$	53.907	0.695	0.0082	50.498	0.721	0.0077
$\text{end} + 1$	50.582	0.707	0.0080	45.392	0.745	0.0071
Furthest	56.374	0.691	0.0089	57.695	0.702	0.0084

Table 6.4: **Condition sampling study.** To investigate the impact of different context views, we condition our methods using different context sampling strategies. *(end,-1]* represents the sampling strategy used through our evaluations, while *end+1* provides context with the maximal amount of information and *furthest* with the minimal amount of information.

The improved FVD scores for both methods under the *end+1* strategy indicate that the models effectively leverage the additional context. The *Explicit Model* benefits most, with an FVD improvement of $\sim$ 5.1 points. In Figure 6.5 it is visible that the *Eplicit Model* leverages the context to generate the chandelier above the table. The *Implicit Model* also incorporates visible context but through the weak geomtric supervision misplaces it in the scene.

In contrast, the *Furthest* strategy degrades the visual quality of the *Explicit Model* more significantly than that of the *Implicit Model*. This sensitivity arises due to the 3D Cache employing a multi-view depth estimator whose performance depends strongly on co-visibility between the reference and context frames. Low co-visibility results in poor registration and ultimately introduces artifacts into the conditioning views, which are propagated to the generated results. The *Implicit Model*, which does not depend on an explicit multi-view estimation stage, is less affected and shows only a modest degredation when the context is temporally distant. Overall, these results

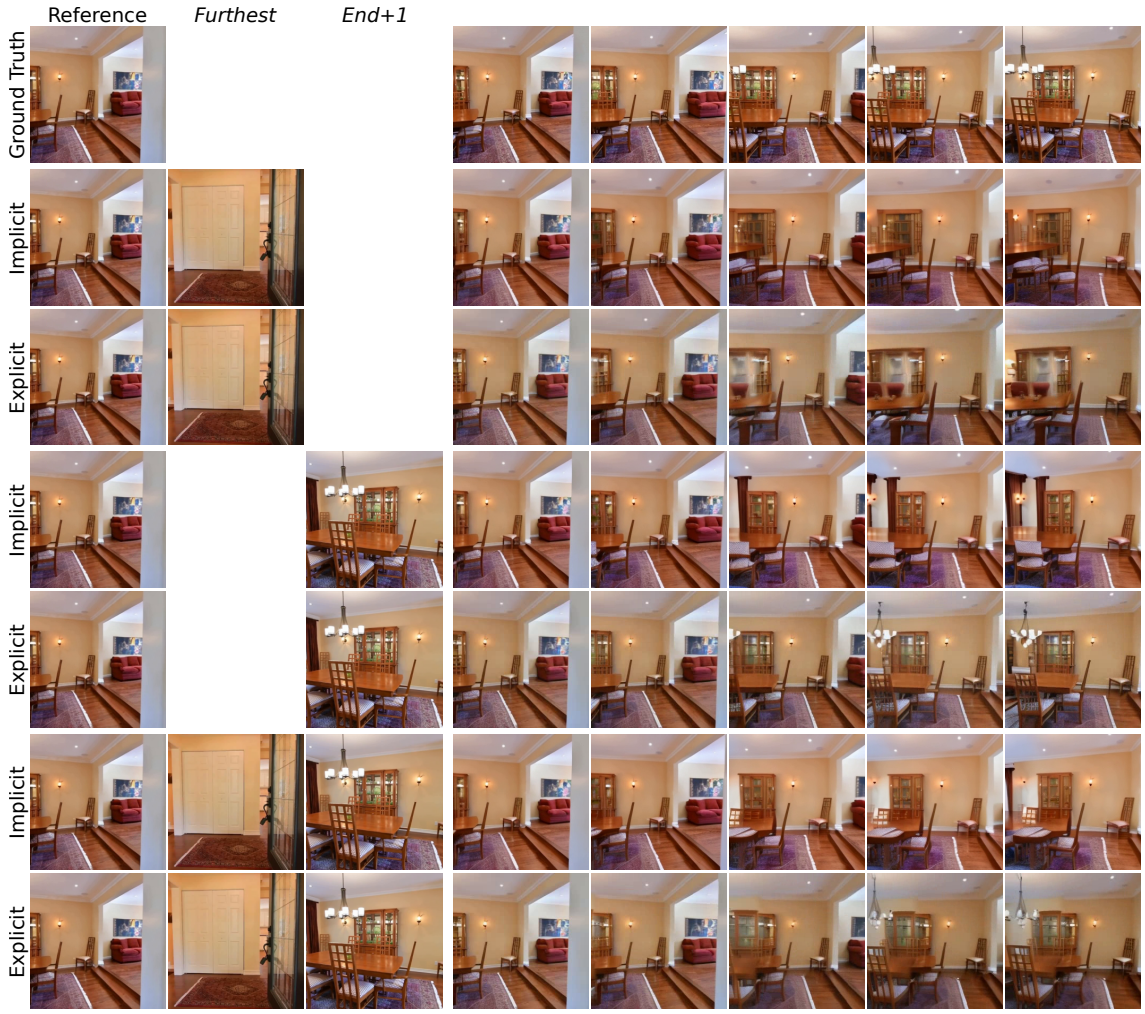


Figure 6.5: **Qualitative results of different sampling strategies.** We generate samples conditioned on the furthest frame providing minimal context and the frame immediately following the video providing maximal context. Our method is able to reject unrelated features from the *furthest* frame and only aggregate features from the *end + 1* frame providing additional information to the diffusion process.

suggest that our models can selectively incorporate useful context while remaining robust to less informative inputs.

6.5 Novel View Synthesis Comparison

To evaluate our method against a state-of-the-art novel view synthesis (NVS) method, we compare our method against FrugalNeRF [82], a sparse-vew NeRF-based method. Such NeRF-based methods typically rely on an initial sparse 3D reconstruction step,

limiting their applicability to diverse scenes. In a two-view setup, the COLMAP pipeline [83], typically employed in this step, only achieves a reliable registration in about 80% of the cases if the frames distance is between 1 s and 8.5 s, heavily limiting such approaches if the context frames are too close or too distant.

We train FrugalNeRF on 100 test scenes from the RealEstate10K dataset using the first and 17th frame as training frames and the intermediate ones as test frames, similar to the *end+1* evaluation setup of the context sampling study in Section 6.4. Accordingly, we define the reference and context frame for our approaches.

Table 6.5 shows that both our methods outperform FrugalNeRF while being more broadly applicable, as it does not require a prior reconstruction step, with our *Explicit Model* having a slight edge over the implicit variant. Moreover, since NeRF-based methods typically employ a test-time optimization training scheme, the time needed to learn and render the 3D representation (1265.87 s) is substantially larger than the pure rendering time for our approaches (8.01 s and 12.68 s), which can be directly applied to novel scenes after training.

Method	SSIM ↑	LPIPS ↓	PSNR ↑	Time (s)
FrugalNeRF [82]	0.737	0.156	23.76	1265.87
Implicit Model	<u>0.741</u>	<u>0.128</u>	<u>24.32</u>	**8.01**
Explicit Model	**0.762**	**0.101**	**25.27**	<u>12.68</u>

Table 6.5: **Novel view synthesis comparison.** We evaluate our models against a state-of-the-art sparse-view 3D reconstruction method that is geometrically grounded to faithfully represent the scene. All methods are tested at a resolution of 512 × 512 using the first and 17th frames as conditioning signals. Our approaches achieve slightly better image quality while requiring only a fraction of the compute time.

6.6 Computational Demand

We quantify the additional GPU memory footprint and the added latency introduced by the context encoders used in our methods. The results are summarized in Table 6.6. Our *Implicit Model* adds 97.4M parameters which amounts to only $\sim 3.3\%$ of CamI2V's total parameter counter. This yields a small memory overhead of ~ 0.47 Gib and a modest increase in forward time of 0.23s, with the visual stream being the main contributor to the extra latency. The added latency stems from the larger set of learnable context tokens, which aggregate context features to form a

Table 6.6: **Computational demand analysis.** Parameter count, GPU memory footprint and end-to-end latency of our methods compared to the baseline models. Latency is measured over the full generation process with 25 DDIM steps using classifier-free guidance (CFG). The *Implicit Model* adds only minimal overhead, whereas the *Explicit Model* introduces a larger costs because its context encoder runs at each denoising step. Both models fit on consumer-grade GPUs with < 16 GiB VRAM.

Method	# Parameters		GPU Memory (GiB)	Latency (s)
	Trainable	Total		
DynamiCrafter	—	2.6 B	10.43	5.143
MotionCtrl	—	2.6 B	10.49	4.849
CameraCtrl	—	2.8 B	11.28	4.898
CamI2V	—	2.9 B	11.21	7.976
Implicit Model	97.4 M	2.9 B	11.68	8.208
– Semantic Stream	50.9 M	50.9 M	0.19	0.004
– Visual Stream	46.5 M	46.5 M	0.17	0.043
Explicit Model	543.1 M	3.4 B	13.15	12.336

timestep- and pixel-wise representation.

In contrast, our *Explicit Model* employs the diffusion U-Net's encoder path adding $543\,M$ parameters and increasing memory usage by ~ 1.94 GiB. Despite this increase, the total requirement remains < 16 GiB, such that the model runs on consumer-grade desktop GPUs. Because the *Explicit Context Encoder* runs in parallel with the diffusion U-Net at each sampling step, it poses a noticable latency overhead during inference, adding about 35.3% runtime for a single video.

6.7 Ablation Studies

In the following, we perform separate ablation studies on the *Implicit* and *Explicit Model* to motivate our design choices.

6.7.1 Implicit Model

To thoroughly evaluate the design choice to embed the context in our *Implicit model*, we report several model variants in Table 6.7 excluding specific modules in the *Implicit Context Encoder*.

Conditioning Pixel	Conditioning Sem.	Epipolar	Time	FVD ↓	SSIM ↑	LPIPS ↓	PSNR ↑	MSE ↓
				71.014	0.656	0.219	20.628	0.0103
✓		✓	✓	76.023	0.648	0.245	19.487	0.0126
	✓	✓	✓	70.441	0.652	0.227	19.739	0.0119
✓	✓			63.817	0.667	0.211	20.316	0.0098
✓	✓		✓	61.613	0.678	0.204	20.759	0.0092
✓	✓	✓		<u>58.156</u>	<u>0.687</u>	<u>0.196</u>	<u>21.063</u>	<u>0.0087</u>
✓	✓	✓	✓	**53.907**	**0.695**	**0.190**	**21.196**	**0.0082**

Table 6.7: **Implicit model ablation studies.** We compare our design choices in different studies showing that our two-stream design complementarily embeds the context and guides the diffusion process. Adding epipolar attention and temporal embeddings to the *Implicit Context Encoder* equips it with explicit 3D and temporal awareness, improving context retrieval and further boosting performance.

Semantic and visual stream. First, we examine the individual contributions of the semantic and visual streams to the diffusion process. We train two model variants, each utilizing only one of the streams to inject additional context. Despite both variants being provided with an extended context, neither improves upon the baseline results. This limited improvement likely stems from DynamiCrafter being originally trained under matching conditions. In contrast, combining both semantic and visual streams significantly enhances performance, highlighting their complementary interaction.

3D awareness. Next, we evaluate the effectiveness of our method's 3D awareness, achieved through the epipolar cross-attention mechanism. Replacing epipolar cross-attention with standard (vanilla) cross-attention, allowing unrestricted feature aggregation from all tokens, still yields a considerable improvement of 9.5 FVD points over the baseline. This model variant, still, demonstrates a significant improvement on the baseline by 9.5 points in the FVD score but fails to match the performance of the 3D-aware model variant. This can be attributed to the model still leveraging the additional context for the generation but failing to reject features from invalid positions, especially when context frames provide minimal additional information due to them being sampled from distant regions.

Temporal awareness. Further, we assess the effect of temporal embeddings integrated into semantic and visual streams. Removing temporal embeddings results in a performance decline, although still outperforming CamI2V considerably. The temporal embeddings, particularly within the visual stream, explicitly guide the temporal attention of the U-Net to properly interpret timestep-specific context. Without this guidance, the epipolar cross-attention timestep-wise embedded context may be interpreted freely, resulting in impaired performance.

6.7.2 Explicit Model

We evaluate the *Explicit Model* by analyzing different 3D scene representations and their suitability. Further, we study different configurations of the *Explicit Context Encoder* injecting the cache renderings to motivate our design choices.

Approach	SSIM	LPIPS	PSNR	Time (s)	Memory (GiB)
Point cloud	0.70	0.20	22.03	0.73	3.13
TSDF Voxel Grid	0.61	0.42	19.05	0.27	1.64
3D Gaussians	0.83	0.12	25.62	0.24	2.39

Table 6.8: **3D representation.** Quality-resource trade-offs across different scene representations. "Time" describes the full forward pass of updating and rendering the 3D representation. 3D Gaussians achieve the best image quality with the fastest forward time and moderate memory requirement. TSDF voxel grids use the least memory but fail to produce high-quality frames.

3D Representation. As identified in Section 2.5, an effective 3D scene representation must deliver high rendering quality, fast inference suitable for training-time use and low memory overhead. We evaluate three representations: 3D Gaussians [53], raw point clouds and TSDF voxel grids. The 3D Gaussian representation is built with MVSplat [54] and rendered with standard Gaussian splatting [53]. Point clouds are rendered using the GPU-accelerated PyTorch3D pipeline [84] and TSDF voxel grids are constructed using NVBlox [85]. Following our evaluation protocol, we render 16 frames from 1-4 context images sampled from outside the 16-frame segment. Results for image quality, inference speed and memory consumption are summarized in Table 6.8.

3D Gaussians yield the highest image quality (SSIM 0.83, LPIPS 0.23, PSNR 25.62) and the fastest forward time with moderate memory usage. We attributed this to the lightweight pretrained prediction module of MVSplat and the efficiency of Gaussian

Figure 6.6: **3D Cache renderings.** Given two context frames (start and end), we render intermediate viewpoints using three 3D caches: a TSDF voxel grid, a point cloud, and 3D Gaussians. While the voxel grid and point cloud renderings exhibit misregistration artifacts like misaligned point clouds and holes, 3D Gaussians are able to faithfully represent the scene.

splatting. The point cloud approach yields the largest runtime and memory footprint due to maintaining full point clouds across all context views. TSDF voxel grids reduce runtime and memory by integrating points into a compact volume but introduce integration artifacts that degrade rendering quality.

In Figure 6.6 we show example renderings from each representation with the context view placed at the end of the sequence. Voxel-grid and point cloud renderings suffer from lighting inconsistencies and color shifts between views, whereas the 3D Gaussian representation is notably more robust. The voxel grid also exhibits integration artifacts at edges and view overlaps, where color discrepancies produce gray voxels. Imperfect depth maps and camera poses from our annotation pipeline further degrade quality, leading to misaligned point clouds, holes and slight scene shifts. In contrast, MVSplat employs a dedicated multi-view depth estimator with a refinement stage, which substantially improves registration. Overall, 3D Gaussians deliver the highest rendering quality with minimal overhead, making them our scene representation of choice.

3D Cache. In Table 6.9, we further investigate our model's performance in conjunction with the different 3D representations. Due to a CUDA-side memory leak in the garbage collection of NVBlox, we ran into out-of-memory issues after approximately $1,000$ training iterations. We therefore report results for 3D Gaussians and point clouds only.

Method	FVD ↓	SSIM ↑	LPIPS ↓	PSNR ↑	MSE ↑
Point cloud	61.739	0.665	0.221	20.789	0.0098
3D Gaussians	50.498	0.721	0.172	22.559	0.0077

Table 6.9: **3D Cache ablation study.** We train the *Explicit Model* with point clouds and 3D Gaussians as scene representations. The 3D Gaussian variant delivers consistently better video quality across all metrics due to its superior rendering quality.

Within our model, 3D Gaussians outperform direct rendering from point clouds on every metric. As illustrated in Figure 6.6, the advantage is primarily due to the superior rendering quality of Gaussian splatting. Point-cloud renderings suffer from misregistration and associated artifacts that do not contribute to the rendering masks. Such artifacts are hard to correct because they are invisible to the network and can only be partially addressed via cross-attention feature aggregation. Moreover, since rendering masks encode visibility, misregistration can hide truly visible regions in the context views and thus preventing correction through the cross-attention layers.

Context encoding. In our final ablation, we study the impact of the injected cross-attention and rendering mask used by the *Explicit Context Encoder*. In Table 6.10 we report three model variants in addition to the final model. The variants are a plain encoder, the encoder with injected cross-attention only and the encoder only employing the rendering masks. The final model uses both mechanisms.

The base model with a plain encoder attain an FVD of 58.513 and performs similarly to the *Implicit Model*. Adding cross-attention enables the model to correct rendering artifacts using the context frames as reference. This variant improves video quality across all metrics (FVD 53.731). Using only the rendering mask zeros features from unobserved regions before injecting context into the skip connections. This forces the denoising U-Net to synthesize such regions without guidance of the context encoder. Quality improves slightly but falls short of the cross-attention variant.

Combining cross-attention and masking corrects artifacts and handles unobserved regions effectively. The combined model achieves an SSIM of 0.721 and an FVD of

Cross-Attn.	Mask	FVD ↓	SSIM ↑	LPIPS ↓	PSNR ↑	MSE ↑
		58.513	0.687	0.194	20.936	0.0096
✓		<u>53.731</u>	<u>0.713</u>	<u>0.181</u>	<u>21.874</u>	<u>0.0084</u>
	✓	55.381	0.696	0.187	21.395	0.0089
✓	✓	**50.498**	**0.721**	**0.172**	**22.559**	**0.0077**

Table 6.10: **Context encoding ablation studies.** Impact of the added cross-attention layers and rendering masks on the video quality. While each technique improves the overall video quality, their conjunction enables our model to effectively recover from cache rendering artifacts, resulting in the overall best video quality.

50.498 achieving the highest quality visual performance. This shows the effectiveness of integrating the context views directly into the context encoding and preventing the propagation of features from unobserved regions.

7 Conclusion

This thesis demonstrates that additional scene context in the setting of context-to-video (C2V) benefits camera-controlled diffusion models that, in conventional image-to-video (I2V) setups, are restricted by the context of a single image reference. Extending the available context through additional posed context views improves both video fidelity and adherence to prescribed camera trajectories.

We explored two distinct approaches for injecting context into a pre-trained diffusion model (i.e., DynamiCrafter [19]) under weak versus strong 3D supervision. The *Implicit Model* fuses context in latent space through temporally and 3D-aware semantic and visual streams, relying only on weak 3D cues. The *Explicit Model* imposes stronger geometric grounding by rendering trajectory-aligned views from a 3D Gaussian scene representation and injecting them into the denoising U-Net in a ControlNet-like fashion.

Under the evaluation protocol in Section 6.1, both models outperform strong camera-controlled baseline across FVD, SSIM, LPIPS, PSNR, and MSE (Section 6.3). Gains are most pronounced at later timesteps, where I2V models' performance typically degrade without sufficient visual context, indicating the importance of added missing context for the generative quality. Both models also reduce rotation and translation errors in recovered camera trajectories relative to CamI2V [18], steming from improved 3D consistency induced by the additional scene constraints.

We further analyzed the impact of differently sampled context on the generative process (Section 6.4). Expressive context, placed at the end of the target sequence, improves fidelity, showing that our models effectively leverage the context. Uninformative context, distant from the target sequence, only slightly degrades the video quality of our models. The *Explicit Model* is most affected due to its dependence on a multi-view depth estimator whose performance degrades with low co-visibility between frames.

Between the two designs, the *Explicit Model* attains the best overall results (FVD of 50.498 and SSIM of 0.721) by leveraging explicit geometry and selectively correcting artifacts via masking and cross-attention. The *Implicit Model* tracks closely while adding only modest parameter and latency overhead, yielding a favorable quality-

compute trade-off. Overall, our findings demonstrate that (i) augmenting missing context improves video quality and trajectory faithfulness and (ii) explicit geometric grounding further enhances 3D consistency and video fidelity.

Limitations and future work. Failure cases occur when the provided context is unrelated to the target sequence. Incorporating mechanisms that evaluate co-visibility and filter or reweight unhelpful context could mitigate these issues. Fine-tuning the denoising U-Net may resolve some errors but requires a carefully designed training pipeline which is not provided in DynamiCrafter's release.

Our evaluation focuses on RealEstate10K [20], which consists primarily of indoor and outdoor house tours and thus offers limited visual diversity. Achieving broader generalization beyond RealEstate10K will likely require training on more diverse datasets (e.g., DL3DV [86], Waymo Open Dataset [87]), with a corresponding increase in compute demands. Because most training data depict static scenes, our methods treat context as an instantaneous snapshot and are agnostic to time. Disentangling static and dynamic scene context, with separate conditioning pathways, is a promising direction for controllable video editing and dynamic-scene generation.

Finally, recent DiT-style backbones [25], [32], [33], show impressive results but differ architecturally from U-Net diffusion models and will require specialized conditioning mechanisms. Extending the C2V paradigm to DiT backbones is a natural next step and may further improve fidelity, especially for longer target sequences.

In summary, additional scene context and explicit geometric grounding substantially advance camera-controlled video generation, while our two approaches outline practical routes for extending I2V models to C2V models.

List of Figures

List of Tables

Bibliography

[1] K. Perlin, "An image synthesizer", *SIGGRAPH '85: ACM SIGGRAPH 1985 Conference Papers*, 1985.

[2] D. Terzopoulos and A. Witkin, "Physically based models with rigid and deformable components", *IEEE Computer Graphics and Applications*, 1988.

[3] "Sora 2 system card", OpenAI, Tech. Rep., 2025.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[7] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", in *International Conference on Learning Representations (ICLR)*, 2014.

[8] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[9] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with CLIP latents", 2022. arXiv: 2204.06125.

[10] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding", 2022. arXiv: 2205.11487.

[11] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, and T. Salimans, "Imagen video: High definition video generation with diffusion models", 2022. arXiv: 2210.02303.

[12] J. Yu, Y. Xu, J. Y. Koh, T. Luong, G. Baid, Z. Wang, V. Vasudevan, A. Ku, Y. Yang, B. K. Ayan, B. Hutchinson, W. Han, Z. Parekh, X. Li, H. Zhang, J. Baldridge, and Y. Wu, "Scaling autoregressive models for content-rich text-to-image generation", 2022. arXiv: 2206.10789.

[13] Z. Wang, Z. Yuan, X. Wang, T. Chen, M. Xia, P. Luo, and Y. Shan, "MotionCtrl: A unified and flexible motion controller for video generation", in *SIGGRAPH '24: ACM SIGGRAPH 2024 Conference Papers*, 2024.

[14] P. Esser, J. Chiu, P. Atighehchian, J. Granskog, and A. Germanidis, "Structure and content-guided video synthesis with diffusion models", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[15] C. Wu, J. Liang, L. Ji, F. Yang, Y. Fang, D. Jiang, and N. Duan, "NÜWA: Visual synthesis pre-training for neural visual world creation", in *European Conference on Computer Vision (ECCV)*, 2022.

[16] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[17] H. He, Y. Xu, Y. Guo, G. Wetzstein, B. Dai, H. Li, and C. Yang, "CameraCtrl: Enabling camera control for video diffusion models", in *International Conference on Learning Representations (ICLR)*, 2025.

[18] G. Zheng, T. Li, R. Jiang, Y. Lu, T. Wu, and X. Li, "CamI2V: Camera-controlled image-to-video diffusion model", 2024. arXiv: 2410.15957.

[19] J. Xing, M. Xia, Y. Zhang, H. Chen, W. Yu, H. Liu, X. Wang, T. Wong, and Y. Shan, "DynamiCrafter: Animating open-domain images with video diffusion priors", in *European Conference on Computer Vision (ECCV)*, 2024.

[20] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, "Stereo magnification: Learning view synthesis using multiplane images", *ACM Transactions on Graphics*, 2018.

[21] 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", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer", *Journal of Machine Learning Research*, 2020.

[23] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "Sdxl: Improving latent diffusion models for high-resolution image synthesis", 2023. arXiv: 2307.01952.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[25] W. Peebles and S. Xie, "Scalable diffusion models with transformers", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[26] Z. Li, J. Zhang, Q. Lin, J. Xiong, Y. Long, X. Deng, Y. Zhang, X. Liu, M. Huang, Z. Xiao, D. Chen, J. He, J. Li, W. Li, C. Zhang, R. Quan, J. Lu, J. Huang, X. Yuan, X. Zheng, Y. Li, J. Zhang, C. Zhang, M. Chen, J. Liu, Z. Fang, W. Wang, J. Xue, Y. Tao, J. Zhu, K. Liu, S. Lin, Y. Sun, Y. Li, D. Wang, M. Chen, Z. Hu, X. Xiao, Y. Chen, Y. Liu, W. Liu, D. Wang, Y. Yang, J. Jiang, and Q. Lu, "Hunyuan-dit: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding", 2024. arXiv: 2405.08748.

[27] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li, "Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis", in *International Conference on Learning Representations (ICLR)*, 2023.

[28] 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. arXiv: 2311.15127.

[29] H. Chen, M. Xia, Y. He, Y. Zhang, X. Cun, S. Yang, J. Xing, Y. Liu, Q. Chen, X. Wang, C. Weng, and Y. Shan, "VideoCrafter1: Open diffusion models for high-quality video generation", 2023. arXiv: 2310.19512.

[30] S. Zhang, J. Wang, Y. Zhang, K. Zhao, H. Yuan, Z. Qin, X. Wang, D. Zhao, and J. Zhou, "I2VGen-XL: High-quality image-to-video synthesis via cascaded diffusion models", 2023. arXiv: 2311.04145.

[31] H. Lu, G. Yang, N. Fei, Y. Huo, Z. Lu, P. Luo, and M. Ding, "VDT: General-purpose video diffusion transformers via mask modeling", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[32] S. Chen, M. Xu, J. Ren, Y. Cong, S. He, Y. Xie, A. Sinha, P. Luo, T. Xiang, and J. Perez-Rua, "GenTron: Diffusion transformers for image and video generation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[33] Z. Yang, J. Teng, W. Zheng, M. Ding, S. Huang, J. Xu, Y. Yang, W. Hong, X. Zhang, G. Feng, D. Yin, X. Gu, Y. Zhang, W. Wang, Y. Cheng, T. Liu, B. Xu, Y. Dong, and J. Tang, "CogVideoX: Text-to-video diffusion models with an expert transformer", in *International Conference on Learning Representations (ICLR)*, 2025.

[34] T. Wang, L. Li, K. Lin, Y. Zhai, C.-C. Lin, Z. Yang, H. Zhang, Z. Liu, and L. Wang, "DisCo: Disentangled control for realistic human dance generation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[35] Z. Xu, J. Zhang, J. H. Liew, H. Yan, J.-W. Liu, C. Zhang, J. Feng, and M. Z. Shou, "MagicAnimate: Temporally consistent human image animation using diffusion model", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[36] X. Wang, H. Yuan, S. Zhang, D. Chen, J. Wang, Y. Zhang, Y. Shen, D. Zhao, and J. Zhou, "VideoComposer: Compositional video synthesis with motion controllability", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[37] D. Xu, W. Nie, C. Liu, S. Liu, J. Kautz, Z. Wang, and A. Vahdat, "CamCo: Camera-controllable 3d-consistent image-to-video generation", 2024. arXiv: 2406.02509.

[38] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models", in *International Conference on Learning Representations (ICLR)*, 2022.

[39] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models", in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[40] L. Hu, "Animate anyone: Consistent and controllable image-to-video synthesis for character animation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[41] M. Li, T. Yang, H. Kuang, J. Wu, Z. Wang, X. Xiao, and C. Chen, "ControlNet++: Improving conditional controls with efficient consistency feedback", in *European Conference on Computer Vision (ECCV)*, 2024.

[42] B. Peng, J. Wang, Y. Zhang, W. Li, M. Yang, and J. Jia, "ControlNeXt: Powerful and efficient control for image and video generation", 2024. arXiv: 2408.06070.

[43] C. Mou, X. Wang, L. Xie, Y. Wu, J. Zhang, Z. Qi, Y. Shan, and X. Qie, "T2I-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models", in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[44] Z. Xing, Q. Dai, H. Hu, Z. Wu, and Y.-G. Jiang, "SimDA: Simple diffusion adapter for efficient video generation", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[45] X. Guo, M. Zheng, L. Hou, Y. Gao, Y. Deng, P. Wan, D. Zhang, Y. Liu, W. Hu, Z. Zha, H. Huang, and C. Ma, "I2V-adapter: A general image-to-video adapter for diffusion models", 2024. arXiv: 2312.16693.

[46] Y. Guo, C. Yang, A. Rao, Z. Liang, Y. Wang, Y. Qiao, M. Agrawala, D. Lin, and B. Dai, "AnimateDiff: Animate your personalized text-to-image diffusion models without specific tuning", in *International Conference on Learning Representations (ICLR)*, 2024.

[47] S. Yang, L. Hou, H. Huang, C. Ma, P. Wan, D. Zhang, X. Chen, and J. Liao, "Direct-a-Video: Customized video generation with user-directed camera movement and object motion", in *SIGGRAPH '24: ACM SIGGRAPH 2024 Conference Papers*, 2024.

[48] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand, "Light field networks: Neural scene representations with single-evaluation rendering", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[49] S. Thrun, "Learning occupancy grids with forward sensor models", *Autonomous Robots*, 2003.

[50] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. I. Nieto, and R. Y. Siegwart, "Signed distance fields: A natural representation for both mapping and planning", in *Robotics: Science and Systems (RSS)*, 2016.

[51] B. Curless and M. Levoy, "A volumetric method for building complex models from range images", in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, ACM, 1996.

[52] 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", in *European Conference on Computer Vision (ECCV)*, 2020.

[53] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering", *ACM Transactions on Graphics*, 2023.

[54] Y. Chen, H. Xu, C. Zheng, B. Zhuang, M. Pollefeys, A. Geiger, T. Cham, and J. Cai, "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images", in *European Conference on Computer Vision (ECCV)*, 2024.

[55] D. Charatan, S. L. Li, A. Tagliasacchi, and V. Sitzmann, "Pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[56] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[57] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics", in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[59] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[60] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model", in *International Conference on 3D Vision (3DV)*, IEEE, 2018.

[61] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras", *IEEE Transactions on Robotics*, 2017.

[62] Z. Liu, L. Zhu, B. Shi, Z. Zhang, Y. Lou, S. Yang, H. Xi, S. Cao, Y. Gu, D. Li, X. Li, Y. Fang, Y. Chen, C. Hsieh, D. Huang, A. Cheng, V. Nath, J. Hu, S. Liu, R. Krishna, D. Xu, X. Wang, P. Molchanov, J. Kautz, H. Yin, S. Han, and Y. Lu, "Nvila: Efficient frontier visual language models", 2024. arXiv: 2412.04468.

[63] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "VGGT: Visual geometry grounded transformer", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[64] J. Wang, N. Karaev, C. Rupprecht, and D. Novotny, "Vggsfm: Visual geometry grounded deep structure from motion", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[65] X. Wei, Y. Zhang, Z. Li, Y. Fu, and X. Xue, "Deepsfm: Structure from motion via deep bundle adjustment", in *European Conference on Computer Vision (ECCV)*, 2020.

[66] H. Jun and J. Bailenson, "Temporal RVL: A depth stream compression method", in *Proceedings of IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops(VRW)*, 2020.

[67] D. Li, J. Li, H. Le, G. Wang, S. Savarese, and S. C. H. Hoi, "LAVIS: A one-stop library for language-vision intelligence", in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023.

[68] G. Ilharco, M. Wortsman, R. Wightman, C. Gordon, N. Carlini, R. Taori, A. Dave, V. Shankar, H. Namkoong, J. Miller, H. Hajishirzi, A. Farhadi, and L. Schmidt, *Openclip*, 2021.

[69] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Révaud, "DUSt3R: Geometric 3d vision made easy", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[70] B. Horn, H. Hilden, and S. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices", *Journal of the Optical Society of America A*, 1988.

[71] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver: General perception with iterative attention", in *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[72] L. Denninger, S. M. Azar, and J. Gall, "Camc2v: Context-aware controllable video generation", in *International Conference on 3D Vision (3DV)*, 2025.

[73] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[74] Y. He, T. Yang, Y. Zhang, Y. Shan, and Q. Chen, "Latent video diffusion models for high-fidelity long video generation", 2023. arXiv: `2211.13221`.

[75] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "Zero: Memory optimizations toward training trillion parameter models", in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

[76] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", in *International Conference on Learning Representations (ICLR)*, 2015.

[77] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[78] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", *IEEE Transactions on Image Processing*, 2004.

[79] 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.

[80] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[81] L. Pan, D. Baráth, M. Pollefeys, and J. L. Schönberger, "Global structure-from-motion revisited", in *European Conference on Computer Vision (ECCV)*, 2025.

[82] C. Lin, C. Wu, C. Yeh, S. Yen, C. Sun, and Y. Liu, "Frugalnerf: Fast convergence for extreme few-shot novel view synthesis without learned priors", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[83] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[84] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d", 2020. arXiv: 2007.08501.

[85] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart, "Nvblox: Gpu-accelerated incremental signed distance field mapping", 2024. arXiv: 2311.00626.

[86] L. Ling, Y. Sheng, Z. Tu, W. Zhao, C. Xin, K. Wan, L. Yu, Q. Guo, Z. Yu, Y. Lu, X. Li, X. Sun, R. Ashok, A. Mukherjee, H. Kang, X. Kong, G. Hua, T. Zhang, B. Benes, and A. Bera, "Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision", 2023. arXiv: 2312.16256.

[87] J. Mei, A. Z. Zhu, X. Yan, H. Yan, S. Qiao, Y. Zhu, L.-C. Chen, H. Kretzschmar, and D. Anguelov, "Waymo open dataset: Panoramic video panoptic segmentation", 2022. arXiv: 2206.07704.