Depth-Conditioned Video Generation via ControlNet + AnimateDiff

Extending Spatial Conditioning to Video Diffusion Models

Jayin Khanna

CSD722: Computer Vision

Contents

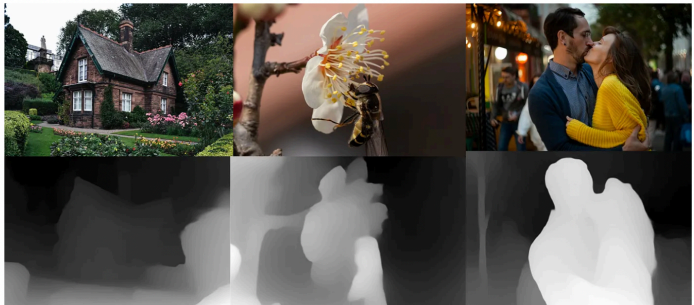
Motivation: The Control Problem in Generative Video

Text is an Underspecified Spatial Signal

- Text-to-video models are powerful but lack explicit spatial control.
- The same prompt:

  “a misty mountain lake at dawn”

  can generate very different layouts.
- Text captures semantics, not geometry.
- We introduce geometric conditioning using depth maps.



Motivation: Depth-Based Geometric Conditioning

Why Depth Maps?

- Depth maps encode spatial layout unambiguously.
- They provide:
  - view consistency
  - resolution-agnostic structure
  - model-agnostic conditioning
- Depth can be estimated from monocular RGB using MiDaS and DPT.
- No specialized hardware is required.

Conditioning Formulation

$$p(x \mid y_{\text{text}})$$

$$p(x \mid y_{\text{text}},\ c_{\text{depth}})$$

$$c_{\text{depth}} \in \mathbb{R}^{B\times 1\times F\times H\times W}$$

Text-to-Image Models Supporting Depth Conditioning

Base Model	Depth Conditioning	Notes
Stable Diffusion 1.5	ControlNet Depth	Most widely used and easiest to fine-tune
Stable Diffusion XL	SDXL ControlNet Depth	Higher quality, larger VRAM usage
Stable Diffusion 2.1	Native depth + ControlNet	Includes depth-aware variants
Stable Diffusion Depth2Img	Native depth conditioning	Official depth-guided pipeline
Flux.1	Depth adapters / ControlNet	Emerging ecosystem
PixArt-α	T2I Adapters	Transformer-based architecture
Kandinsky 2.2	ControlNet-style conditioning	Supports structural guidance
Playground v2	SDXL-compatible ControlNet	Built on SDXL tooling

Motivation: Why Video Makes This Harder

Video Adds a Third Dimension of Difficulty: Time

Challenge 1: Temporal Coherence

- Applying image ControlNet independently to each frame fails.
- Each frame is denoised independently:
  - no information shared across time
  - objects flicker between frames
  - spatial consistency is not preserved
- Depth maps are temporally smooth, but per-frame generation is not.

Challenge 2: Memory and Compute Scaling

	Image	Video (8 Frames)
Latent shape	$[B, 4, 64, 64]$	$[B, 4, F, 64, 64]$
ControlNet params	563M	563M
Frozen UNet params	1313M	1313M
Activation memory	~ 3 GB	~ 23 GB
24GB GPU feasible?	?	? (naively)

Motivation: Temporally Structured Conditioning

Challenge 3: Conditioning Must Be Temporally Structured

In image ControlNet, the condition is a single tensor:

$$c \in \mathbb{R}^{1 \times H \times W}$$

In our setting, the condition becomes a sequence:

$$c_{\text{depth}} \in \mathbb{R}^{B \times 1 \times F \times H \times W}$$

This introduces new architectural requirements:

- The condition encoder must process temporal sequences.
- Features must align with AnimateDiff temporal attention modules.
- Temporal consistency must be preserved across frames.

Key Insight

$$\text{Video Generation} \neq \text{Image Generation} \times F$$

Background: Generative Models

Goal of Generative Modeling

We observe a dataset of images:

$$x_1, x_2, x_3, \ldots, x_n$$

sampled from an unknown real data distribution:

$$p_{\text{data}}(x)$$

- We never observe $p_{\text{data}}(x)$ directly.
- We only observe samples drawn from it.
- The objective is to learn a parameterized model:

$$p_\theta(x)$$

$$p_\theta(x) \approx p_{\text{data}}(x)$$

After training, we want to sample:

$$x \sim p_\theta(x)$$

Background: Denoising Diffusion Probabilistic Models (DDPMs)

Main Idea

Directly sampling from a complex distribution is difficult.

Instead, diffusion models learn a simpler task:

Denoising

The model learns how to progressively remove Gaussian noise from corrupted images.

Forward Noising Process

Starting from a clean image x_0:

$$x_0 \to x_1 \to x_2 \to \cdots \to x_T$$

clean pure Gaussian noise

At each timestep:

- a small amount of Gaussian noise is added
- after many steps, the image becomes pure noise

The model is then trained to reverse this process.

DDPM Training and Inference

Forward Diffusion Process

$$q(z_t \mid z_0) = \mathcal{N}\left(z_t; \sqrt{\bar{\alpha}_t} z_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

Training reparameterization:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$$

$$\varepsilon \sim \mathcal{N}(0, \mathbf{I})$$

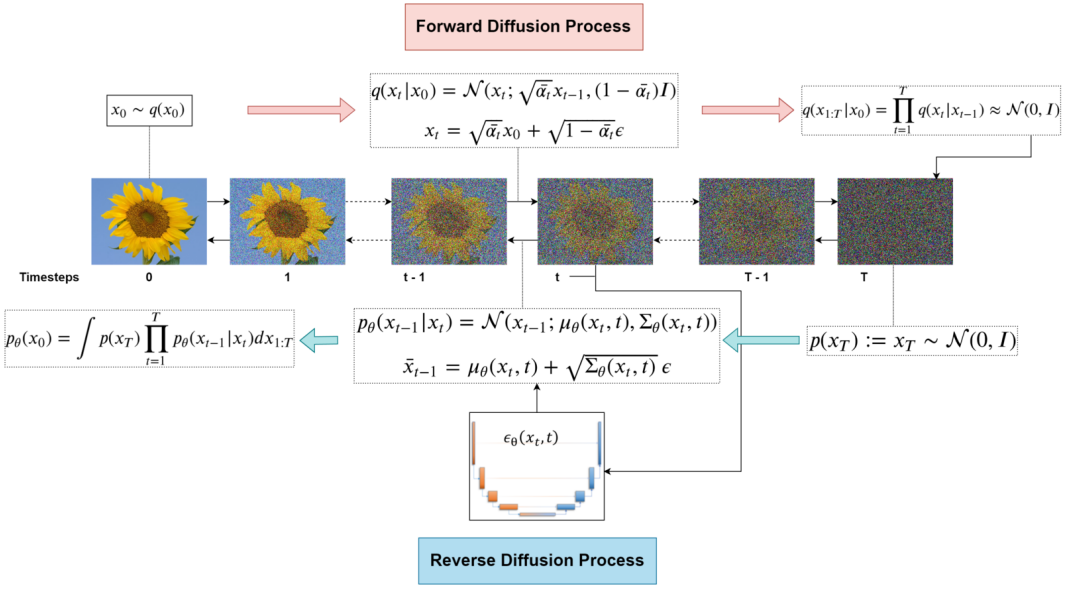
$$\bar{\alpha}_t = \prod_{s=1}^{t} (1 - \beta_s)$$

Training Objective

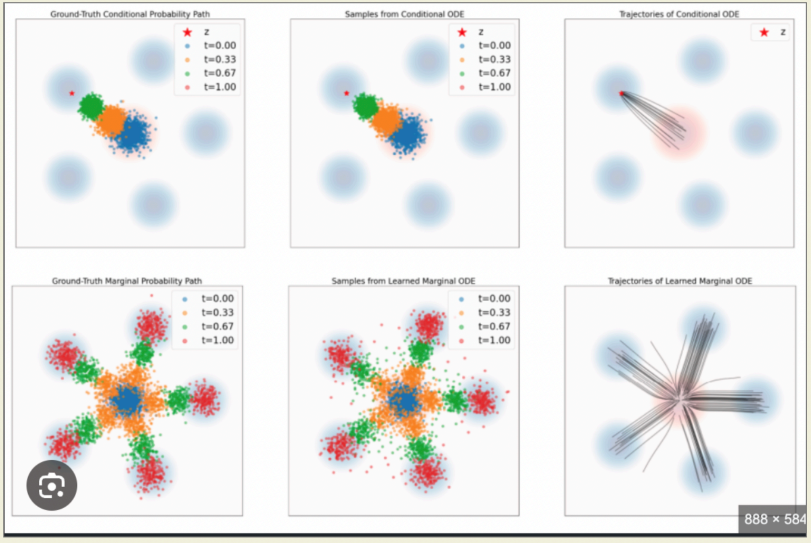
$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}_{z_0, \varepsilon, t}\left[\left\|\varepsilon - \varepsilon_\theta(z_t, t, \tau(y))\right\|^2\right]$$

- $\tau(y)$: CLIP text embedding
- UNet predicts added noise

DDPM Framework Overview



Latent space view



Dimensionality Problem

Why Pixel-Space Diffusion Fails

- Images live in high-dimensional space
- Example:
  - $64 \times 64 \times 3 = 12{,}288$
  - $256 \times 256 \times 3 = 196{,}608$
  - $1024 \times 1024 \times 3 = 3{,}145{,}728$
- Attention operates on:

$$N = h \times w$$

- Computational complexity:

$$\mathcal{O}(N^2) = \mathcal{O}((h \times w)^2)$$

Attention Scaling

Res	N	N^2	Memory
32^2	1K	1M	4 MB
64^2	4K	16M	67 MB
128^2	16K	268M	1.07 GB
256^2	65K	4.3B	17 GB
512^2	262K	68B	275 GB

Training Cost

- ADM (ImageNet):
  - 916 V100-days
- 50K samples:
  - $\sim$ 5 days on A100

Latent Diffusion Models (LDM)

Main Idea

Compress → Diffusion in latent space → Decode to image space

Stage 1: Autoencoder

- $$x \to \mathcal{E} \to z \to \mathcal{D} \to \tilde{x}$$
- z is significantly smaller than x
- $\tilde{x}$ remains perceptually similar to x
- Encoder and decoder are frozen after training

Latent Compression

$$512 \times 512 \times 3$$

to

$$64 \times 64 \times 4$$

Approximately: 48 times dimensionality reduction

Stage 2: Diffusion in Latent Space

- Forward process:

$$z \to z_T \sim \mathcal{N}(0, I)$$

- Learn reverse denoising process:

$$z_T \to z_0$$

- Decode once at the end:

$$z_0 \to \mathcal{D} \to \tilde{x}$$

Diffusion now operates on compact latent tensors:

$$z \in \mathbb{R}^{4 \times 64 \times 64}$$

Related Work II: ControlNet

Injecting Spatial Control into Frozen Diffusion Models

The Core Problem

- Stable Diffusion is extremely expensive to retrain

$$\sim 256 \text{ A100-days}$$

- We cannot fine-tune the entire model for every new conditioning signal

ControlNet Insight

- Freeze the pretrained UNet entirely
- Train only a parallel conditioning encoder
- Inject learned residual corrections into the frozen backbone

Why Naive Methods Fail

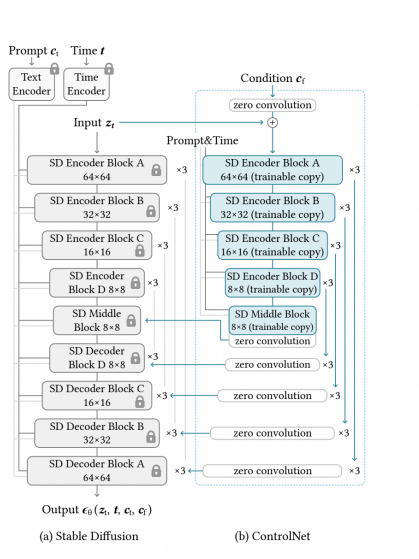
Fine-tune Entire UNet Catastrophic forgetting of pretrained knowledge
Concatenate Depth Channels Destroys pretrained channel statistics
Train From Scratch No access to pretrained diffusion features

Key Principle

$$\varepsilon_\theta^{\text{controlled}} = \varepsilon_\theta^{\text{frozen}} + \sum_{l=1}^{L} \mathcal{Z}_l(f_l(c))$$

Frozen diffusion backbone + trainable residual conditioning

ControlNet Architecture



Frozen Stable Diffusion backbone with trainable residual conditioning pathways.

$$y_c = \mathcal{F}(x;\Theta) + \mathcal{Z}\Big(\mathcal{F}(x+\mathcal{Z}(c;\Theta_{z1});\Theta_c);\Theta_{z2}\Big)$$

- $\mathcal{F}(x;\Theta)$: Frozen pretrained backbone
- $\mathcal{Z}(\cdot)$: Zero-initialized convolution layers
- c: Conditioning input (depth, edges, pose, etc.)
- Residual branch injects spatial control without disturbing pretrained weights

Zero-Convolutions and Training Stability

Why Zero-Convolutions Are Necessary

At initialization, ControlNet features are unstable.
If injected directly into the frozen decoder:

- generated images collapse into noise
- pretrained representations are destroyed
- gradients become unstable

Critical Stability Property

At training step 0:

$$\mathcal{Z}_l(\cdot) = 0$$

Therefore:

- residuals initially vanish
- frozen decoder behaves exactly like pretrained Stable Diffusion

Zero-Initialized Residual Injection

$$\text{residual}_l = \mathcal{Z}_l\left(f_l^{\text{ControlNet}}(c)\right)$$

where:

$$W_{\mathcal{Z}_l} = 0$$

$$b_{\mathcal{Z}_l} = 0$$

Training Dynamics

As training progresses:

- residual pathways gradually move away from zero
- only useful conditioning information is injected
- pretrained representations remain preserved

Controlled Diffusion Formulation and Limitations

Controlled Noise Prediction

$$\varepsilon_\theta^{\text{controlled}}(z_t, t, \tau(y), c) = \varepsilon_\theta^{\text{frozen}}(z_t, t, \tau(y)) + \sum_{l=1}^{L} \mathcal{Z}_l(f_l(c))$$

Interpretation:

- Frozen Stable Diffusion backbone remains unchanged
- ControlNet injects learned residual corrections
- Residuals are initialized at zero and learned progressively

What ControlNet Does NOT Solve

- Conditions only on:

$$c \in \mathbb{R}^{1 \times H \times W}$$

- No temporal dimension; No notion of frame ordering; No motion modeling
- Applying independently per frame causes temporal flickering

ControlNet provides spatial controllability for images, but not temporally coherent video generation.

Related Work III: AnimateDiff

Temporal Coherence via Motion Modules in Frozen Spatial Layers

Core Insight

AnimateDiff uses the same idea ControlNet does, but in the temporal direction:

Freeze Spatial Layers + Train Temporal Layers

- Stable Diffusion already generates high-quality individual frames
- Spatial layers have no temporal memory
- Temporal consistency must be learned separately

Motion Modules

AnimateDiff inserts trainable temporal attention blocks after spatial layers.

Each frame attends to all other frames:

$$f \to \{1, \ldots, F\}$$

This enables:

- object consistency
- motion continuity
- temporal coherence

Only motion modules are trained on video data.

Frozen Spatial Backbone + Trainable Temporal Attention

Motion Module: Temporal Self-Attention

Input Feature Representation

After a spatial block:

$$h \in \mathbb{R}^{(B \cdot F) \times C \times H \times W}$$

Frames are batch-fused because spatial layers process each frame independently.

Step 1: Expose Temporal Dimension

$$h \to h' \in \mathbb{R}^{B \times (C \cdot H \cdot W) \times F}$$

Each spatial position now becomes a temporal sequence of length F.

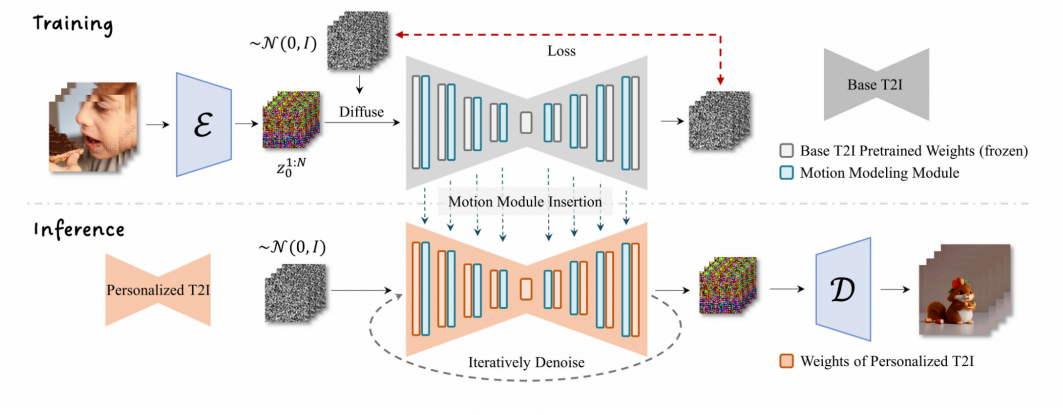
Step 2: Temporal Self-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V$$

Frame f attends to all frames:

$$\{1, \ldots, F\}$$

AnimateDiff Pipeline



Where Motion Modules Are Inserted

AnimateDiff UNet Structure

- Motion modules are inserted after every spatial block
- Spatial layers remain frozen
- Motion modules are trained on WebVid-10M
- They learn temporal evolution, not spatial appearance

Critical Shape Detail: Text Embedding Expansion

The Shape Problem

Because frames are batch-fused:

$$B \cdot F$$

the text embedding must also expand across frames.

Original text embedding:

$$\tau(y) \in \mathbb{R}^{B \times 77 \times 768}$$

Expanded embedding:

$$\tau(y) \to \mathbb{R}^{(B \cdot F) \times 77 \times 768}$$

Why This Matters

Each frame requires its own copy of the caption embedding.

Passing shape:

$$[B, 77, 768]$$

instead of:

$$[B \cdot F, 77, 768]$$

causes cross-attention shape mismatches inside the UNet.

This implementation detail is undocumented in the paper and official repository.

Key Insight

The same caption applies to every frame in the clip.

What AnimateDiff Does NOT Solve

AnimateDiff Limitation

AnimateDiff generates temporally coherent video, but provides no explicit spatial control.

- text-conditioned generation only
- no depth maps
- no edge maps
- no controllable scene layout

The Gap

AnimateDiff: $p(x_{\mathbf{1}:F} \mid y_{\text{text}})$

Our Work: $p(x_{\mathbf{1}:F} \mid y_{\text{text}}, c_{\text{depth}}^{\mathbf{1}:F})$

Temporally coherent and spatially controlled video generation

Related Work IV: MiDaS / DPT

Pseudo-Ground-Truth Depth via Dense Prediction Transformers

The Core Problem

WebVid videos contain:

- RGB frames
- text captions
- no depth sensor measurements

We therefore generate:

$$c_{\text{depth}}^{1:F}$$

using a pretrained depth estimator.

These depth maps are:

- pseudo-ground-truth labels
- model predictions; a form of self-supervised labeling

Why DPT Uses a ViT Backbone

DPT-Large uses global self-attention from the first layer.

Image → Patch Tokens → ViT-Large → Depth Map

- Patch size: 16×16
- $512 \times 512 \rightarrow 1024$ tokens
- Hidden dimension: $d = 1024$
- 24 transformer layers
- Global receptive field from layer 1

Unlike CNNs, every image patch can immediately attend to every other patch.

Pseudo-Ground-Truth Depth Maps

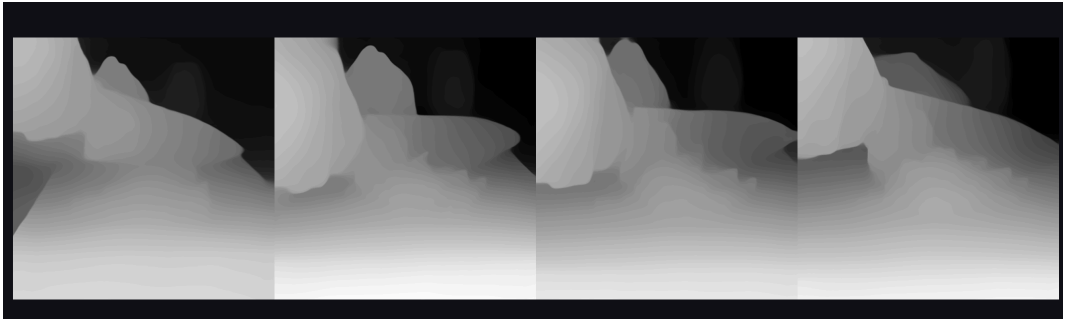


Figure: Depth Maps

Classifier-Free Guidance (CFG)

Amplifying Conditioning Signals at Inference

The Core Problem

The frozen UNet is already extremely strong at:

text-conditioned generation

As a result:

- Depth conditioning becomes weak; outputs mostly follow text priors

CFG Solution

Classifier-Free Guidance amplifies conditioning signals at inference without retraining.

Why 15% Dropout?

- Too low: model never learns unconditioned prediction

Conditioning Dropout During Training

With probability:

$$p = 0.15$$

the depth condition is removed:

$$c_{\text{depth}} \leftarrow \mathbf{0}$$

This forces the model to learn:

Conditioned Path

$$\varepsilon_\theta(z_t, t, \tau(y), c_{\text{depth}})$$

Unconditioned Path

$$\varepsilon_\theta(z_t, t, \tau(y), 0)$$

CFG at Inference

Dual UNet Evaluation

At each DDIM step, the UNet is evaluated twice.

Conditioned Prediction

$$\varepsilon_{\text{cond}} = \varepsilon_\theta(z_t, t, \tau(y), c_{\text{depth}})$$

Unconditioned Prediction

$$\varepsilon_{\text{uncond}} = \varepsilon_\theta(z_t, t, \tau(y_\emptyset), 0)$$

Guided Prediction

$$\varepsilon_{\text{guided}} = \varepsilon_{\text{uncond}} + s \cdot (\varepsilon_{\text{cond}} - \varepsilon_{\text{uncond}})$$

where:

$$s = \text{guidance scale}$$

This extrapolates beyond the conditioned prediction, strengthening conditioning effects.

Geometric Interpretation

$$\varepsilon_{\text{uncond}} \longrightarrow \varepsilon_{\text{cond}} \longrightarrow \varepsilon_{\text{guided}}$$

Guidance Scale Selection

Why We Use $s = 12.0$

Standard text-to-image diffusion commonly uses:

$$s \in [7, 9]$$

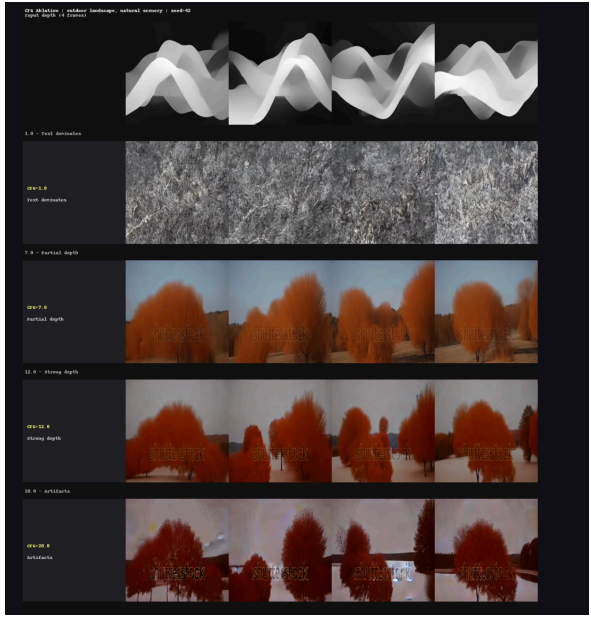
What I realised through trial and error: Our setting requires stronger guidance because

- text conditioning is heavily pretrained
- geometry must take over text priors

Guidance Scale Ablation

Scale s	Observed Behavior
1.0	Text dominates, depth barely visible
7.0	Partial depth adherence, layout mismatch remains
12.0	Strong depth adherence, distinct outputs per depth map
20.0	Artifacts, oversaturation, color collapse

CFG Abalation results



Novel Contributions: ConditionEncoder + Auxiliary Losses

Preventing Condition Encoder Collapse

Why a ConditionEncoder Is Needed

Original ControlNet conditioning was designed for:

$$c \in \mathbb{R}^{1\times H\times W}$$

Our video depth conditioning is:

$$c_{\text{depth}} \in \mathbb{R}^{B\times 1\times F\times 512\times 512}$$

This requires:

- spatial downsampling
- channel expansion
- temporal structure preservation

ConditionEncoder Objectives

Transform depth maps into latent-compatible tensors:

$$[B, 1, F, 512, 512]$$

$$\Downarrow$$

$$[B, 4, F, 64, 64]$$

Output becomes directly compatible with:

- latent diffusion tensors
- ControlNet injection layers
- AnimateDiff temporal modules

Condition Encoder Collapse

Observed Failure Mode

For two different depth maps:

$$c_1 \neq c_2$$

the encoder produced nearly identical normalized features:

$$\frac{f(c_1) \cdot f(c_2)}{\|f(c_1)\| \|f(c_2)\|} \approx 0.9999$$

Geometric Interpretation

Different depth maps collapsed to nearly the same point on the unit hypersphere in feature space.

Why Collapse Happens

The frozen UNet already performs strong text-conditioned generation.

Therefore:

- gradients reaching the ConditionEncoder are extremely small
- easiest optimization path is constant output

Result: low diffusion loss + zero feature diversity

Auxiliary Reconstruction Loss

Goal

Force the ConditionEncoder to preserve depth-specific information.

Decoder Architecture

A lightweight decoder reconstructs the original depth map:

$$f_\theta(c) \to g_\phi(f_\theta(c)) \approx c$$

Reconstruction Objective

$$\mathcal{L}_{\text{recon}} = \mathbb{E}\left[\|g_\phi(f_\theta(c_{\text{depth}})) - c_{\text{depth}}\|^2\right]$$

Why This Prevents Collapse

If all depth maps map to the same latent code:

$$f_\theta(c_1) = f_\theta(c_2)$$

then the decoder cannot reconstruct distinct depth maps.

Therefore reconstruction loss forces the encoder to preserve depth-specific structure.

Auxiliary Diversity Loss

Motivation

Reconstruction loss alone is not sufficient.

The encoder can still produce nearly constant spatial features.

Diversity Objective

$$\mathcal{L}_{\text{diversity}} = \mathbb{E}\left[\text{ReLU}\left(\tau - \text{Var}_{f,h,w}(\text{cond\_feat})\right)\right]$$

Interpretation

- high feature variance:
$$\text{Var} > \tau \Rightarrow 0 \text{ penalty}$$
- low feature variance:
$$\text{Var} \approx 0 \Rightarrow \text{maximum penalty}$$

Constant feature outputs are explicitly penalized, independent of diffusion loss behavior.

Total Training Objective

Combined Optimization Objective

$$\boxed{\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{diffusion}} + 0.5 \cdot \mathcal{L}_{\text{recon}} + 1.0 \cdot \mathcal{L}_{\text{diversity}}}$$

Loss Components

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}\left[\|\varepsilon - \varepsilon_\theta(z_t, t, \tau(y), f_\theta(c))\|^2\right]$$

$$\mathcal{L}_{\text{recon}} = \mathbb{E}\left[\|g_\phi(f_\theta(c)) - c\|^2\right]$$

$$\mathcal{L}_{\text{diversity}} = \mathbb{E}\left[\text{ReLU}(\tau - \text{Var}_{f,h,w}(f_\theta(c)))\right]$$

Failures faced During Training

Failure 1: Model not learning from Depth Maps

I faced no problem:

- diffusion loss decreased smoothly
- no NaNs or instability ; no GPU memory failures

But Inference showed Collapse

Different depth maps produced nearly identical outputs.

Same prompt + different geometry $\rightarrow$ lead to same generated landscape

How I debugged:

Cosine similarity between residuals:

$$\text{sim}(r_A, r_B) = 0.999$$

The Residuals had become nearly identical for all depth inputs.

The model learned to ignore depth conditioning entirely.

Root Causes of Condition Collapse

Root Cause 1

Dataset Mismatch

Training clips contained:

- people outdoors
- sports scenes
- cafes and crowds

Depth maps represented human silhouettes, not landscape geometry.

The model had learnt the easiest optimization:

ignore depth maps entirely

Root Cause 2

No Conditioning Dropout

The model never learned an unconditional pathway.

Therefore:

- CFG had nothing meaningful to amplify
- text conditioning took over the training
- depth conditioning was ignored

Model Collapse During Inference

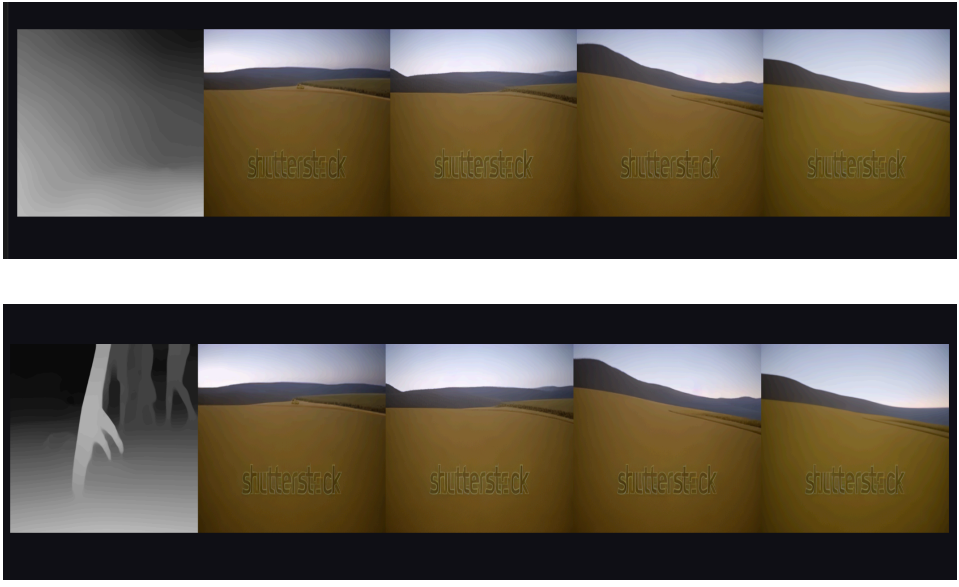


Figure: Mode collapse images

Fixing the Dataset: Landscape-Only Training

The Core Problem

Human-centric depth maps and landscape geometry were structurally incompatible.

Landscape Filtering

Included keywords:

- mountain
- aerial
- river
- valley
- drone
- terrain

Human Removal

Excluded keywords:

- person
- crowd
- portrait
- interview
- sport
- face

Quality Constraint

$$\text{depth std} > 0.18$$

Ensured meaningful spatial depth variation in every clip.

Impact of the Dataset Fix

What Improved

- depth maps became geometrically meaningful
- training supervision became consistent
- landscape layouts aligned with depth structure

What Still Failed

Despite the improved dataset:

$$\text{sim}(r_A, r_B) \approx 0.999$$

ConditionEncoder outputs remained nearly identical.

Critical Insight

The dataset problem and encoder collapse were independent failure modes.

Improving data quality alone was not sufficient.

Failure 2: Feature-Space Collapse

Dataset and CFG Were Fixed

Training was now stable and geometrically meaningful.

However, the cosine similarity diagnostic still showed severe collapse:

```
step 500: sim(clip1, clip2)=0.999978

step 500: sim(clip1, clip3)=0.999981

step 500: sim(clip2, clip3)=0.999974
```

What This Meant

Different depth maps produced feature vectors pointing in nearly identical directions.

Mountains, valleys, plains, and canyons all mapped to nearly the same region in feature space.

The encoder learned a:

fixed-bias solution

rather than meaningful geometric encoding.

Why Diffusion Loss Could Not Prevent Collapse

The Optimization Problem

The frozen UNet minimizes:

$$\mathcal{L}_{\text{diffusion}} = \mathbb{E}\left[\|\varepsilon - \varepsilon_\theta(z_t, t, \tau(y), f_\theta(c))\|^2\right]$$

Gradient Path

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \varepsilon_\theta} \cdot \frac{\partial \varepsilon_\theta}{\partial \text{residuals}} \cdot \frac{\partial \text{residuals}}{\partial f_\theta} \cdot \frac{\partial f_\theta}{\partial \theta}$$

Problem

The gradient must pass through the frozen UNet.

Because the pretrained UNet already performs strong text-conditioned generation:

- depth features weakly affect loss
- gradients reaching the encoder become extremely small
- the loss surface becomes nearly flat

The easiest optimization strategy becomes:

constant feature direction

Fix: Auxiliary Losses with Direct Gradient Paths

Reconstruction Loss

A lightweight decoder reconstructs the original depth map:

$$g_\phi(f_\theta(c)) \approx c$$

$$\mathcal{L}_{\text{recon}} = \mathbb{E}\left[\|g_\phi(f_\theta(c)) - c\|^2\right]$$

This creates a direct gradient path:

$$\text{Loss} \to g_\phi \to f_\theta$$

No frozen UNet involved.

Diversity Loss

Directly penalize low feature variance:

$$\mathcal{L}_{\text{diversity}} = \text{ReLU}\left(\tau - \text{Var}_{f,h,w}(f_\theta(c))\right)$$

This forces:

- spatially varying features
- depth-specific structure
- non-constant feature maps

Constant outputs receive maximum penalty.

Final Objective

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{diffusion}} + 0.5\mathcal{L}_{\text{recon}} + 1.0\mathcal{L}_{\text{diversity}}$$

ConditionEncoder Similarity During Training

Pairwise Cosine Similarity

Training Step	Similarity
500	0.9948
1000	0.9943
2000	0.9925
5000	0.9870
10000	0.9820
20000	0.9780
30000	0.9736

Interpretation

Feature vectors gradually moved away from collapse and began encoding distinct directions.

$$0.999 \rightarrow 0.97$$

showed meaningful diversification in feature space.

Feature Diversity After Auxiliary Losses

What Changed

The encoder now produced genuinely different feature vectors for different depth maps.

- mountains encoded differently from valleys
- canyons encoded differently from flat terrain
- spatial geometry now influenced feature direction

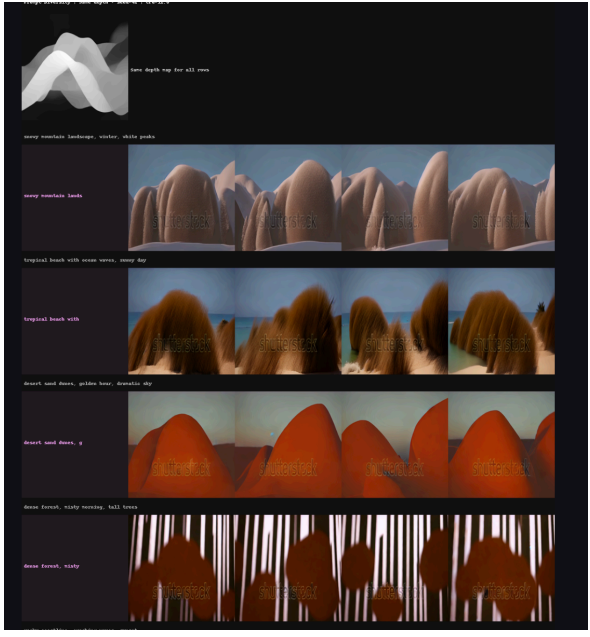
Outcome

Depth structure was no longer treated as a weak bias term.

The ConditionEncoder learned a meaningful geometric representation space.

Final Improved Results

Prompt Abaltion



Future Work

1. Longer Video Generation

- Current system generates only 4 frames
- AnimateDiff motion modules were originally trained on 16-frame clips
- Longer sequences would better utilize temporal modeling capacity

2. Multi-Condition Control

- Current model uses depth conditioning only
- Future systems could combine:
  - depth + edges
  - depth + pose
  - depth + segmentation

3. Expanded Scene Categories

- Current training focused exclusively on landscape clips
- Generalization to other domains remains unexplored:
  - indoor scenes
  - urban environments
- Future work may require category-balanced training or specialized conditioning modules

4. Higher Resolution Video

- Current system operates in 512×512 latent space; Extending to higher resolutions would improve fine geometric detail
- Requires memory-efficient attention and scalable temporal modules

Thank You

Questions?

"The future depends on some graduate student who is deeply suspicious of everything I have said."

— Geoffrey Hinton