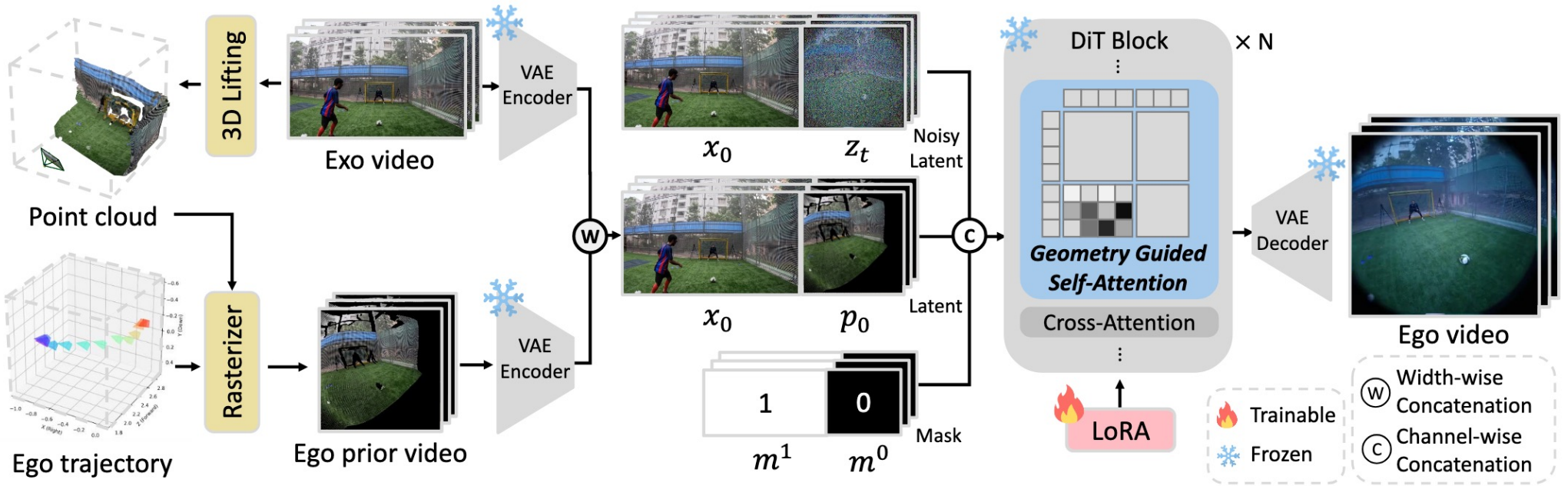

# EgoX: Deep Dive

Egocentric Video Generation from a Single Exocentric Video [project url]

Taewoong Kang, Kinam Kim, Dohyeon Kim, Minho Park, Junha Hyung, Jaegul Choo [slides by Leo Ho]

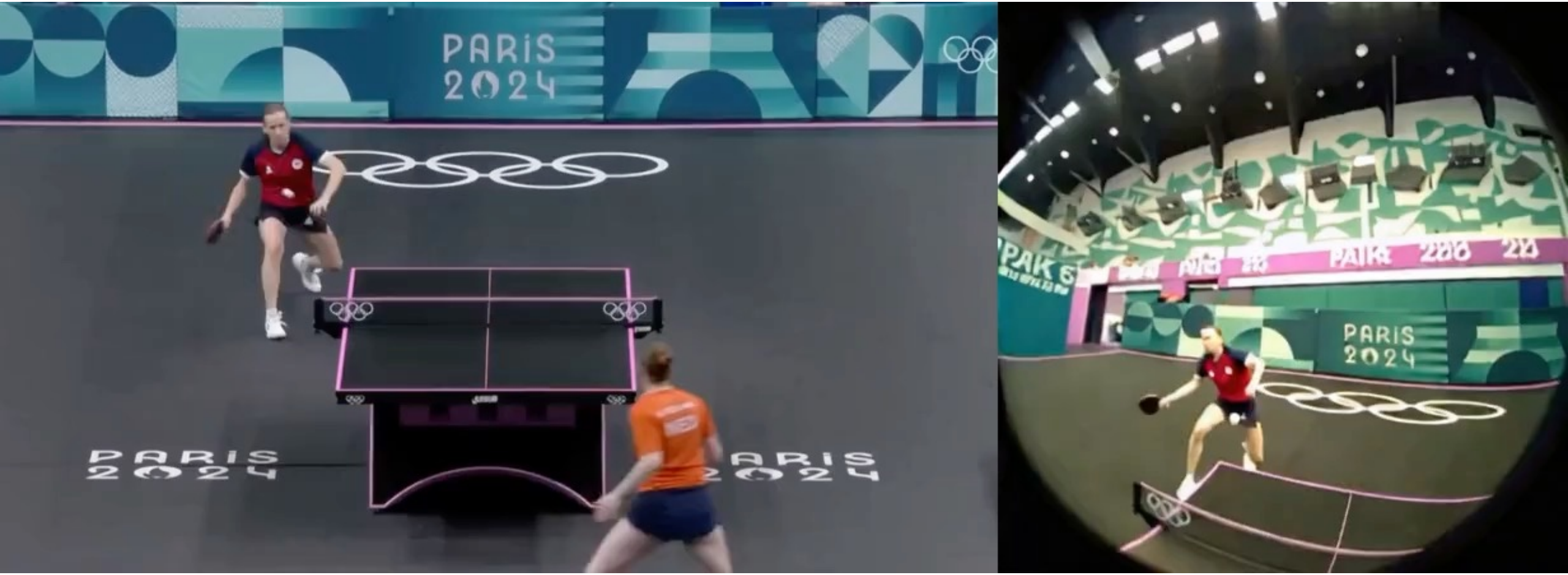
# EgoX




A pipeline that transforms exocentric (3rd person) video into egocentric (1st person) using a novel conditioning technique and trainable LoRA for Wan2.1-Inpaint

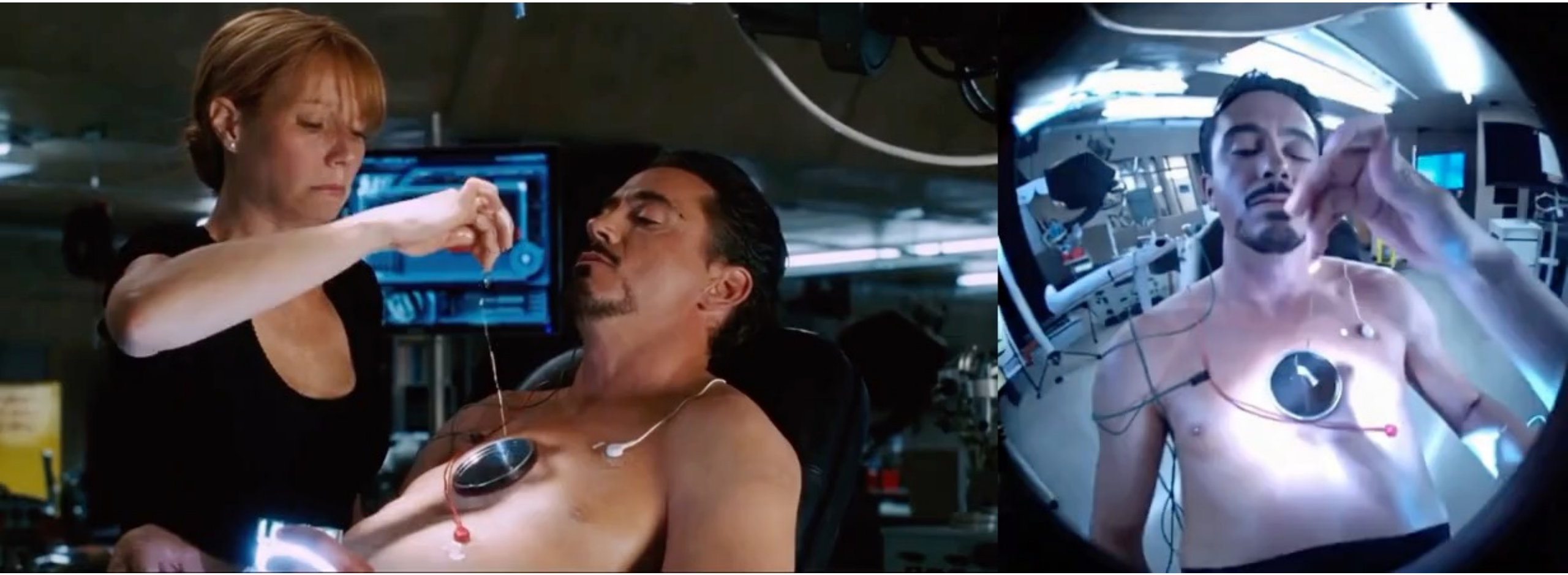
# Demo

# Demo



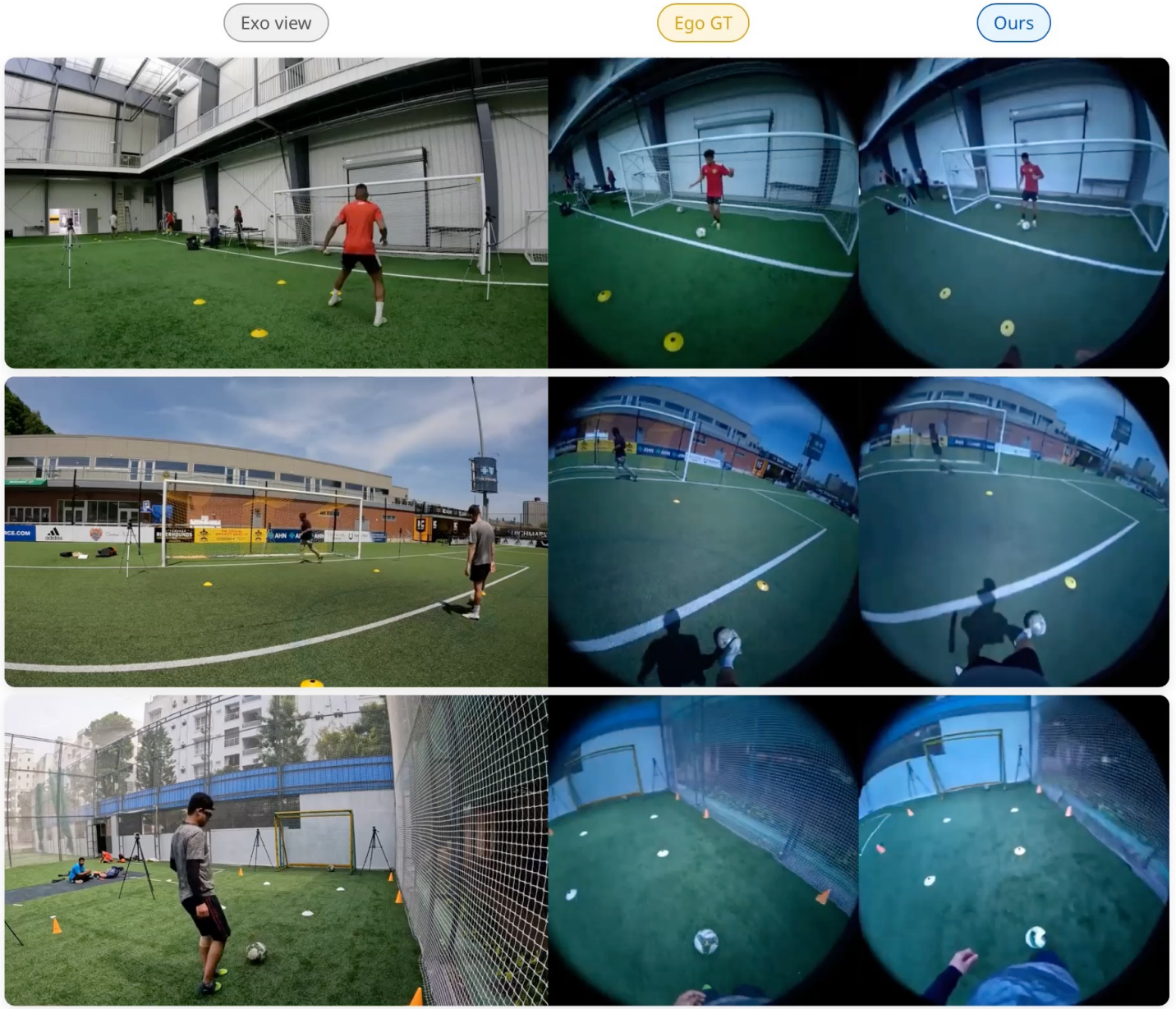

Some artifacts (e.g. “disintegrated” balls, double balls present)

# Demo



# Demo

# Comparison vs GT

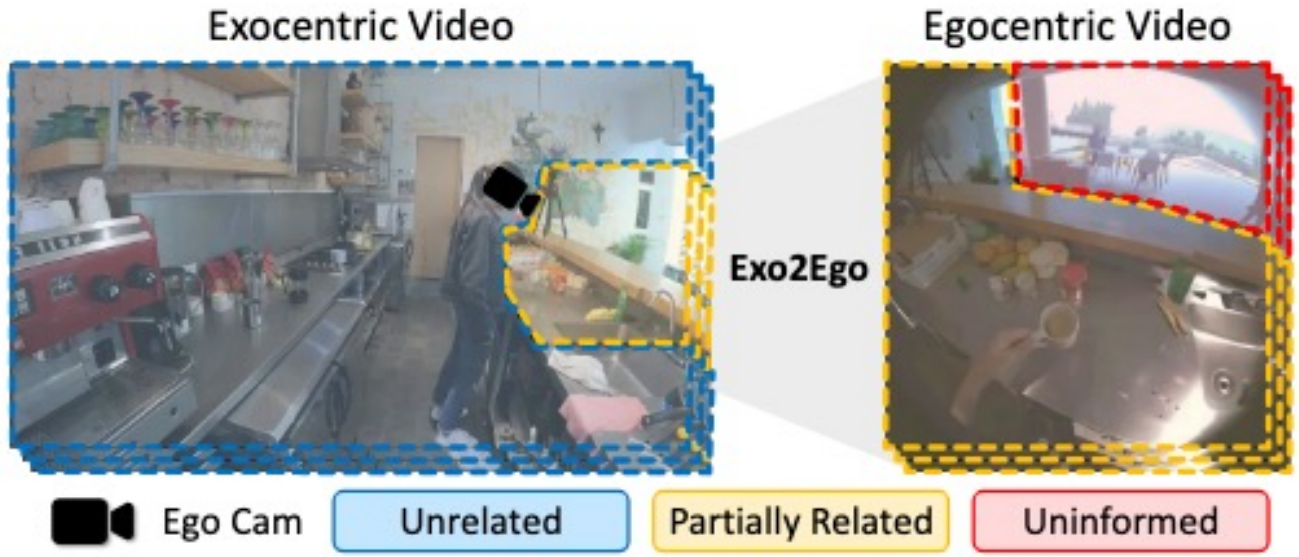


# Use Cases

- Create new angles of movie and TV scenes
  - Can save VFX effort to finish one scene and just “rotate” to get a new view
- Generate ego humanoid training data from exo Internet videos
- For athletes to better visualize their opponent before the real match
- Accident reconstruction / forensics from CCTV angles

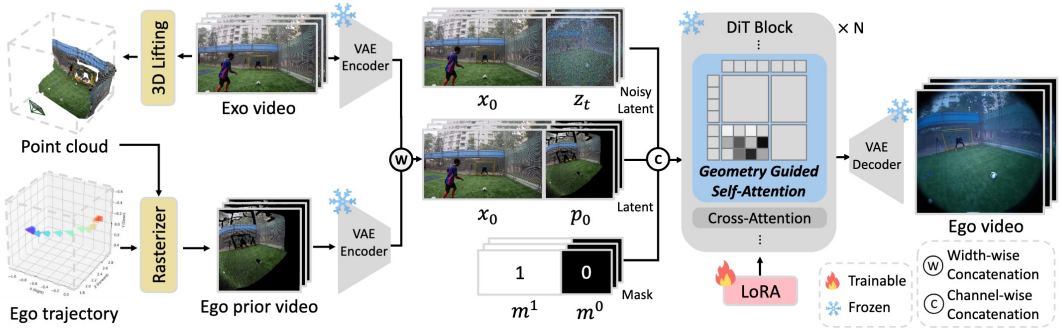
# What makes the problem challenging?

- Drastically alters visible field of view
- Requires extreme camera pose translation
- Only a small portion of exo video is related to ego view
- Uninformed parts must be inpainted



# Why does the model work?

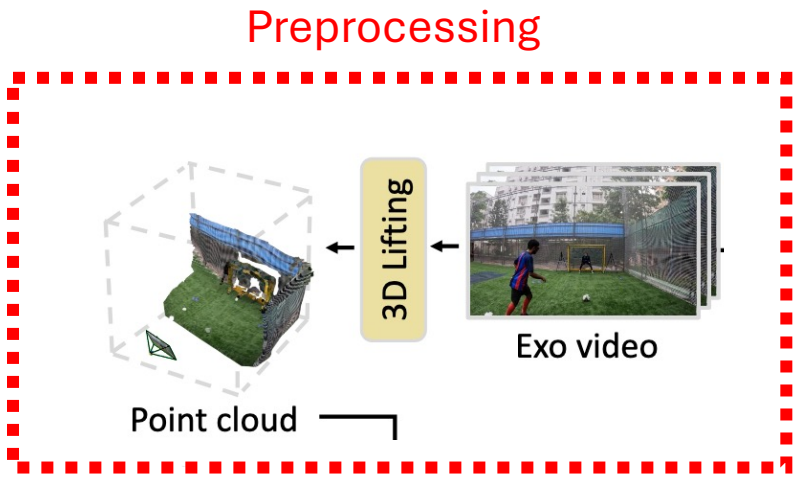
1. Using point clouds as hints for generation
2. Good attention formulation that directs model attention to relevant areas in exo video
3. Wan2.1 is really good at inpainting!
4. Dataset of exo-ego pairs help in training LoRA



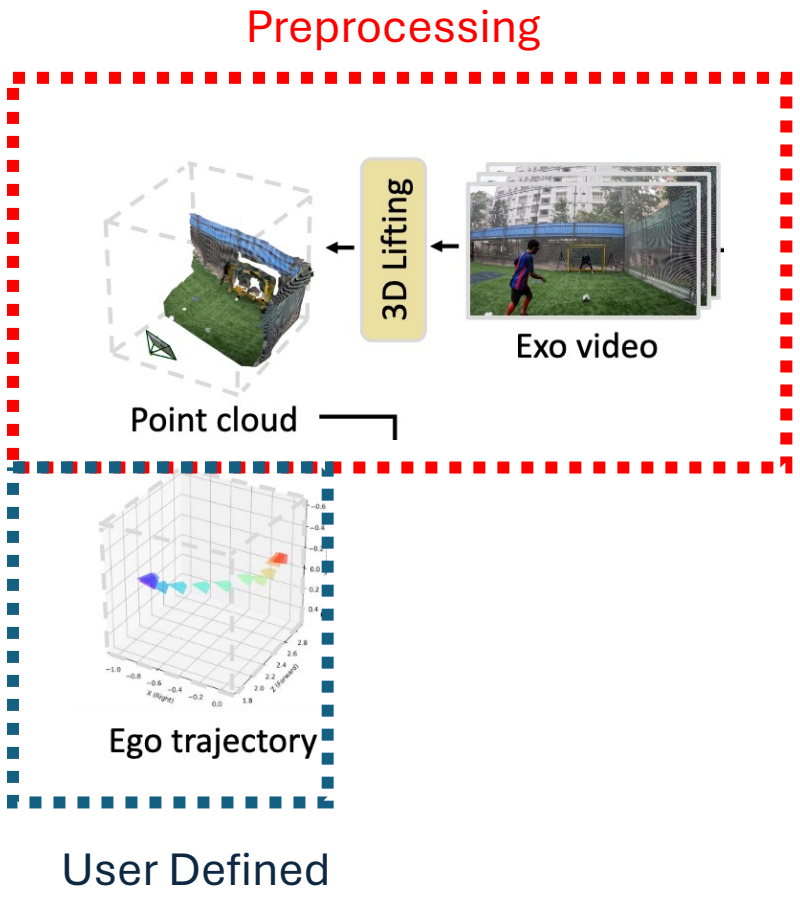
# Inference Pipeline

Exo video

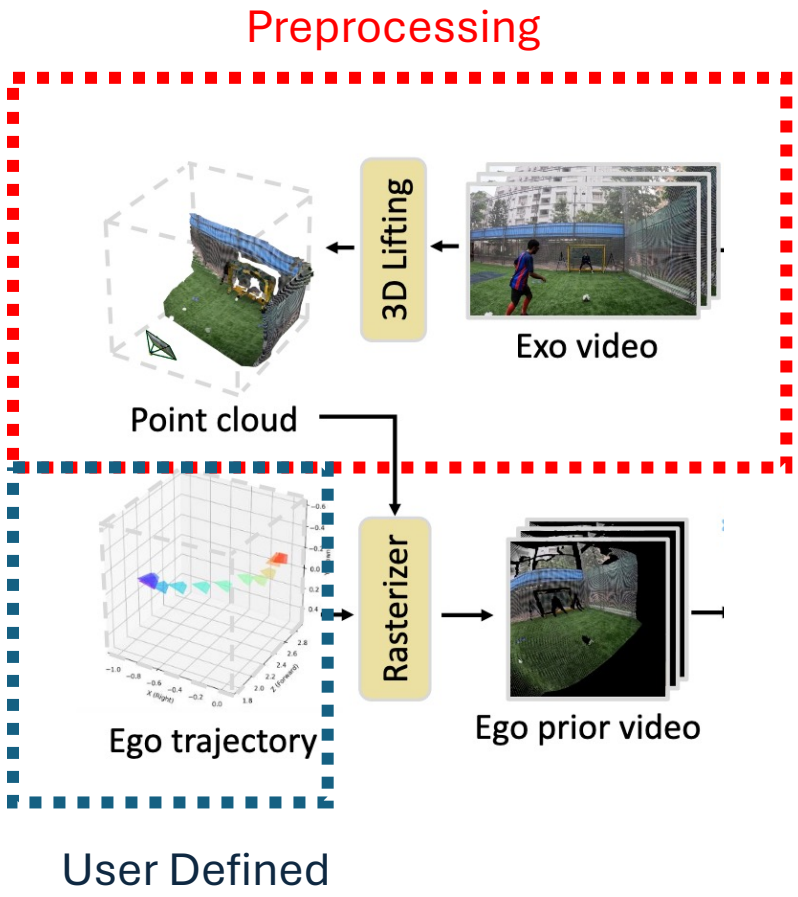
# Inference Pipeline



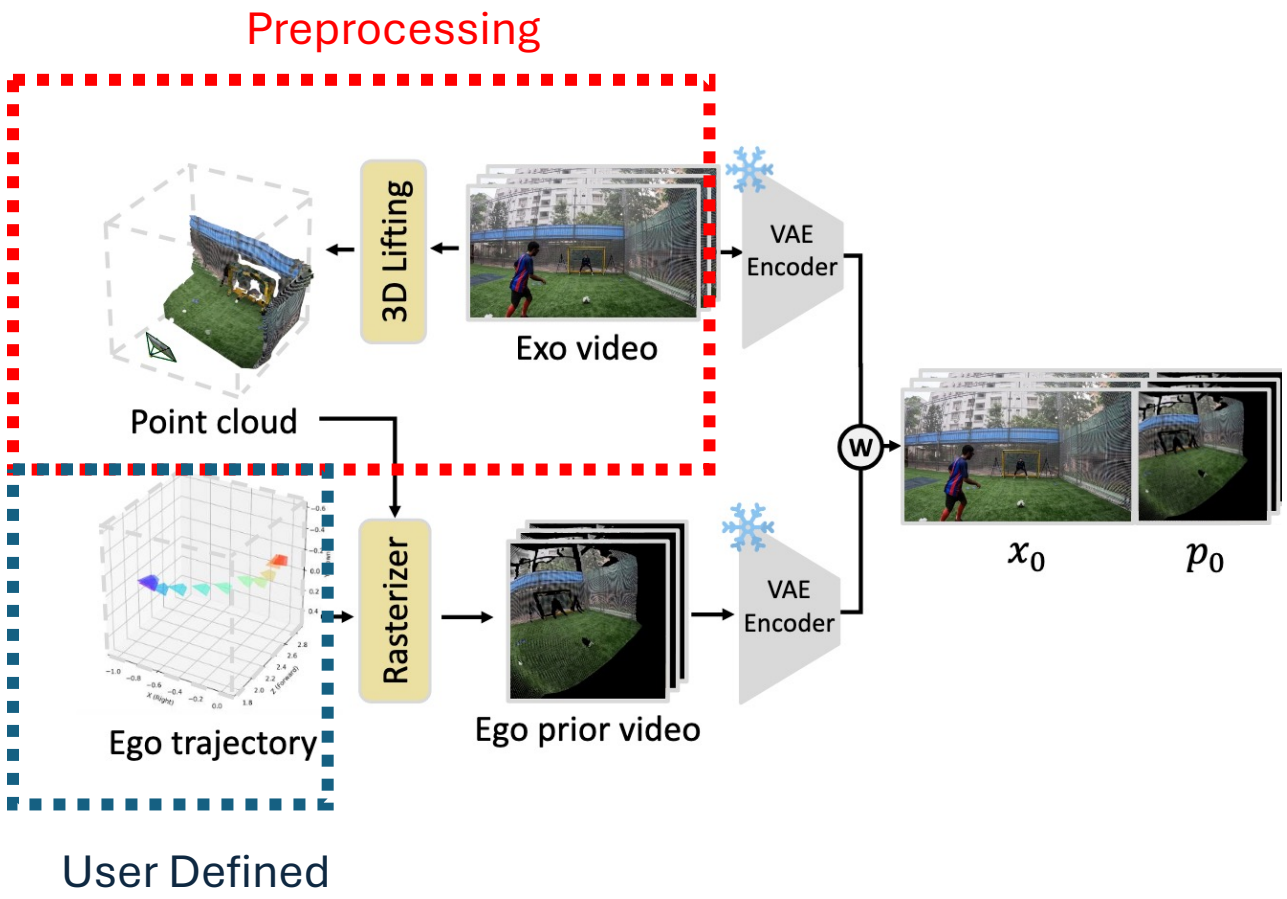
# Inference Pipeline



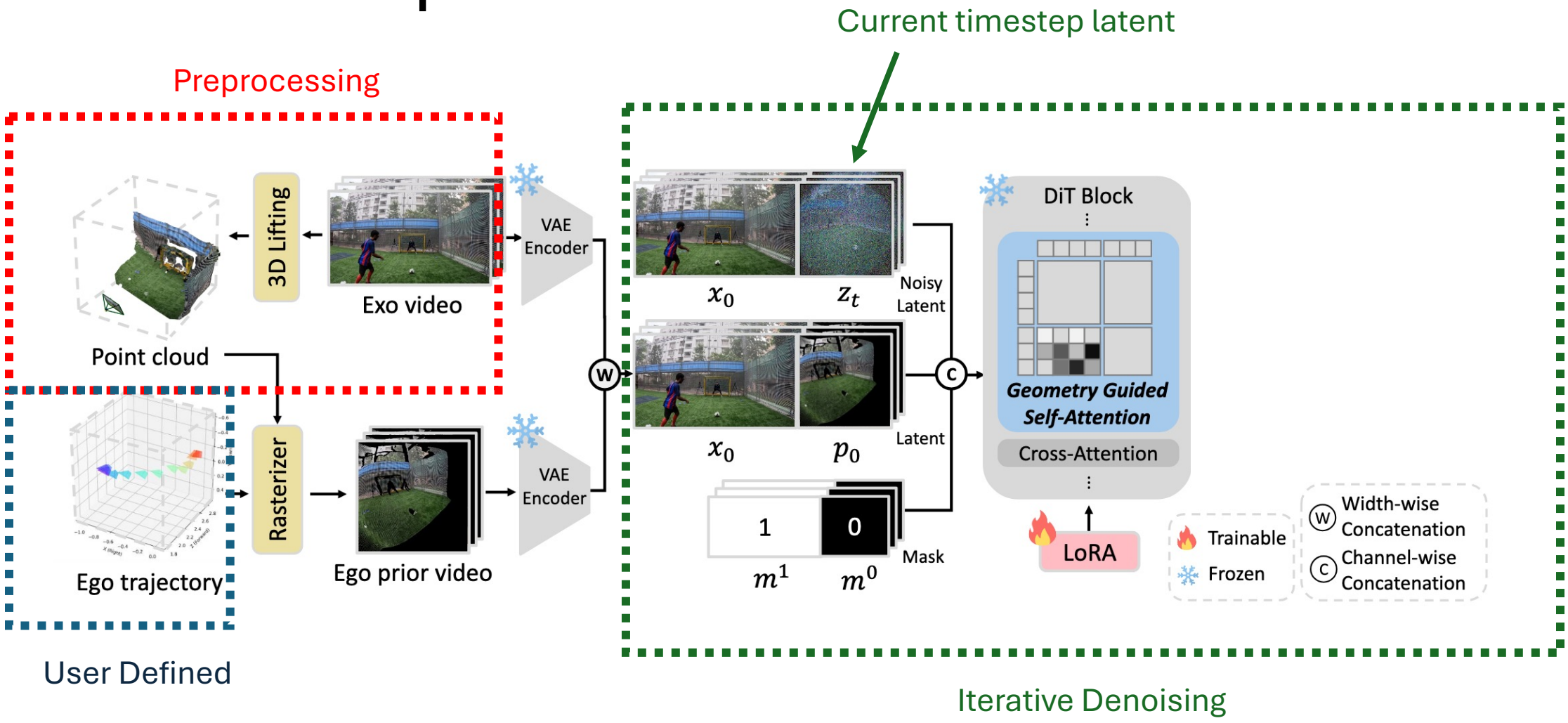
# Inference Pipeline



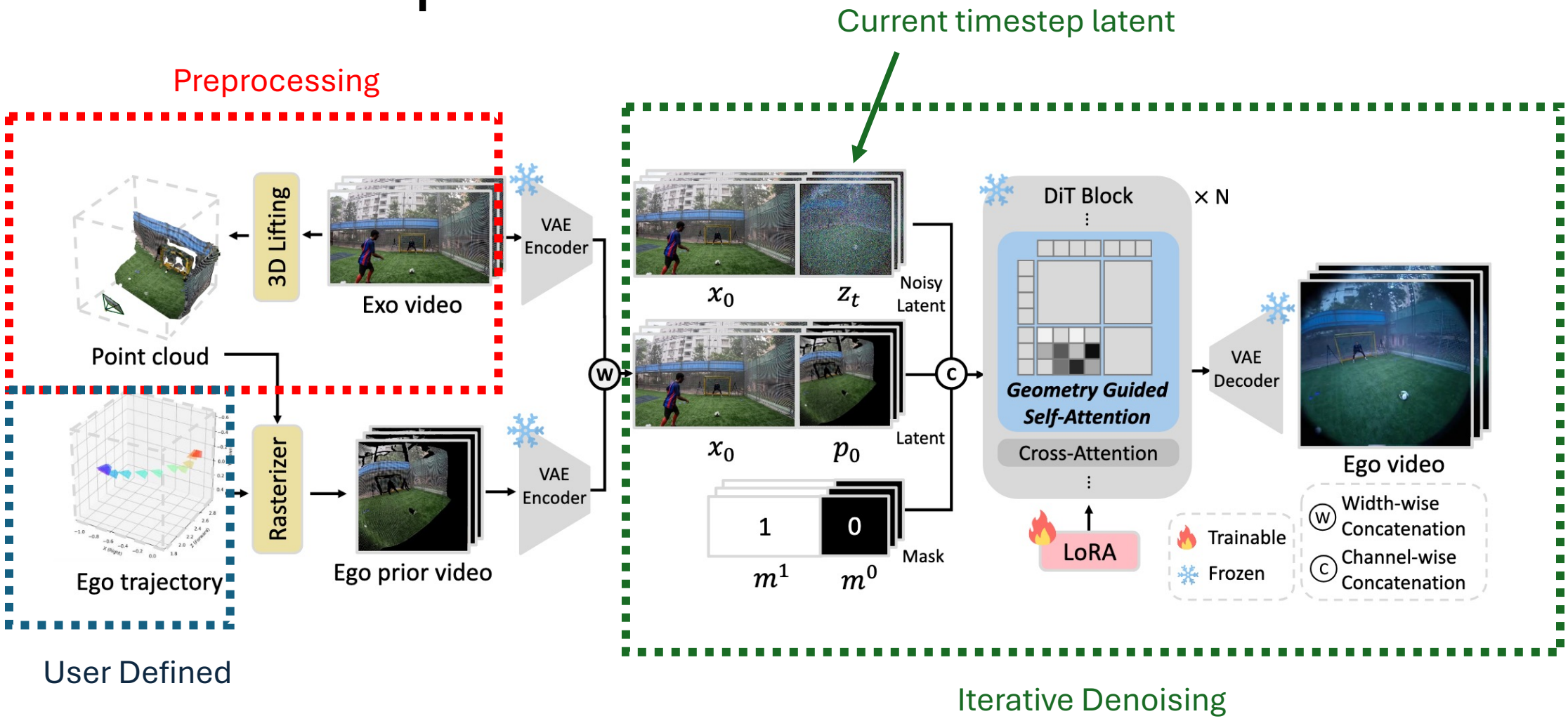
# Inference Pipeline



# Inference Pipeline

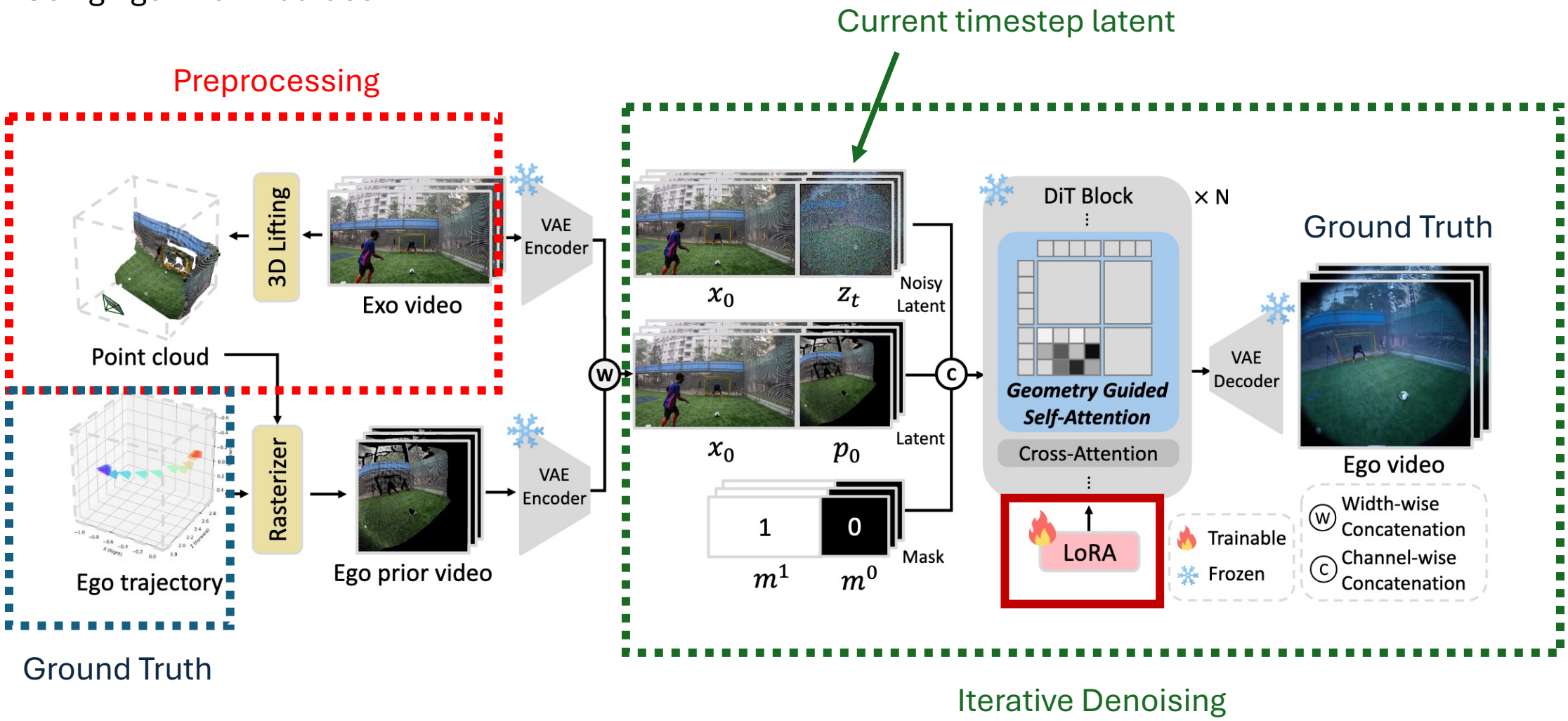


# Inference Pipeline



# Training Pipeline

Using Ego-Exo4D dataset



# Ego-Exo4D Dataset

Used 4000 clips from Ego-Exo4D (Train/Test: 3600/400), plus 100 additional clips to assess generalization

× 5035 (# takes) × 740 (# participants) × 123 (# sites) × 1286.3 (# ego+exo hours)

| | | | | | |
|---|---|---|---|---|---|
| Cooking | × 678 | × 173 | × 60 | × 564.13h |
| Music | × 276 | × 59 | × 8 | × 180.08h |
| Soccer | × 282 | × 78 | × 14 | × 66.97h |
| Health | × 397 | × 122 | × 24 | × 114.5h |
| Basketball | × 910 | × 113 | × 5 | × 78.01h |
| Dance | × 728 | × 93 | × 7 | × 106.57h |
| Bike Repair | × 363 | × 32 | × 8 | × 82.15h |
| Rock Climbing | × 1401 | × 98 | × 2 | × 93.90h |


Bouldering in Minneapolis


Jump Shooting in Chapel Hill


Layups in Burnaby


CPR in Queenstown


Salsa in Bogotá


Soccer in Pittsburgh


Replacing a Bike Tube in Atlanta


Removing a Wheel in Menlo Park


Guitar Freeplay in Philadelphia

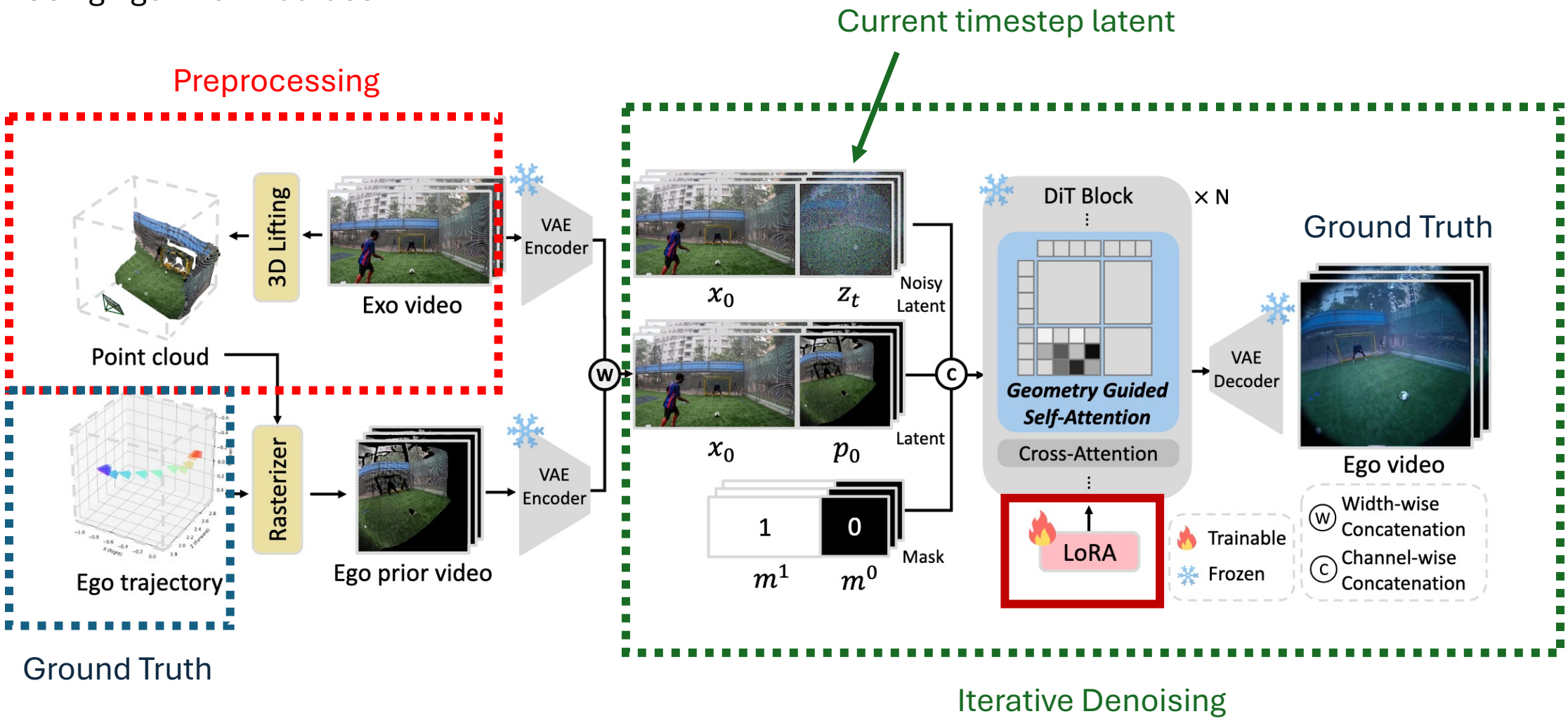

Piano Playing in Bloomington


Making Chai in Hyderabad

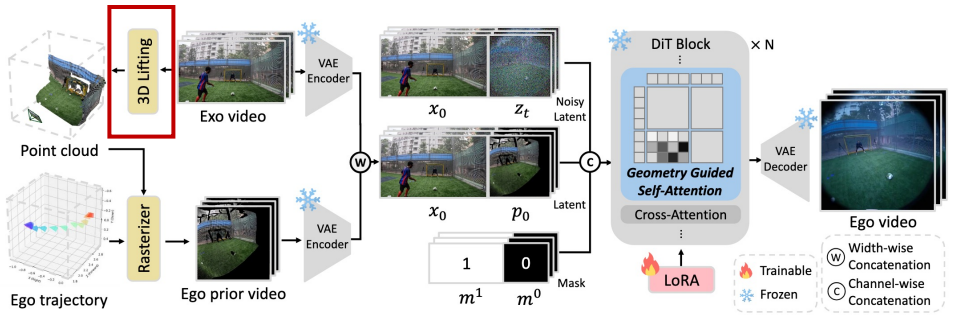

Salad Preparation in Tokyo

# Training Pipeline

Using Ego-Exo4D dataset

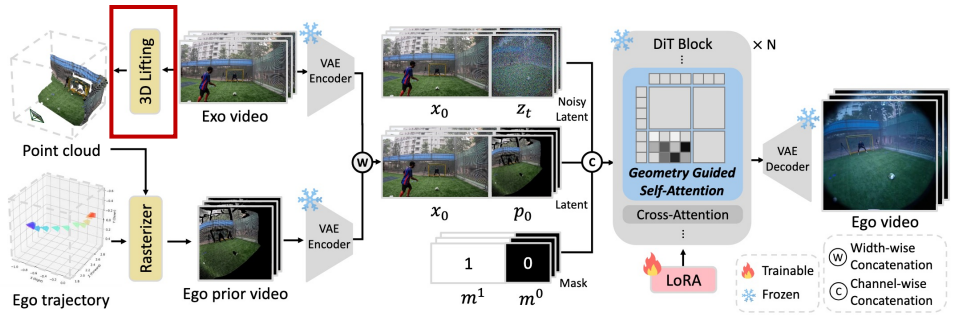


# 3D Lifting of Exo view




- If you can estimate the camera intrinsics and a depth map, you can generate a 3D point cloud
- Camera intrinsics: **ViPE** paper (Video Pose Engine)
  - Tracks correspondence via optical flow and feature matching
  - Uses learned priors in optimization
  - Refer to ViPE main paper for details
- Depth map is more complicated

# 3D Lifting of Exo view

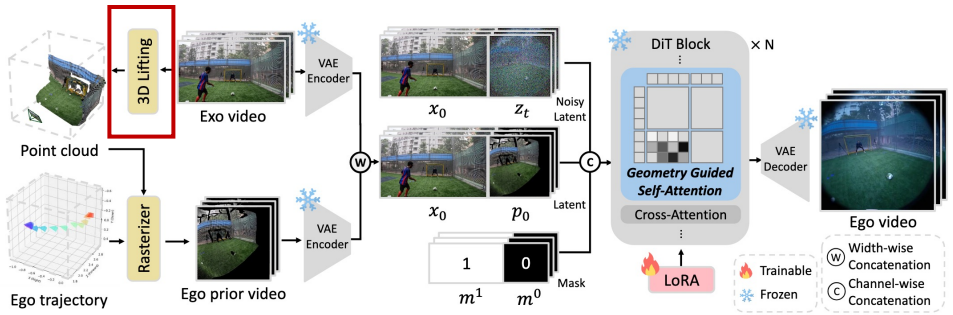



- Estimators of depth is based on **disparity**, i.e. 1/depth
- Only estimate on static background regions to avoid point cloud artifacts
- Two estimators of disparity used:

1. Image-based estimator *(MoGE-2)*
   yields exact estimations but temporally jittery

2. Video-based estimator *(Video Depth Anything)*
   yields relative estimations (affine invariant) but temporally stable

# 3D Lifting of Exo view




- Disparity from image estimator: $D_m$
- Disparity from video estimator: $D_v$
- Optimize $\alpha, \beta$ for every frame,

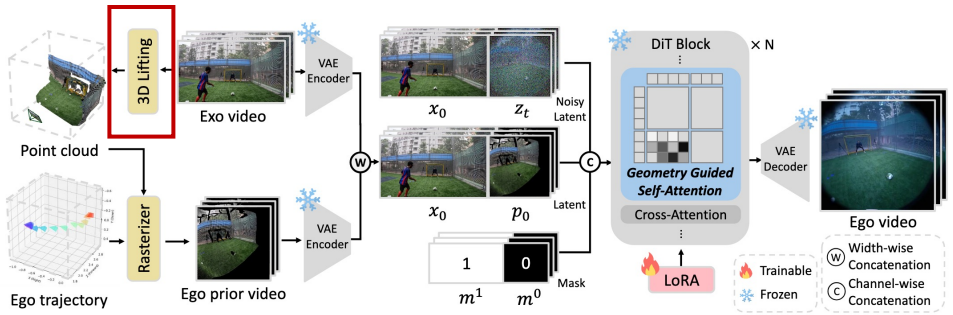
$$\min_{\alpha,\beta} \sum_{p \in \text{pixels}} \rho\big(\alpha D_v(p) + \beta - D_m(p)\big)$$ , where $\rho$ is a loss fn. (e.g. Huber/L1)

- Update $\alpha, \beta$ by momentum across frames,

$$\alpha_f \leftarrow \mu\alpha_{f-1} + (1-\mu)\alpha_f^{\text{optimal}}$$

$$\beta_f \leftarrow \mu\beta_{f-1} + (1-\mu)\beta_f^{\text{optimal}}$$

# 3D Lifting of Exo view

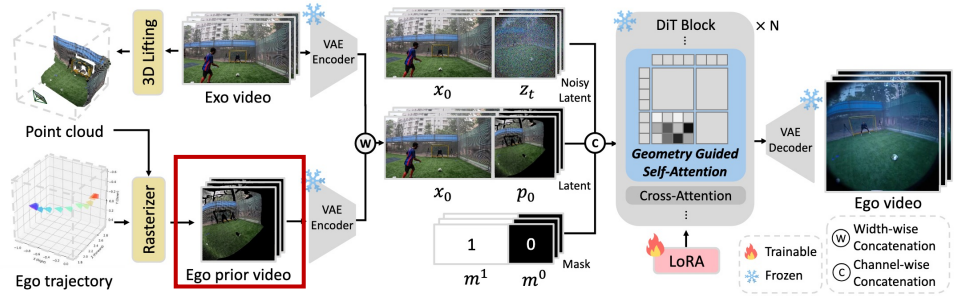



- Final depth per pixel

$$\frac{1}{\alpha D_v(p) + \beta}$$

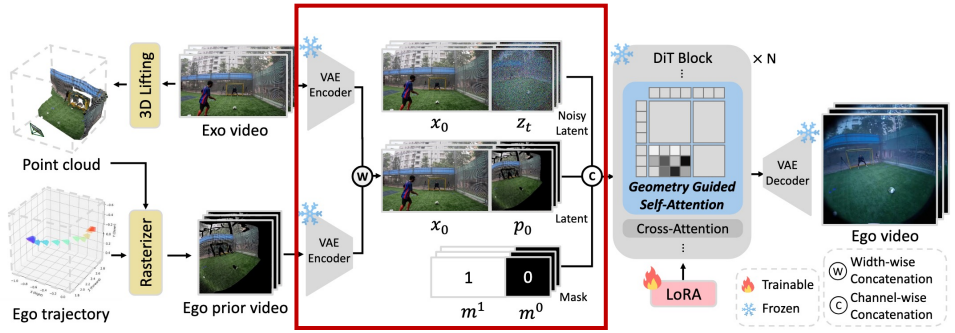
- Use un-projection from camera origin to obtain point cloud

# Ego prior video

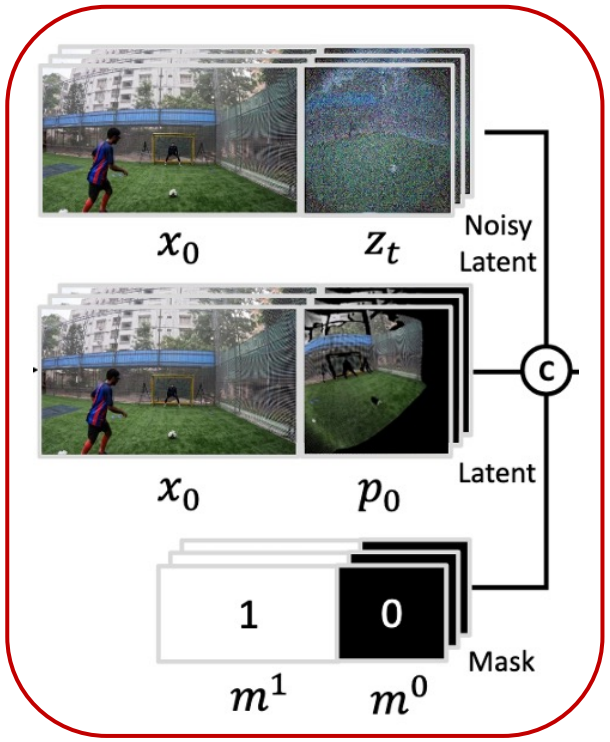


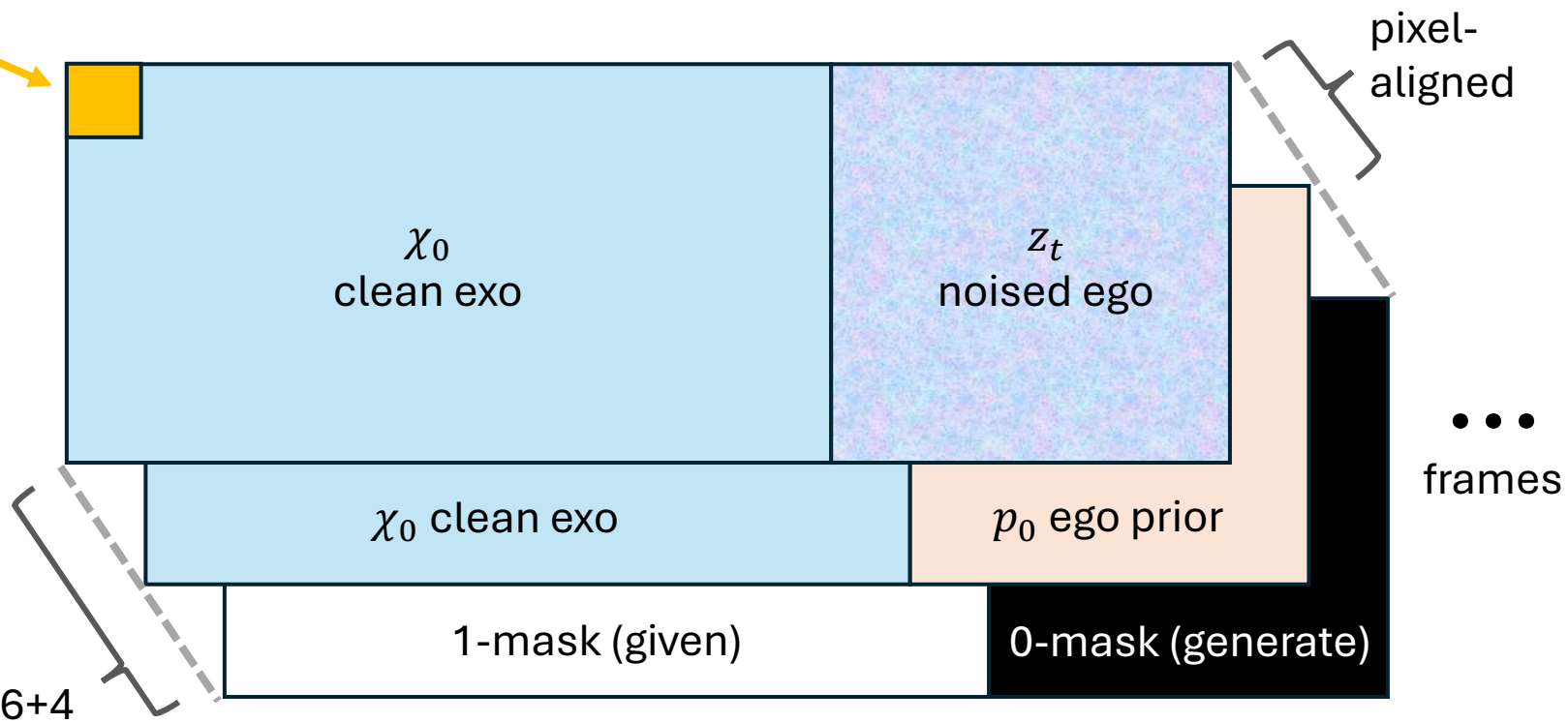

- A square RGB video that is obtained by rendering the point cloud along the user-defined / GT ego trajectory
- Rendered with fisheye camera (intrinsics matched to training dataset)

# Condition Input

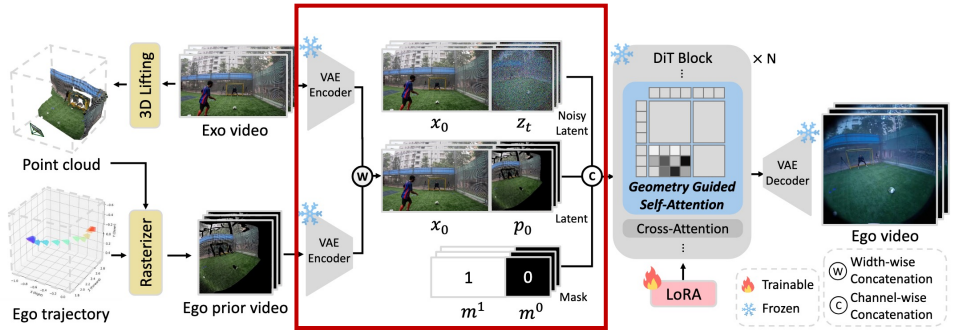



- Each VAE latent corresponds to 4 frame x (8 x 8 pixel) area, outputs 16-dim vector



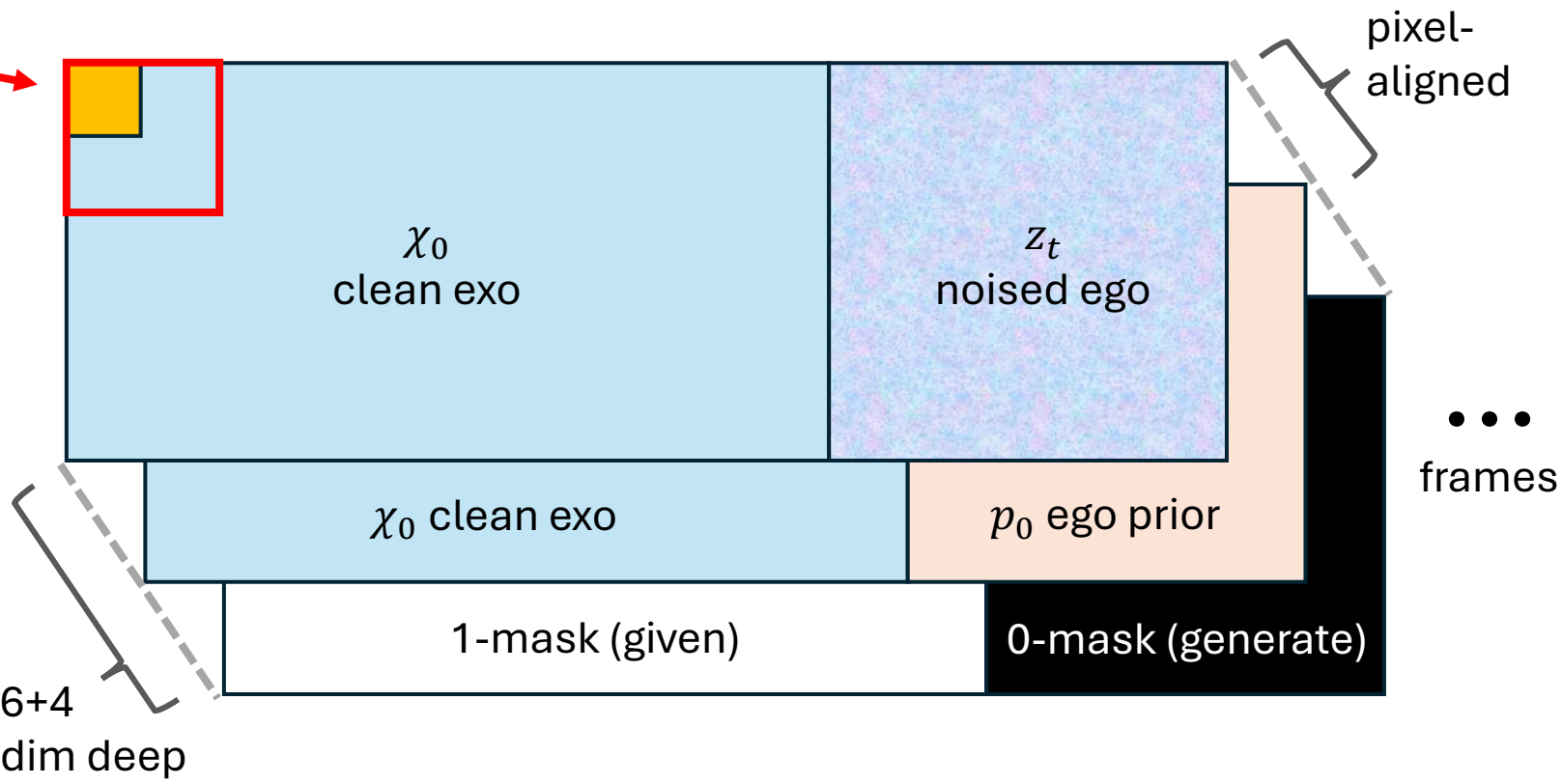
# Tokenization



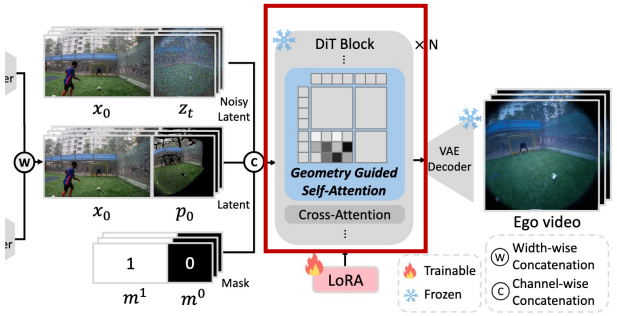

- Each transformer token corresponds to a 2 x 2 latent patch,

  i.e. 4f x16x16

- Position embedding (f, h, w) with RoPE
- Format aligns with Wan2.1-Inpaint tokens
- Tok count: ≈28K



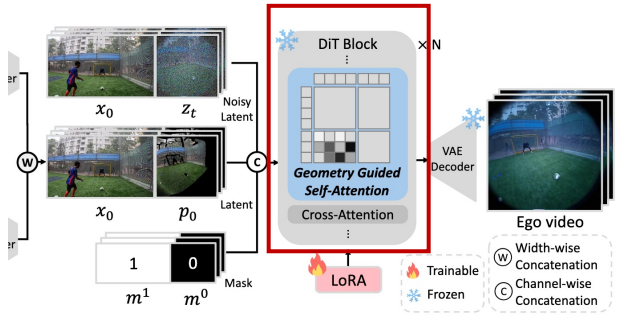
# Geometry Guided Self-Attention (GGA)




- Goal: During self-attention, we want to pair patch tokens of **ego view** with relevant patch tokens from **exo view**

- Observation: The two red vectors below have the same direction in world space; can we use this information for pairing?

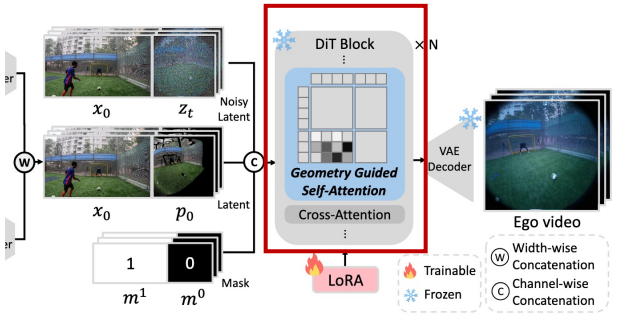
# Geometry Guided Self-Attention (GGA)




- For every token in the **exo** section, because corresponding points $P_c$ in point cloud is known, can calculate avg. direction vector $v$ between **ego** camera origin and $P_c$ in world space

- Then, match an **ego query** token with the **exo key** tokens with most similar direction vector $v$

# Geometry Guided Self-Attention (GGA)



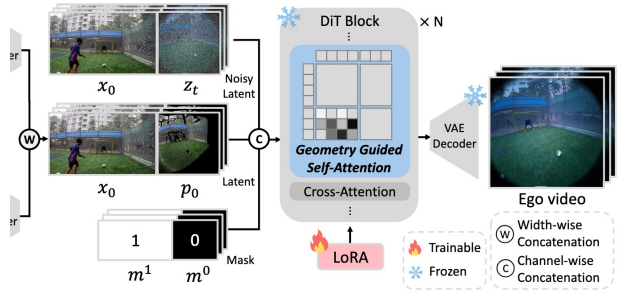

- Modified attention logits:

$$s'_{m,n} = s_{m,n} + \log\left(g\left(\widehat{q_m}, \widehat{k_n}\right) \cdot \lambda_g\right)$$

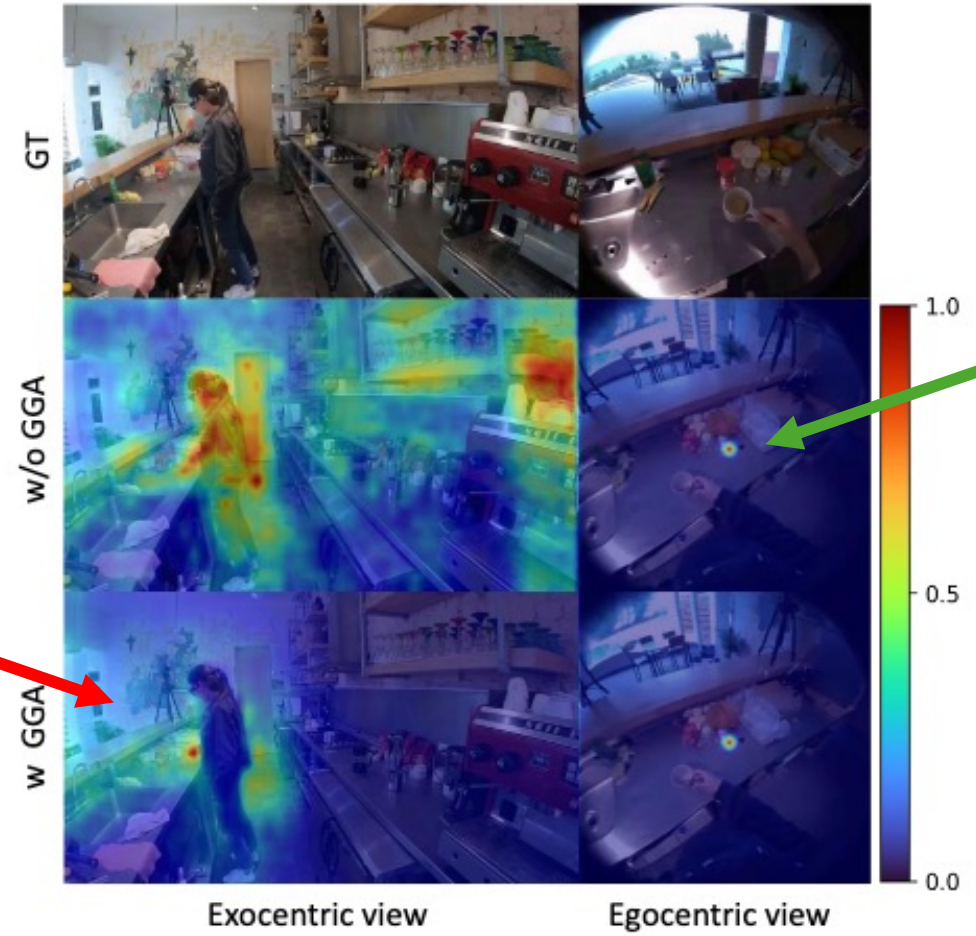
$$g(a,b) = \cos\_\mathrm{sim}(a,b) + 1$$

where $s_{m,n}$ is the standard attention logit and $\lambda_g$ is a hyperparameter

# Geometry Guided Self-Attention (GGA)



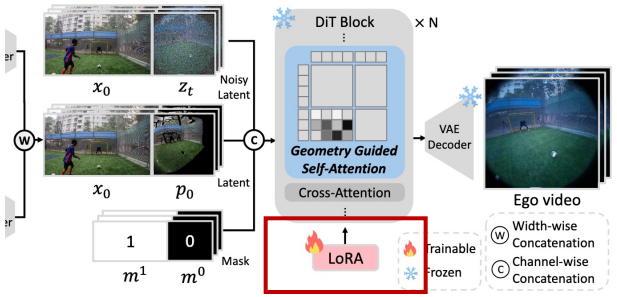

## Results




**With GGA,** an ego query patch attends only to the exo keys in front of the subject

**Without GGA,** an ego query patch is attended by many irrelevant exo keys

# LoRA




- Only trains the attention projections $W_q, W_k, W_v$
- Rank 256, higher than usual for a LoRA (8-64)
- Training time (batch size 1, 14B params): 1 day on 8 x H200

# Ablations


good perspective alignment

hallucinations

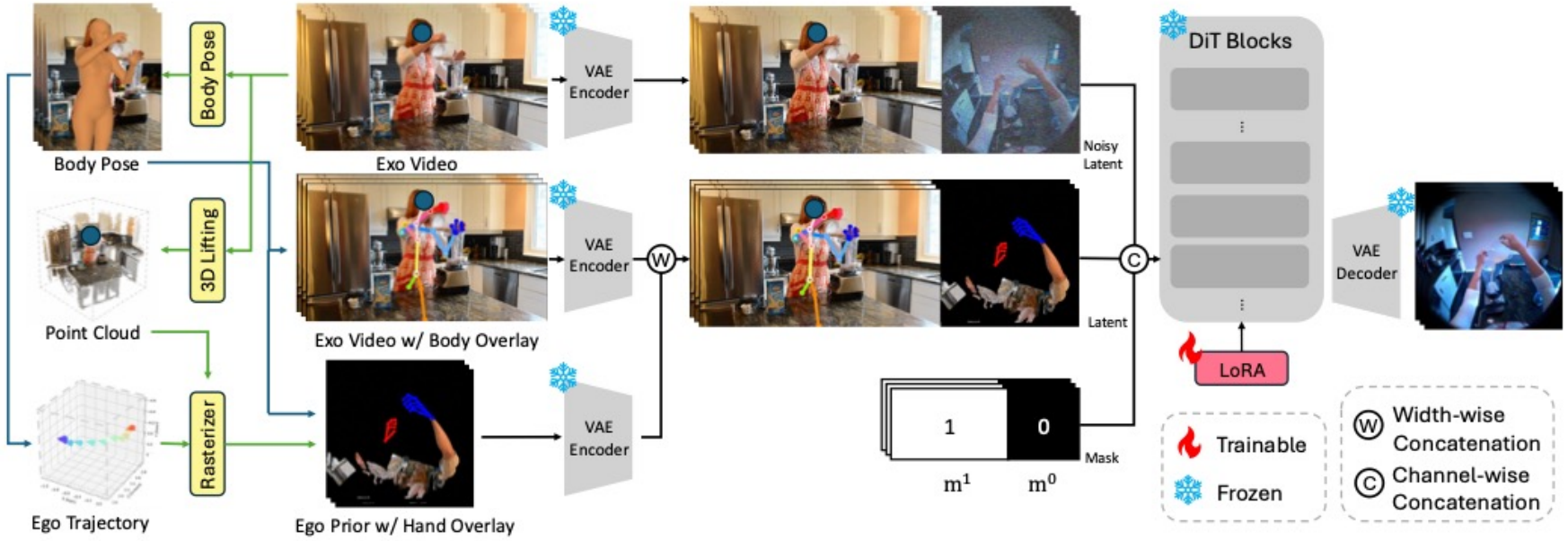
exo latent predicted noise allowed to accumulate

# Limitations

- The egocentric camera trajectory is given, not estimated
  - Mostly an engineering problem
- Weak priors for dynamic objects, only relies on GGA
  - e.g. humans, vehicles, pets, balls
- GGA formulation only allows an ego frame to look at ONE corresponding exo frame, unable to reference an object's information across frames
- Coupling of scene and view generation, lots of details gets recomputed across frames repeatedly, requiring large models
  - Full scene pre-computation can be done first

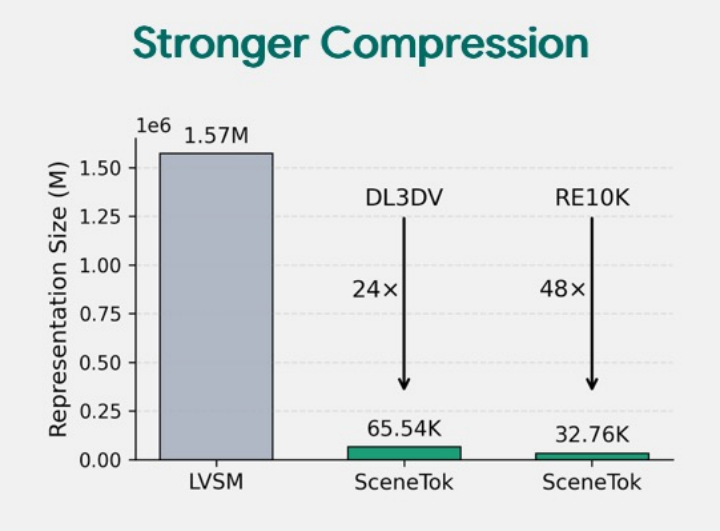
# EgoExo-WM

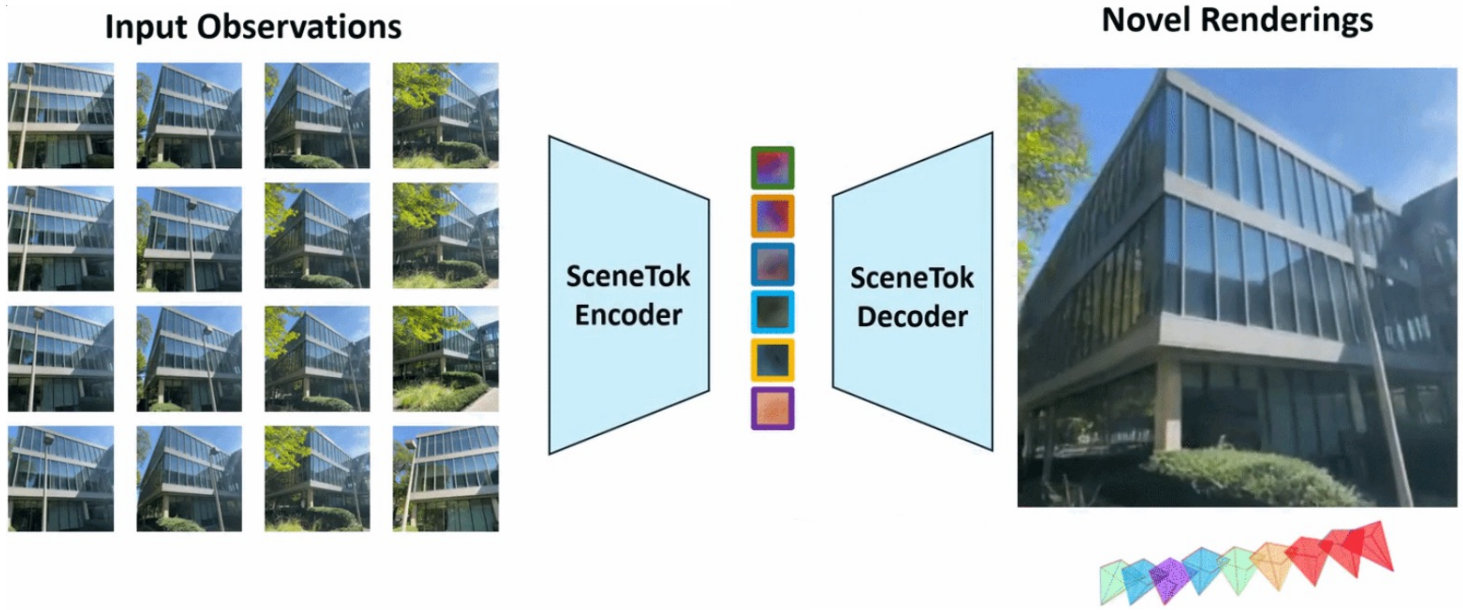
- Addresses the “weak prior” for human body and hands
- Very similar to EgoX pipeline



# SceneTok (CVPR 2026)

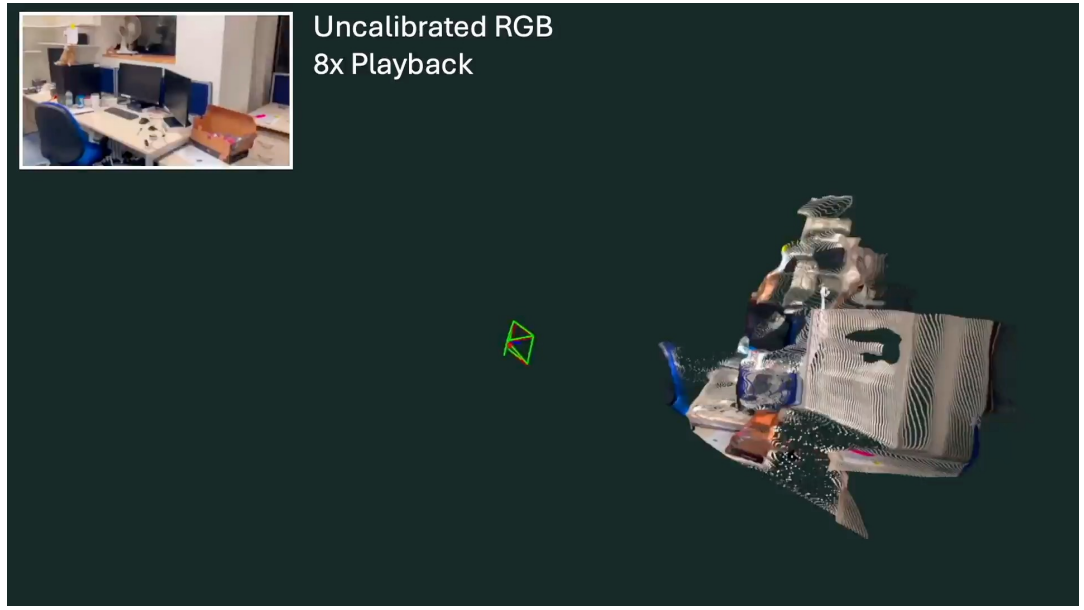
- Highly compressed, token representations for 3D scenes
- Decouples scene generation from view rendering
- Fast: 192 views synthesized in 5 seconds on RTX 4090



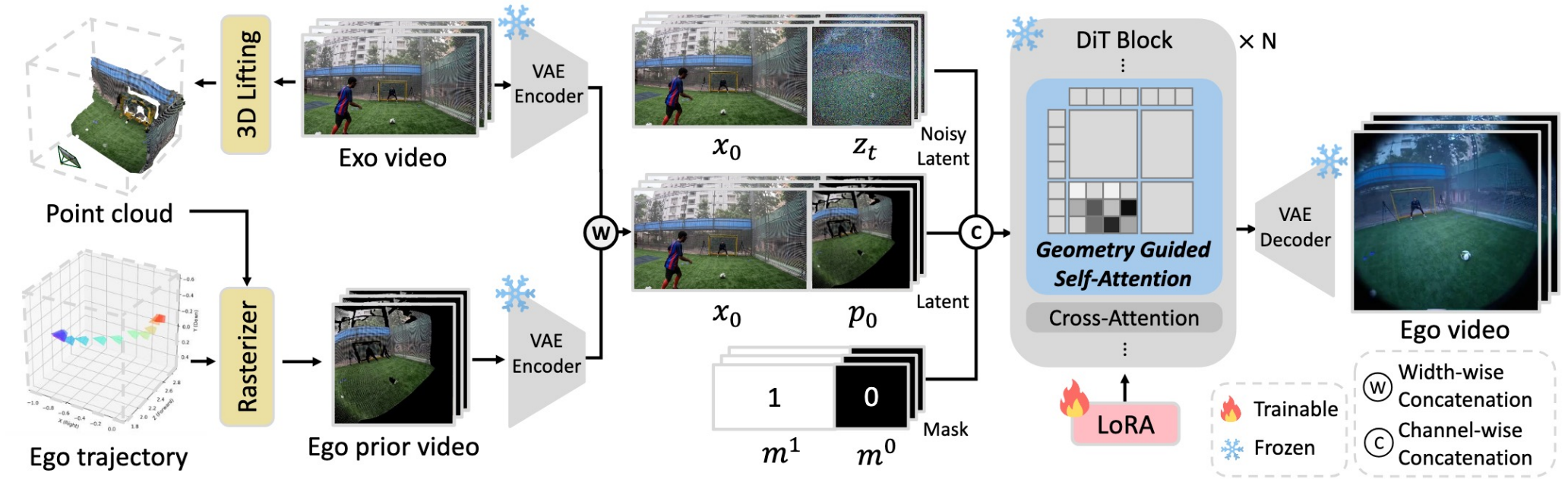
# Extensions

- Streaming generation
  - Current pipeline limited by offline point cloud creation
  - Idea: use visual SLAM methods, e.g. MASt3R-SLAM
- Use Gaussian Splats instead of point clouds for more expressive 3D scene priors
  - e.g. SplaTAM, MonoGS
- Use gradient masking for inpainted areas



# Extensions

- Instead of GGA only referencing exo pixels, attend to specially created key tokens that contain information on the object
- Extend to use in VR
  - Native generation in 3D space and render
  - Problem: cannot utilize pretrained knowledge of large video models
- Modify LoRA training methodology
  - Similar to DreamFusion (score distillation sampling)
  - Vary camera location and generate samples, minimize geometry consistency loss between output vs expected




# Thanks for listening

EgoX: Deep Dive

[slides by Leo Ho]