A Experiments Details

In this section, we describe the details of the implementations (Appendix A.1), the hyperparameters (Appendix A.2), and the task settings (Appendix A.3).

A.1 Implementations

We build our framework based on PyTorch [46] and use the implementations of Transformer tricks from the codebase x-transformers[5]. We pre-train a VQGAN with the images extracted from BAIR and downstream tasks based on the released code[6]. We use 8×A100 Nvidia GPU and 64 CPU cores for the pre-training run, while using one single A100 Nvidia GPU and 16 CPU cores for each training run. 10 hours are required to pre-train PVDR. For the online stage, 12 hours are required to train on each Meta-World [88] task, and 18 hours are required to train on each RLBench [29] task. The details of the implementations are as follows.

PVDR Structure. We implement all modules in PVDR as Transformer-based structures. Overall, the structures of the modules in PVDR, including a visual dynamics encoder, a visual dynamics prior, a visual dynamics decoder, an action alignment module, and a critic network, are shown in Figure 8. These modules use a pre-trained VQGAN to compress the raw frames into grids of visual tokens and subsequently employ the Spatial-Temporal Transformers [79] to process spatial-temporal visual token sequences. The modules differ from each other in the input data type and the post-processing form of the hidden states. Specifically, the visual dynamics representations are in the same shape as visual token grids and are viewed as grids of prompts that will be concatenated with the visual token grids. During the decoding process, the hidden state of each visual token will be projected onto a vector, whose dimension equals the size of the VQGAN codebook. The softmax value of each dimension is viewed as the probability of the corresponding code. For a quick decoding process, we do not use the beam search. Instead, we recurrently use the code with the highest probability to form a grid and directly decode one visual frame with VQGAN.

ST Transformer. In our PVDR implementation, the core structure is the Spatial-Temporal Transformer (ST Transformer). For a lower attention computation burden, an ST Transformer uses cross-stacked spatial and temporal attention blocks to, respectively, process the information in spatial and temporal sequences. Generally, we design spatial and temporal attention blocks following MaskViT [24]. As shown in Figure 9, the spatial attention blocks process the attention map of visual tokens at the same timestep, while the temporal attention blocks process the attention map of visual tokens in a small local spatial window alongside the temporal sequence.

[5] https://github.com/lucidrains/x-transformers
[6] https://github.com/CompVis/taming-transformers

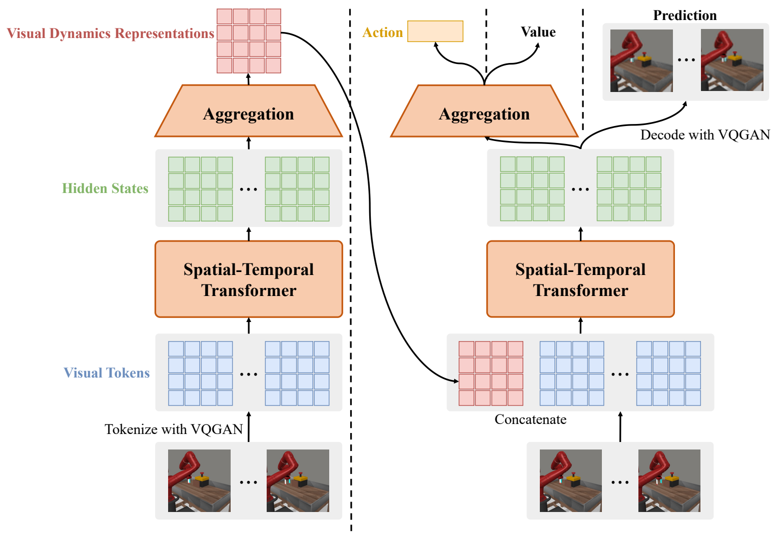

Fig. 8: Illustration of module structures in PVDR. The visual dynamics encoder and the visual dynamics prior share the structure on the left. The visual dynamics decoder, the action alignment module, and the critic network share the main network on the right, with distinct post-processing forms of the hidden states.

Formally, $(l+m) \times h \times w$ visual tokens are processed by the ST Transformer, which are numbered $\{\mathcal{E}_{i,j,k} \mid 0 \leq i < h, 0 \leq j < w, 0 \leq k < l+m\}$. Take a certain visual token, $\mathcal{E}_{i^\star,j^\star,k^\star}$, for example. The visual tokens in the set $\{\mathcal{E}_{i,j,k^\star} \mid 0 \leq i < h, 0 \leq j < w\}$ are included in its spatial attention map. And the visual tokens in the set $\{\mathcal{E}_{i,j,k} \mid \lfloor \frac{i}{b} \rfloor = \lfloor \frac{i^\star}{b} \rfloor, \lfloor \frac{j}{d} \rfloor = \lfloor \frac{j^\star}{d} \rfloor, 0 \leq k < l+m\}$ are included in the temporal attention map, where $b \times d$ are the local attention window size. Additionally, the temporal attention block used in the visual dynamics decoder employs a causal mask. That is, the visual tokens in the set $\{\mathcal{E}_{i,j,k} \mid \lfloor \frac{i}{b} \rfloor = \lfloor \frac{i^\star}{b} \rfloor, \lfloor \frac{j}{d} \rfloor = \lfloor \frac{j^\star}{d} \rfloor, 0 \leq k < k^\star\}$ are included in the causal-masked temporal attention calculation of $\mathcal{E}_{i^\star,j^\star,k^\star}$.

Baselines. For PVDR without pre-training and PVDR-based algorithms for ablation, the structures of the modules are consistent as described above. For PPO, the actor network is in the same structure as the action alignment module in PVDR, while the critic network is identical to the one in PVDR. The setups of APV, FICC, and STG are consistent with the original setups. In particular, FICC requires a discrete action space for search. Thus, we evenly divide the continuous action space of environments into 16 subspaces and uniformly sample one action in each space for each round of search. As PVDR does not learn a reward function, we replace the reward function in FICC with $r_t = -\left\|o_{t+1} - g\right\|_1$ during inference for a fair comparison. All algorithms use $r_t = -\left\|o_{t+1} - g\right\|_1$ as an intrinsic reward for the goal-conditioned setting.

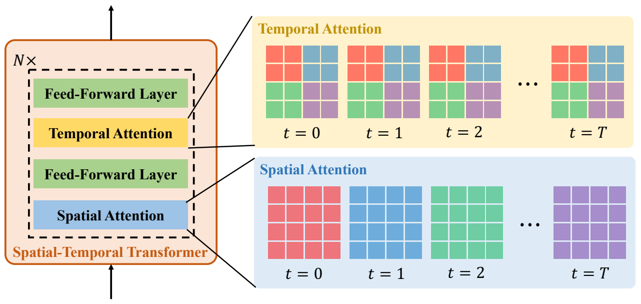

Fig. 9: Illustration of Spatial-Temporal Transformer in PVDR. An ST Transformer (left) is composed of N cross-stacked Spatial Attention and Temporal Attention blocks, with interleaved Feed-Forward Layers. The visual tokens (right) in the same color are processed in one attention map.

Table 1: Hyperparameters of the VQGAN used in PVDR.

Parameter	Setting
Resolution	128×128
Visual token grid size	16×16
Codebook size	256
Code dimension	1024
Codebook loss weight	1.0
Discriminator loss weight	0.8
Discriminator loss start step	10000
Number of downsampling blcoks	4
Number of residual blocks	2
Channel multiplier	$(1, 2, 2, 4)$
Channel in & out	in:160 out:64
Minibatch size	1600
Optimizer	Adam
Optimizer: learning rate	$4.5\mathrm{e} - 6$
Dropout rate	0.1
Training steps	4e5

A.2 Hyperparameters

In this section, we list the hyperparameters in our PVDR implementation. The hyperparameters of the pre-trained VQGAN we used are shown in Table 1. The hyperparameters on structures and learning of ST Transformers in PVDR are shown in Table 2. The hyperparameters of the PPO training are shown in Table 3. In fact, the hyperparameters of the PPO baseline are identical to those shown in Table 3. In particular, there are slight differences in our PPO training of different Meta-World tasks. Here, we list the tasks of 3 different hyperparameter sets. **Task Set 1**: button press topdown, dial turn, drawer close, reach, window

Table 2: Hyperparameters of the ST Transformer used in PVDR.

Parameter	Setting
Visual token grid size	16×16
Local window size	4×4
Context frames length m	2
Future frames length l	6
Visual dynamics representation shape	$16 \times 16 \times 32$
Attention dimension	512
Feedforward dimension	512
Token embedding dimension	1024
Visual token vocabulary size	256
Encoder Transformer depth	6
Encoder Transformer heads	4
Prior Transformer depth	6
Prior Transformer heads	4
Decoder Transformer depth	3
Decoder Transformer heads	4
Action alignment Transformer depth	3
Action alignment Transformer heads	4
Critic Transformer depth	3
Critic Transformer heads	4
Loss weight λ_1	$1\text{e}-3$
Loss weight λ_2	2.5 (RLBench); 0.8 (Meta-World)
Reward weight λ_3	4.5 (RLBench); 1.0 (Meta-World)
Minibatch size	1025
Optimizer	Adam
Pre-training learning rate	$1\text{e}-4$
Fine-tuning learning rate	$1\text{e}-5$
Downstream training rate	$4\text{e}-5$
Pre-training steps	3.5e5

open. **Task Set 2**: button press, plate slide, plate slide back, window close. **Task Set 3**: plate slide side, hand insert, drawer open.

A.3 Task Settings

We select 12 tasks from Meta-World and 3 tasks from RLBench for our experiments. As we consider goal-based tasks, we replace the original dense reward in Meta-World with the sparse reward. A positive r_{suc} will be given when the goal is achieved. Concretely, r_{suc} is 100 in RLBench tasks and 60 in Meta-World tasks. And the visual goals of the tasks are shown in Figures 10 and 11.

Table 3: Hyperparameters of PPO in PVDR.

Parameter	RLBench	Task Set 1	Task Set 2	Task Set 3
λ	0.92			
γ	0.99			
ϵ	0.4	0.6	0.2	0.5
ϵ_{value}	10			
$c_{entropy}$	$3e-4$	$1e-2$	$2e-4$	$5e-3$
c_{value}	0.5			
Max gradient norm	1.0			
Minibatch size	100			
Actor learning rate	$3e-4$			
Critic learning rate	$1e-3$			
Training Epochs	500	200	200	200

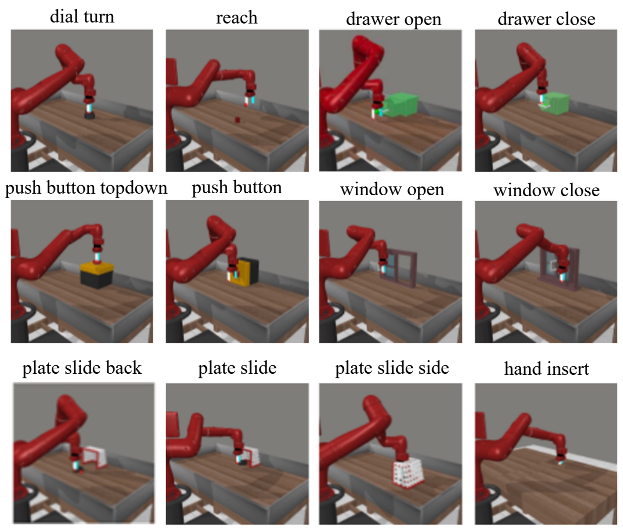

Fig. 10: Illustration of the image goals in Meta-World tasks.

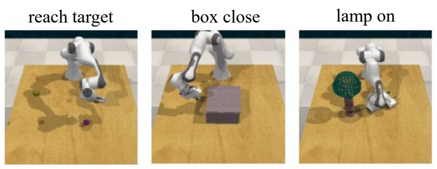

Fig. 11: Illustration of the image goals in RLBench tasks.

B Additional Experiments

B.1 Hyperparameters Influence

Given that PVDR contains three weight factor hyperparameters ($\lambda_1, \lambda_2, \lambda_3$), we conduct additional experiments to explore the impact of these factors on the experimental results. Specifically, we experiment with a range of values for these three factors on four Meta-World tasks, and the learning curves are shown in Figures 12 to 14. The performance of PVDR is shown to be quite sensitive to the value of λ_1, relatively stable to the value of λ_2, and moderately influenced by the value of λ_3.

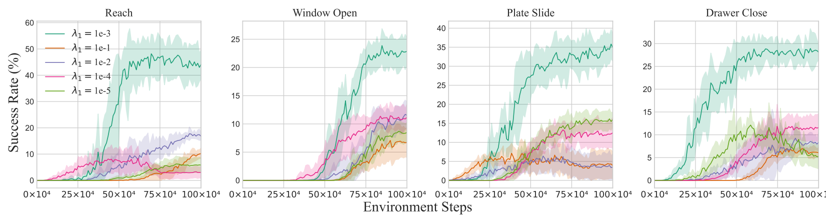

Fig. 12: Learning curves of different λ_1 value in PVDR on four Meta-World tasks measured on success rate. The solid line and the shaded regions represent the mean and variance of performance across five runs with different seeds.

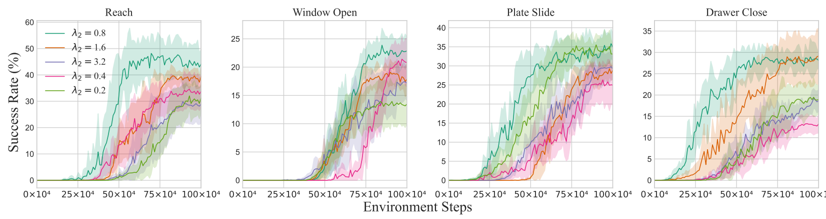

Fig. 13: Learning curves of different λ_2 value in PVDR on four Meta-World tasks measured on success rate. The solid line and the shaded regions represent the mean and variance of performance across five runs with different seeds.

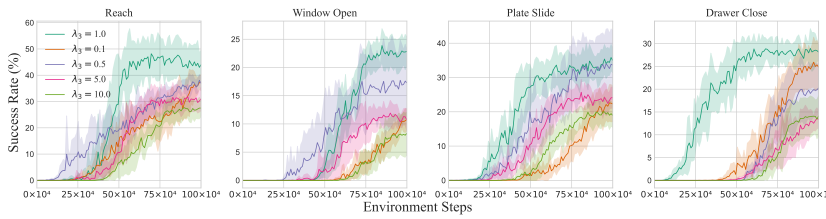

Fig. 14: Learning curves of different λ_3 value in PVDR on four Meta-World tasks measured on success rate. The solid line and the shaded regions represent the mean and variance of performance across five runs with different seeds.

Table 4: Success rates (%) of PVDR with various horizon lengths in 2 Meta-World tasks. The means and variances of the performance over five runs with different seeds are reported.

Horizon Length $(\mathbf{m}+\mathbf{n})$	$\mathbf{2+6}$	$\mathbf{1+1}$	$\mathbf{2+2}$
Reach	45.0 ± 1.3	7.8 ± 1.5	11.0 ± 4.1
Plate Slide	35.2 ± 4.2	7.6 ± 2.2	11.4 ± 3.4

In addition, we conduct ablation studies on the horizon length $(m+n)$ in PVDR. As shown in Table 4, we find that shorter horizons negatively impact the performance. This indicates that encoding more abstract and compressed information from longer horizons is beneficial.

B.2 Various Dataset

We also conduct additional experiments to evaluate the generalization ability of PVDR using various datasets: trajectory dataset collected in Meta-World, Open-X Embodiment datasets [45] (USC Jaco Pay and Berkeley Cable Routing). The size of each used dataset is consistent with the BAIR dataset to ensure fairness. For the four datasets (including BAIR) we conduct experiments on, they represent different quality of the pre-training datasets:(1) **Meta-World Dataset** is in-domain. (2) **BAIR Dataset** is task-relevant but has a visual gap. (3) **USC Jaco Pay Dataset** uses Jaco robot in less relevant tasks. (4) **Berkeley Cable Routing Dataset** uses Franka robot in totally different tasks. The results are shown in

Table 5, indicating different pre-training datasets impact the performance and generalization, and PVDR works effectively across various datasets. The benefits of pre-training are limited by the increasing gap, which necessitates capturing the more generalizable dynamics priors. Our results indicate that PVDR effectively bridges the gap across different pre-training datasets and downstream tasks.

B.3 Discussion about ContextWM

In our baseline selection, we choose APV instead of ContextWM, which is somehow an extension of APV. ContextWM introduces a representation of the visual context to 'facilitate knowledge transfer between distinct scenes' and thus enables the leverage of videos from multiple sources. This mechanism can be incorporated in PVDR by adding an extra condition (context representation) during the video prediction. We don't compare PVDR with ContextWM because BAIR dataset doesn't contain distinct scenes. Instead, the single-scene dataset may hinder the learning of meaningful context representation. We conduct additional experiments with ContextWM and PVDR with context representation. The results in Table 6 show that incorporating context representation indeed worsens performance in the current setting.

Table 5: Success rates (%) of PVDR in Meta-World with 4 various pre-training datasets. The datasets are represented with the following numbers: (1) **Meta-World Dataset**, (2) **BAIR Dataset**, (3) **USC Jaco Pay Dataset**, and (4) **Berkeley Cable Routing Dataset**. The means and variances of the performance over five runs with different seeds are reported.

Dataset		(1)	(2)	(3)	(4)
Reach	PVDR	$\mathbf{65.4 \pm 5.9}$	$\mathbf{45.0 \pm 1.3}$	$\mathbf{47.2 \pm 3.2}$	$\mathbf{39.0 \pm 4.1}$
	APV	64.2 ± 4.4	26.8 ± 3.4	35.8 ± 2.6	21.2 ± 1.6
	FICC	36.4 ± 3.8	5.4 ± 2.3	4.2 ± 2.8	0.0 ± 0.0
Plate Slide	PVDR	$\mathbf{56.0 \pm 2.5}$	$\mathbf{35.2 \pm 4.2}$	$\mathbf{30.8 \pm 2.7}$	$\mathbf{22.2 \pm 2.2}$
	APV	38.8 ± 1.6	20.4 ± 2.6	26.6 ± 2.2	16.8 ± 0.8
	FICC	22.4 ± 3.7	5.2 ± 3.7	3.0 ± 1.1	0.0 ± 0.0
Hand Insert	PVDR	$\mathbf{45.8 \pm 3.7}$	$\mathbf{28.6 \pm 3.0}$	$\mathbf{27.2 \pm 1.3}$	$\mathbf{17.8 \pm 2.0}$
	APV	43.6 ± 2.4	27.6 ± 1.4	26.0 ± 0.6	12.6 ± 6.1
	FICC	14.2 ± 1.2	4.6 ± 2.0	2.4 ± 1.4	0.0 ± 0.0

Table 6: Success rates (%) of PVDR, PVDR with context representation and ContextWM in 2 Meta-World tasks. The means and variances of the performance over five runs with different seeds are reported.

	PVDR	PVDR with context representation	ContextWM
Reach	45.0 ± 1.3	31.0 ± 2.4	26.0 ± 1.7
Plate Slide	35.2 ± 4.2	22.0 ± 2.6	20.6 ± 2.2

C Cases Visualization

In this section, we showcase the visualization of some cases from the pre-training video dataset and downstream tasks in Figures 15 to 17. Additionally, we provide some examples in Figure 18, showing the effectiveness of the online adaptation. As is shown in Figure 18.(a), online adaptation helps to generate more reasonable visual plans. Furthermore, the gradually converging action alignment reward curves in Figure 18.(b) indicate the effectiveness of the action alignment mechanism.

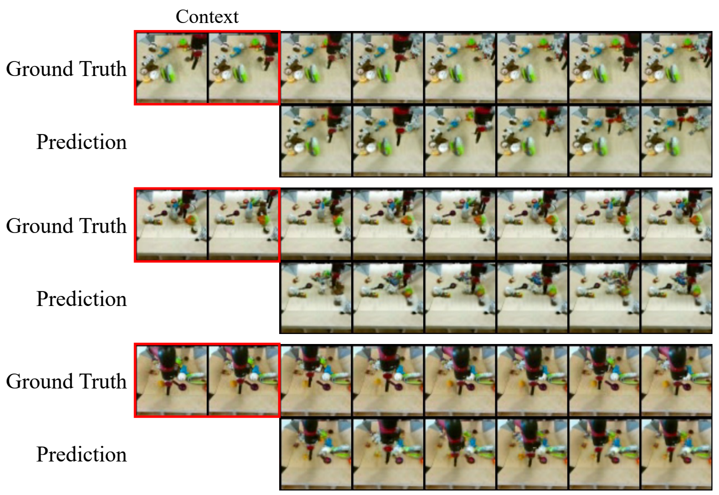

Fig. 15: Future frames predicted by the pre-trained visual dynamics model on three cases from the BAIR dataset. The pre-trained model is capable of capturing the dynamics prior in the pre-trainig video datasets.

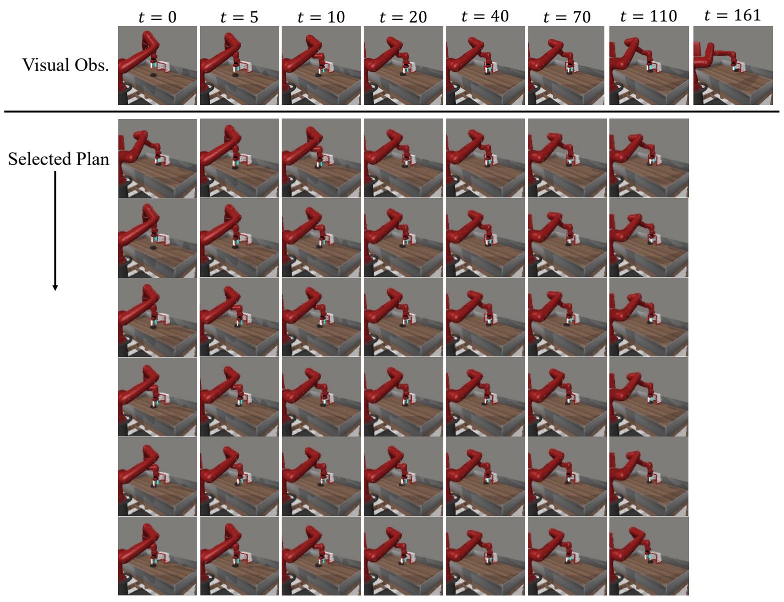

Fig. 16: Visual observations and selected plans alongside the interaction of PVDR in the *plate slide* task from Meta-World environment. PVDR is capable of generating meaningful plans with dynamics information and executing relevant actions.

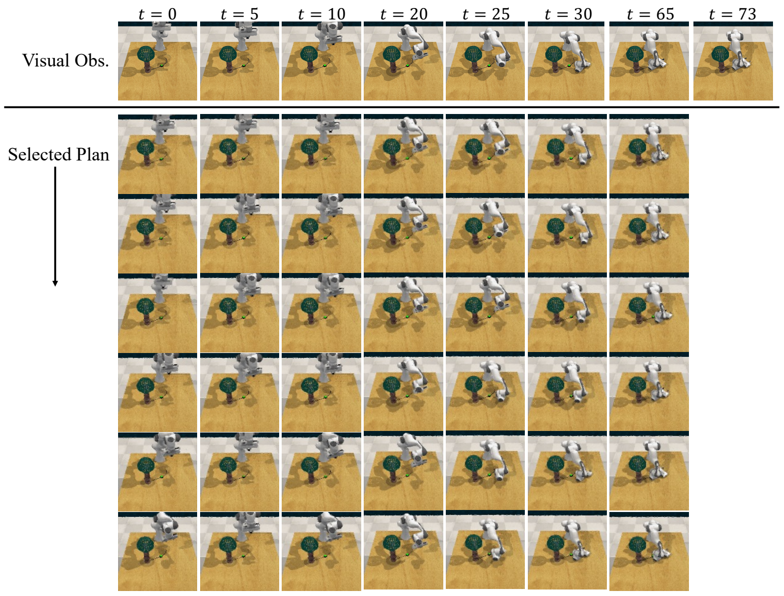

Fig. 17: Visual observations and selected plans alongside the interaction of PVDR in the *lamp on* task from RLBench environment. PVDR is capable of generating meaningful plans with dynamics information and executing relevant actions.

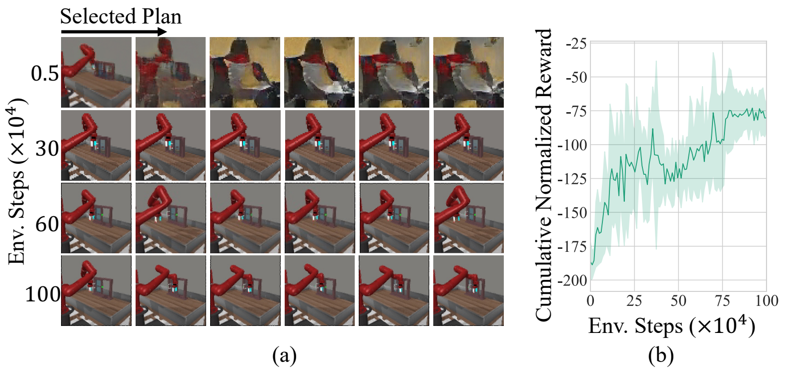

Fig. 18: Examples from the adaptation in *window close* task: selected visual plans (a) and action alignment reward curves (b).

D Extended Related Works

Our work involves video prediction as well as goal-conditioned RL in addition to RL pre-training with videos. We are here to discuss the works related to these two topics. In addition, some works [15, 26, 31, 60, 75] pre-train modules with aligned video-text pairs to assist in solving tasks conditioned on textual goals, which is not within the scope of our work.

Video Prediction. As a complex amalgamation of visual comprehension and temporal sequence prediction, video prediction tasks aim to predict future frames given context frames. Generally, a series of generative models, prominently generative adversarial networks [17, 34], variational autoencoders [1, 72, 73], autoregressive models [80, 81, 86], and diffusion models [8, 28, 36], have been shown instrumental in surmounting this challenge. Furthermore, many works [8, 63, 71, 74] consider the incorporation of text prompts/descriptions to aid in video generation. Our work mainly builds up the video prediction structure through a combination of earlier VAE-style works [1, 72] and some recent works [24, 79] based on spatial-temporal attention.

Goal-Conditioned Reinforcement Learning. Our work uses the goal-conditioned RL framework. Unlike the setting based on a reward function, agents are provided with a behavior goal of the task in goal-conditioned RL. The goals can be images, feature vectors, language, commands, intrinsic skills, and so on, which are more general and flexible than the sometimes hard-to-define reward function. Many works address the challenge from various perspectives, such as optimization based on the divergence to the goal [39, 44, 53, 68], sub-goal generation/ selection [41, 42], and relabeling techniques [2]. Our work uses the most common and direct approach [19, 49, 77] to measure the distance between observation and goal. Our choice accommodates the nature of the tasks in our experiments, and more delicate methods may help when handling more complex tasks.

E Limitation Discussion

Despite being an effective method for *pre-training with videos*, PVDR has some limitations. Firstly, the learning of the reward function is not included in PVDR, and instead a preset reasonable reward function is assigned. Such a direct designation may not provide valid guidance for more complex tasks. In fact, some works have developed the usage of the foundation model as a reward function, which has the potential to be integrated to address the lack of a learned reward function. In addition, the pre-training videos are limited to a single source currently, and incorporating Internet videos from a wider range of sources may lead to a more general and capable agent. ContextWM [76] actually already provides an effective way to extend single-source video data to large wide-source datasets through the introduction of context representations, and the introduction of this mechanism to PVDR may address this limitation.