APPENDIX

Table of Contents

A CODE

The code for reproducing our results is included in `./codebase_AVDC` directory of the attached supplementary .zip file.

B EXTENDED QUALITATIVE RESULTS

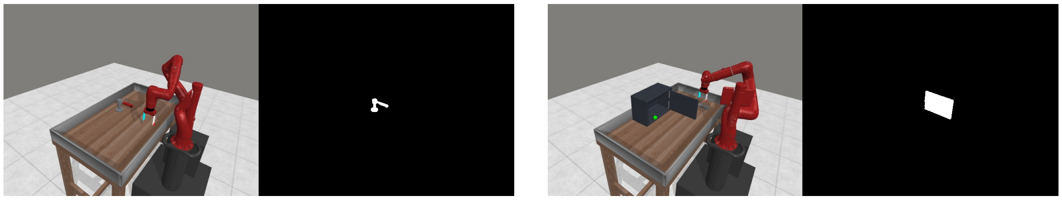
Our supplementary website presents additional qualitative results, including

- **Synthesized Videos**: Meta-World, iTHOR, Visual Pusher, and Bridge.
- **Task Execution Videos**: Meta-World, iTHOR, Visual Pusher, and the real-world Franka Emika Panda tasks.

C ZERO-SHOT GENERALIZATION ON REAL-WORLD SCENES

While most tasks in the Bridge data were recorded in toy kitchens, we found that the video diffusion model trained on this dataset already can generalize to complex real-world kitchen scenarios, producing reasonable videos given RGB images and textual task descriptions. Examples of the synthesized videos can be found on our supplement website.

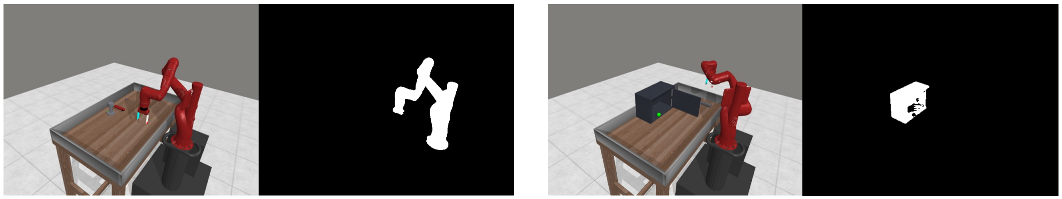
Successful mask prediction



Input text: small faucet handle

Input text: gray door

Failed mask prediction



Input text: small faucet handle

Input text: gray door

Figure 11: **Object Mask with Segmentation models.** Successful and failed object masks extracted by Language Segment Anything.

D OBJECT MASK WITH SEGMENTATION MODELS

We experimented with utilizing existing segmentation methods to extract the object mask for our action regression algorithm. We employed Language Segment-Anything (Medeiros, 2023), which is based on GroundingDINO (Liu et al., 2023) and Segment-Anything (Kirillov et al., 2023). The experiment was conducted in our Meta-World setting, using the predicted object mask instead of the GT object mask. In this setup, the achieved success rate averaged over all 11 tasks is 34.5%, which is 8.6% lower than the success rate using the GT object mask (43.1%). The performance drop is attributed to incorrect object masks produced by the object segmentation model. Qualitative object segmentation results are presented in Figure 11.

E ADDITIONAL ABLATION STUDIES

E.1 FIRST-FRAME CONDITIONING

To study the effectiveness of our first-frame conditioning strategy, we conducted a quantitative experiment that compares our RGB-channel-wise concatenating strategy with a trivial baseline: frame-wise concatenate, which concatenates the input frame before the first frame of the noisy video. We calculated the mean squared error (MSE) between the last frame of the ground video and the last frame of the synthesized video to evaluate the quality of synthesized videos. We show the results in Figure 12. Each data point is an average MSE calculated with 4000 samples of video generation, and the error bar shows the standard error of MSEs. Our method (cat_c) consistently outperforms frame-wise concatenating (cat_t) in the early stages of training on the Bridge dataset. Some qualitative generation results can be found on our supplementary website.

E.2 TEXT ENCODER

We compare the video generation quality of our CLIP-text encoder (63M parameters) with the same model but with a T5-base encoder (110M parameters), dubbed AVDC (T5-Base). We used the same pixel-level MSE error as the evaluation metric. Figure 12 shows the result. The difference between the performance of the two text encoders is not significant.

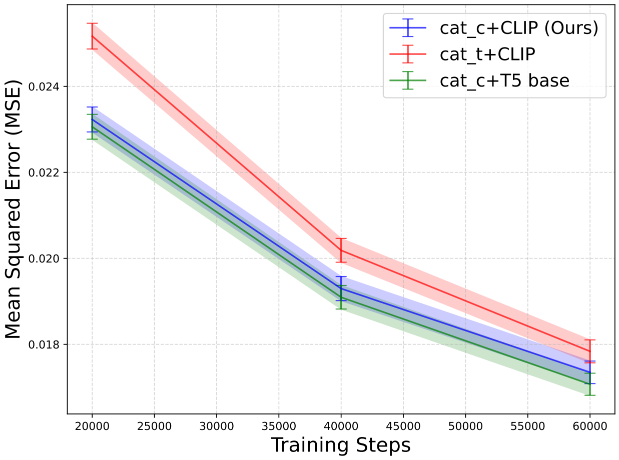


Figure 12: **Additional Ablation Studies on the First Frame Condition Strategy and Text Encoders.** We ablate the first frame conditioning strategy and compare the performance of different text encoders. Specifically, we calculate the MSE between the last frame of a ground truth video and the last frame of a synthesized video. **cat_c (Ours)**: the first frame is concatenated with a noisy video in RGB dimension, which is our proposed method. **cat_t**: the first frame is concatenated with a noisy video in time dimension. **CLIP**: CLIP text encoder (63M). **T5**: T5 base encoder (110M).

F MODEL ARCHITECTURE AND TRAINING DETAIL

F.1 TEXT ENCODER

We used a fixed pre-trained CLIP-Text encoder for encoding text descriptions. After encoding a text description with the encoder, we employ (Perceiver; Jaegle et al., 2021) as our attention-pooling network to aggregate the output tokens from the CLIP-text encoder into one single vector and added it to the time embedding of the diffusion model. This simple condition mechanism has been successful in our experiments. We did not use cross-attention on text inputs throughout this work. The hyperparameters of Perceiver are listed in Table 3.

Parameter	Value
layers	2
num_attn_heads	8
num_head_channels	64
num_output_tokens	64
num_output_tokens_from_pooled	4
max_seq_len	512
ff_expansion_factor	4

Table 3: Model parameters for our Perceiver.

F.2 VIDEO DIFFUSION MODEL

For all models, we use dropout=0, num_head_channels=32, train/inference timesteps=100, training objective=predict_v, beta_schedule=cosine, loss_function=l2, min_snr_gamma=5, learning_rate=1e-4, ema_update_steps=10, ema_decay=0.999. We list all other hyperparameters in Table 4.

G HARDWARE AND COMPLEXITY ANALYSIS

G.1 TRAINING COST

We train all models on 4 V100 GPUs with 32GB memory each. Our claim for training video policies within a day refers to training a diffusion model from scratch on small, environment-specific datasets

	Meta-World	**iTHOR**	**Bridge**
num_parameters	201M	109M	166M
resolution	(128, 128)	(64, 64)	(48, 64)
base_channels	128	128	160
num_res_block	2	3	3
attention_resolutions	(8, 16)	(4, 8)	(4, 8)
channel_mult	(1, 2, 3, 4, 5)	(1, 2, 4)	(1, 2, 4)
batch_size	16	32	32
training_timesteps	60k	80k	180k

Table 4: Comparison of configuration parameters for Meta-World, iTHOR, and Bridge.

like in Meta-World and iTHOR experiments. As for a much larger dataset like Bridge, we have to train for longer to obtain consistent results. Despite the large size of Bridge data compared to the datasets we used in the other two experiments, we can generate high-quality and consistent results with just two days of training.

- Meta-World: about 24 hours of training (165 videos)
- iTHOR: about 24 hours of training (240 videos)
- Real world experiment: 48 hours of pre-training on Bridge and 4 hours of fine-tuning on human data. (40000 Bridge + 20 Human videos)

G.2 INFERENCE COST

We conducted our experiments (inference) on a machine with an RTX 3080Ti as GPU. We provide a detailed run time breakdown on Meta-World experiment of each step of our method below.

- **Text-Conditioned Video Generation**: Synthesizing a video of predicted execution is the most time-consuming step of our method, which takes roughly **10.57** seconds (**1.51** seconds per video frame on average).
- **Flow Prediction**: Predicting optical flow between a pair of two subsequent frames takes **0.28** seconds on average.
- **Action Regression from Flows and Depths**: Inferring the action from optical flow prediction **1.31** seconds on average.
- **Action Execution**: Running an inferred action using the controller in the environment takes **1.53** seconds on average.

G.3 REPLANNING COST

In Meta-World experiments, AVDC used about 18 seconds for each round of action planning. Since the maximum number of replans is set to 5, the number of action planning rounds within an episode varies from 1 to 6. Therefore, the total planning cost ranges from about 18 to 108 seconds on a Nvidia GeForce RTX 3080Ti GPU.

G.4 IMPROVING INFERENCE EFFICIENCY WITH DENOISING DIFFUSION IMPLICIT MODELS

We can incorporate various techniques into our method to improve its inference efficiency. To speed up the video synthesis step, we can progressively distill the diffusion models for faster sampling (Salimans & Ho, 2022). Also, we can leverage lighter-weight optical flow prediction models to increase efficiency. To accelerate action prediction from flows, we can design more sophisticated techniques for sampling and optimizing actions or parallelizing them using GPUs.

This section investigates the possibility of accelerating the sampling process using Denoising Diffusion Implicit Models (DDIM; Song et al., 2021). To this end, instead of iterative denoising 100 steps, as reported in the main paper, we have experimented with different numbers of denoising steps (*e.g.*, 25, 10, 5, 3) using DDIM. The qualitative results of synthesized videos are presented on our supplementary website.

We have found that reducing the number of denoising steps to 10 still leads to satisfactory generated video quality while resulting in a 10x speedup in video generation. Specifically, the overall mean success rate across tasks in Meta-World with 10-step DDIM is **37.5%**, which is competitive with our original method with 100 denoising steps with an overall success rate of **43.1%**. That said, when the task is running time-critical, we can speed up the video generation step by ten times with only **5.6%** drop in task performance.

H DETAILS ON EXPERIMENTAL SETUP

This section describes experimental details, including learning diffusion models, inferring actions, replanning strategies, etc.

H.1 META-WORLD EXPERIMENTAL SETUP

This section describes the details of the Meta-World experiments.

H.1.1 LEARNING THE DIFFUSION MODEL

We aim to learn a video diffusion model that can synthesize a video, showing a robot fulfilling a task, from an initial frame and a the task described in natural language. We found that in most goal-conditioned manipulation tasks, the final frame of the whole video is often highly correlated to the text description when the current (first) frame is given. In other words, the model can easily synthesize the last frame given the current frame and text description, while the model is often more uncertain about intermediate frames and therefore performs poorly in synthesizing intermediate frames.

To take advantage of this finding, we propose an *adaptable frame sampling technique* to sample frames from the whole video for training. In particular, we first randomly sample a frame from the whole video dataset as the current frame. We then uniformly sample $T - 2$ frames from the current (*i.e.*, initial) frame to the final frame from the same video. Then, we use these T frames (1 current/initial frame, $T - 2$ intermediate frames, and 1 final frame) to train our video diffusion model. We empirically found that this *adaptable frame sampling technique* significantly improves the learning efficiency of the video diffusion model, enabling the training to finish within a single day using just 4 GPUs.

H.1.2 CALCULATING OBJECT RIGID TRANSFORMATIONS AND INFERRING ACTIONS

Given the predicted optical flow between each pair of frames of a synthesized video and the initial frame, we aim to infer a robot's actions to follow the synthesized video.

Tracking Object and Determining Contact Point. Since Meta-World focuses on manipulating objects, we propose to track an object of interest by extracting an object mask. To determine the contact point for the robot to grasp an object, we simply sample $N = 500$ points from the object mask and compute the centroid of an object as the contact point. Note that more sophisticated methods for determining contact points can be employed to further improve the proposed method.

Calculating Object Rigid Transformation Object and Computing Subgoal. Given the optical flow computed from the synthesized video frames, we can use it to compute the 2D correspondence between two frames. We use the RANSAC algorithm to find an optimal 2D transformation that produces the most inliners from the 2D correspondences. We only use these inliner points for the computations in the current and the following steps for better robustness. We apply our method as described in Section 3.3 to obtain a sequence of 3D rigid transformations. Then, we apply these transformations on the sampled grasp sequentially to obtain a sequence of subgoals, indicating how the object should be moved.

Inferring Actions. Given each subgoal, we decide whether to use "`grasp`" action or "`push`" action to interact with the object by checking if the maximum magnitude of vertical displacement exceeds 10cm based on the heuristic that pick-and-place tasks usually require the robot to lift the object, which produces a vertical displacement; on the other hand, optimal object trajectories of pushing tasks do not exhibit such vertical displacement.

Once we decide if the robot should "`grasp`" or "`push`" the object, we determine the robot arm's action as follows. For the `grasp` mode, we simply control the robot to take a grasping action (closing the grippers) at the grasp point and then move toward the subgoals. For the `push` mode, we put the robot arm in a specific direction to the object that allows pushing before moving the robot towards the subgoals. We calculate such direction by extrapolating the line between the grasp point and the first subgoal more than 10cm away from the grasp.

H.1.3 REPLANNING STRATEGY

In Meta-World, we replan (*i.e.*, perform the closed-loop control) by synthesizing a video with the current observed state as the initial frame for the video diffusion model. Then, we use the current object mask and depth information to compute a new sequence of subgoals. Note that we do not re-decide the interaction mode. For the `grasp` mode, we simply move the gripper toward the new subgoals. For the `push` mode, we re-initialize the gripper as described above and then move the gripper toward the new subgoals.

H.2 ITHOR EXPERIMENTAL SETUP

This section describes the details of the iTHOR experiments.

H.2.1 LEARNING THE DIFFUSION MODEL

We aim to learn a video diffusion model that can synthesize a video that shows an agent navigating to a target object in first-person point of view in iTHOR indoor scenes. To sample video segments for training, we first randomly sample a frame from the video demonstration dataset, and then we retrieve $T - 1$ consecutive future frames from the same video. We do not skip any intermediate frames (*i.e.*, we do not apply the *adaptable frame sampling technique* used in Meta-World). If the number of subsequent frames is less than $T - 1$ in the sampled video, we duplicate the last frame to compensate for missing frames. This allows the model to recognize that the target object is found and the agent should stop moving.

H.2.2 CALCULATING SCENE TRANSFORMATIONS AND INFERRING ACTIONS

Tracking Scene and Calculating Scene Transformation. In the navigation setup, instead of tracking the correspondences of a particular object, we track the correspondences of the entire scene. To this end, instead of generating an object mask, we initialize a scene mask by thresholding out moving points (magnitude of optical flow > 1). Then, to calculate the scene transformation, we apply a similar procedure to the Meta-World experiment for computing the object transformation. However, we do not use the RANSAC algorithm to obtain inliners; instead, at each time step, we simply remove the key points that move out-of-bound (*i.e.*, outside the image) and keep the rest as the inliner points to calculate the scene transformation.

Inferring Actions. Given calculated the scene transformation at each step, we design a procedure to infer an action (`MoveForward`, `RotateLeft`, `RotateRight`, or `Done`). We propose to observe an *imaginary point* located 1 meter in front of the robot. We apply the calculated scene transformation on this *imaginary point*. We then decide the action based on the translation of this *imaginary point* before and after applying the transformation. Specifically, if the translation is close to 0 ($<$ 1mm), since the agent stays still, we choose `Done`. Otherwise, we check the horizontal displacement of this *imaginary point*. If the magnitude of the horizontal displacement is less than 25cm, we choose `MoveForward`. Otherwise, we select `RotateLeft` or `RotateRight` depending on the direction of the displacement.

H.2.3 REPLANNING STRATEGY

In iTHOR, we replan when we lose track of most correspondences from the initial frame. Specifically, if the number of inliners we keep is less than 10% of the original number we sampled, we re-synthesize a video, predict optical flow, and infer actions.

H.3 REAL-WORLD FRANKA EMIKA PANDA ARM EXPERIMENTAL SETUP

This section describes the details of the real-world Franka Emika Panda arm experiments.

Hardware. Our arrangement comprises a Franka Emika Panda arm and an Intel Realsense D435 RGBD camera mounted at a fixed frame relative to the table. The robot arm is equipped with a parallel motion two-jaw gripper, and the robot arm is in joint position control mode. Meanwhile, the camera has calibrated intrinsic and extrinsic. Therefore, any object motion predicted in the camera frame can be directly transformed into the world frame.

Dataset Collection. After training on the Bridge dataset, we fine-tuned the model with 20 human demonstrations in a real-world tabletop manipulation setting. These videos are collected by humans using their hands to move objects on the table and accomplish tasks. Our object set includes plates, bowls, a few categories of fruits (apples, oranges, bananas, peaches, and mangoes), and utensils such as forks and knives as distractors. The task is to pick up fruits from their initial locations and place them in the specified container, a plate or a bowl.

Action Prediction and Execution. In our real-world evaluation, we assume that the target object can be grasped using a top-grasp and that no re-orientation of the target object is needed. Therefore, in order to compute the target object poses, we first manually specify the segmentation of the object (in principle, it can be done using other object segmentation models, too), extract the corresponding optical flow, and compute the sequence of the object poses. We extract its initial pose (in the first frame) and the target pose (in the last frame), and generate a robot arm trajectory using an inverse-kinematics (IK) solver. In practice, we found that since we are using only a small set of tracking points (objects are small in our camera view), the reconstruction of 3D rotations is not robust. This can be potentially addressed by leveraging a higher-resolution video generative model or simply, different camera configurations.

Failure Mode Analysis In our real-world experiments, we found that our approach failed in 8 of the 10 tested trials. We found that 75% of the failures were caused by the wrong plan from the video diffusion model. It either picked the wrong object or placed it at the wrong target. The other 25% of the failures were caused by the discontinuity of video generation. The generated plan seems to be correct, but the object disappeared in some intermediate frame, which eventually led to the failure.