Chain of World: World Model Thinking in Latent Motion

Supplementary Material

1. Implementation Details

1.1. Datasets

We collected high-quality robot manipulation data for fine-tuning the Latent Motion Extractor (LME) and training the VLA, with the datasets summarized in Table 1. Most of the data comes from the OXE [14] dataset, and we additionally include the Calvin [13] and Libero [12] simulation datasets. For LME fine-tuning, we use only episode frames. In the VLA pre-training stage, we use both episode frames and text instructions. Following UniVLA [22], we adopt different sampling intervals for each dataset to ensure that the temporal gap between keyframes is approximately one second. We then uniformly sample 16 frames from the continuous frames covered by six keyframes for pre-training. Throughout this stage, only third-person view data is used, excluding wrist-camera views.

During the VLA co-fine-tuning stage, we train on the benchmark-specific training sets using text instructions, frames, and actions. For example, the BridgeV2 dataset [18] is used for the SimplerEnv-Bridge evaluation [11], while the Libero [12] evaluation uses the mixed data of four Libero task suites processed by OpenVLA [10]. In addition, the appendix includes extended experiments using the Fractal dataset [1] for the Simpler-Google Robot [11] evaluation and the Calvin dataset [13] for the Calvin evaluation, covering both ABCD→D and ABC→D task settings. Across the co-fine-tuning experiments, Bridge and Google Robot training use only third-person views, while Libero and Calvin use both third-person and wrist views.

Table 1. Training datasets.

Dataset Name	Count
Berkeley Autolab Ur5 [4]	892
Bridgev2 [18]	24879
Cmu Play Fusion [3]	576
Fractal [1]	65530
Kuka [9]	84202
Maniskill [7]	30029
Taco Play [16]	3242
Toto [24]	899
Utaustin Mutex [17]	1500
Viola [25]	135
Calvin [13]	22966
Libero [12]	1693
Total	**236543**

1.2. Training Details

For LME fine-tuning, we start from the VidTwin [21] pre-trained model and fine-tune it on the video data from the datasets listed in Table 1. We use 4 A800 GPUs with a per-GPU batch size of 4, randomly sampling 16 frames per video. Each frame is resized to 224×224. The KL loss weight is set to 1e-6, and the reconstruction loss is reduced using the mean over all elements rather than the default reduction over the batch dimension only. We randomly sample 1000 videos from the training set as a validation set and select the checkpoint with the lowest reconstruction loss. The final model corresponds to the checkpoint trained for one epoch plus 20k iterations.

For VLA pre-training, we initialize from the 8.5B Emu3 [20] pretrained checkpoint and train on the datasets in Table 1. The training is performed on 32 A800 GPUs with a per-GPU batch size of 8. Image observations are resized to 256×256. We use the first and last frames of each video clip together with one learnable motion query, and the maximum sequence length is set to 2500 tokens. We train for 10k iterations in total, which takes roughly 24 hours.

For VLA co-fine-tuning, we follow the evaluation protocols from UniVLA [22] for each benchmark. We load the checkpoint from the VLA pre-training stage and train with 16 A800 GPUs, using a batch size of 8 per GPU and full-parameter fine-tuning. The maximum sequence length is set to 3200 tokens. For SimplerEnv-Windowx [11], we use BridgeV2 [18] data with images resized to 256×256 and train for 12k iterations. For SimplerEnv-Google Robot [11], Fractal [1] images are resized to 240×192, and training continues for 16k iterations. For Libero [12], images are resized to 200×200, and training runs for 8k iterations. For Calvin [13], third-person views are resized to 200×200 and wrist views to 80×80, with training conducted for 12k iterations. The per-iteration training time across these configurations is similar; for example, Libero training takes about 25 hours for 8k iterations. Overall, each configuration requires roughly one to two days of training.

1.3. Interpretation of the World Model and the Latent Motion Chain

Our method combines a world model formulation with latent action modeling. The world model component consists of two stages: pre-training and co-fine-tuning. During pre-training, the world model is not action-conditioned. This follows the representation adopted by UniVLA [22] and FlowVLA [23], where the world model predicts future

Table 2. Long-horizon robotic manipulation evaluation on the CALVIN [13] benchmark. Methods marked with † are from our re-implementation.

Method	Task	Tasks Completed in a Row					**Avg. Len ↑**
		1	2	3	4	5	
UniVLA† [22]	ABCD→D	**0.988**	0.934	0.883	0.829	0.764	4.398
Ours		0.972	**0.939**	**0.894**	**0.859**	**0.809**	**4.473**
TLA [2]	ABC→D	0.955	0.858	0.754	0.669	0.565	3.800
Dita [8]		0.945	0.825	0.728	0.613	0.500	3.610
UniVLA† [22]		**0.972**	0.902	0.826	0.741	0.661	4.102
Ours		0.968	**0.912**	**0.844**	**0.779**	**0.708**	**4.211**

Table 3. Evaluation on SimplerEnv-Google Robot [11] across various manipulation tasks.

Model	Pick	Move	Drawer	Place	**Average**
OpenVLA [10]	0.180	0.563	0.630	0.000	0.343
SpatialVLA [15]	0.860	**0.779**	0.574	0.090	0.576
MoTo [6]	0.740	0.604	0.431	0.000	0.444
villa-X [5]	**0.987**	0.750	**0.593**	0.056	0.597
UniVLA [22]	0.870	0.565	0.194	0.167	0.449
Ours	0.923	0.676	0.428	**0.407**	**0.609**

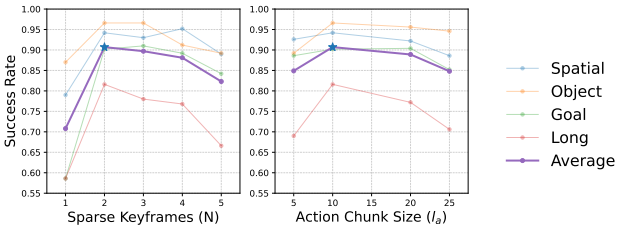


Figure 1. Sensitivity analysis of N and l_a on LIBERO.

environment evolution given a language instruction and an initial state, rather than explicit actions. During the co-fine-tuning stage, we introduce an action-conditioned formulation: $p(v^{t+1} \mid v^t, A^t)$.

Our latent motion does not explicitly perform multi-step rollouts. Instead, it provides a continuous and decoupled motion encoding over a temporal window, which can be interpreted as an implicit motion chain.

2. Additional Results

2.1. Analysis of keyframes and action chunk size

We evaluate the number of sparse keyframes $N \in \{1, 2, 3, 4, 5\}$ and action chunk sizes $l_a \in \{5, 10, 20, 25\}$ on LIBERO to understand the temporal granularity required by latent motion reasoning. As shown in Figure 1, both hyper-parameters exhibit a clear inverted-U trend. The best performance is achieved at $(N = 2, l_a = 10)$, corresponding to a ∼20-frame (≈2 s) temporal horizon.

When using only one keyframe ($N = 1$), performance drops significantly across all suites, especially on long-horizon tasks, indicating that the latent motion becomes under-constrained. Increasing N to 2 provides sufficient visual anchoring and yields the largest improvement. However, further increasing N gradually degrades performance. With dense observations, the model can rely on short-term visual matching instead of inferring motion dynamics, weakening the benefit of latent temporal reasoning.

A similar phenomenon appears for action chunk size. Small chunks ($l_a = 5$) reduce temporal abstraction and make the policy closer to step-wise imitation. Large chunks ($l_a \geq 20$) introduce high uncertainty in future evolution, particularly harming the long-horizon tasks. The intermediate chunk size ($l_a = 10$) achieves the best trade-off between predictability and abstraction.

Overall, the results suggest that the proposed model performs best when sparse observations provide partial constraints while still requiring the model to infer continuous evolution. This supports our design motivation: the latent motion token serves as a dynamics aggregator over a medium temporal window rather than dense frame tracking or one-step prediction.

2.2. Comparison with other Video VAE

To further analyze the role of latent motion representations, we replace VidTwin with the VAE from Wan 2.1 [19] and

Table 4. Comparison between our latent motion representation and Wan 2.1 VAE latent **z** on LIBERO.

Variant	**Pre-training**	**Co-fine-tuning**	Spatial	Object	Goal	Long	**Average**
Ours	latent motion + terminal frame	+ latent motion	0.948	0.974	0.958	0.906	**0.947**
Wan2.1 VAE [19]	latent **z** + terminal frame	+ latent **z**	0.938	0.950	0.922	0.868	0.920

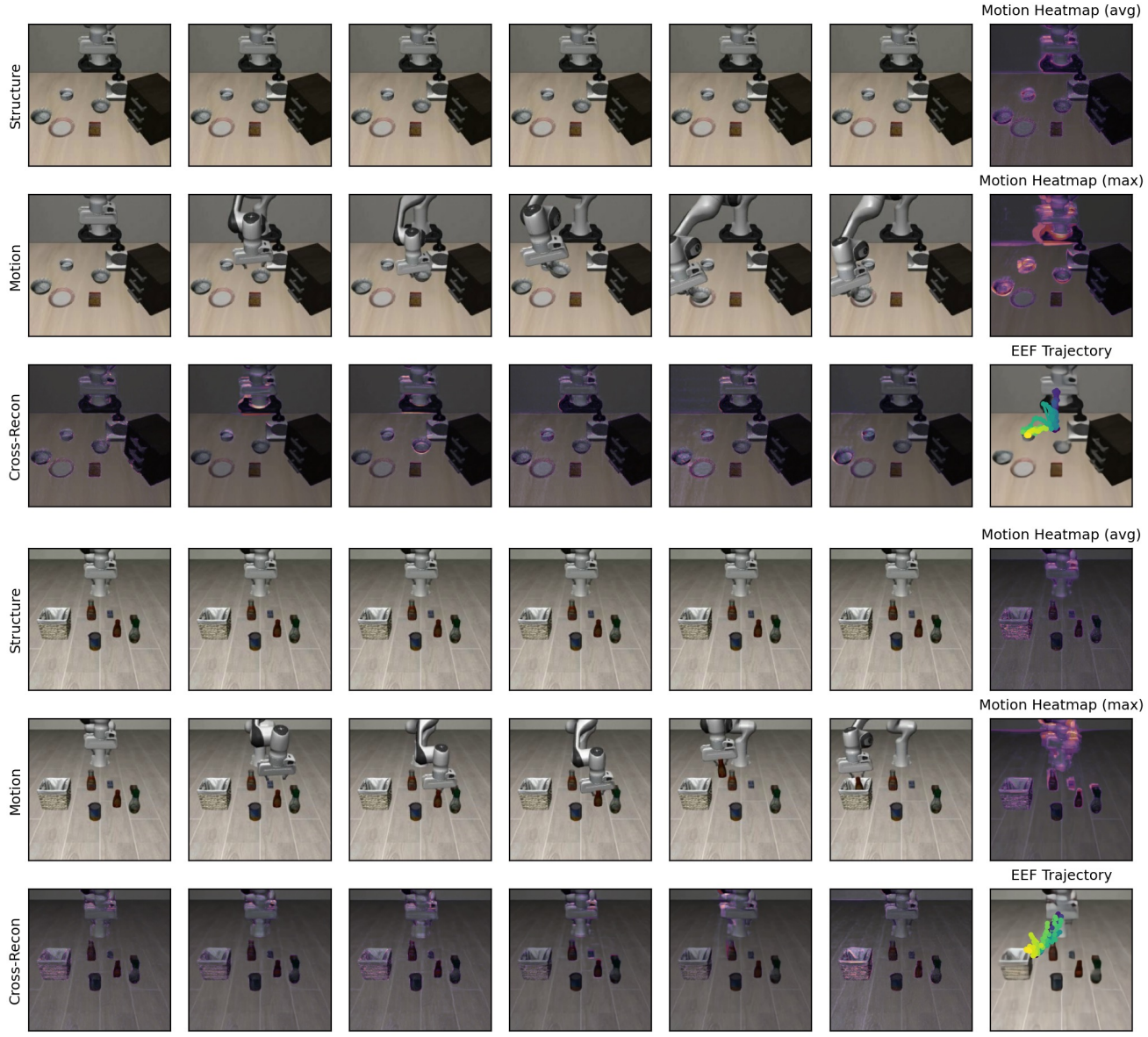


Figure 2. Cross-Recon visualization on LIBERO [12]. The first six columns show temporally sampled frames from three rows: Structure (top), Motion (middle), and Cross-Recon (bottom). The Cross-Recon videos are generated by combining the static appearance from the Structure video with the motion representation extracted from the Motion video, revealing the transferred motion patterns. Each Cross-Recon frame is overlaid with a motion heatmap to highlight dynamic regions. The last column presents three summary maps: motion heatmaps obtained by averaging and maximizing per-frame absolute differences between Cross-Recon and Structure, and the end-effector trajectory estimated from the motion regions.

conduct a controlled comparison. Specifically, we use the latent **z** extracted by the Wan 2.1 VAE as auxiliary supervi-

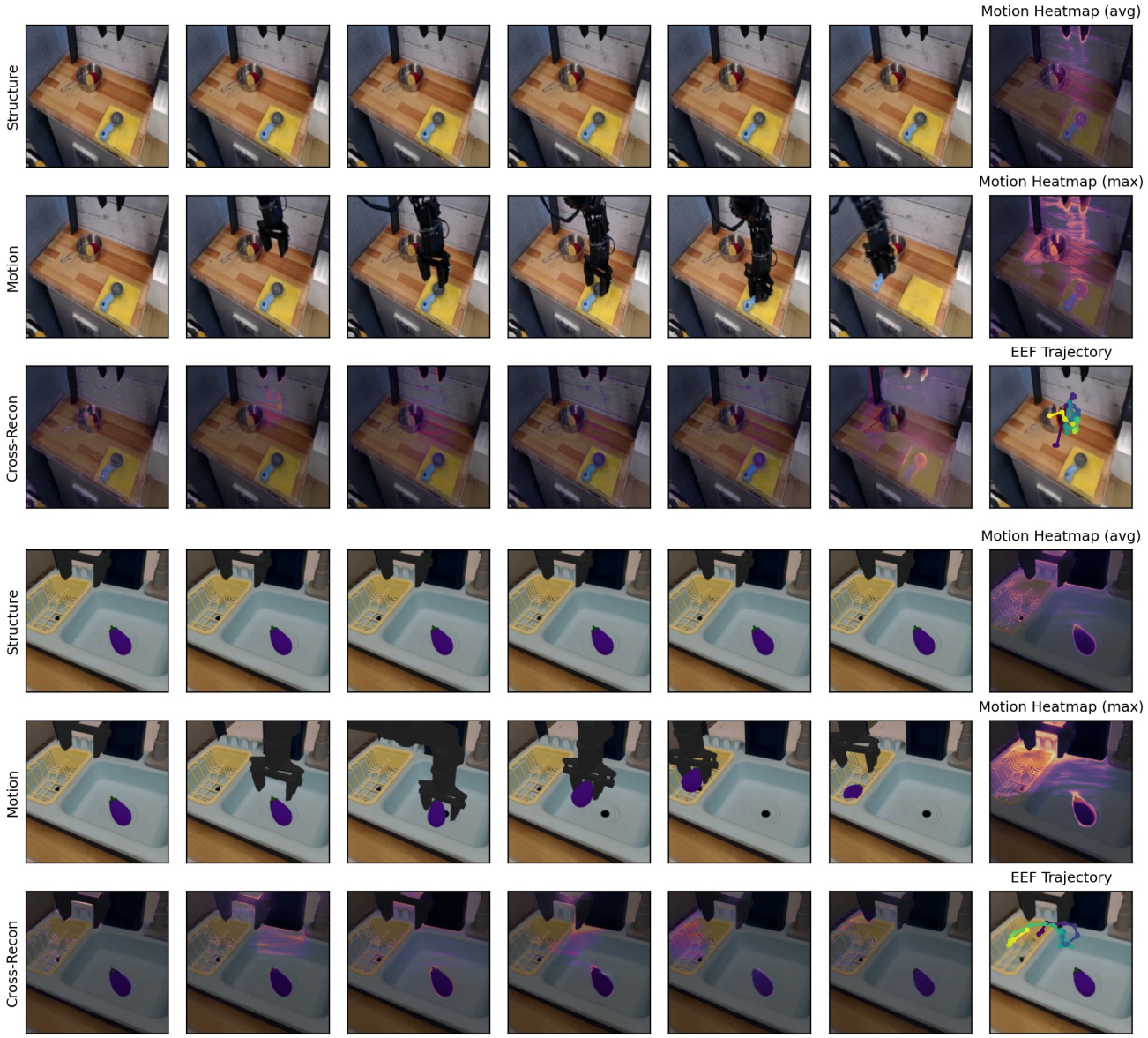


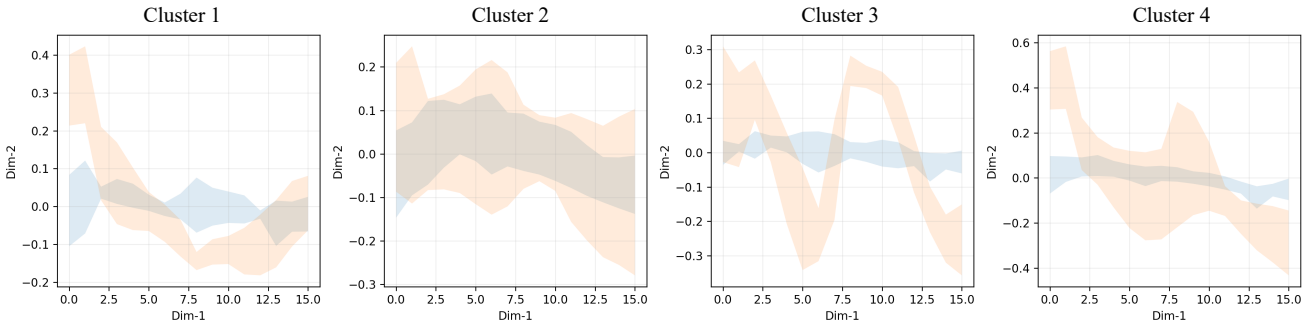
Figure 3. Cross-Recon visualization on SimplerEnv [11] and Bridgev2 [18].

sion during both pre-training and co-fine-tuning.

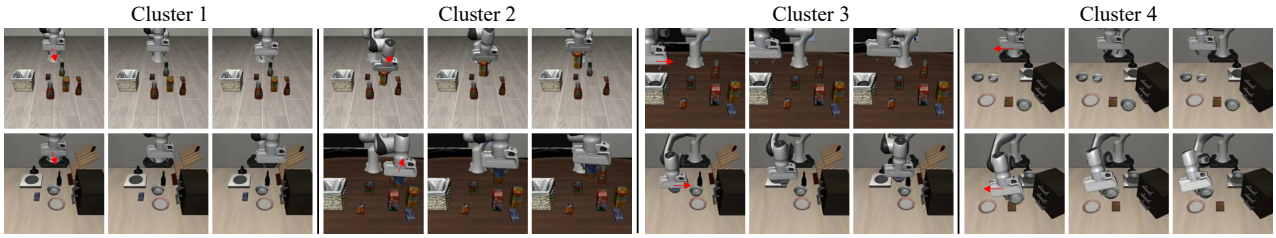
The Wan 2.1 VAE is trained on large-scale video data and therefore incorporates rich generic video priors. As shown in Table 4, this variant achieves an average success rate of 0.920 on LIBERO. While competitive, it remains inferior to our latent motion design (0.947).

2.3. CALVIN

Calvin [13] is an open-source simulated benchmark built on PyBullet, designed for learning long-horizon, language-conditioned robotic manipulation tasks. It provides a tabletop simulation environment containing 23 types of manipulation skills, such as lifting, pushing, rotating, and object relocation. These skills must be executed in sequence to complete multi-step tasks, introducing substantial uncertainty and randomness, which makes Calvin a highly challenging evaluation benchmark. The dataset includes a large number of expert demonstrations and is organized into multiple subsets. In our experiments, we use the ABCD→D and ABC→D subsets, and during training, we only utilize demonstrations that include natural language descriptions of the actions. Following the official evaluation protocol, all tests consist of 1000 episodes, each containing a sequence of five sub-tasks specified by natural language instructions.



(a) Unsupervised clustering of latent-motion trajectories.



(b) Representative video clips sampled from motion clusters.

Figure 4. Visualization of latent-motion clusters and corresponding video examples. (a) Unsupervised clustering results of clip-level motion trajectories. Each subplot shows the average 2D motion trajectory (obtained from the first two PCA components of the accumulated frame-wise motion deltas) for one cluster. (b) Representative video examples from clusters. Cluster 1 and 2 correspond to monotonic downward-like or upward-like motions, whereas Cluster 3 and 4 exhibit rightward-like or leftward-like behaviors.

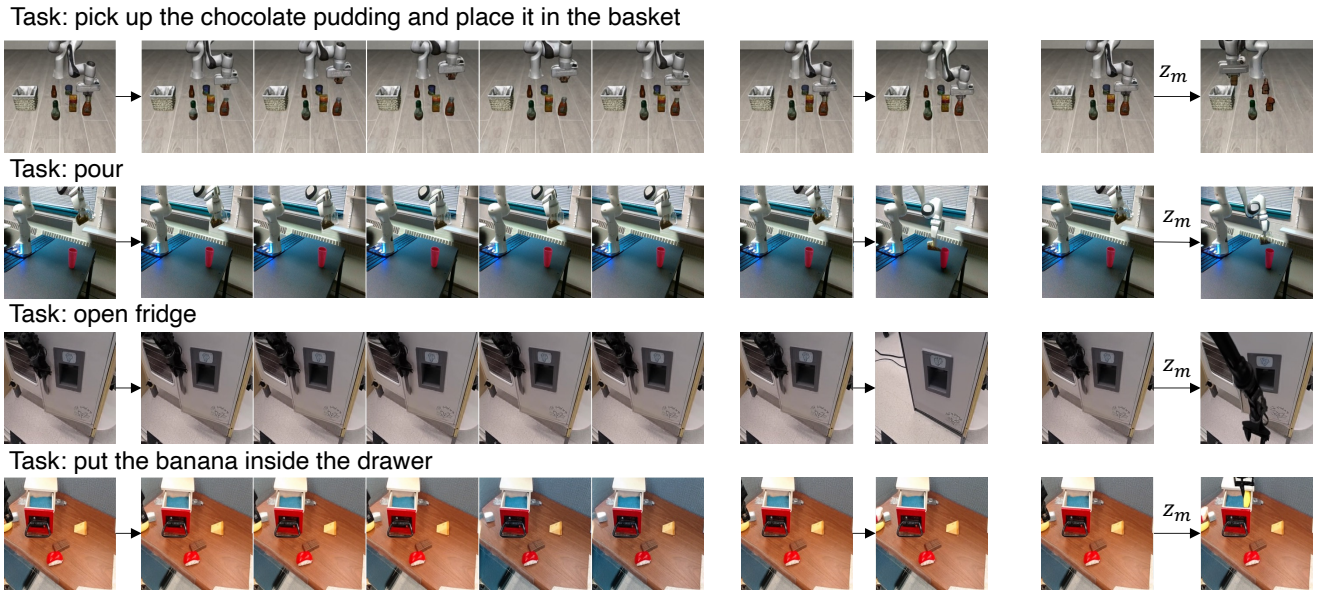


Figure 5. Comparative visualization of future-frame prediction strategies.

Figure 6. An Intel RealSense camera and a Realman RM75B robot.

The main results are presented in Table 2. Our method achieves an average success length of 4.473 on the ABCD→D task and 4.211 on the ABC→D task. For a fair comparison, we reproduced UniVLA [22] using the training sets listed in Table 1, and followed a fine-tuning setup with 16 A800 GPUs and a per-GPU batch size of 8. Under the same training configuration, our approach outperforms UniVLA [22].

2.4. SimplerEnv-Google Robot

We also evaluate our method on the SimplerEnv-Google Robot benchmark. The evaluation primarily follows the visual matching protocol, which assesses the alignment between real and simulated visual appearances by overlaying real-world images onto simulated backgrounds and adjusting the textures of foreground objects and the robot within the simulator. This benchmark includes four tasks: *pick coke can*, *move near*, *open/close drawer*, and *place in closed drawer*.

The main results are shown in Table 3. Our method achieves an average success rate of 0.609, outperforming UniVLA [22], villa-x [5], MoTo [6], and other baselines. Here, UniVLA refers to our reproduction. Our method surpasses UniVLA on all four tasks and shows a particularly large improvement on the *place in closed drawer* task.

2.5. More Visualization

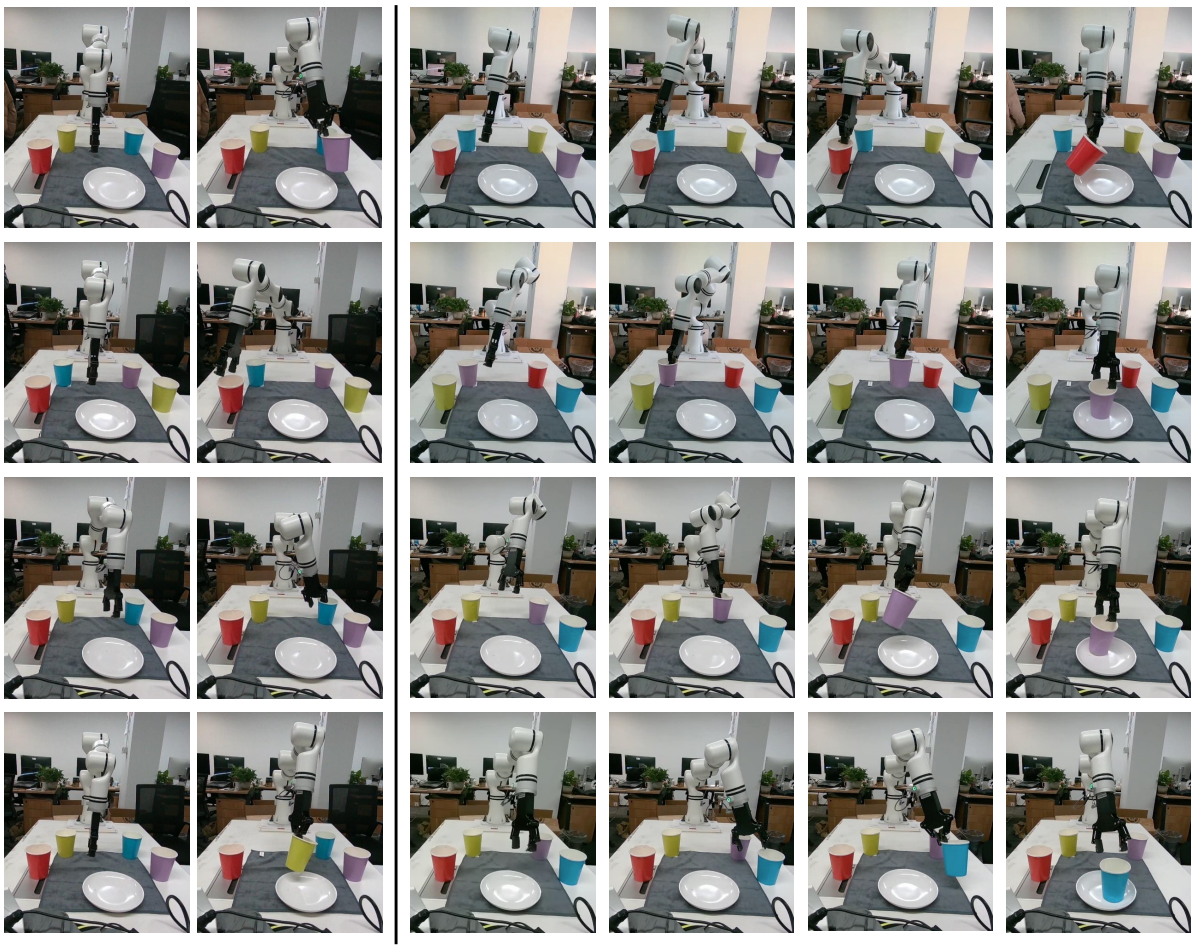
We provide extended visualizations for the latent motion analysis presented in Section 4.4, with the main results shown in Figures 2, 3, 4, and 5.

Effective decoupling of structure and motion latents. Figures 2 and 3 analyze representative samples from the Libero and Bridge datasets. The first six columns display temporally sampled frames from three rows: Structure (top), Motion (middle), and Cross-Reconstruction (bottom). The Cross-Recon videos are synthesized by combining the static appearance from the Structure video with the motion representation extracted from the Motion video, thereby revealing transferred motion patterns. Each Cross-Recon frame is overlaid with a motion heatmap to highlight dynamic regions. The final column summarizes three diagnostic maps: motion heatmaps computed by averaging and maximizing the per-frame absolute differences between Cross-Recon and Structure, as well as the end-effector trajectory estimated from the activated motion regions. As shown, the highlighted areas consistently follow the movement of the robot arm in the Motion video. In the video results, these regions fluctuate over time; for clarity in static visualization, we display aggregated highlights in the figures.

We further analyze the distribution of motion latents, as shown in Figure 4. To derive an interpretable trajectory representation from high-dimensional motion latents, we first extract per-frame motion features from each video clip and accumulate framewise differences to obtain a temporal sequence describing the overall motion trend of the clip. These sequences are then resampled to a fixed length across all clips and standardized globally. We subsequently apply PCA to the sequence features and take the first two principal components as a 2D trajectory for each clip. This representation preserves the dynamic structure encoded in the latent space while enabling clear comparison across clips.

Figure 4 (a) shows unsupervised clustering of all motion trajectories in the 2D PCA space. To obtain cluster-level canonical shapes, we temporally align trajectories within each cluster via resampling and plot their mean curves along with 95% confidence intervals. Distinct trajectory patterns emerge across clusters—such as monotonic rises, two-stage reversals, and multi-phase back-and-forth motions—indicating that the model's motion latent captures high-level motion semantics. To further validate the semantic consistency within each cluster, we randomly sample two video clips per cluster and visualize three uniformly sampled frames from each clip, as shown in Figure 4 (b). The clips within the same cluster exhibit highly similar motion trends in appearance, confirming that the structure of the motion-latent space yields meaningful discrimination among different action patterns.

Motion latent enhances dynamic modeling for future frame prediction. As shown in Figure 5, we further visualize future frame predictions under different pretraining strategies. From top to bottom, the examples correspond to four tasks: i) pick up the chocolate pudding and place it in the basket, ii) pour, iii) open the fridge, and iv) put the banana inside the drawer. In Figure 5 (a), world-model-based approaches suffer from reconstructing redundant background pixels, which can draw attention away from critical interactions and motion cues. As a result, the predicted future frames sometimes remain nearly unchanged, such as in tasks (ii) and (iii). Figure 5 (b) shows that predicting only the target frame often leads to unsta-



(a) collected data (b) real-world deployment

Figure 7. Comparison between data collection and real-world deployment during testing.

ble generation due to the absence of intermediate evolution steps: in task (i), the target frame nearly collapses back to the initial frame, and in task (iii), only one door of the fridge is generated. In contrast, our method leverages the motion latent z_m as a chain-of-thought for motion, providing stronger guidance for future-frame prediction. The generated final frames align more accurately with the intended task instructions.

3. Real-Robot Experiments

Experimental Setup. As shown in Figure 6, we use the Realman RM75B robot, which is equipped with 7 degrees of freedom and a single gripper. An Intel RealSense camera is used to capture RGB images. We set up a cup-grasping experiment and collected a total of 127 episodes, consisting of 65,382 frames with corresponding actions. Each episode contains an average of 515 frames, corresponding to approximately 20 seconds in the real world. The dataset mainly includes grasping cups of four different colors, with the number of episodes per color as follows: red 31, blue 39, yellow 24, and purple 33. Figure 7 (a) shows some collected data.

During training, all images are cropped and resized to 256×256. The action chunk size is set to 10. We train the model for 2k steps using 16 GPUs with a per-GPU batch size of 8. The data were collected in the afternoon and evening and then used for model training. Testing was conducted the following day. As shown in Figure 7, the lighting conditions have some differences between data collection compared and during real-world deployment. We found that the model was still able to correctly execute instructions under different lighting conditions. Figure 7 (b) shows in the first two rows two test cases: grasping a red/purple cup and placing it on a plate. Their background lighting differs from the training data, but the model is still able to execute the tasks successfully.

References

[1] Anthony Brohan, Noah Brown, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 1

[2] Qingwen Bu, Yanting Yang, Jisong Cai, et al. Learning to act anywhere with task-centric latent actions. In *RSS*, 2025. 2

[3] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *CoRL*, pages 2012–2029, 2023. 1

[4] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset, 2024. 1

[5] Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, et al. villa-X: enhancing latent action modeling in vision-language-action models. *arXiv preprint arXiv:2507.23682*, 2025. 2, 6

[6] Yi Chen, Yuying Ge, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for robot manipulation. In *ICCV*, 2025. 2, 6

[7] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, et al. Maniskill2: A unified benchmark for generalizable manipulation skills. *arXiv preprint arXiv:2302.04659*, 2023. 1

[8] Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *ICCV*, 2025. 2

[9] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *CoRL*, pages 651–673, 2018. 1

[10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, et al. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2024. 1, 2

[11] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, et al. Evaluating real-world robot manipulation policies in simulation. In *CoRL*, 2024. 1, 2, 4

[12] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023. 1, 3

[13] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *RA-L*, 7(3):7327–7334, 2022. 1, 2, 4

[14] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration. In *ICRA*, pages 6892–6903, 2024. 1

[15] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, et al. Spatialvla: Exploring spatial representations for visual-language-action model. In *RSS*, 2025. 2

[16] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *CoRL*, pages 1838–1849, 2023. 1

[17] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. *arXiv preprint arXiv:2309.14320*, 2023. 1

[18] Homer Rich Walke, Kevin Black, Tony Z Zhao, et al. Bridgedata v2: A dataset for robot learning at scale. In *CoRL*, pages 1723–1736, 2023. 1, 4

[19] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 2, 3

[20] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024. 1

[21] Yuchi Wang, Junliang Guo, Xinyi Xie, Tianyu He, Xu Sun, and Jiang Bian. Vidtwin: Video vae with decoupled structure and dynamics. In *CVPR*, pages 22922–22932, 2025. 1

[22] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *ICLR*, 2026. 1, 2, 6

[23] Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, et al. Flowvla: Visual chain of thought-based motion reasoning for vision-language-action models. *arXiv preprint arXiv:2508.18269*, 2025. 1

[24] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, et al. Train offline, test online: A real robot learning benchmark. *arXiv preprint arXiv:2306.00942*, 2023. 1

[25] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *CoRL*, pages 1199–1210, 2023. 1