GEM: A Generalizable Ego-Vision Multimodal World Model for Fine-Grained Ego-Motion, Object Dynamics, and Scene Composition Control

Supplementary Material

7. Frequently Asked Questions

Q1: Will you share the code and dataset publicly?

All codes and model checkpoints are publicly available at our GitHub repo including all scripts used for pseudo-labeling for reproducibility. We additionally intend to release the pseudo-labels of the dataset.

Q2: How accurate are the trajectories pseudo-labeling?

We evaluate our pseudo-labeling pipeline which consists of calibration, depth estimation, and finally SLAM, on Nuscenes. We show the results in Sec. 8.1 based on the Average Displacement Error (ADE) which are 0.48 meters with scale compensation and 1.68 meters without scale compensation. Although both these numbers are acceptable, this difference marks the inherent error in the monocular depth estimator model we use. We find the accuracy of the trajectories high enough to use them as control signals for GEM.

Q3: Are the depth and images synced?

By observing the generations, it is evident that the modalities are aligned. This alignment can be attributed to several factors: (1) both modalities are encoded using the same model, ensuring similar latent representations and preserving spatial correspondences; (2) they are processed simultaneously through the network, allowing it to learn and model relationships between the modalities; and (3) the same sampler is used for both. These factors collectively contribute to the alignment observed between the two modalities. Refer to our website to see many examples on both modalities.

Q4: Why are different controls added with different techniques?

For ego motion, we empirically find that it is sufficient to incorporate the trajectories using additional cross-attention layers. However, this was not the case for the other controls that did not show similar effectiveness when added through additional cross attention layers. For object and human pose controls, where fine-grained details in the encoding of the scene composition is needed, it is necessary to incorporate spatial information. Therefore, we use specific networks to project these controls and add these features to the output of the backbone's input blocks.

Q5: What's the strategy used to evaluate the different controls?

Our evaluation strategy for all control techniques involves applying the control, detecting it in both the generated video and the ground truth, and comparing the results using a specific metric. For example, for ego motion control, we apply the trajectory control and get the generated video. For datasets without ground-truth labels, we use our pseudo-labeling pipeline to detect the trajectories in both the generated video and the ground-truth video. We then use Average Displacement Error (ADE) for comparison. We use similar technique with other controls, each having a specific evaluation metric. If no similar world model can perform the same control strategy, we compare our conditional generations against the unconditional ones to show the effectiveness of our control.

Q6: Is data curation needed? What's the motivation behind it?

We incorporate a large amount of uncurated data into the dataset. Many samples suffer from poor camera distortions, extreme blurriness, or are completely black. Therefore, a quality filtering step is necessary. Furthermore, the dataset contains numerous videos with minimal activity, *e.g.*, long highway drives. To enhance training efficiency, we filter the dataset based on diverse scene characteristics. Furthermore, to achieve precise control over object movements within a scene, the training data must: (1) include diverse interactions and dynamics, and (2) capture fine-grained details of the objects. This ensures the training process supports accurate control mechanisms while maintaining efficiency.

Q7: Are there any evidence of knowledge transfer to the other domains ?

To assess cross-domain knowledge transfer for GEM, finetuned on human and drone domains, we finetune SVD for the same 5 epochs. GEM provides two key benefits: (1) faster convergence in depth generation—SVD completely fails to generate depth as shown in Fig. 8, (2) improved controllability as shown in Fig. 9.

Drones-SVD Drones-GEM Human-SVD Human-GEM

Figure 8. Depth outputs of finetuning GEM v.s. finetuning SVD.

Q8: Why does GEM have higher FID and lower FVD than Vista?

FID difference between GEM and Vista in unconditional generations stem from their training strategies. GEM's fine-grained control necessitates learning spatio-temporal dependencies from the control signals. Therefore, GEM uses controls in both training stages, while Vista is trained

Figure 9. **Video frames** showing control behavior. Top: Control-GEM (human), Bottom: finetuned Control-SVD (human). GEM shows better controllability when finetuned on human egocentric domain in comparison to finetuning SVD.

on OpenDV without controls and finetuned with LoRA on Nuscenes, likely improving FID by directly adapting to distributions. Furthermore, GEM's data curation is designed to enhance controllability by emphasizing diverse dynamics, likely contributing to its superior FVD.

Q9: Can DINO features be used to insert objects with different appearances?

Indeed, DINOv2 encode appearance; their strong representation capabilities ensure visual consistency of inserted objects. We show examples of inserted objects with different appearances, via extracting DINOv2 features of specific vehicles, in Fig. 16.

Q10: What happens when the control signal is Out of Distribution (OOD)? For instance, what happens if a car is placed in the sky?

GEM typically aims to remain in distribution with the given control. As illustrated in Fig. 10, positioning a car on the sidewalk results in a car being generated on the bike lane. We observe that as the control becomes increasingly OOD, the quality of the generations deteriorates. For entirely OOD controls, such as placing a car in the sky, GEM's outputs result in chaotic generations.

Figure 10. **Video frames** on Out of Distribution (OOD) control signals. Each row shows four frames from one video.Top: Inserting car on sidewalk. Bottom:Inserting car on train rails.

8. Method

8.1. Pseudo-labeling

Depth. We generate depth information for (1) trajectory pseudo-labeling and (2) generating the spatial information of the scene. For depth estimation, we utilize the metric version of Depth Anything V2-Small [73], a state-of-the-art depth estimator known for its accuracy on the KITTI dataset and per-frame consistency.

Ego-trajectories. To estimate ego-trajectories, we first determine the camera's intrinsic parameters with GeoCalib [58], using a pinhole camera model. For videos with radial distortion, we empirically find that radial camera calibration yields improved results. Using the estimated intrinsics and the RGB-depth output from Depth Anything V2, we then apply DroidSlam [56], an RGB-D SLAM algorithm. The use of metric depth is crucial to help with scale ambiguities. The output of the SLAM algorithm consists of a sequence of camera-to-world matrices $A_i = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$ for $i \in \{1, \ldots, N\}$. For driving scenes, we extract X and Z displacements to have Bird-eye-view trajectories, and for ego-centric domains, we include Y displacement and the rotations in the Ortho6D format [86].

We evaluate our trajectory pipeline on the NuScenes dataset using ground truth trajectories as a benchmark. As shown in Tab. 4, the Average Displacement Error (ADE) is 1.64 m when scale is not compensated, relying solely on the depth pseudo-labels to guide the scale. However, when we compensate for the scale using ground truth labels, the ADE is reduced to 0.48 m. This result highlights the potential value of improving depth annotations, as better depth quality could further enhance trajectory accuracy. Despite this limitation, our pseudo-labeled trajectories are sufficiently accurate to guide the model in controlling the motion of the ego vehicle. In our use case, the primary requirement is for the trajectories to approximate the motion of the ego agent closely enough to enable the model to generalize and control the vehicle in new scenarios effectively.

	Nuscenes ADE (m) ↓
With Scale Compensation	0.48
Without Scale Compensation	1.63

Table 4. Trajectory pipeline evaluation on Nuscenes

Human Pose. We generate human poses using DW-Pose [75]. The annotation of each human is 17 keypoints describing all the body joints.

8.2. Sampling Algorithm

Algorithm 1 introduces the sampling technique used with dynamic noise schedule. The scheduling matrix S governs the progression of noise levels across frames, with values adjusted based on the temporal relationship between the scheduling index and frame indices. The noise schedule dynamically adjusts to three different phases: initialization, autoregressive and termination. The initialisation starts denoising the frames at different timesteps till the first frame

Algorithm 1 Sampling with Dynamic Noise Schedule

Require: Initial noisy frames $x \in \mathbb{R}^{F\times H\times W}$, noise schedule $\{\sigma_t\}_{t=1}^{T}$, chunk size C.

Compute the scheduling matrix $S \in \mathbb{R}^{H\times F}$:

$$S(m,t) = \begin{cases} \sigma_0 & \text{if } t > m \\ \sigma_{m-t} & \text{if } m - t < |\sigma| \\ \sigma_{|\sigma|-1} & \text{otherwise} \end{cases}$$

i = 0 , f = 0 ▷ Set row index and frame index to 0

while frames remain to be denoised **do**

Apply denoise step

$$x[f:f+H] = x[f:f+H] + \Delta_t,$$
$$\Delta_t = \text{DenoiseStep}(x, S[i,:], S[i+1,:])$$

if $f = 0$ (first iteration) **then**

Initialization Phase:

Frames $x[f : f + H]$ begin denoising with scheduling matrix S.

else if $F - f > C$ (frames are appended) **then**

Autoregressive Phase:

Update scheduling matrix S by shifting columns and adding a new column:

$$S = \text{ShiftLeft}(S), \quad S[:,-1] = \{\sigma_t\}_{t=1}^{T}$$

else

Termination Phase:

Stop appending new frames. Continue denoising with the remaining columns of S:

$$S = S[:, : F - f]$$

end if

if fully denoised frame **then**

Save the fully denoised frame $x[f]$.

Increment $f = f + 1$.

end if

i = i+1

end while

Return fully denoised frames x_{denoised}.

is fully denoised and the last frame just started a few denoising steps. Autoregressive phase gets a fully denoised frame at each step which gets saved and a new column is appended for a new frame. Once we cannot append any more frames, termination starts and the rest of the frames are progressively denoised without appending new ones.

8.2.1. Time complexity

Here we discuss the time complexity of our sampling algorithm. Assume we want to generate a video of F frames, each denoised in $d = 25k$ steps. In the initialization phase, frame $1 \le i \le 25$ gets denoised $(25 - i + 1)k$ times, requiring $25k$ forward passes of the model. After this phase, the first frame is clean and the 25th frame is denoised for k steps. In the autoregressive phase, we remove the clean frame at the beginning of the window, and append a new noisy frame at the end. This is followed by k denoising steps, yeilding a new clean frame, hence each new frame needs only k forward passes of the model and the autoregressive phase needs $(F - 25)k$ forward passes. Finally, in the termination phase all the frames currently in the window get fully denoised. The last frame already being denoised k steps, it takes $24k$ forward passes to finish the termination phase. Summing these phases, our method requires $\frac{F+24}{25}d$ forward passes of the model to generate a F frame video with each frame denoised through d steps. On a GH200 GPU, each forward pass takes around 1 second, initializing the sampler around 20 seconds, and decoding the denoised latent features takes 0.25 seconds per frame. One could use the above explanation and estimates to calculate the inference time based on their needs. Tab. 5 provides specific examples, illustrating the time required to generate videos with 25, 50, and 150 frames, each frame undergoing $d = 50$ denoising steps.

8.3. Data Curation

Since our focus is on learning a world model, we emphasize curating data that ensures in-distribution samples with reliable control rather than prioritizing aesthetically pleasing generations. To achieve this, we carefully select filtering methods and thresholds to balance efficiency, quality, and adherence to the desired data distribution.

However, even after filtering based on the aesthetic score, several undesirable samples remain, including overly blurry videos, night recordings with minimal visibility, or clips affected by dirty camera lenses. To address these issues, we additionally utilize PIQE as a distortion detector [59]. While a PIQE score above 50 typically indicates poor quality, the diversity of our dataset—including scenes such as urban environments, rural highways, and night recordings—necessitates a higher threshold to minimize false positives. We therefore set the threshold to 70–80, achieving a balance that minimizes false positives (e.g., retaining valid night driving scenes) while removing the problematic clips mentioned earlier.

Figure 11 presents both high-quality samples based on the PIQE and aesthetic scores, as well as examples with low-quality scores. Additionally, Figure 11c shows examples of images with a high aesthetic score (indicating good quality) but also a high PIQE score. These results demonstrate that incorporating PIQE into the quality filtering pipeline effectively removes additional unwanted samples.

Frames F	Init time (s)	Sampling time (s)	Decoding time (s)	Total time (s)
25	20	98	6	124
50	20	148	12	180
150	20	348	36	404

Table 5. Inference time calculation examples for different number of frames.

For both levels of diversity filtering, we employ DINOv2 (large), which we found to outperform alternatives such as CLIP and SSCD [46] in representing diversity within and across video clips.

For cross-clip diversity filtering, we compute the DINO feature vector of the middle frame of each video and calculate the cosine similarity between all resulting vectors. On our dataset, even high thresholds of 0.80 filtered out entire videos with monotone highway drives featuring little diversity. Consequently, we opted for thresholds between 0.90 and 0.98 for our training. Example frames with cross-similarity ≥ 0.9 are shown in Figure 12a.

For intra-clip diversity, we aim to measure meaningful changes within a clip. In driving videos, the typically high ego-motion makes a motion score based solely on optical flow unsuitable (see examples in Figure 12d).

To address this, we process the start and end frames through DINO, extract the feature maps, and compute the cosine similarity between the feature vectors of these frames. We then count the number of tokens with cosine similarity ≤ 0.5 and normalize by the total number of spatial features. This results in small thresholds (ranging from 0 to 0.05) that effectively capture intra-clip diversity for training.

Finally, we observed that DINO occasionally failed to compute meaningful features for certain samples, allowing some static videos to evade filtering. To mitigate this, we additionally apply a motion score based on the average optical flow magnitude between the start and end frames, using a low threshold of 0.02 to further filter such cases.

9. Implementation Details

As a baseline, we employ GH200 GPUs with 100 GB of memory. Due to the increased size of our network, we incorporate activation checkpointing and optimizer sharding to mitigate memory constraints, utilizing DeepSpeed [50].

9.1. Training Stages

Our training process builds upon the SVD video model [4] and the EDM framework [36]. To achieve fine-grained, high-quality control, we employ a two-stage training regime, detailed as follows:

Stage	Filter Type	Threshold	Data (%)
Stage 1	Aesthetic Score	≤ 4.0	91%
	PIQE	≥ 70	89%
	Intra- Similarity	≤ 0.02	80%
	Motion Score	0.02	79%
	Cross- Similarity	≥ 0.98	76%
Stage 2	Aesthetic Score	4.2	91%
	PIQE	70	89%
	Intra- Similarity	0.02	80%
	Motion Score	0.02	79%
	Cross- Similarity	0.95	68%

Table 6. Percentage of remaining data after each filter step, starting from 100%.

9.1.1. Control Learning Stage

In this stage, diverse control signals and modalities are introduced. External modules that inject new information into the network are initialized to zero. Given the wide variety of information and tasks across spatial and temporal layers, the entire network is trained without freezing layers or using custom learning rates.

For DINO control, 0 to 10 frames are randomly sampled, a region within these frames is selected, and the regions are encoded using DINOv2 [44]. Following [14], tokens are randomly masked to produce 0 to n_{tokens} per frame, with n_{tokens} set to 16 to maintain sparsity. For identity training, the same frames are used, with 0 to 4 source frames randomly selected and 0 to 3 target frames sampled per source frame, as described in Sec. 3.2.2. Optical flow for the identity training is obtained using RAFT [55].

The initial resolution is 320×576, with a learning rate of 8×10^{-5} and an effective batch size of 1024. Training spans two epochs (15k steps), with control learning verified as detailed in Sec. 5.4. Weak filtering thresholds are applied to maximize training throughput while emphasizing intra-clip diversity to enhance variability in control signals.

9.1.2. High-Resolution Fine-Tuning

This stage aims to refine the quality of the control. Training is conducted at a higher resolution of 576×1024. As DINO control operates at a downsampling factor of 16, this resolution allows for four times more opportunities for token placement. To maintain sparsity, the number of retained

tokens after masking is increased to $n_{\text{tokens}} = 32$.

Training continues with a reduced learning rate of 4×10^{-5} and an effective batch size of 512 for one epoch (6k steps). Stricter filtering thresholds are applied during this stage to ensure higher-quality outputs. The thresholds and corresponding data retention percentages for the different training stages are summarized in Tab. 6.

10. Additional Evaluation

Depth Generation. Tab. 7 presents GEM's depth evaluation using AbsRel and δ, compared to DepthAnything V2's small and large models. Interestingly, while the training labels are from the small model, results indicate GEM's depth generations align more closely with the large model on OpenDV, more accurate model in the OpenDV dataset, demonstrating improved depth accuracy over the input.

	Nuscenes		OpenDV	
	AbsRel ↓	δ ↑	*AbsRel* ↓	δ ↑
GEM (vs ViT-S)	**0.17**	**0.79**	0.17	0.8
GEM (vs ViT-L)	0.2	0.75	**0.13**	**0.84**

Table 7. Depth generation quality comparison. Our model, despite being trained on pseudo labels from the smaller model, GEM generates slightly closer quality to the estimates of the larger DepthAnything model [72].

11. Ablation Studies

Identity Evaluation. Showing the significance of adding ID embeddings to the DINO tokens of different objects is challenging. This is because the ID is primarily beneficial in scenarios with ambiguous actions (e.g. two very close objects or when moving an object and inserting another). Therefore, we randomly chose a subset of 100 videos where we can test the importance of adding ID labels. As shown in Tab. 8, adding ID embeddings resulted in a slight decrease in the Controllability of Object Manipulation metric (COM) error, from 22.4 pixels to 21 pixels. However, COM is not an ideal metric for evaluating the role of ID embeddings in these scenarios. To better illustrate their importance, we provide examples in Fig. 13. These highlight the critical role of ID embeddings in resolving ambiguities and enabling more precise control when managing interactions with adding different controls on different objects.

Depth. To investigate the significance of the depth modality during training, we compare the quality and controllability results when training GEM with and without depth. We use FID and FVD as video quality metric and COM for object motion controllability. As shown in Tab. 9, incorporating depth leads to slight improvements in both quality and controllability, highlighting its significance especially for controllability.

	OpenDV
	COM ↓
With object ID	21.0
Without object ID	22.4

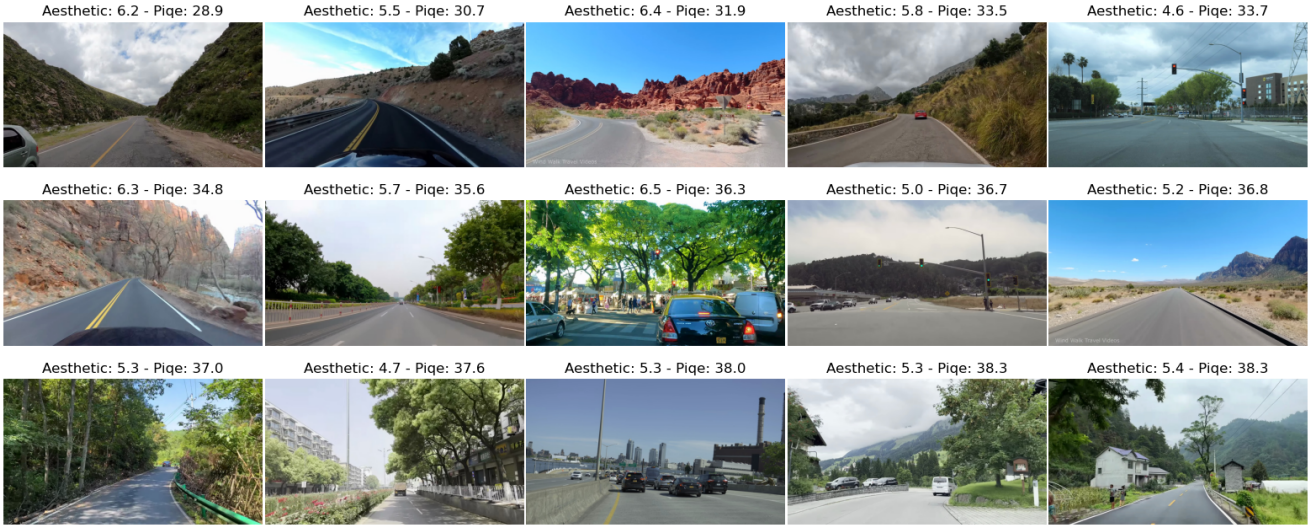
Table 8. Comparison of adding ID embeddings for DINO tokens.

	FID ↓	FVD ↓	COM ↓
GEM (w/ depth)	**10.5/6.27**	**158.5**/130.5	**12.2/11.5**
GEM (w/o depth)	13.3/6.36	179/**129.13**	13.6/12.4

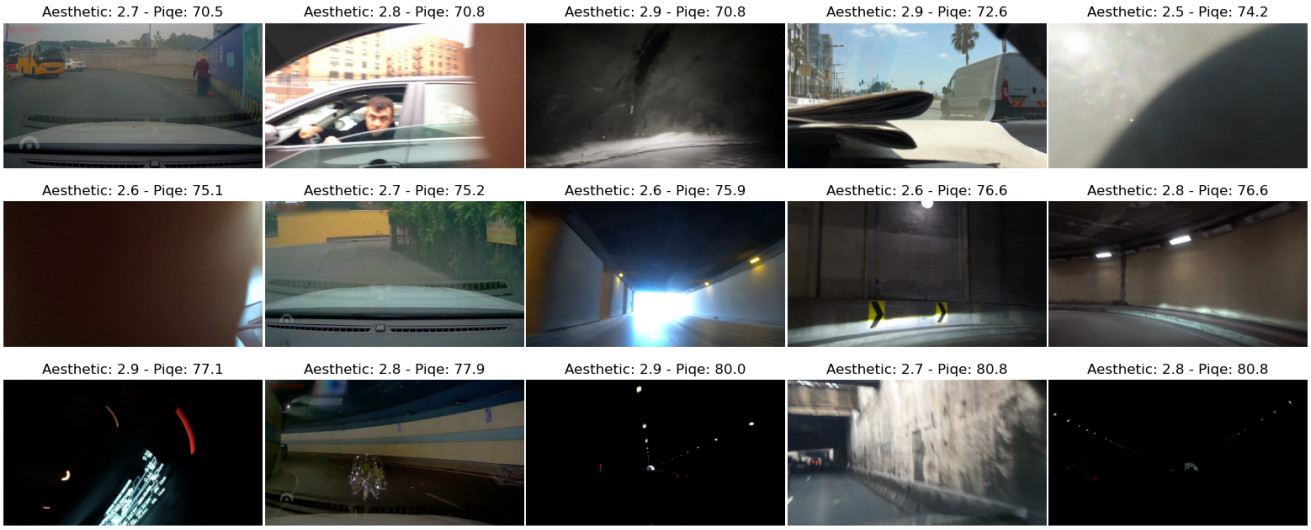
Table 9. Comparison for GEM w/ and w/o depth. Training GEM with depth slightly improves quality and controllability.

12. Qualitative Results

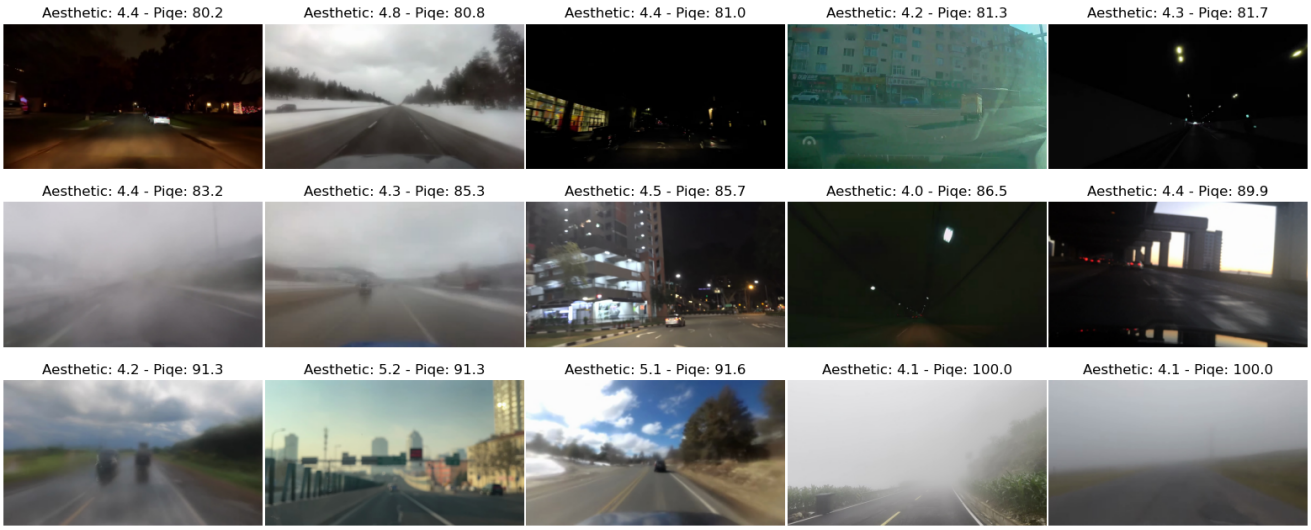
Figs. 14 to 17 show qualitative examples of our generations, our controls, long generation and multimodal outputs.



(a) High-quality images, with high aesthetic score (≥ 4) and low Piqe score (≤ 50).



(b) Images filtered with aesthetic score ≤ 3 and Piqe score ≥ 70.



(c) Images with high aesthetic score (≥ 4) but high Piqe score (≥ 80).

Figure 11. Visual examples for quality filtering.

(a) Images with a cross similarity ≥ 0.90.

(b) Video clip with high intra-diversity of 0.24.

(c) Video clip with low intra-diversity of ≤ 0.02.

(d) Video clip with low intra-diversity of ≤ 0.02i, but high motion score (0.12).

Figure 12. Visual examples for diversity filtering.

(a) We move the car to the right while inserting another car to the left.

(b) We move the car to the left while inserting another car to the right.

Figure 13. Demonstration of moving an object while simultaneously inserting a new one nearby. We utilize DINO tokens of the car from the initial frame and replicate them at specified locations and times (e.g., $T = 0$ and $T = 10$). Identity is added to tokens corresponding across time. The DINO control is shown on the left, and the resulting generation is displayed on the right.

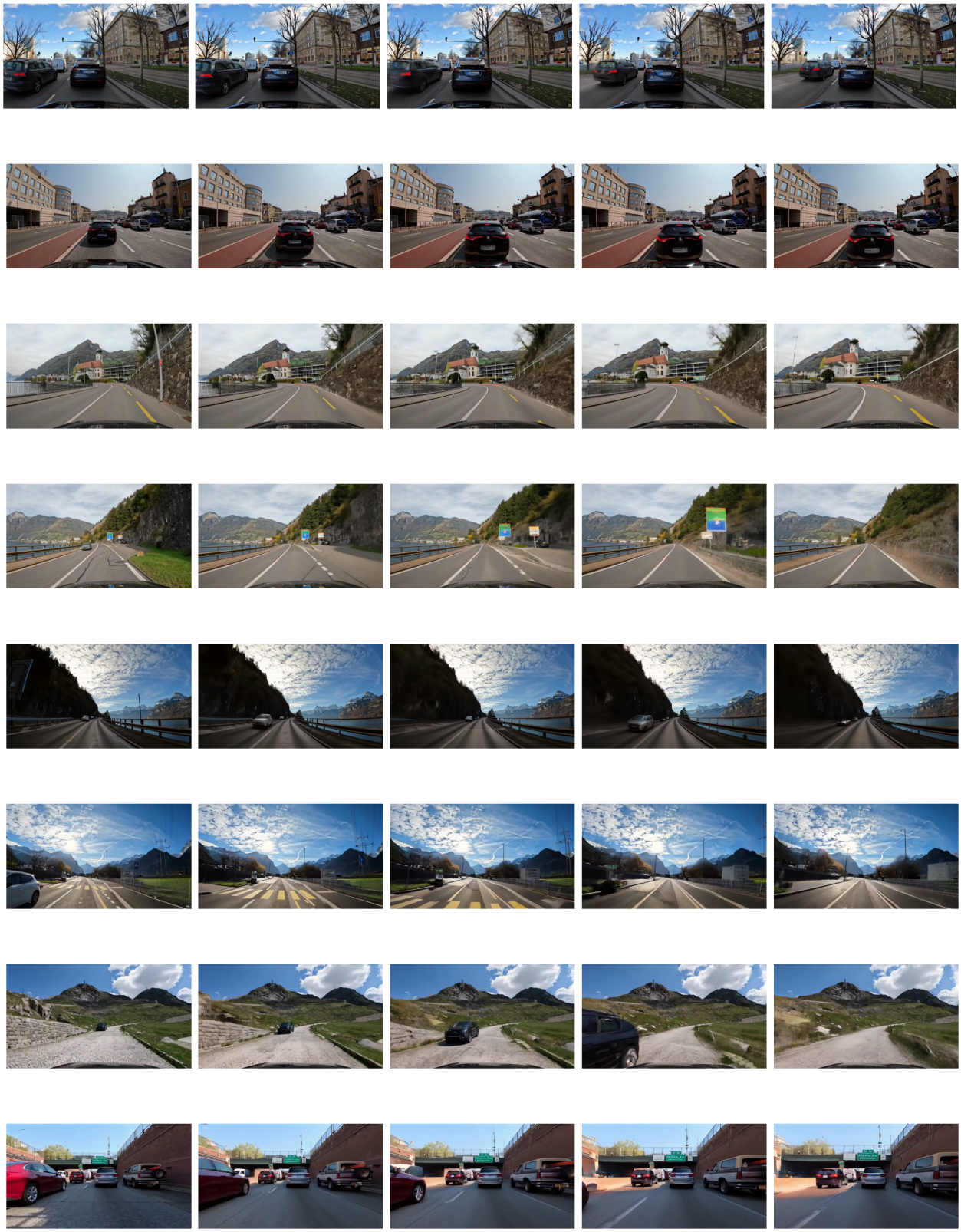


Figure 14. Example visualizations of GEM's generated videos with lengths equal to the training horizon (2.5s, 25 frames) from OpenDV validation set.

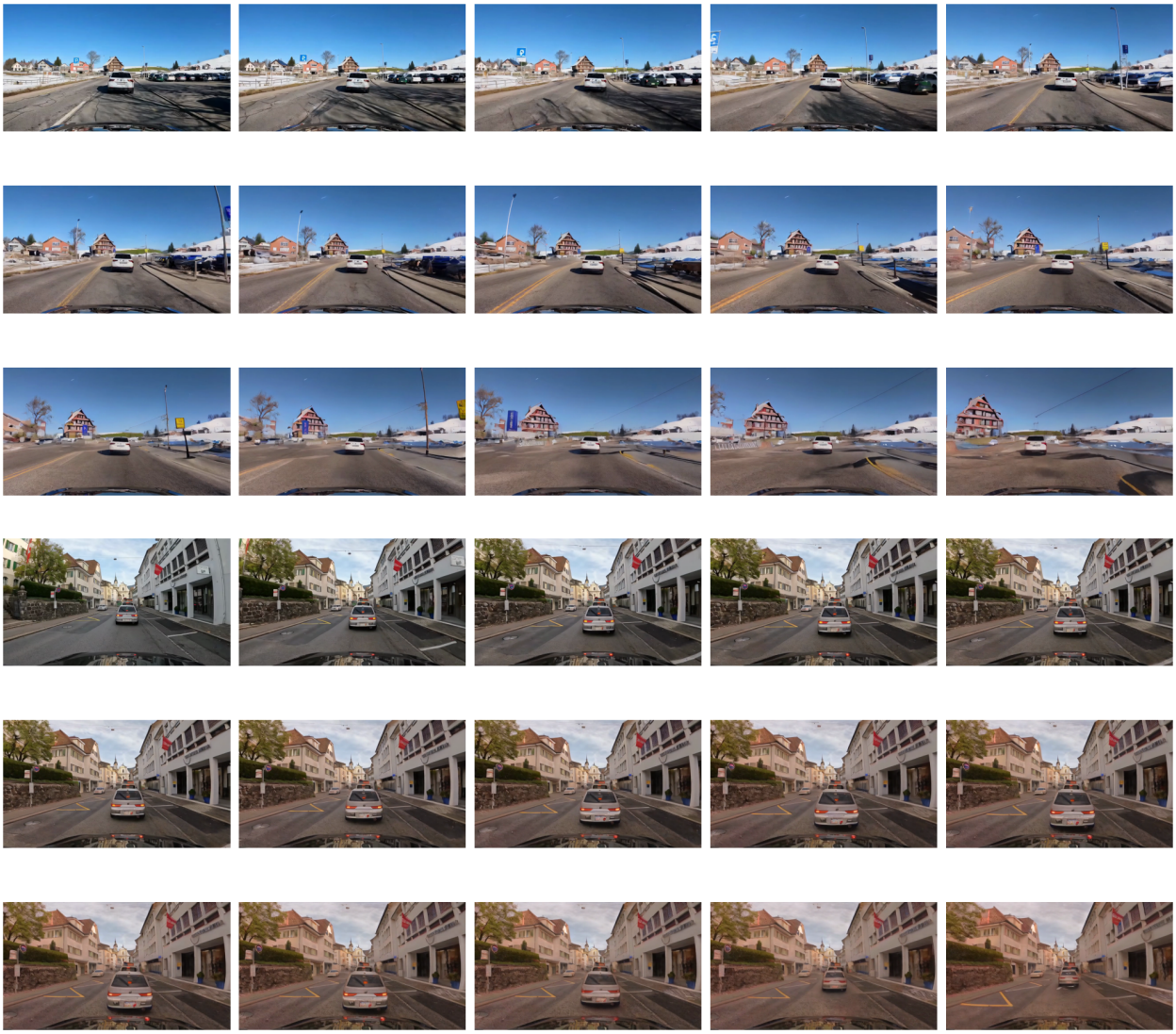


Figure 15. Example visualizations of GEM's generated videos with lengths 6 times greater than the training horizon from OpenDV validation set. The generations are 150 frames which is 15s long

(a) Moving two cars.

(b) Inserting a car on the left.

(c) Inserting a truck on the left.

Figure 16. Examples of moving and inserting objects with DINO control.

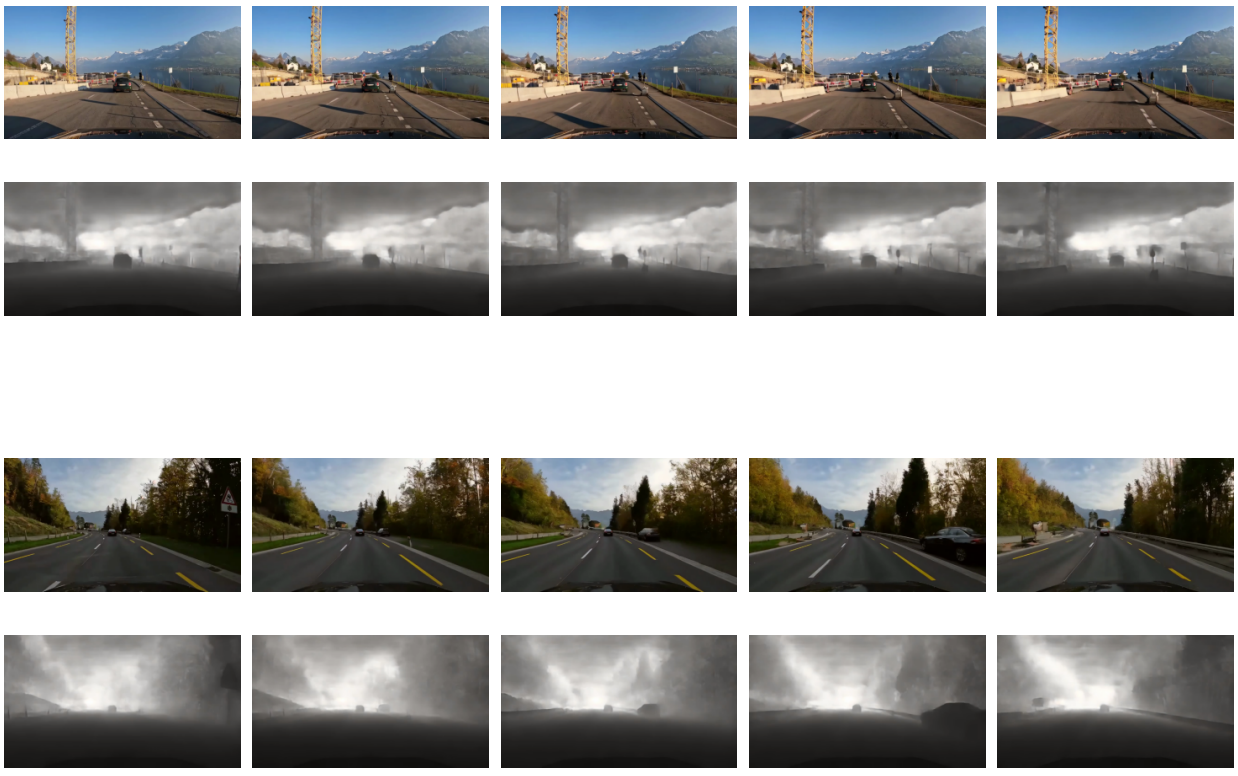


Figure 17. Visualizations of GEM's multimodal generations showing paired depth and RGB frames from OpenDV validation set.