ACTION-SPACE VELOCITY CORRECTION FOR VISUAL CONSISTENCY IN ROBOTIC FLOW POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Visuomotor policies trained under one visual appearance degrade in performance when lighting, color, or background change during inference. Retraining or augmenting the policy for every new appearance is costly. In this paper, we introduce Action-Space Velocity Correction (ASVC), an inference-time correction to the velocity of a flow-matching policy that corrects based on a reference view of the observation. Transport conditioned on reference views provides consistency of visual variation in action space. ASVC modifies only the integration of a frozen policy, requiring no retraining and applying as an add-on to any flow-matching action policy. Our method is validated on a toy reaching task and on the DMC-GB2 benchmark across three flow-matching base policies: ReinFlow, GFP, and Diffusion Policy. We further deploy ASVC on a Trossen WidowX AI robot for a cube-stacking task with variation in nearby distractor objects and lighting. Across these settings, ASVC recovers much of the return lost under color and video shifts while performance on the training appearance is preserved.

1 INTRODUCTION

Visuomotor policies trained under one visual appearance degrade in performance when lighting, color, or background change at inference (Packer et al., 2018; Cobbe et al., 2019; Stone et al., 2021). Across these variations, the task is unchanged. So dynamics, rewards, and optimal actions are determined by the underlying state of the scene rather than appearance. Thus loss in performance is attributed to the observation's visual shift, due to the encoder mapping the observation to a representation the policy was never fitted on. Retraining or augmenting the policy for every new appearance is costly, and a policy already deployed has to remain fixed.

Existing solutions reshape training through augmentation or auxiliary objectives for ensuring consistency of a policy. Augmenting the observation distribution with random crops, shifts, and convolutions widens the appearance representation of the encoder (Laskin et al., 2020b; Yarats et al., 2021; 2022), while auxiliary objectives shape the latent space for visual invariance (Zhang et al., 2021; Laskin et al., 2020a). Both require retraining the policy for each new objective or data set. Adapting at inference avoids the cost, although the parameters acting during rollout are different from the parameters after training (Hansen et al., 2021). Mapping the observation back to the source appearance leaves the policy fixed (James et al., 2019), although errors of the mapping propagate directly to the action. Each of these options changes either the policy or the observation it reads, leaving the action itself untouched.

In this paper, we introduce Action-Space Velocity Correction (ASVC), an inference-time method that corrects the velocity field of a flow-matching policy using a reference view of each observation. During inference, a flow-matching policy generates an action by integrating over a learned velocity field for several Euler steps, correcting each transport step for consistency. Given a reference map trained to re-render the observation with the source appearance, we evaluate the field a second time under the reference view and obtain the direction in which visual variation moves the transport. Where the observed transport opposes the reference, we correct it toward the reference transport, otherwise leave it unchanged. Transport conditioned on reference views provides consistency of visual variation in action space. ASVC modifies only the integration of a frozen policy, requires no retraining, and is applied as an add-on to flow-matching action policies.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We validate ASVC on a toy reaching task and on the DMC-GB2 benchmark, using both ReinFlow and GFP as base policies. Additionally, we deploy ASVC on a Trossen WidowX AI robot for a cube-stacking task with variation in nearby distractor objects and lighting. Across these settings, ASVC recovers much of the return lost under color and video shifts, while performance over the training appearance is preserved. Our contributions are as follows:

1. Framing visual consistency as a property of the action transport of a flow-matching policy.
2. Introduction of an inference-time correction of the action-space transport from a reference view of a frozen policy.
3. Evaluation on a toy reaching task, three DMC-GB2 tasks across three base policies, and a cube stacking task on a Trossen WidowX AI robot.

2 PRELIMINARY

Flow matching models learn a velocity field transporting a source distribution onto a desired data distribution (Lipman et al., 2023). Given data sample x_1 and noise x_0 drawn from a source distribution, the rectified-flow path between x_0 and x_1 is given by

$$x_t = (1 - t)\, x_0 + t\, x_1, \qquad t \in [0, 1]. \tag{1}$$

For our use case, we choose a unit Gaussian as the source distribution as is common with other papers. The field f_θ is regressed onto the constant velocity $x_1 - x_0$ of the path, thus minimizing the squared error over t, x_0, and x_1. For data sampling, a flow-matching model integrates f_θ from $t = 0$ to $t = 1$, which in a discrete case is performed with K Euler steps of size $1/K$, where after integration x_K is the sample onto the data distribution.

A subset of flow-matching policies use the action space as their target data distribution, thus conditioning the field over the agent's observations. Given observation o, the policy integrates a noise sample over $f_\theta(a, t \mid o)$. A policy samples an action via the endpoint of a transport. Many policies sample using action chunking, thus the transport runs over the full chunk and the agent executes a subset of the chunk.

3 METHOD

In this section, we formulate visual robustness as a consistency property for the action transport of a flow-matching policy. We derive ASVC as a per-step, inference-time correction that corrects for consistency on a frozen policy. At each step, ASVC determines the direction the action transport opposes a reference view, and repairs the transport toward the reference transport. Figure 1 gives an overview.

3.1 PROBLEM SETTING

Our method considers visuomotor policies. We denote the physical state as s, which collects the joint configuration, and the visual appearance as v, which covers factors such as lighting, surface color, background, and texture. The policy receives a rendering of both,

$$o = R(s, v). \tag{2}$$

During training the environment's appearance may remain static, while at test time it can vary in any of these ways. We consider only variations that do not modify the task, so given the same task we would like the policy to perform similarly regardless of visual variation.

Visuomotor policy π is fitted on renderings whose appearance lies in set $\mathcal{V}_{\text{train}}$, our training visual distribution, and deployed within set $\mathcal{V}_{\text{test}}$, our testing visual distribution, which is a superset of $\mathcal{V}_{\text{train}}$. Our task remains unchanged across visual variations, thus actor dynamics, rewards, and optimal actions are dependent on s alone, which we form into the property

$$\pi\big(R(s, v)\big) = \pi\big(R(s, v')\big) \quad \text{for all } v, v'. \tag{3}$$

If we train policy π on $\mathcal{V}_{\text{train}}$, then appearance shifts within $\mathcal{V}_{\text{test}}$ may result in suboptimal performance. Our goal is to enforce consistency at deployment-time on frozen π.

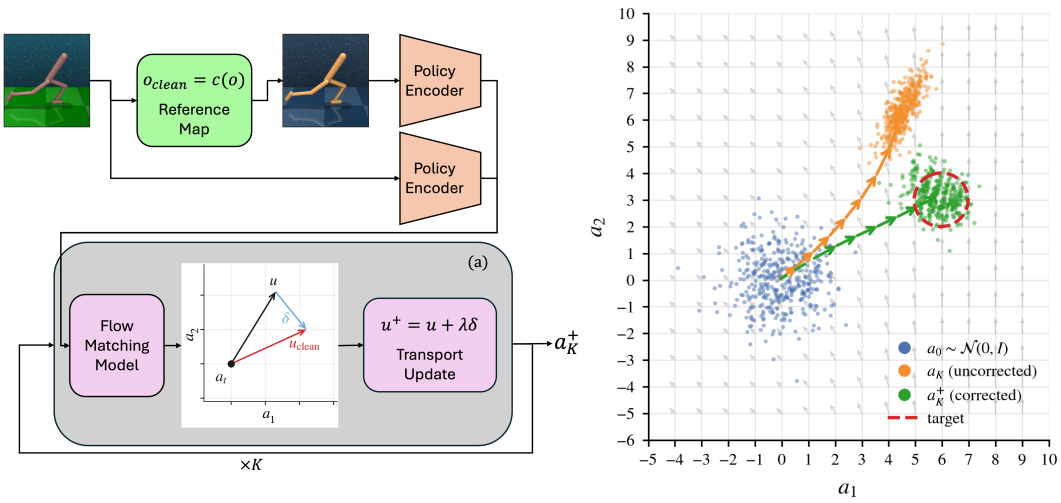

Figure 1: Overview of ASVC. **Left:** the observation and its reference view pass through the frozen policy encoder, and components within (a) show the K integration steps of a flow-matching policy. **Right:** source distribution a_0 integrated over $K = 8$ Euler steps toward the target on a synthetic two-dimensional Gaussian dataset. Under a time-dependent noisy velocity field, the flow is displaced and the uncorrected a_K ends off target. Background arrows indicate the direction of the injected noise. At steps where the observed transport opposes the reference, correction is applied and the step integrates toward the reference transport, thus a_K^+ lands on target.

3.2 Transport Residual

Flow-matching policies. We consider flow-matching policies. The policy samples $a_0 \sim \mathcal{N}(0, I)$ and integrates a_0 over a learned transport field $f_\theta(a, t \mid o)$ over K Euler steps,

$$a_{i+1} = a_i + \tfrac{1}{K} f_\theta(a_i, t_i \mid o), \qquad t_i = i/K. \tag{4}$$

Our policy operates over action a_K, the transport of our initial action in action space conditioned on our observation. As visual variation occurs within our observation, the transport of our flow-matching model is also affected, leading to suboptimal actions. Thus if we correct the transport at each Euler step toward the transport the policy would follow under an appearance from $\mathcal{V}_{\text{train}}$, we can recover the action without modifying the policy.

Reference map. To test consistency, we require a second cleaned rendering of the same state. We train a reference map $c : o \to o_{\text{clean}}$ that re-renders the observation so that $o_{\text{clean}} \approx R(s, v_{\text{clean}})$ with $v_{\text{clean}} \in \mathcal{V}_{\text{train}}$. We apply this map once per control step, and condition a second transport on it at every integration step. In our implementation, we use a small U-Net architecture trained on generic appearance jitter with no access to deployment appearance.

Appearance residual. At integration step t, we calculate both

$$u = f_\theta(a, t \mid o), \qquad u_{\text{clean}} = f_\theta(a, t \mid o_{\text{clean}}). \tag{5}$$

We refer to u as the observed transport, and u_{clean} as the reference transport. Thus their difference $\delta = u_{\text{clean}} - u$ is the appearance residual. Normalizing δ gives $\hat{e} = \delta/\|\delta\|$, which is the residual direction. Because both are evaluated at the same (a, t), the residual is a vector in the action space that separates the effect of appearance from the policy's own dynamics.

3.3 Residual Projection

The orthogonal decomposition of the observed transport is

$$u = \rho\,\hat{e} + u_\perp, \qquad \rho = \text{proj}_{\hat{e}}^{u}, \qquad u_\perp = \text{orth}_{\hat{e}}^{u}. \tag{6}$$

When $\rho > 0$, the observed transport points in the direction of the reference transport, and differs from it only in magnitude. When $\rho < 0$, the observed transport points away from the reference transport, so appearance is driving the policy in the wrong direction, and integrating that step compounds the error over the remaining steps.

To improve alignment, our goal is to minimize how much the observed transport points away from the reference transport. We impose the constraint $\rho \geq \varepsilon_{\text{dir}}$, where ε_{dir} sets the tolerance of transport deviation. This constraint constructs the space

$$\mathcal{C}_{\varepsilon_{\text{dir}}} = \{\, w : \text{proj}_{\hat{e}}^{w} \geq \varepsilon_{\text{dir}} \,\}. \tag{7}$$

Thus the smallest edit that satisfies the constraint is the closest point of $\mathcal{C}_{\varepsilon_{\text{dir}}}$ to u, which has closed form

$$P_{\varepsilon_{\text{dir}}}(u) = \underset{w \in \mathcal{C}_{\varepsilon_{\text{dir}}}}{\arg\min} \tfrac{1}{2}\|w - u\|^2 = u + (\varepsilon_{\text{dir}} - \rho)_+ \, \hat{e}. \tag{8}$$

This edit changes u only along $\hat{e}$, doing nothing when the transport already aligns with the reference transport. The transport is then corrected by

$$u^+ = (1 - \lambda)\, u + \lambda\, P_{\varepsilon_{\text{dir}}}(u) = u + \lambda\, (\varepsilon_{\text{dir}} - \rho)_+ \, \hat{e}, \tag{9}$$

where $\lambda \in (0, 1]$ controls how much we correct by.

3.4 Additive Correction

Each integration step guarantees the observed transport does not oppose the reference, constraining the magnitude along $\hat{e}$. Therefore the transport can still carry residuals within the orthogonal complement of $\hat{e}$. Denoting d as the dimension of the action space,

$$\dim \text{span}(\hat{e}) = 1, \qquad \dim \hat{e}^{\perp} = d - 1, \qquad \text{fraction of corrected directions} = \frac{1}{d} \xrightarrow[d \to \infty]{} 0. \tag{10}$$

Thus as d grows, the portion of the velocity to be corrected decreases. Due to this limitation within a subtractive system, using an additive system corrects the transport over all action space dimensions. During integration steps, the updated observed transport is given by

$$u^+ = (1 - \lambda)\, u + \lambda\, u_{\text{clean}} = u + \lambda\, \delta \tag{11}$$

In implementation, we only update the observed transport when $\rho < \varepsilon_{\text{dir}}$ and leave it otherwise. Using this update, with $\lambda = 1$ the update reduces to $u^+ = u_{\text{clean}}$. For every integration step, the corrected transport is either u or u_{clean}. ASVC is therefore a selector between the observed and reference transports. We fix $\lambda = 1$ in all high-dimensional experiments. Figure 1 (right) illustrates this selection on an analytic two-dimensional transport. Under a corrupted observation the observed transport aims at a displaced target, so the uncorrected endpoint lands off target. Firing on the late steps where the two transports oppose, the correction integrates the reference transport and lands the endpoint on the target.

3.5 Correction Gating

Given our trained reference map can be imperfect, the residual decomposes as $\delta = \delta_{\text{app}} + \eta$, where η is the reference map's error and δ_{app} is the transport change caused by appearance. When $v \in \mathcal{V}_{\text{train}}$, $\delta_{\text{app}} = 0$ and the residual reduces to $\delta = \eta$. Without gating the correction, integrating η at every step would compound an error. Restricting firing over $\rho < \varepsilon_{\text{dir}}$ repairs only steps where the observed transport opposes the reference by some tolerance. Thus the gate at Euler step i is

$$\text{fire}_i = \left[\rho < \varepsilon_{\text{dir}}\right] \tag{12}$$

and no shifted data is required to set it. We report the fraction of steps which the gate fires for every result in our experiments. This fraction distinguishes our method from an uncorrected policy baseline.

3.6 Implementation

Algorithm 1 showcases ASVC as implemented. Relative to an uncorrected policy, our method adds an additional evaluation of the field per Euler step and a pass of the reference map per control step. We keep the policy frozen, thus ASVC is a correction method to be deployed over a trained model.

Algorithm 1 ASVC: consistency-gated transport

Require: frozen field f_θ, reference map c, gain λ, tolerance $\varepsilon_{\mathrm{dir}}$

$o_{\mathrm{clean}} \leftarrow c(o)$
$a \leftarrow a_0 \sim \mathcal{N}(0, I)$
for $i = 0, \ldots, K-1$ **do**
  $t \leftarrow i/K$
  $u \leftarrow f_\theta(a, t \mid o)$; $u_{\mathrm{clean}} \leftarrow f_\theta(a, t \mid o_{\mathrm{clean}})$
  $\delta \leftarrow u_{\mathrm{clean}} - u$; $\hat{e} \leftarrow \delta / \|\delta\|$; $\rho \leftarrow \langle u, \hat{e} \rangle$
  fire $\leftarrow [\rho < \varepsilon_{\mathrm{dir}}]$
  $a \leftarrow a + \frac{1}{K}(u + \lambda\,\delta)$ **if** fire **else** $a + \frac{1}{K}\,u$
end for
return a

4 EXPERIMENTS

We evaluate ASVC through three settings, increasing in either difficulty or in practicality. First, we evaluate within a low-dimensional toy reach task. This task's goal is for an actor to reach a goal state, where during inference we apply visual shifts. This experiment validates our mechanism and also provides ablations for our hyperparameter decisions. Second, we evaluate over the DMC-GB2 benchmark on three separate base policies under color and video background shifts. Lastly, we perform a cube-stacking task on a real robot and test our method under variation in distractor objects and lighting. Throughout experiments, we compare a frozen policy with and without ASVC applied.

4.1 SETUP

Settings. Our toy experiment is a 28×28 grid in which an agent must reach a goal with a two-dimensional action $(\Delta x, \Delta y)$. The policy is trained on a flat gray background and evaluated on held-out background shifts. For DMC-GB2 (Almuzairee et al., 2024), we use the tasks `walker_walk`, `walker_stand`, and `finger_spin`. For visual variations within DMC, policies are evaluated on both color and video shift, using `color_easy`, `color_hard`, and `video_easy` as test-time shifts. For our real robot setting, the policy performs a cube-stacking task on a Trossen WidowX AI robot, described further in Section 4.4.

Base policies. ASVC is applied only at inference-time over the integration of a flow-matching policy, so we use three base policy classes differing in the encoder, training objective, and number of Euler steps. On DMC-GB2, policies include ReinFlow (Zhang et al., 2026a), Guided Flow Policy (GFP) (Nguimatsia Tiofack et al., 2026), and a modified Diffusion Policy (Chi et al., 2025). Diffusion Policy is adapted to our setting by replacing the denoising objective with a rectified-flow action head; otherwise the architecture is the same. GFP deploys a distilled one-step actor, so there are no integration steps for ASVC, thus we run the policy for more than one Euler step during action sampling. Each policy is trained on the source distribution of the tasks and frozen before evaluation. Within the toy experiment, we train the policy via rectified-flow behavior cloning of an expert. On the robot setting, we use the same Diffusion Policy setup as on DMC-GB2, although trained on our expert demonstrations.

Reference map. For each task, we train reference map c as a U-Net mapping procedurally corrupted frames to clean frames using color jitter and channel permutation. Thus no test-time shifts are used. Reference maps are accepted if the base policy using the output of the reference map is within 90% of the returns from the base policy using the current observation.

Protocol. DMC-GB2 policies are evaluated with 3 seeds and 120 episodes per seed and each policy re-plans at every control step, thus all policies share the same evaluation protocol. We report average returns and standard deviation over all seeds, and include the mean firing rate of ASVC. Hyperparameters for all DMC-GB2 tasks can be seen in Appendix A.1.

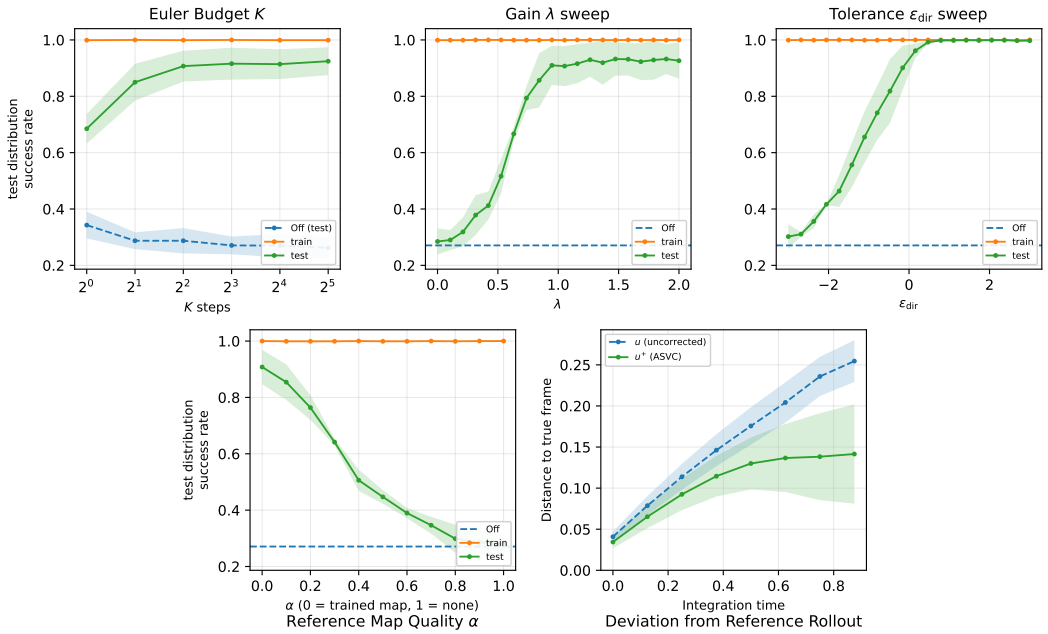

Figure 2: Toy experiment ablations on the test-distribution backgrounds, mean $\pm$ std over 3 seeds. Panels sweep the Euler budget K, the gain λ, the direction tolerance ε_{dir}, and the reference map quality α, where $\alpha = 0$ is our trained map and $\alpha = 1$ is none. The lower right panel gives the deviation from the reference rollout over integration time for the uncorrected and corrected transports.

4.2 Toy

Our toy policy is trained on clean frames, so performance recovery under shift is attributed to our correction mechanism rather than the encoder. We use the default setting $\lambda = 1$, $\varepsilon_{\text{dir}} = -0.1$, and $K = 8$. Using the default settings, success on the test backgrounds rises from 0.27 to 0.90, averaged over 3 seeds, while success remains at 1.00 on the training backgrounds. On these experiments, ASVC is applied at a fire rate of 44% of all integration steps on the test backgrounds and 48% on training backgrounds. Figure 2 shows ablations of sweeping each component of ASVC while other components remain at their default value.

As seen in Figure 2, ASVC improves success at $K = 1$, reaching 0.685, and converges at $K = 4$. As K grows, the step firing rate rises from 0.27 to 0.44, spreading the correction over more, smaller steps. As λ grows, the success rate increases from 0.285 at $\lambda = 0$ to 0.91 at $\lambda = 1$ and changes minimally for $\lambda > 1$. Increasing ε_{dir} increases the success minimally while also increasing the firing rate.

A limitation of ASVC is that the reference map is a bound on recovery. For a poor reference map, the corrected transport can carry error. To measure this, we blend the output of our trained map toward the raw observation by factor α and evaluate at each blend. Success falls from 0.908 at our trained map to 0.279 at $\alpha = 1$, where the map is removed and the correction cannot apply.

4.3 DMC-GB2

Table 1 reports DMC-GB2 results over three base policies. For each base policy and task, we give the average returns over 120 episodes per seed for 3 seeds both with and without ASVC on four different modes. We also report the average correction fire rate for ASVC to show the result is not trivial.

Shifted modes. On shifted modes, ASVC recovers the returns of every base policy for almost all tasks. The exception is GFP on `walker_stand` with shifted mode `color_easy`, where the

Table 1: ASVC on three flow-matching policy classes, DMC-GB2. Mean ± std over 3 seeds at 120 paired episodes. Bold is the better arm within each pair; Fire is the mean firing fraction of the gate.

Task	Mode	ReinFlow base	ReinFlow +ASVC	ReinFlow Fire	GFP base	GFP +ASVC	GFP Fire	Diffusion Policy base	Diffusion Policy +ASVC	Diffusion Policy Fire
walker_walk	train	**826.4 ± 74.6**	747.4 ± 84.5	0.73	**925.8 ± 18.2**	920.1 ± 14.2	0.55	**874.1 ± 38.9**	866.5 ± 44.3	0.55
	color_easy	254.1 ± 37.2	**356.7 ± 80.5**	0.84	757.4 ± 28.1	**804.5 ± 51.1**	0.59	643.4 ± 44.6	**783.3 ± 82.3**	0.57
	color_hard	97.8 ± 17.3	**279.0 ± 62.0**	0.92	474.9 ± 44.9	**713.6 ± 36.9**	0.61	539.7 ± 43.1	**697.9 ± 83.9**	0.68
	video_easy	234.0 ± 80.9	**427.4 ± 118.3**	0.90	499.8 ± 77.8	**658.4 ± 68.2**	0.60	510.6 ± 57.1	**690.5 ± 103.2**	0.55
walker_stand	train	858.7 ± 18.8	**859.8 ± 43.6**	0.61	964.9 ± 12.2	**966.9 ± 7.4**	0.52	**939.5 ± 22.9**	935.4 ± 19.7	0.66
	color_easy	519.2 ± 120.4	**749.8 ± 90.8**	0.79	**947.5 ± 15.9**	946.1 ± 20.4	0.63	889.3 ± 59.8	**917.4 ± 27.0**	0.59
	color_hard	312.9 ± 85.7	**617.9 ± 108.7**	0.92	800.0 ± 91.5	**934.4 ± 16.8**	0.71	837.5 ± 68.7	**908.1 ± 27.8**	0.59
	video_easy	469.4 ± 19.5	**729.1 ± 100.6**	0.81	751.7 ± 84.2	**929.6 ± 17.8**	0.69	821.4 ± 44.5	**910.6 ± 38.4**	0.57
finger_spin	train	**710.2 ± 12.3**	681.8 ± 17.9	0.52	**770.1 ± 9.9**	763.5 ± 8.6	0.51	**747.7 ± 0.9**	739.3 ± 1.8	0.46
	color_easy	272.4 ± 26.8	**394.3 ± 22.4**	0.57	713.5 ± 37.4	**722.2 ± 31.4**	0.49	**700.5 ± 20.7**	691.9 ± 6.9	0.52
	color_hard	96.8 ± 10.9	**259.0 ± 20.7**	0.67	540.0 ± 82.8	**611.7 ± 77.4**	0.47	613.7 ± 43.2	**642.4 ± 22.8**	0.57
	video_easy	184.5 ± 45.9	**260.9 ± 21.5**	0.76	378.1 ± 63.2	**405.3 ± 33.2**	0.56	356.4 ± 11.2	**425.4 ± 4.7**	0.66

policy's return is near its clean return and there is little to recover. The largest recovery is $+305$ on ReinFlow for `walker_stand` with `color_hard` and $+239$ on GFP `walker_walk` with `color_hard`. As shown within Table 1 and our supplementary videos, return averages can recover up to $2.0\times$ with ASVC due to the degradation of the base policy. Diffusion Policy degrades the least under shift, although with ASVC it recovers up to $1.35\times$ its degraded return on `walker_walk`. As the three policies differ in their encoders, training objective, and chunk length, the improvement is not bound to specific designs.

Clean mode. On the clean source distribution mode, ASVC degrades the base policies' performance due to the correction applying. The largest degradation is -79 on ReinFlow `walker_walk` with the correction firing for 73% of integration steps.

4.4 Real Robot

For the robot task, experiments are run on a Trossen WidowX AI stationary robot using only the right arm for stacking a blue cube on a red cube. Within training rollouts, the red cube initial state is the same whereas the blue cube's initial state varies. We collected 100 teleoperated demonstrations, each observed through four cameras at 160×160 resolution. Diffusion Policy is trained with the flow head on these demonstrations for a single seed, re-planning at every control step with temporal ensembling over 16-step chunks and integrating over $K = 3$ Euler steps at deployment. Our reference map is trained on demonstration frames under per-channel color casts and blended natural-image crops, thus no shifted data is used. Evaluation covers the source scene and two shift conditions: distractor objects near the cubes and a change in room lighting. For the distractor shift, office supplies were added to the robot desk to obstruct the camera views, but were not placed so as to obstruct the robot from performing the task. For the lighting shift, the room lights were dimmed and an external light source was placed to cast shadows, with the lights dimmed only so far that the cameras could still see the task. We place the blue cube at 10 fixed start locations shown in Appendix A.5.1, run 3 corrected then 3 uncorrected episodes per location, and judge success when the blue cube rests on the red cube with the gripper released. This process is repeated for the source scene and both shifts.

Table 2 reports success over 30 episodes for each condition, both with and without ASVC. Under distractors, success rises from $12/30$ to $23/30$ with correction, and under the lighting change the base policy never picks the cube up while ASVC succeeds on $11/30$. Figure 3 shows four rollouts, two using ASVC and two without, showing how ASVC helps the policy reach the goal state in the presence of visual shifts. On the source scene the two arms are within one success of each other, thus the correction costs little where there is nothing to correct. As the policy and reference map are a single seed, these results demonstrate the correction on real hardware rather than establish a multi-seed effect. ASVC adds one reference map pass and $2K$ velocity evaluations per action, raising plan latency from 25.9 ms to 39.7 ms, which stays within the 15 Hz control budget, thus the control rate is unchanged. Additional videos of experiments can be viewed within the supplementary material.

Table 2: Success over robot experiment for 30 episodes per shift for with ASVC and without. Recorded with 10 start locations, 3 episodes per location. Fire rate is the fraction of integration steps where the correction fires, averaged over all rollouts.

Method	Clean	Distractors	Lighting
Off	26/30	12/30	0/30
+ASVC	**27/30**	**23/30**	**11/30**
Fire Rate	0.45	0.25	1.00

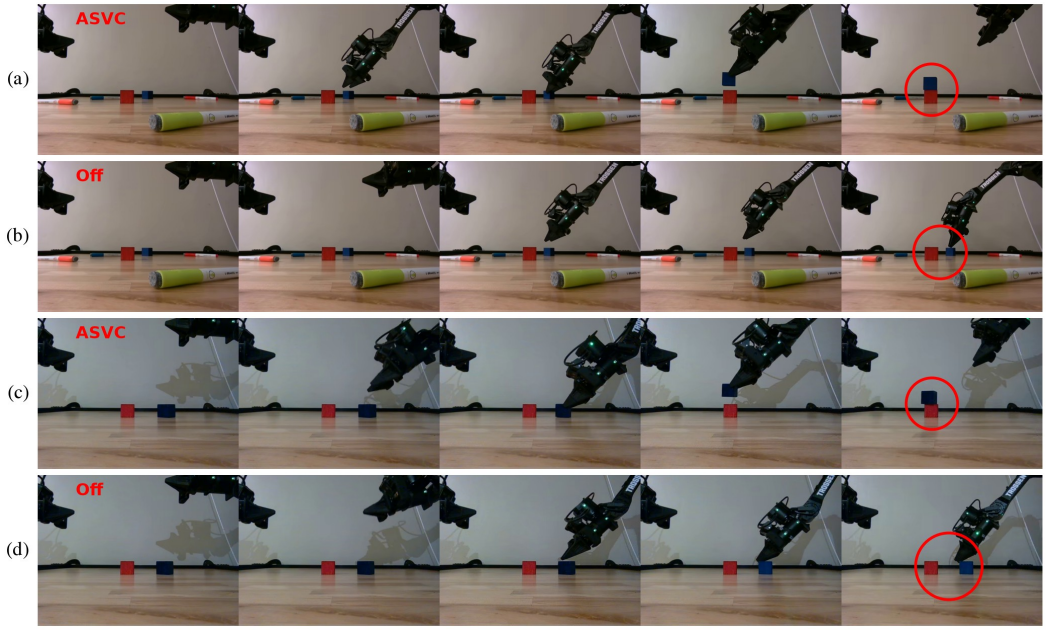

Figure 3: Rollouts from the front camera. (a) Distractors-shift with ASVC, (b) Distractors-shift with ASVC off, (c) Lighting with ASVC, (d) Lighting with ASVC off. In both cases, ASVC is shown succeeding at stacking the blue cube on the red cube, but without ASVC fails due to the visual shift. For more rollout visuals of ASVC, please view our supplementary video.

5 Related Work

Visual Generalization. A common solution for appearance shift is training on more data that captures more visual variation in pixel-based reinforcement learning. RAD, DrQ, and DrQ-v2 apply data augmentation to observations in their models during training through random crops, shifts, and, in RAD, random convolutions (Laskin et al., 2020b; Yarats et al., 2021; 2022). Later work adapts how augmentation is applied, selecting it online from a candidate set, restricting it to task-irrelevant pixels, or keeping augmented observations out of the value targets, to provide consistency (Raileanu et al., 2021; Yuan et al., 2022; Almuzairee et al., 2024). Others add auxiliary objectives collapsing observations sharing rewards and dynamics (Zhang et al., 2021), aligning augmented views of observations (Laskin et al., 2020a), or predicting the agent's own future latent representations (Schwarzer et al., 2021). Correcting for consistency during training using augmentation or auxiliary objectives is costly as policies need to be retrained with the updated objectives or data. Inference-time adaptation instead updates the policy's encoder online against a self-supervised objective computed on target frames (Hansen et al., 2021). Through this, consistency is obtained by updating the policy's parameters either prior to or during deployment. Our method leaves every parameter of our policy fixed and instead corrects the action transport during inference.

Image Translation. Rather than policy retraining, others re-render observations with the training appearance before the policy acts, thus the goal is to learn a mapping from the current observation to the source appearance. RCAN learns a translator whose outputs, alongside the raw frame, the policy is trained and deployed over (James et al., 2019). An unsupervised translation instead maps source frames onto the target appearance and retrains the policy on the translated frames (Rao et al., 2020; Bousmalis et al., 2018). Under both approaches, the policy is fitted to the output of the translator, thus error of the mapping is passed onto the action. ASVC also trains a mapping, although it never feeds the output to the policy. Instead the mapping output enters through the transport over the action space, where analysis of the transport can minimize the error to steps where the reference and observed transports significantly oppose.

Flow Policies. Generating actions by integrating over a learned velocity field, flow-matching (Lipman et al., 2023) policies (Black et al., 2025; Park et al., 2025) are a viable replacement for Gaussian action heads in imitation and reinforcement learning settings. Training such policies with reinforcement learning, ReinFlow injects learnable noise into the integration to recover a tractable likelihood (Zhang et al., 2026a), flow policy gradients replace the likelihood ratio with a ratio of conditional flow-matching losses (McAllister et al., 2026), and GFP couples a multi-step flow policy trained with value-aware behavior cloning to a distilled one-step actor through bidirectional guidance (Nguimatsia Tiofack et al., 2026). With these policies, the transport is unchanged at deployment, so visual variation enters the transport with no mechanism to correct it. ASVC acts over the integration of any action flow-matching policy, correcting the transport at each step where the gate fires rather than the parameters of the policy.

Transport Guidance. Editing the transport of generative models at inference time is established in image generation. Classifier-free guidance evaluates the field with and without the conditioning and shifts the field along their difference at each denoising step (Ho & Salimans, 2022). Within concept erasure, differential vector erasure (DVE) corrects the velocity field during integration along the difference between velocities conditioned on a desired concept and a concept to remove (Zhang et al., 2026b). DVE performs well at concept removal, although visuomotor policies are conditioned on observations rather than text, so no prompt is available to name the concept to erase. Others remove concepts from a representation rather than from a sampler, projecting out the subspace so later linear probes cannot recover it (Ravfogel et al., 2020; Belrose et al., 2023). ASVC works in the action space where prompts cannot easily update the transport and instead uses reference views similar to concepts.

6 Conclusion

We introduced ASVC, an inference-time correction to the velocity of a flow-matching policy that restores consistency under visual variation without retraining. Our correction fires only where observed transports oppose a reference-conditioned transport, and applies as an add-on for flow-matching action policies. Across a toy reach task, DMC-GB2 with three base policy classes, and a real-robot stacking task, ASVC recovers much of the return lost under visual shift while nearly preserving performance on the training appearance.

Limitations. Recovery of ASVC is bounded by the reference map, which must be trained per task and silently fails when visual variation falls heavily outside the training corruptions. On source observations, correction can still apply as shown on DMC-GB2. Learning reference maps that transfer across tasks and prevent corrections on the source observations would make integration-time corrections more stable for flow policies.

7 Reproducibility Statement

We provide source code and configurations for the performed experiments within the supplementary materials. Instructions are also provided for training and evaluating the policies. Code was verified to run end-to-end.

REFERENCES

Abdulaziz Almuzairee, Nicklas Hansen, and Henrik I Christensen. A recipe for unbounded data augmentation in visual reinforcement learning. *Reinforcement Learning Journal*, 2024.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. Leace: Perfect linear concept erasure in closed form. *NeurIPS*, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *RSS*, 2025.

Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor Sampedro, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *ICRA*, 2018.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *IJRR*, 2025.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *ICML*, 2019.

Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. In *ICLR*, 2021.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, 2019.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *NeurIPS*, 2020b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *ICLR*, 2026.

Franki Nguimatsia Tiofack, Théotime Le Hellard, Fabian Schramm, Nicolas Perrin-Gilbert, and Justin Carpentier. Guided flow policy: Learning from high-value actions in offline reinforcement learning. In *ICLR*, 2026.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *ICML*, 2025.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *NeurIPS*, 2021.

Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *CVPR*, 2020.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. In *ICLR*, 2021.

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *ICLR*, 2021.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *ICLR*, 2022.

Zhecheng Yuan, Guozheng Ma, Yao Mu, Bo Xia, Bo Yuan, Xueqian Wang, Ping Luo, and Huazhe Xu. Don't touch what matters: Task-aware lipschitz data augmentation for visual reinforcement learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2021.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *NeurIPS*, 2026a.

Zhiqi Zhang, Xinhao Zhong, Yi Sun, Shuoyang Sun, Bin Chen, Shu-Tao Xia, and Xuan Wang. Differential vector erasure: Unified training-free concept erasure for flow matching models. *arXiv preprint arXiv:2602.01089*, 2026b.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH*, 2018.

A Appendix

A.1 Hyperparameters

Table 3 shows the hyperparameters across all settings of our experiments. We fixed $\lambda = 1$ and $\varepsilon_{\text{dir}} = -0.1$ across all settings. Our method modifies no components of the base policies, so each base policy's hyperparameters are tuned per setting.

Table 3: Hyperparameters per setting. DP denotes our adapted Diffusion Policy with a rectified-flow action head. DP (DMC) policy training is in the order `walker_walk` / `walker_stand` / `finger_spin`.

	Toy	ReinFlow	GFP	DP (DMC)	DP (Robot)
Observation	28×28	84 px, stack 3	84 px, stack 3	84 px, 3 slots, crop 76	4 cameras, 160 px, crop 144
Action dimension	2	task	task	task	7
Action chunk (executed)	1 (1)	4 (1)	1 (1)	16 (1)	16 (1)
Euler steps K	8	10	10	10	3
Gain λ	1	1	1	1	1
Direction tolerance ε_{dir}	-0.1	-0.1	-0.1	-0.1	-0.1
Policy training	2,500 steps	200 epochs	25k steps	45/30/90 epochs	400 epochs
Reference map	U-Net	3-scale U-Net	3-scale U-Net	3-scale U-Net	4-scale U-Net, GroupNorm
Reference map training	2,000 steps	60 epochs	60 epochs	60 epochs	150 epochs
Episodes per cell	384	120	120	120	30

A.2 Reference Maps

We train a U-Net architecture as our reference map for mapping a corrupted frame back to the clean source frame. The U-Net is trained using data from the training visual distribution, thus there is no leakage of the testing environment. On DMC-GB2, synthetic corruptions are drawn per frame using color jitter, channel permutation, background replacement using images from COCO (Lin et al., 2014), and identity frames. DMC-GB2 uses background replacements from RealEstate10K (Zhou et al., 2018) videos, so no leakage occurs here. For the toy experiment, the reference map is trained for $2{,}000$ steps on background replacements using smooth random color fields. For the robot, one map is shared across all four cameras and trained for 150 epochs on demonstration frames. Per-channel color jitter, natural-image crops blended into the frame, and identity images are used to train the reference map within the robot setting.

For reference map selection on DMC-GB2, we accept a reference map when the policy acting over the reference output frames retains 90% of returns against acting over the training frames. For the robot setting, we define acceptance within the action space. Thus, a reference map is accepted when the actions of the policy acting over mapped demonstration frames differ from its actions over the raw demonstration frames by at most 50% of the policy's own error against the demonstrations. In our experiments, the robot map differed by 0.264% of the joint range with acceptance requiring at most 1.911%.

A.3 Toy Experiments

On the toy experiment, our policy's success rate stays near 1.00 across every sweep of Figure 2 on the training backgrounds. Thus, our method has not been shown to reduce performance when the policy is already performing well. Each figure below is over a single seed, although results within Section 4.2 are reported over 3 seeds.

Training Samples. Figure 4 shows the training data of our toy experiment. Our policy is trained over gray backgrounds, while our reference map includes background shifts using smooth random color fields.

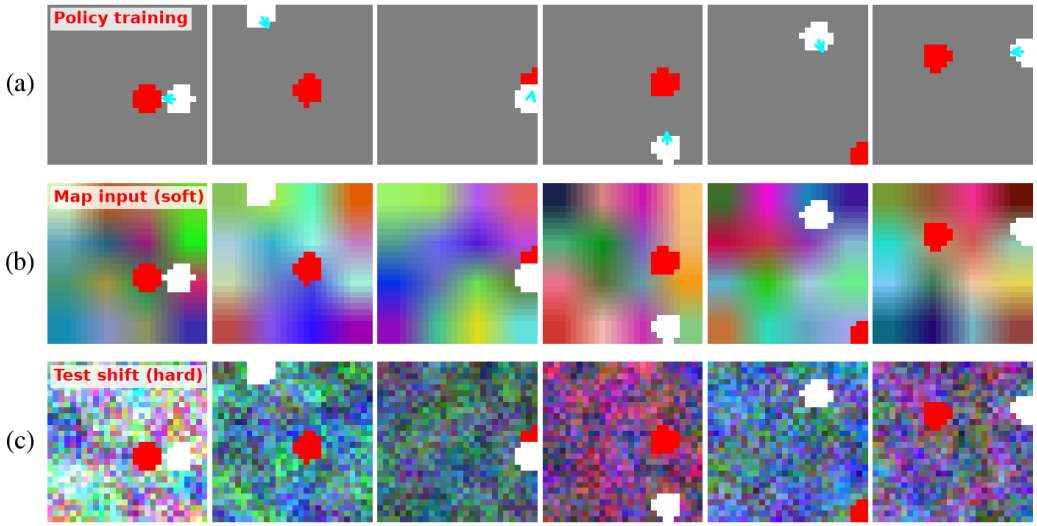

Figure 4: Six states from the toy training data. (a) Policy training frames, expert action is shown with an arrow. (b) Reference map training inputs. (c) Test frames.

Rollout Samples. Figure 5 shows three test episodes with and without ASVC. Frames are at steps $\{0, 2, 4, 6, 20\}$.

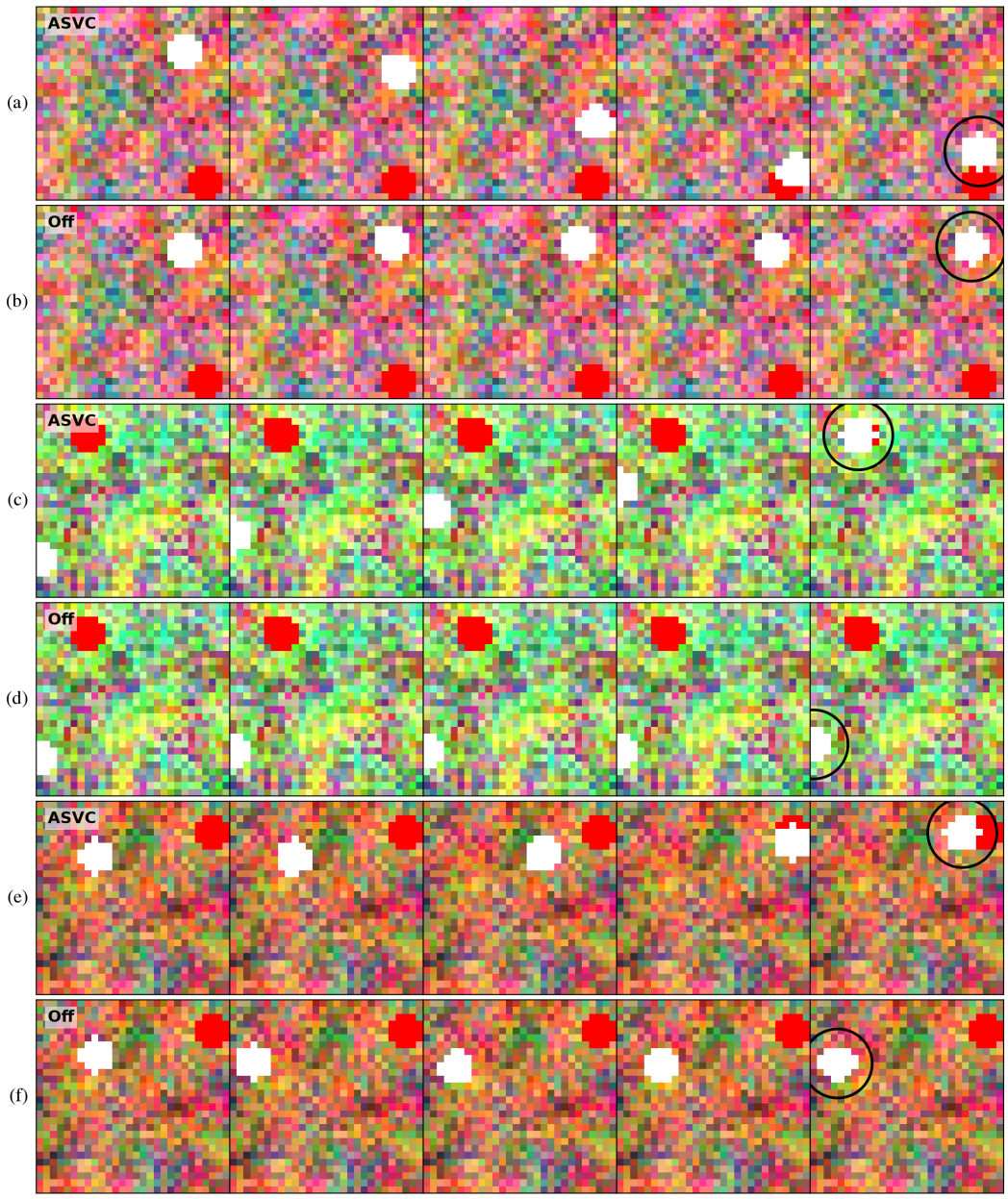

Figure 5: Three test episodes with and without ASVC.

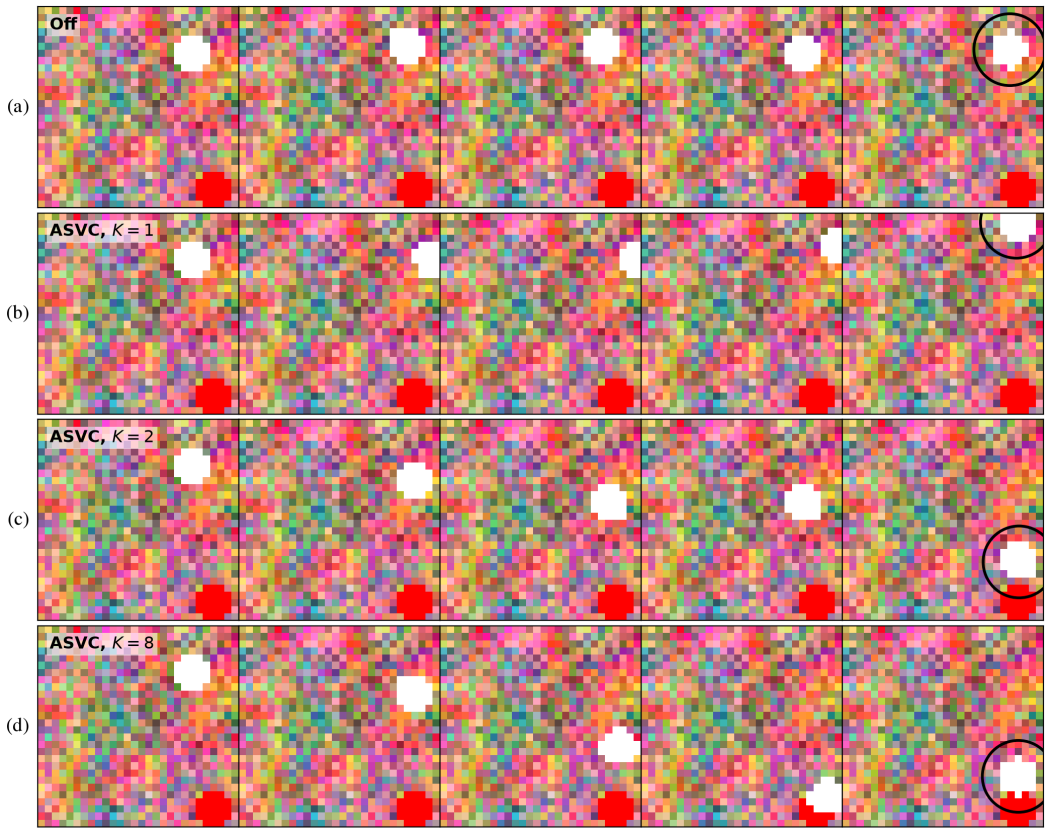

Figure 6: Test episode without ASVC and with ASVC at $K \in \{1, 2, 8\}$.

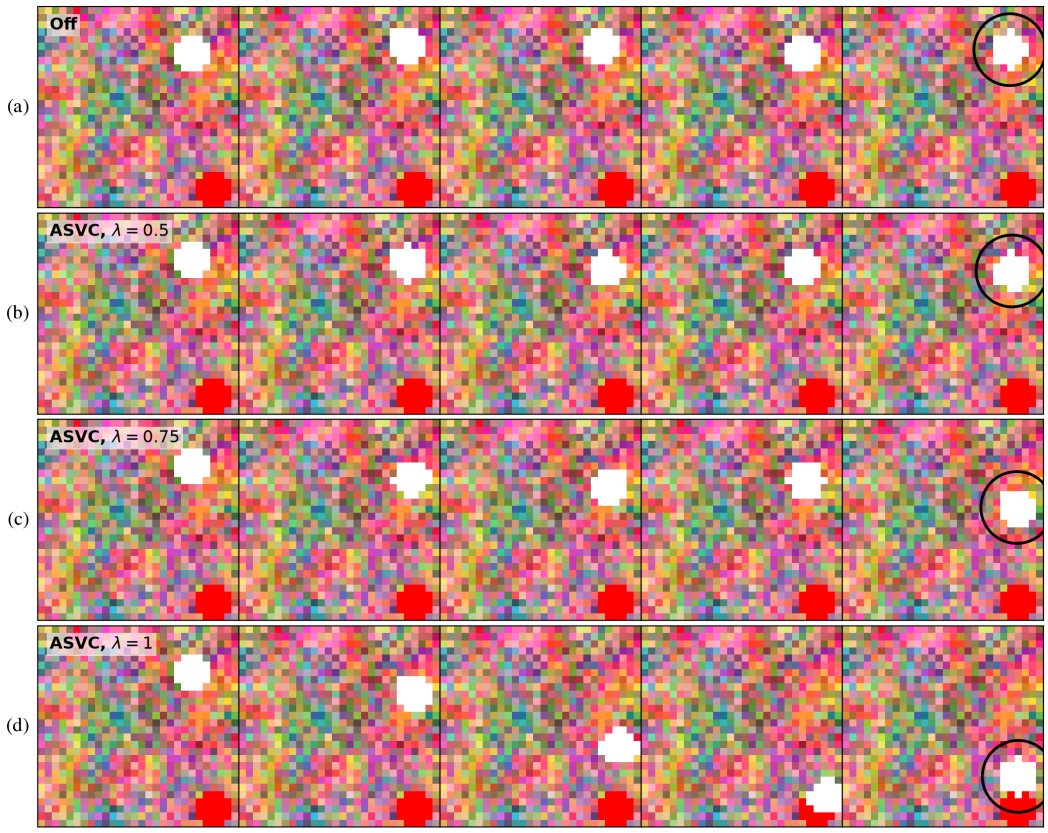

Figure 7: Test episode without ASVC and with ASVC at $\lambda \in \{0.5, 0.75, 1\}$.

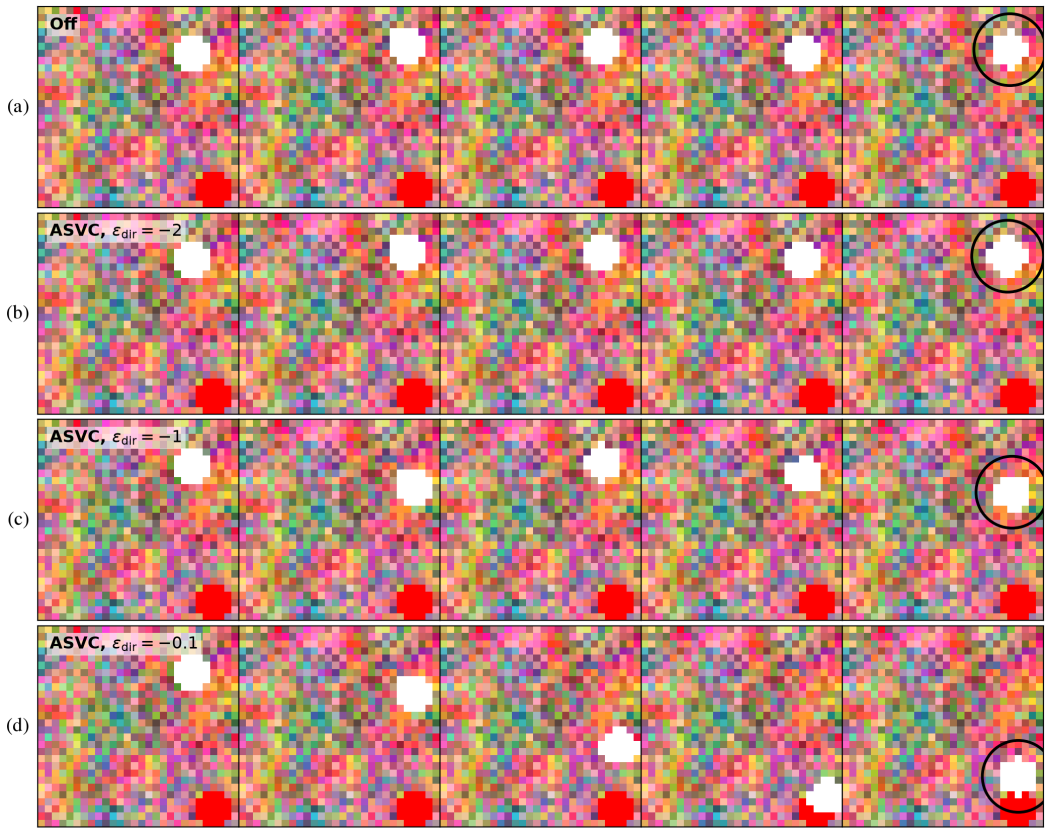

Figure 8: Test episode without ASVC and with ASVC at $\varepsilon_{\text{dir}} \in \{-2, -1, -0.1\}$.

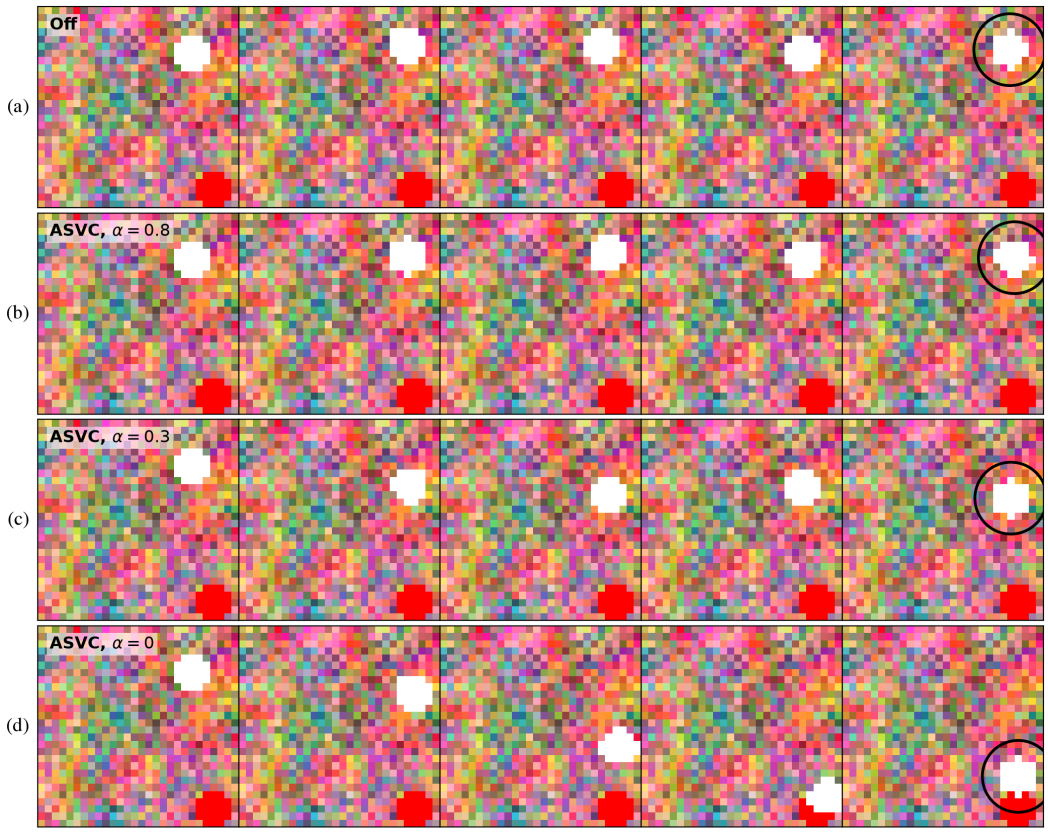

Figure 9: Test episode without ASVC and with ASVC at $\alpha \in \{0.8, 0.3, 0\}$.

Reference Map Samples. Figure 10 shows the outputs of our reference map over train and test backgrounds. Test background outputs recover the agent and goal states with minimal residual noise.

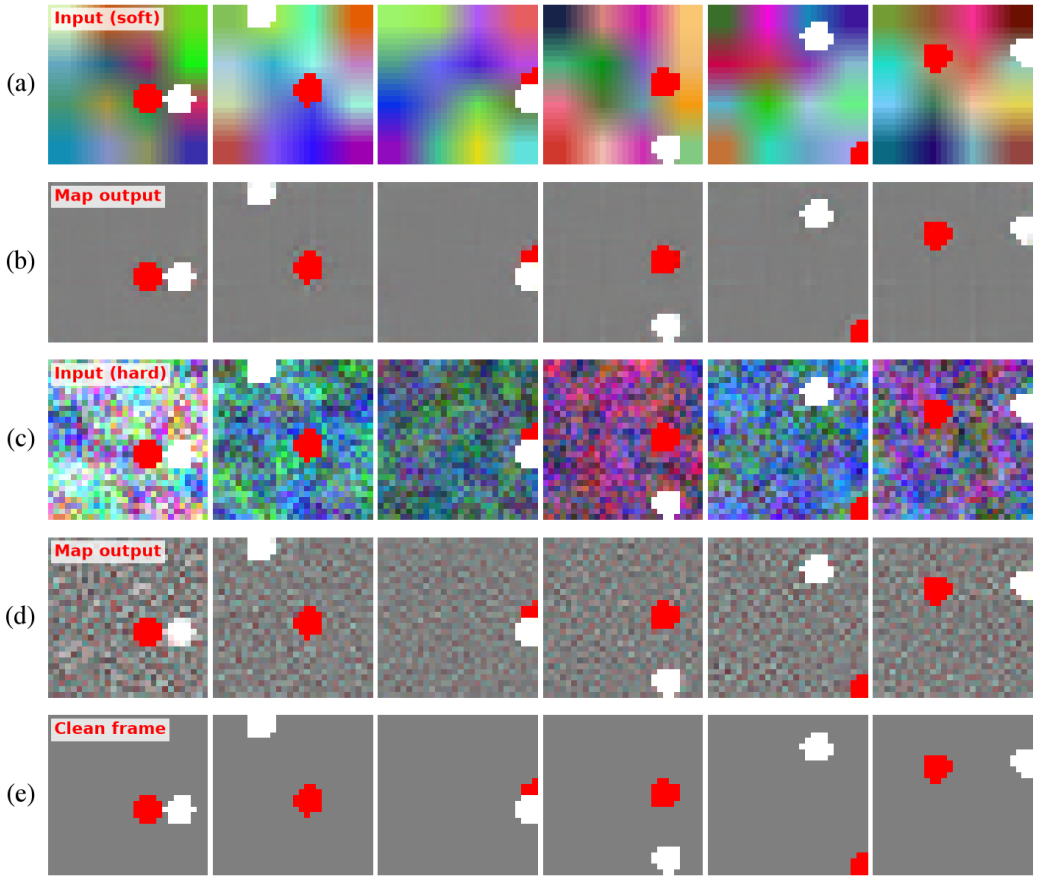

Figure 10: Toy reference map samples. (a) Training inputs and (b) their outputs. (c) Test inputs and (d) their outputs. (e) Training frames.

A.4 DMC-GB2 Experiments

On DMC-GB2, we show example training demonstrations and rollouts of each base policy under shift. Rollouts are drawn from a single seed and show the two episodes with the largest gap between arms. Frames are chosen at control steps $\{0, 3, 6, 10, 250\}$ to show the progression of the agent.

Reference Map Samples. Figure 11 shows the reference map outputs on a shifted frame for each task and shift. Our reference map keeps the agent and its pose while minimal residual noise remains.

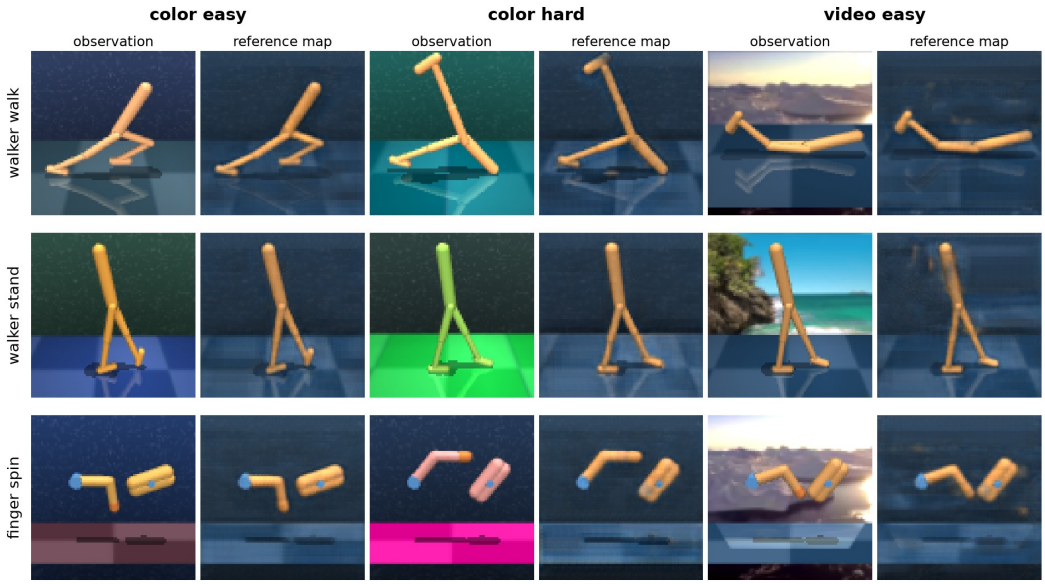

Figure 11: DMC-GB2 reference map samples.

A.4.1 WALKER_WALK

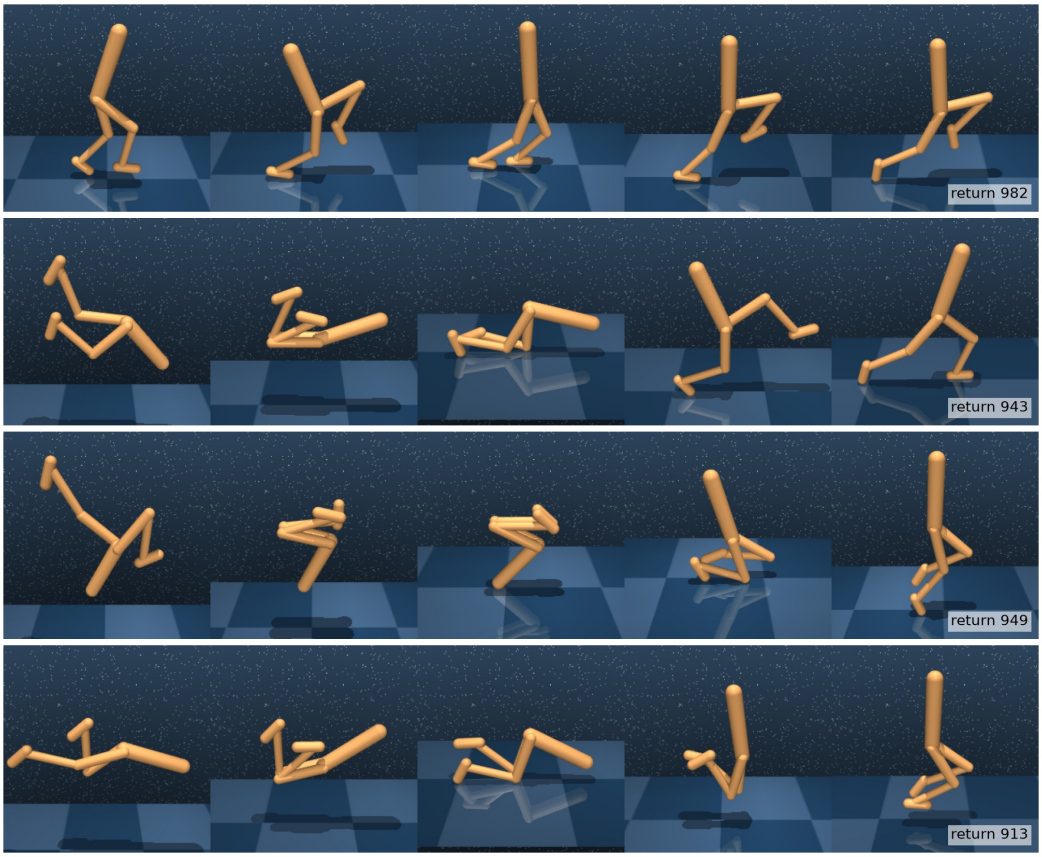

Figure 12: Example walker_walk training demonstrations.

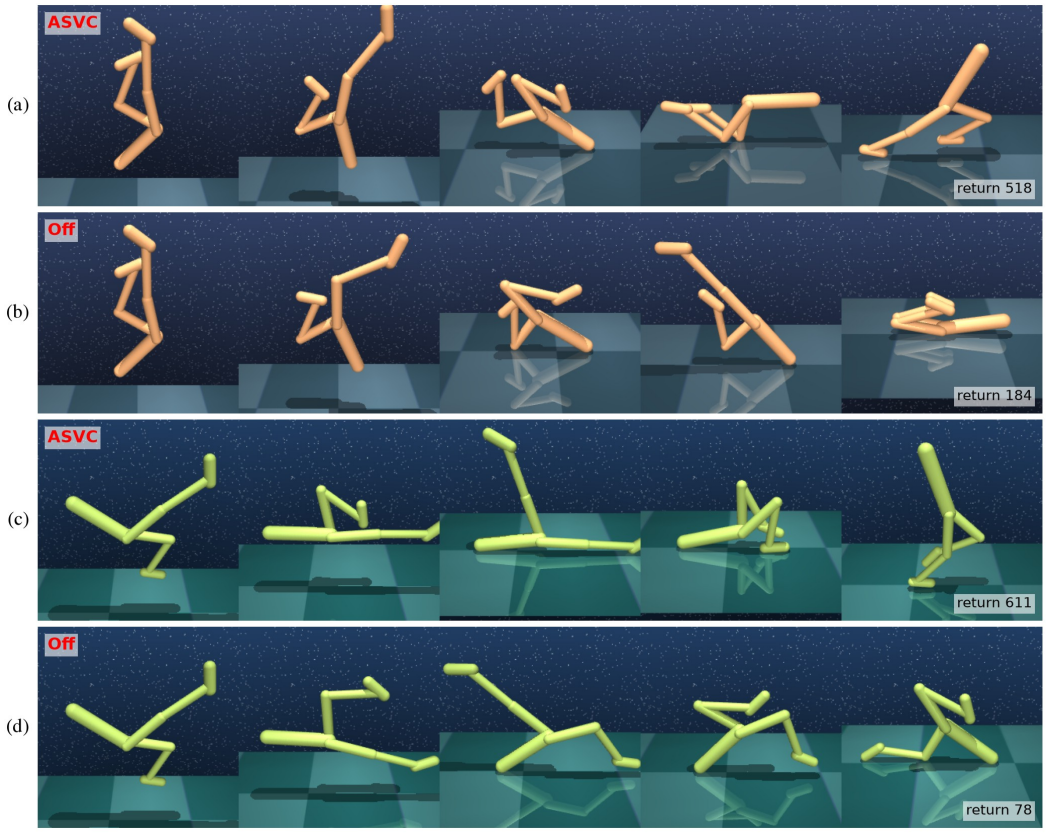

Figure 13: ReinFlow on `walker_walk` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

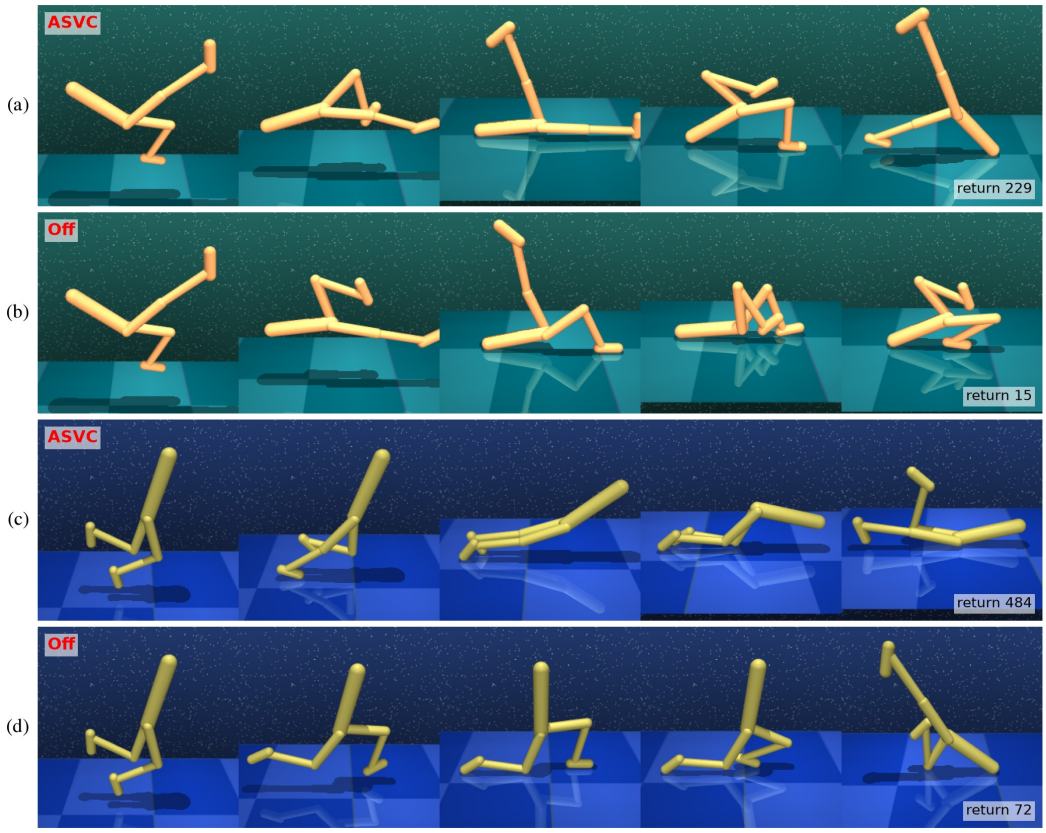

Figure 14: ReinFlow on `walker_walk` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

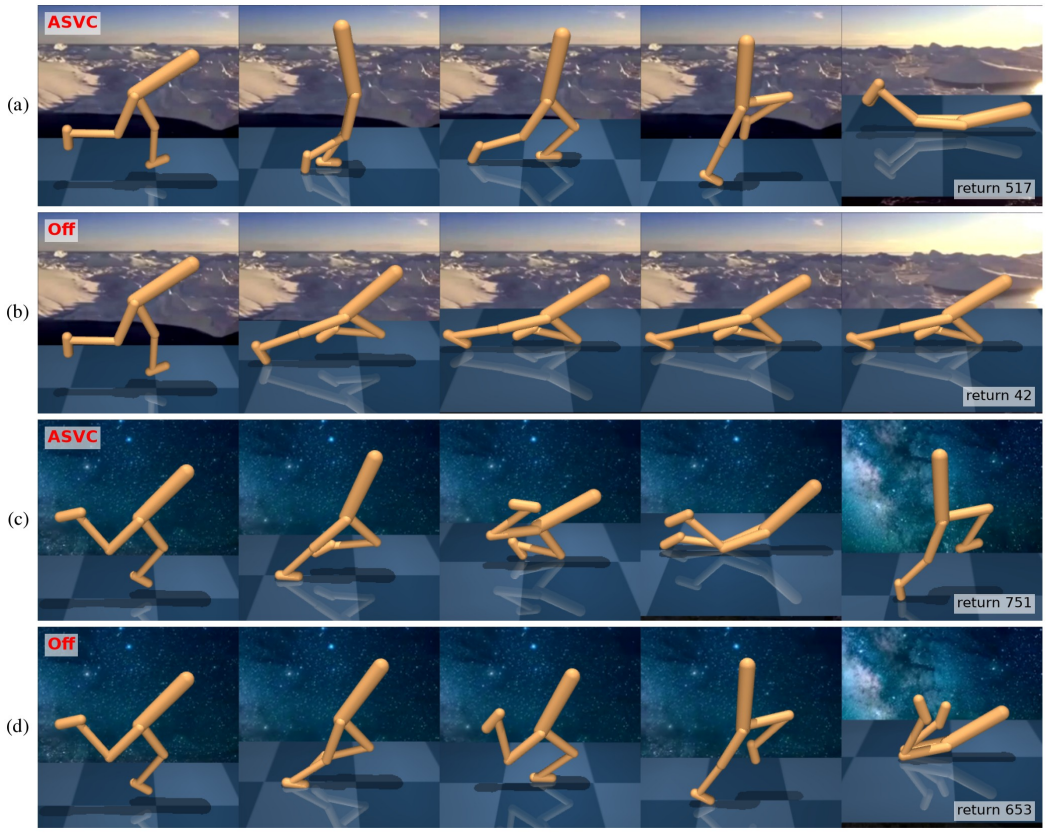

Figure 15: ReinFlow on `walker_walk` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

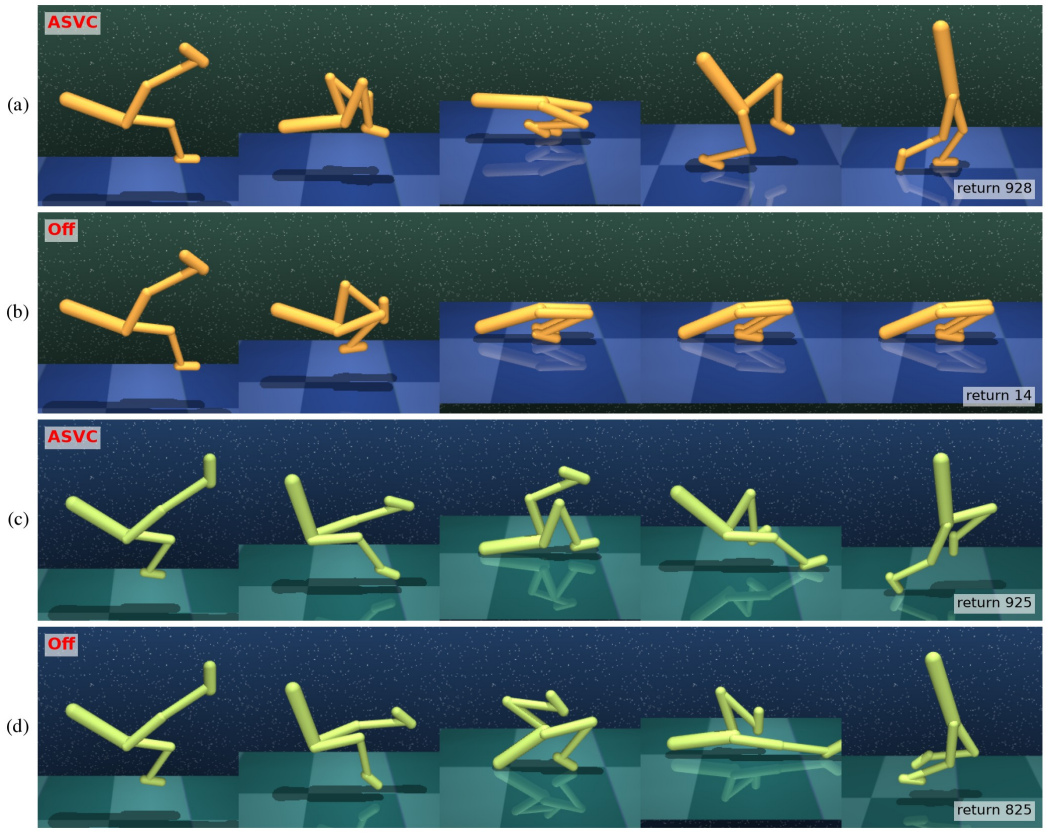

Figure 16: GFP on `walker_walk` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

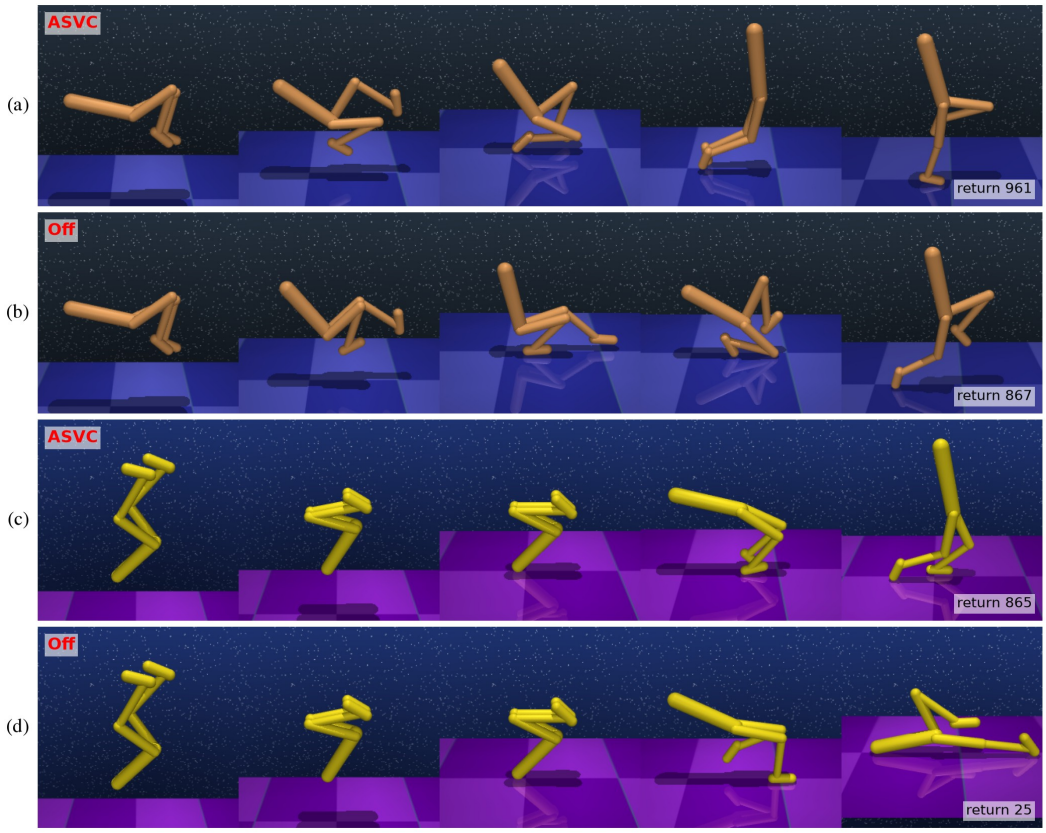

Figure 17: GFP on `walker_walk` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.



Figure 18: GFP on `walker_walk` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

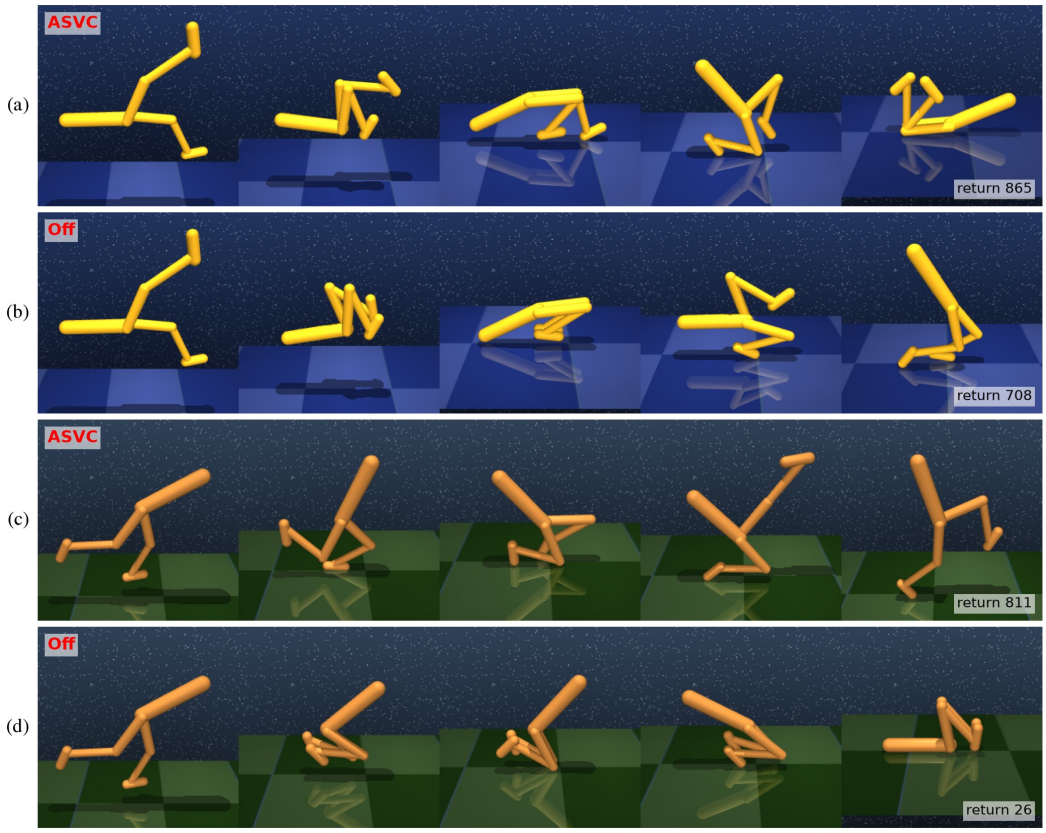

Figure 19: Diffusion Policy on `walker_walk` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

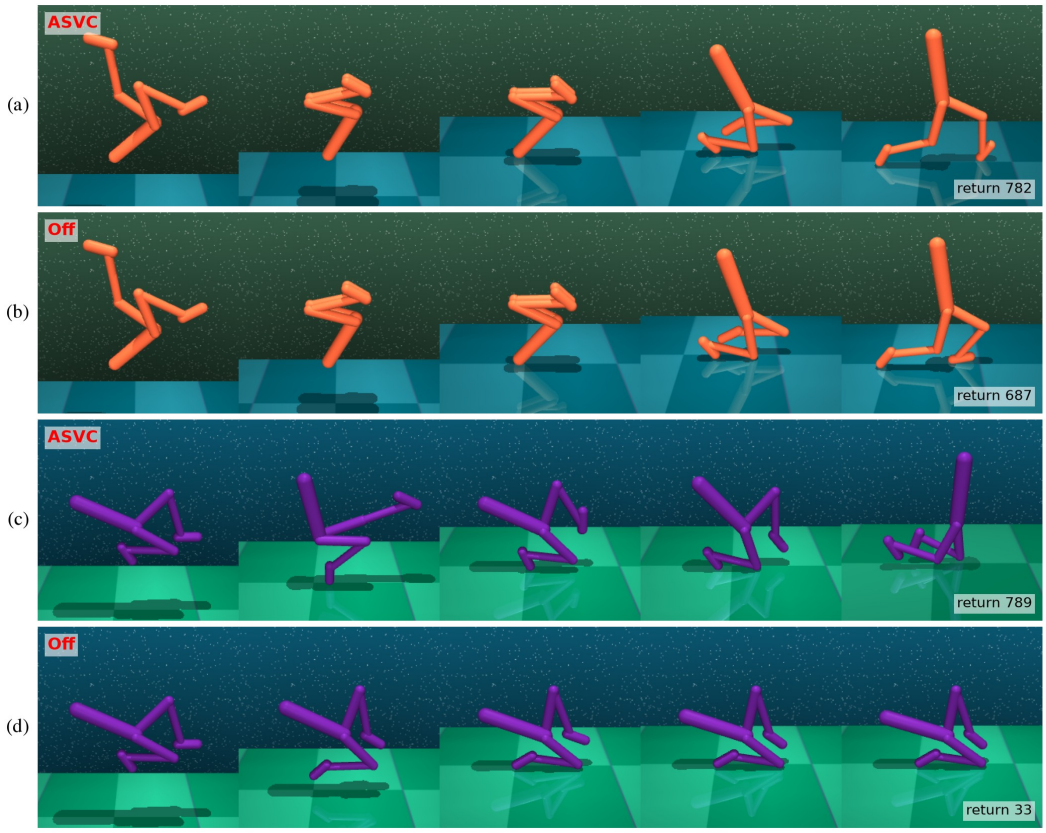

Figure 20: Diffusion Policy on `walker_walk` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.



Figure 21: Diffusion Policy on `walker_walk` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

A.4.2 WALKER_STAND

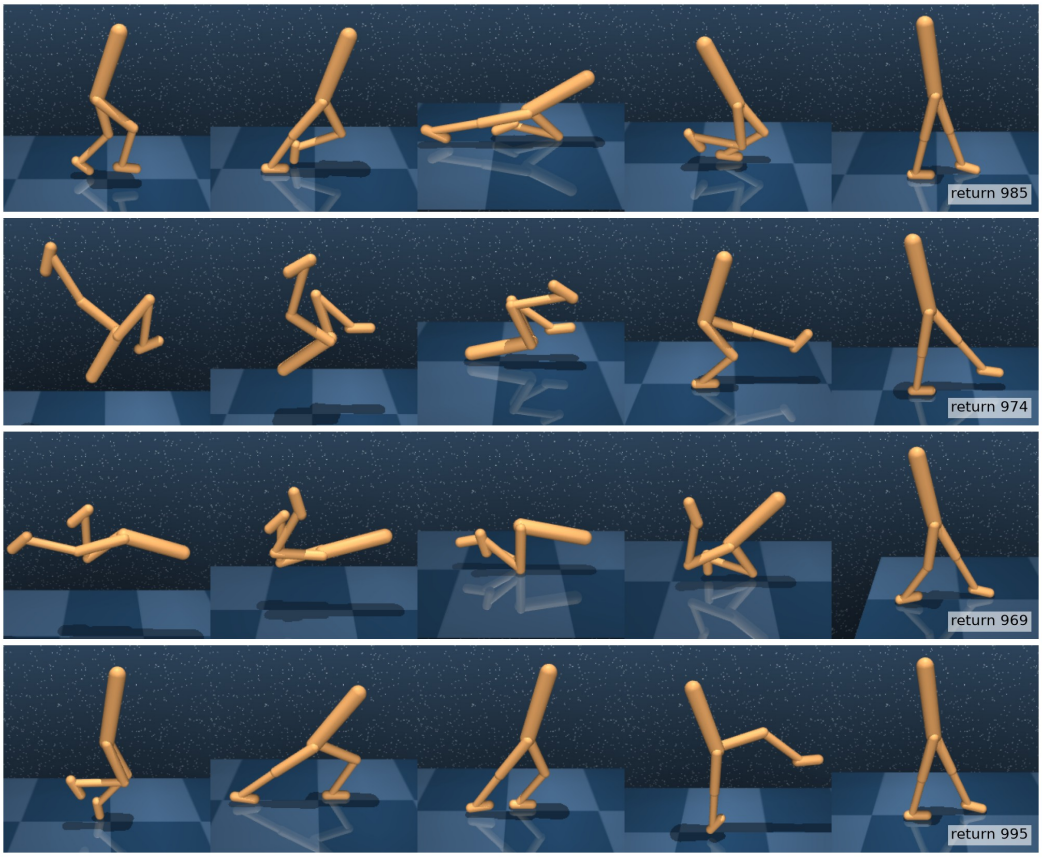

Figure 22: Example `walker_stand` training demonstrations.

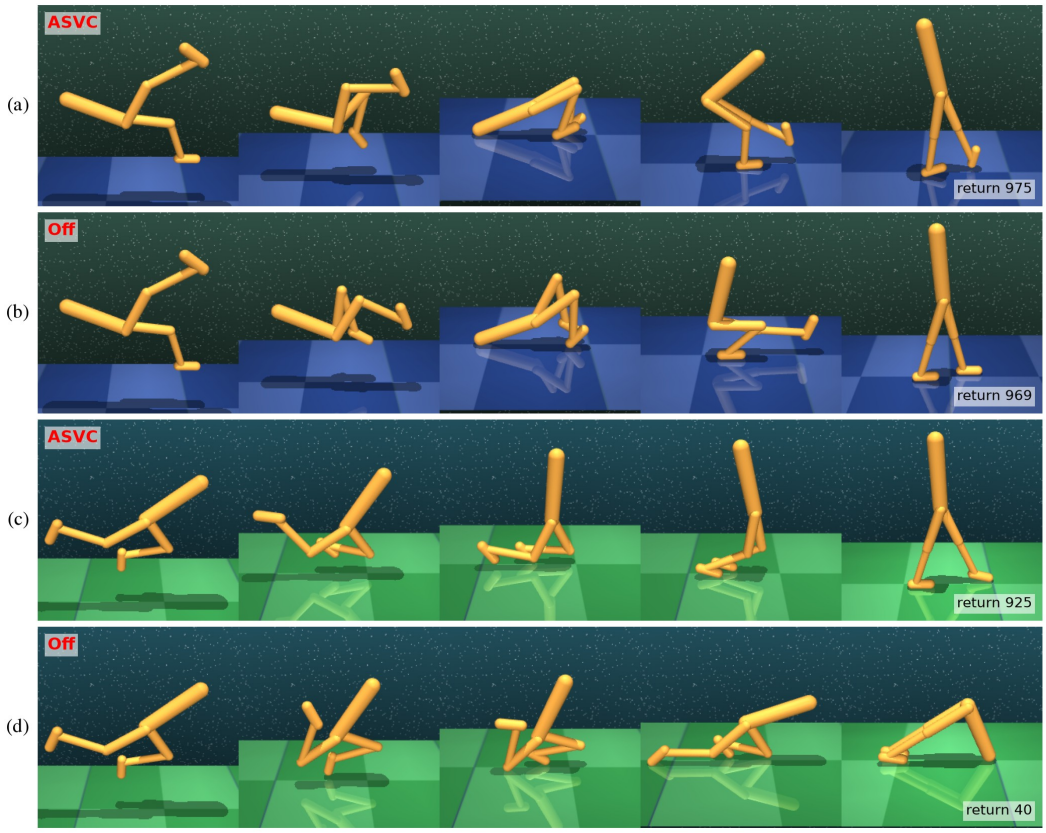

Figure 23: ReinFlow on `walker_stand` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

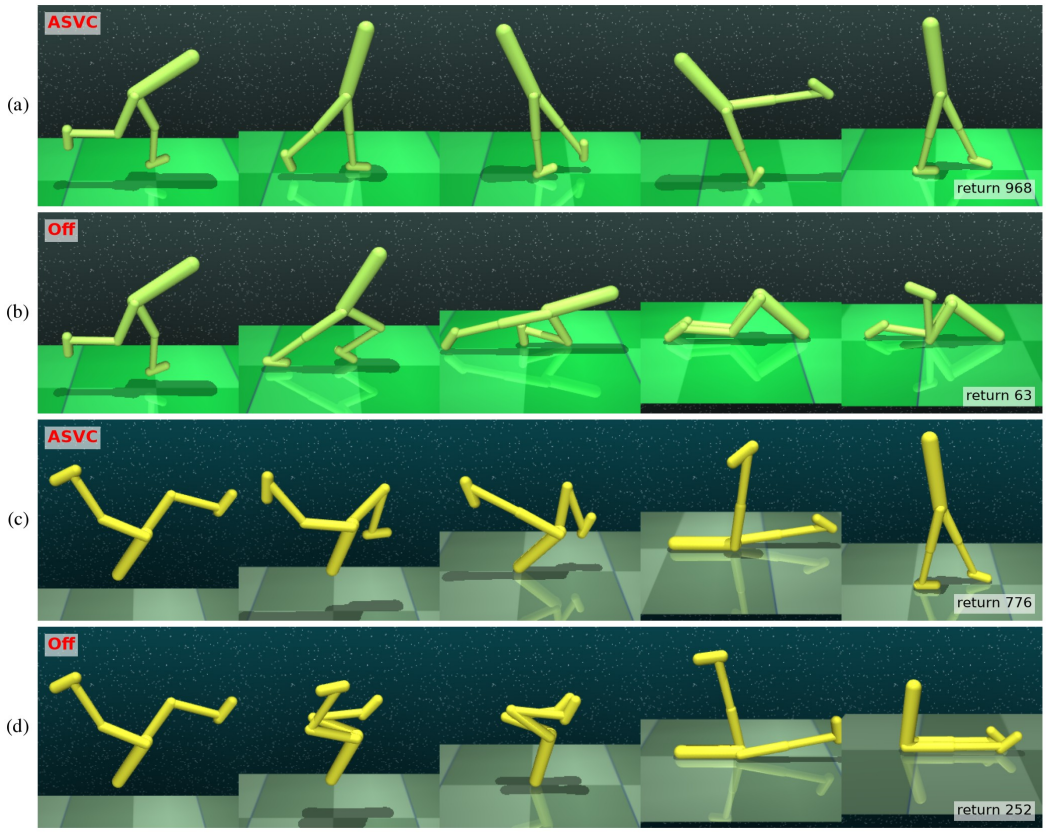

Figure 24: ReinFlow on `walker_stand` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.



Figure 25: ReinFlow on `walker_stand` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

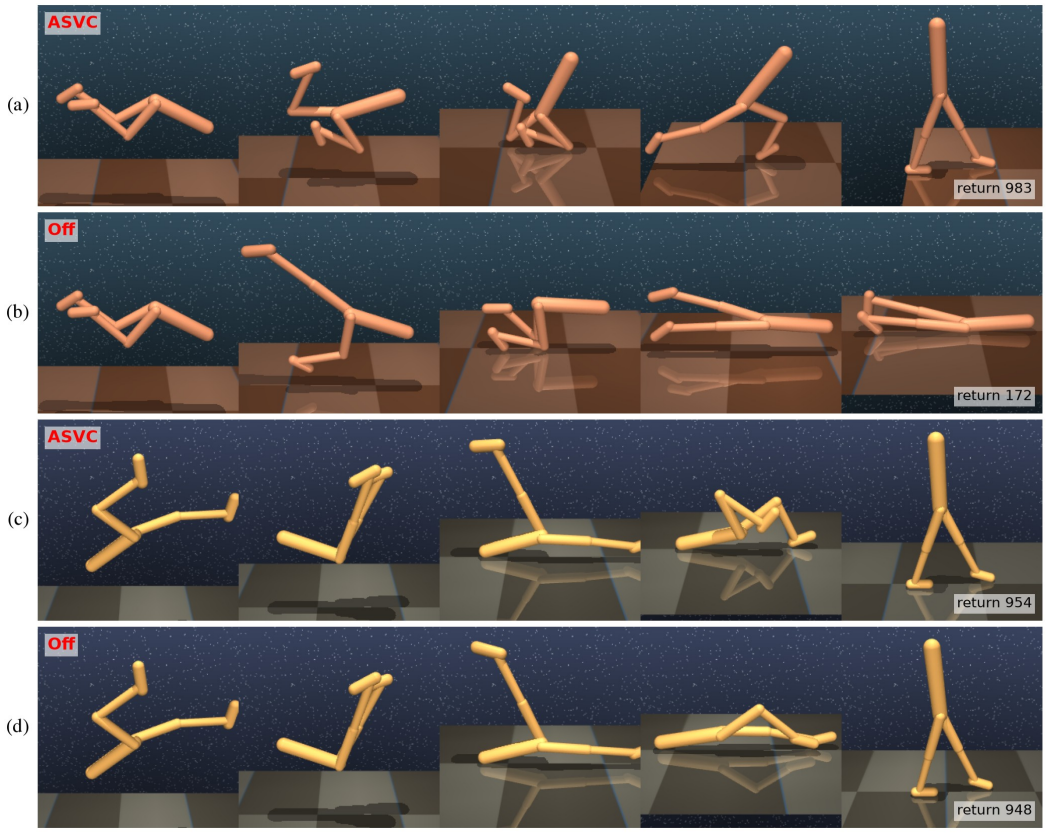

Figure 26: GFP on `walker_stand` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

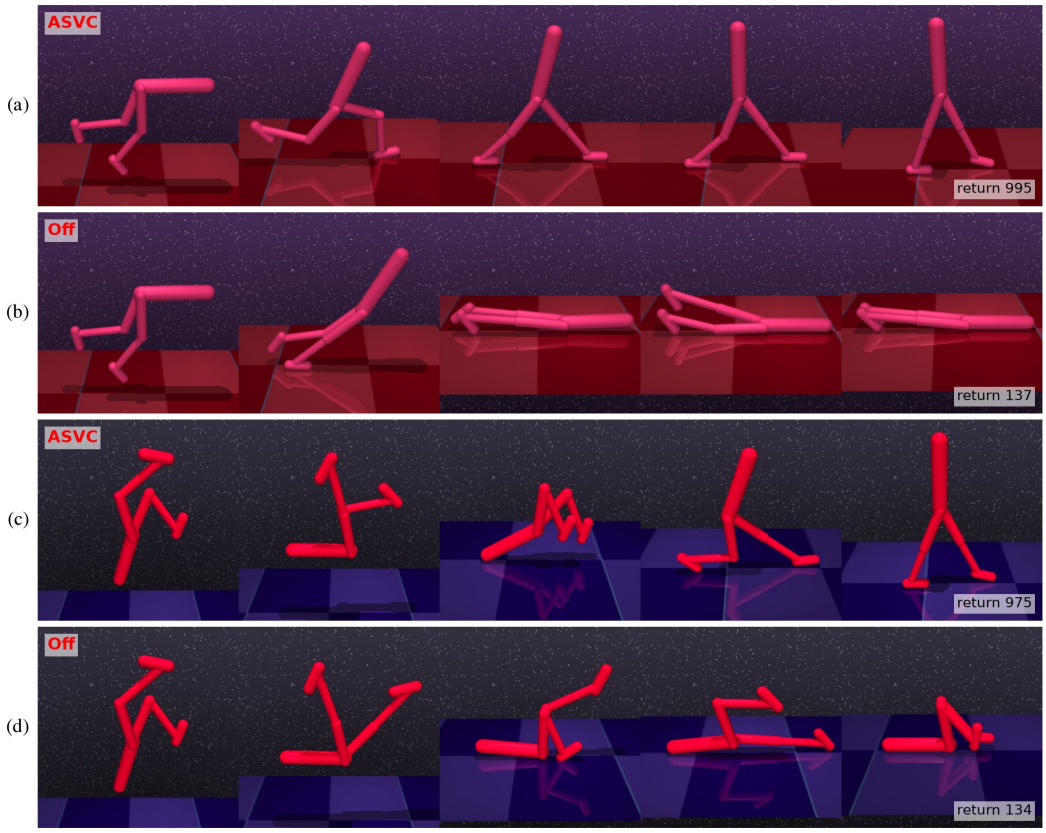

Figure 27: GFP on `walker_stand` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

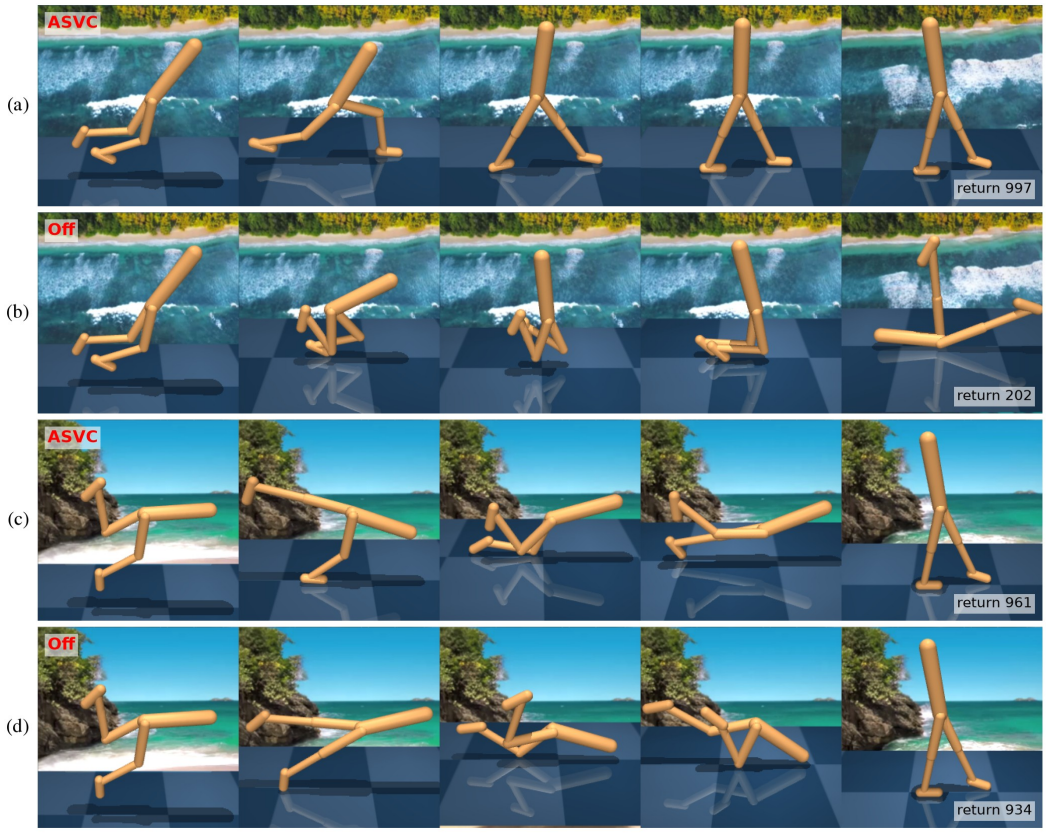

Figure 28: GFP on `walker_stand` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

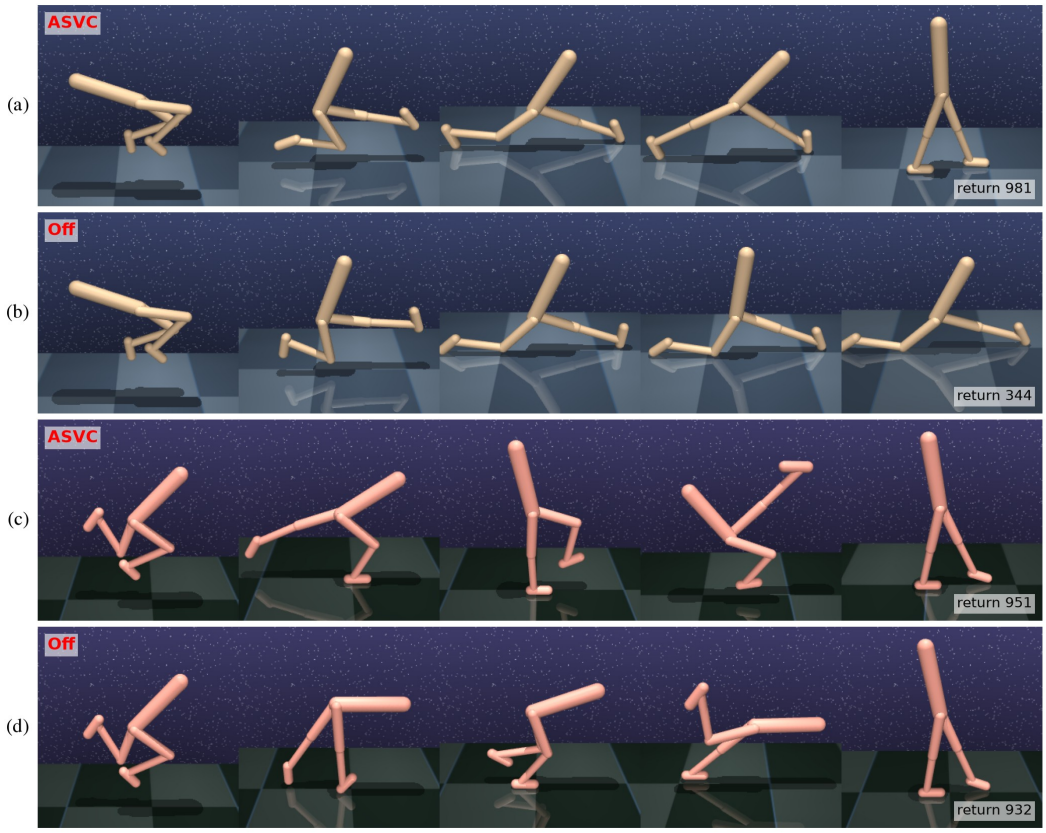

Figure 29: Diffusion Policy on `walker_stand` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

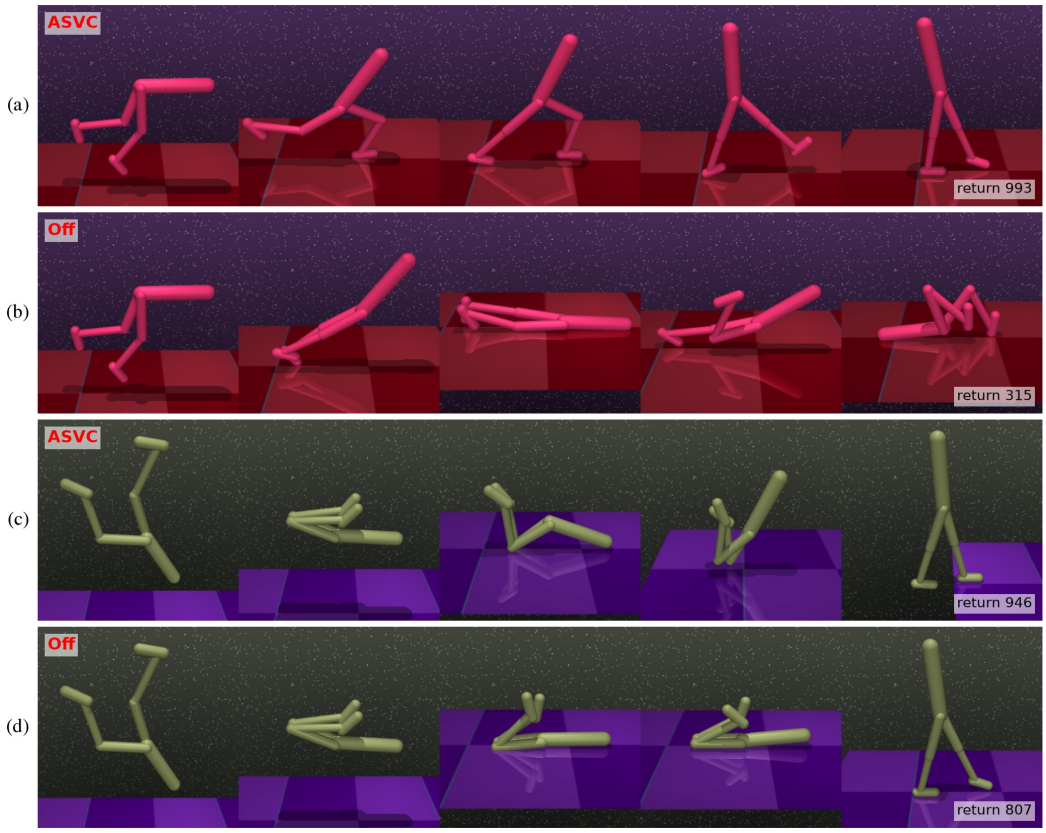

Figure 30: Diffusion Policy on `walker_stand` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.



Figure 31: Diffusion Policy on `walker_stand` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

A.4.3 FINGER_SPIN



Figure 32: Example `finger_spin` training demonstrations.

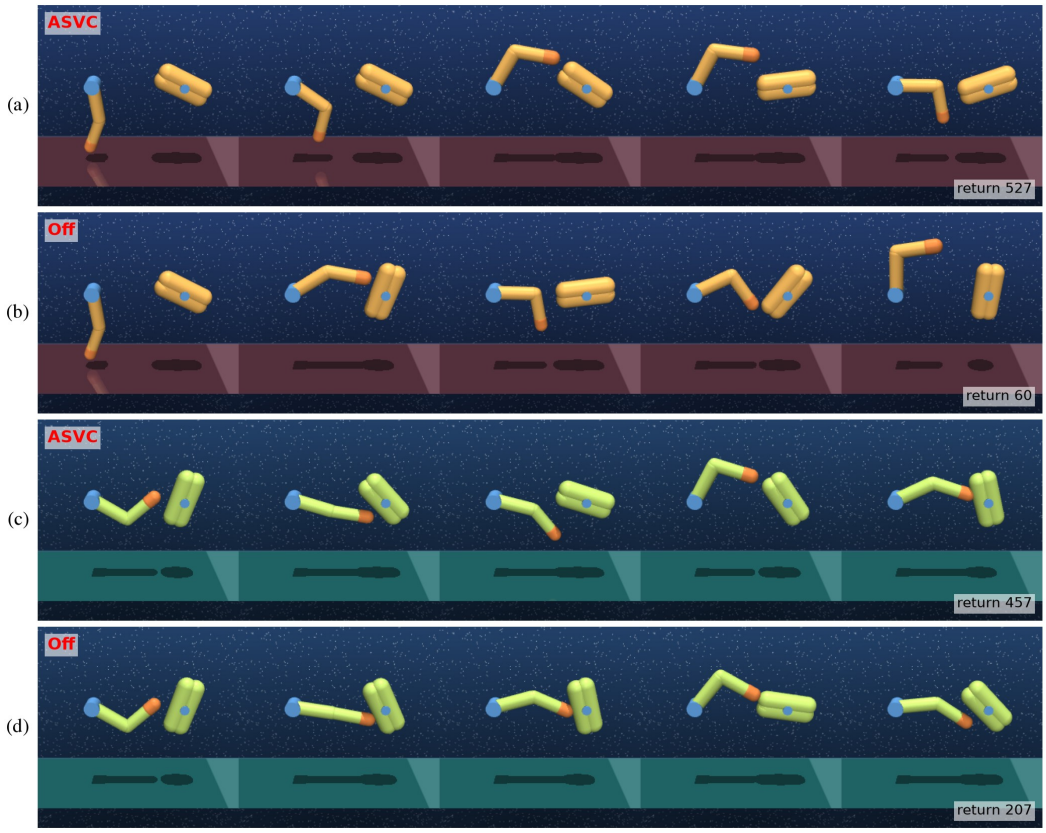

Figure 33: ReinFlow on `finger_spin` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

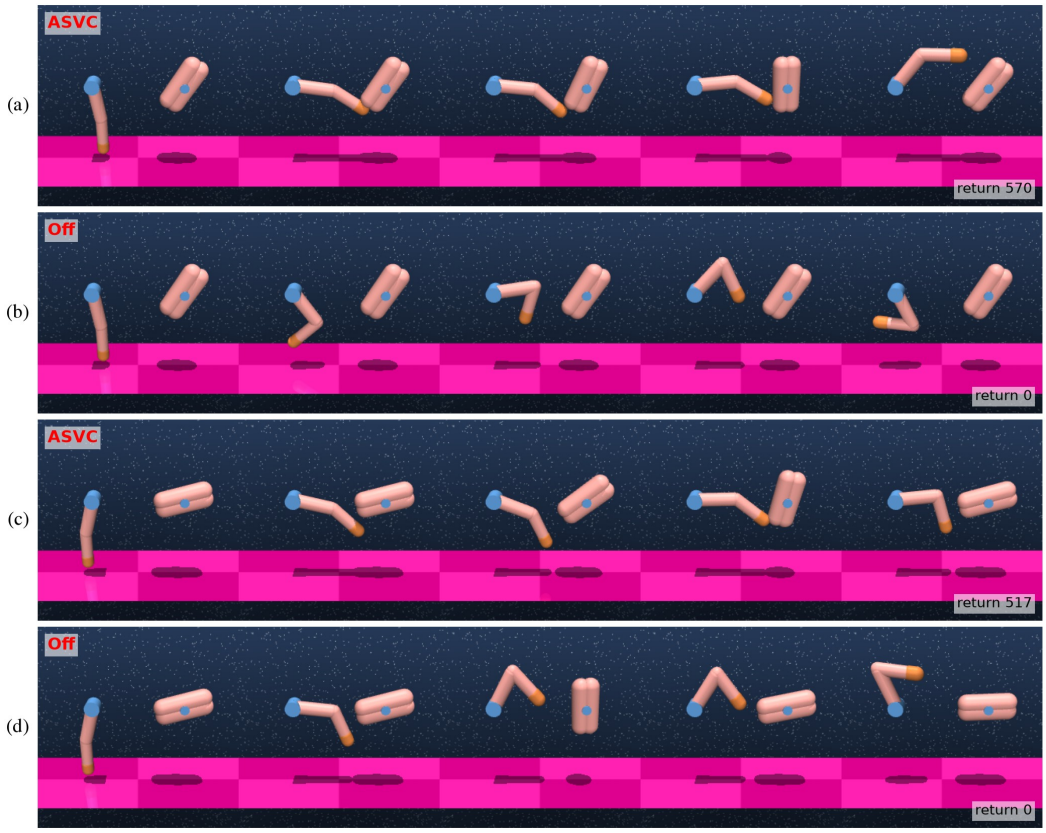

Figure 34: ReinFlow on `finger_spin` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

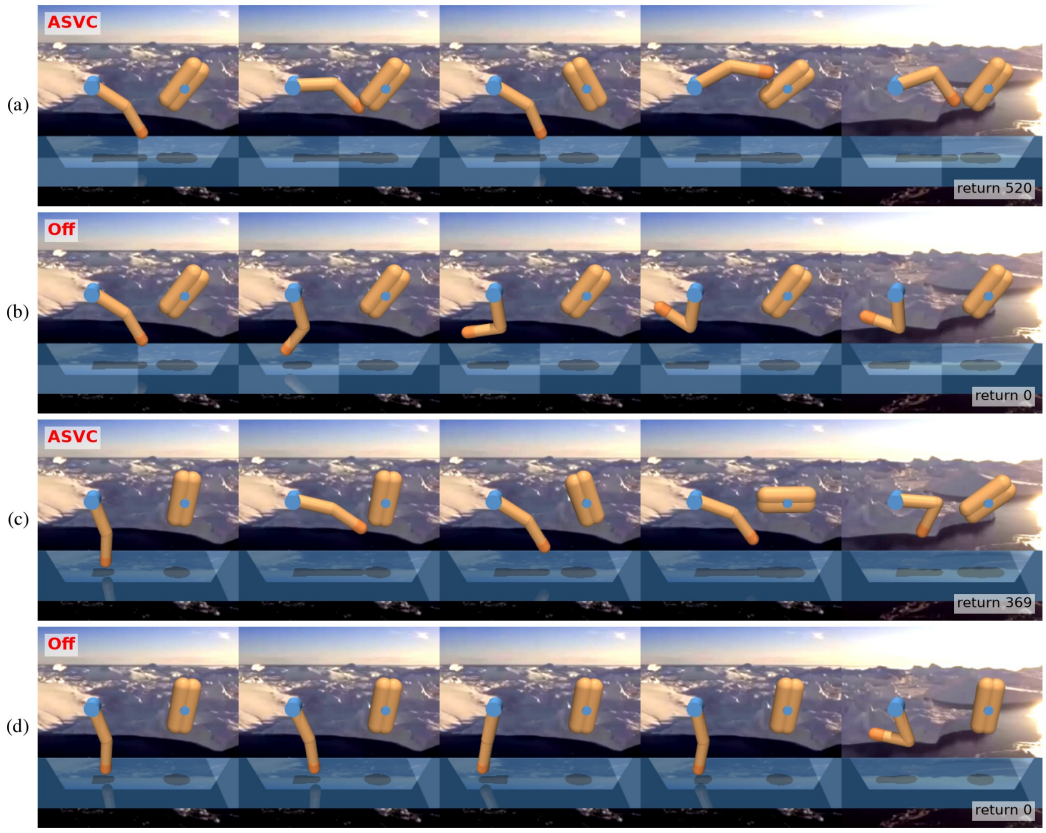

Figure 35: ReinFlow on `finger_spin` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

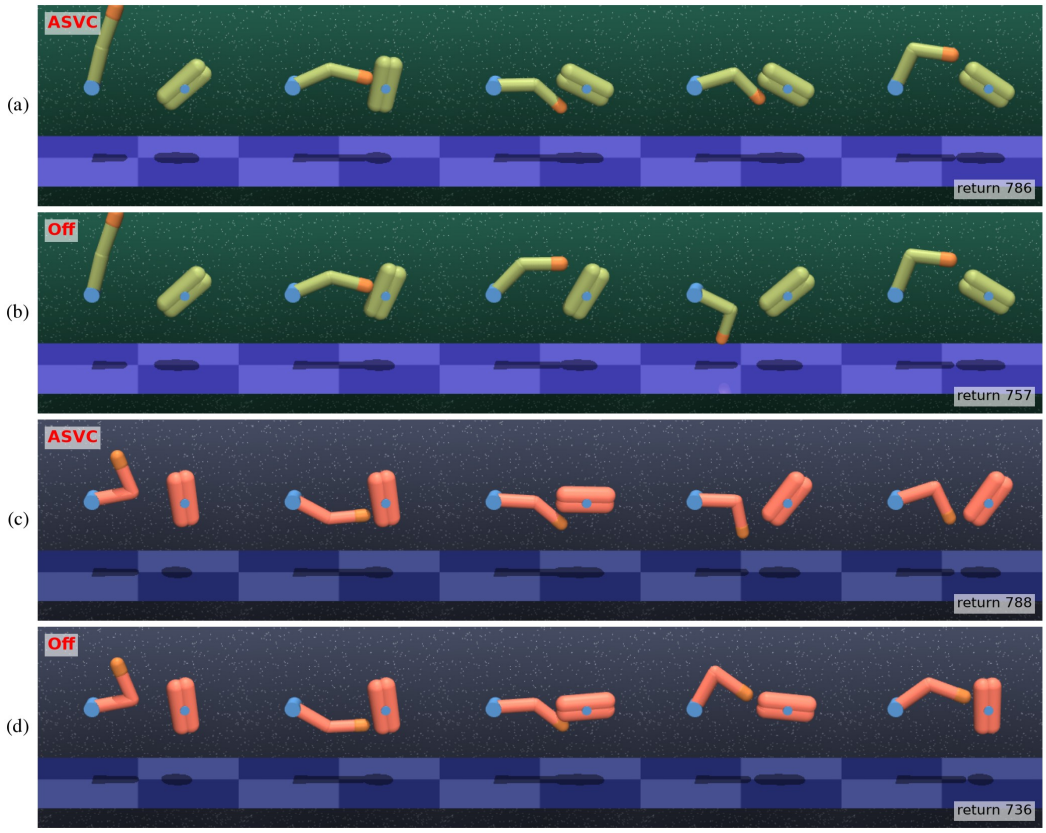

Figure 36: GFP on `finger_spin` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

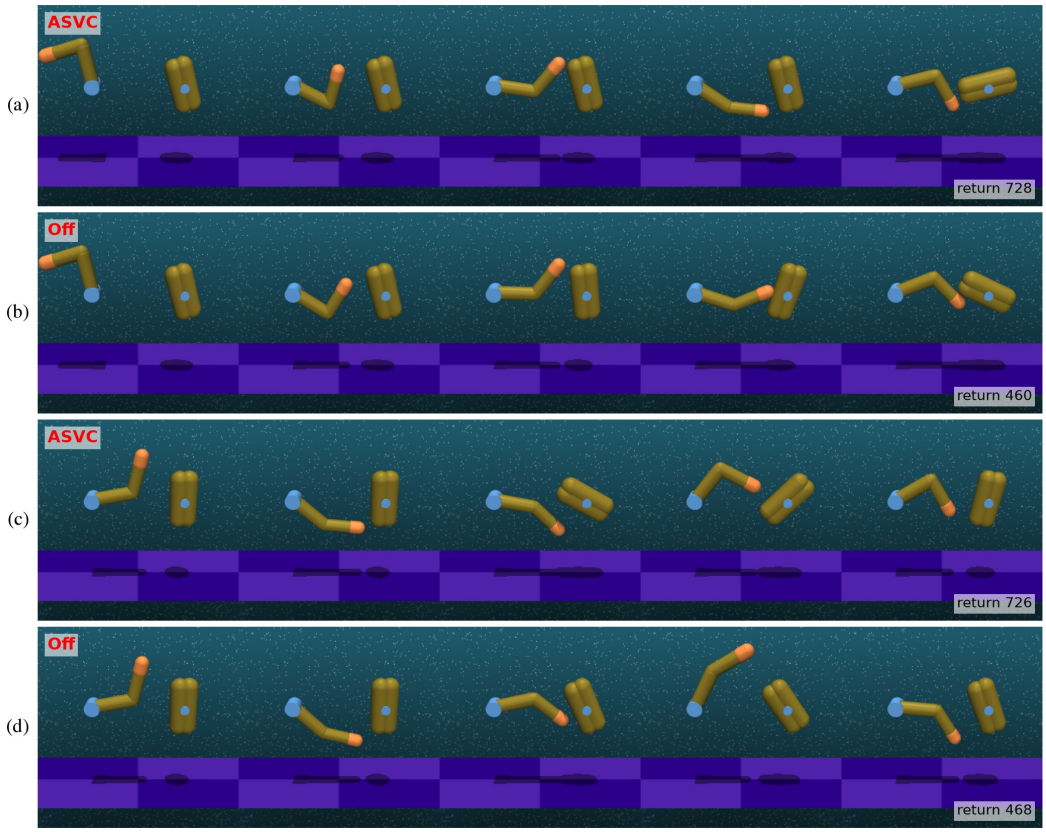

Figure 37: GFP on `finger_spin` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

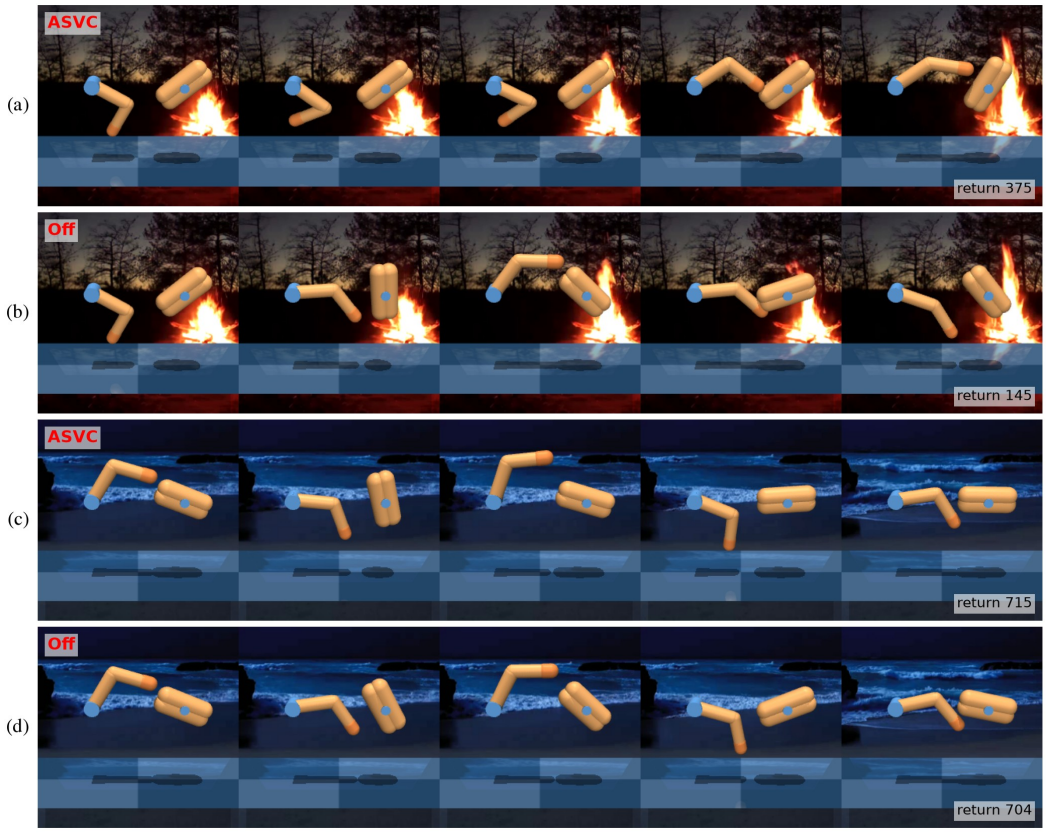

Figure 38: GFP on `finger_spin` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

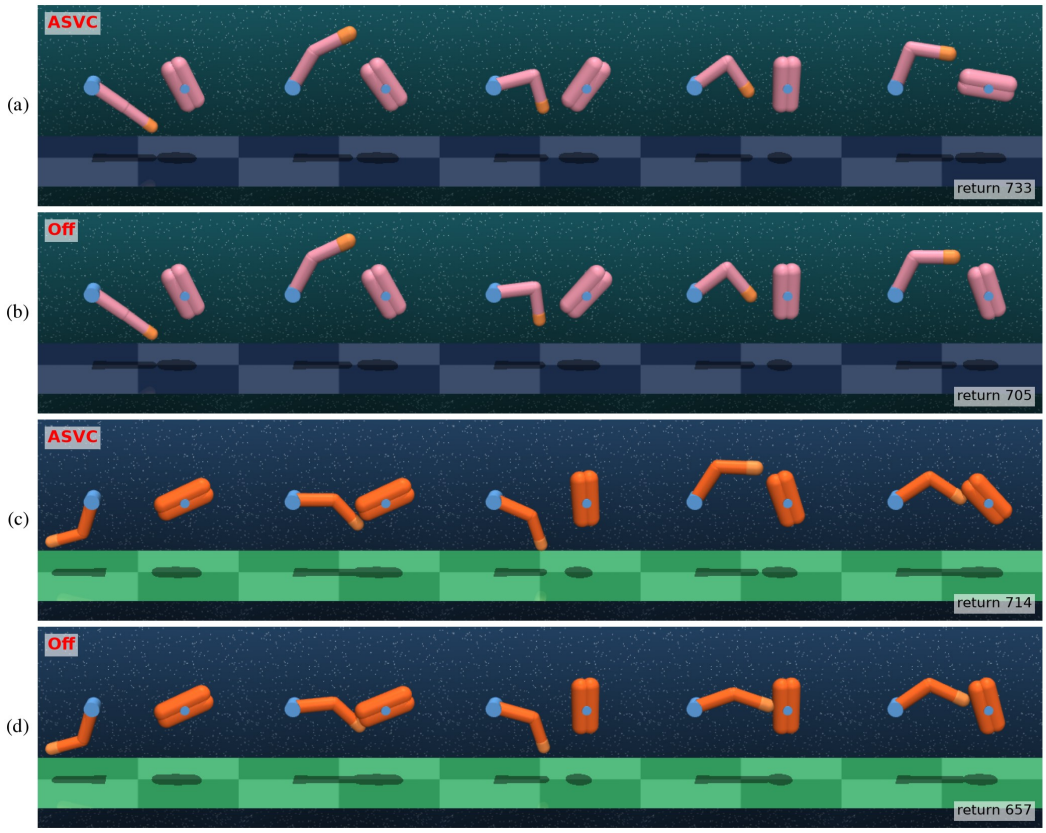

Figure 39: Diffusion Policy on `finger_spin` under `color_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

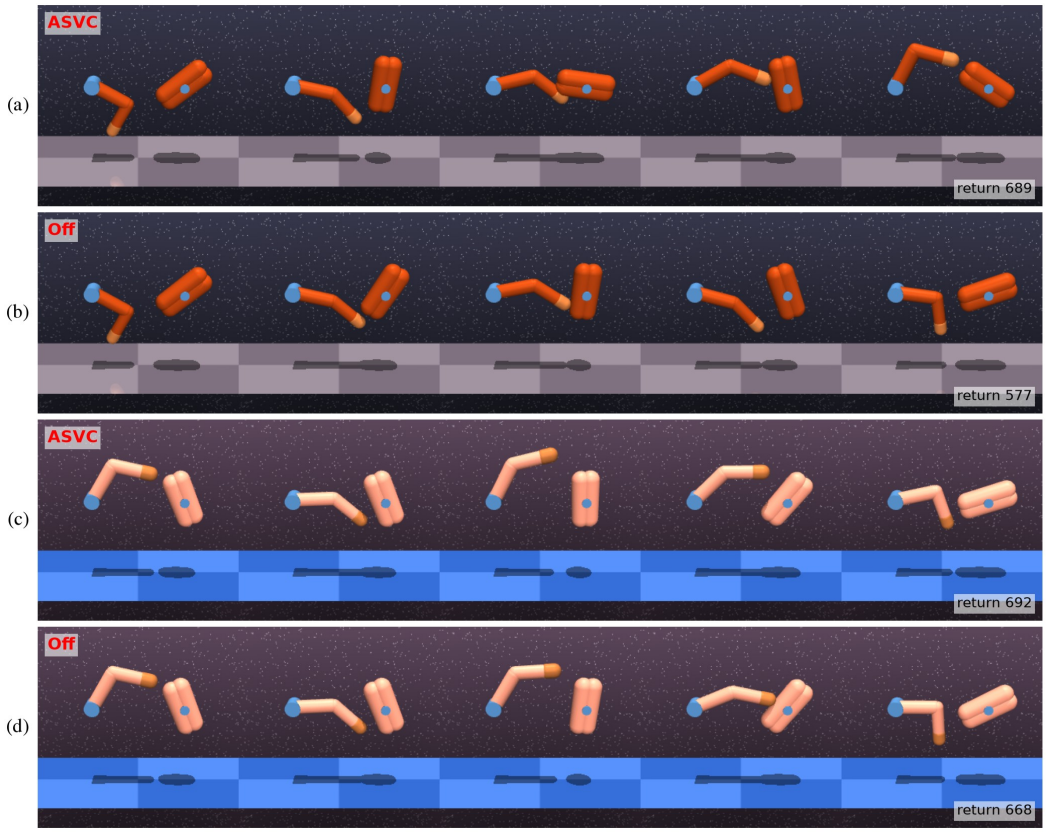

Figure 40: Diffusion Policy on `finger_spin` under `color_hard`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

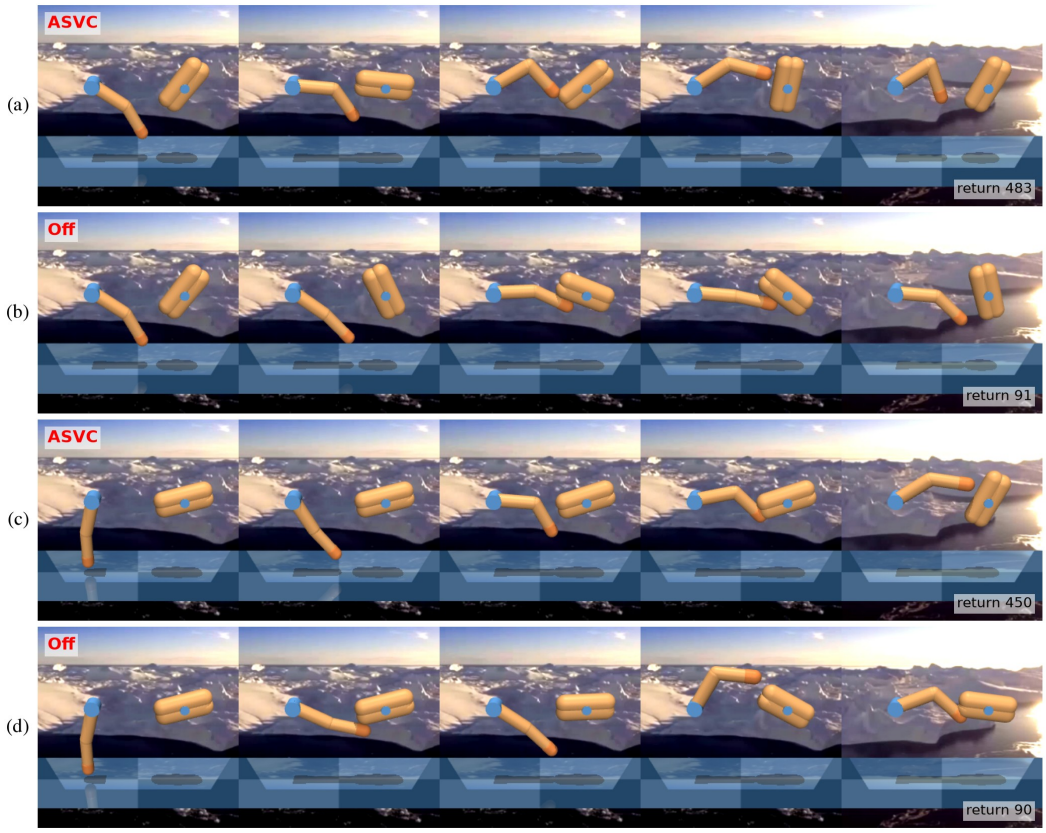

Figure 41: Diffusion Policy on `finger_spin` under `video_easy`. Rows (a) and (b) depict an episode with and without ASVC, and rows (c) and (d) show another episode in the same format.

Table 4: Per-seed DMC-GB2 returns for ReinFlow, mean over 120 paired episodes per cell. Fire is the mean firing fraction of the gate for the corrected arm.

		base			+ASVC			Fire		
Task	Mode	s0	s1	s2	s0	s1	s2	s0	s1	s2
walker_walk	train	742.4	852.3	884.6	740.6	666.6	835.1	0.73	0.77	0.69
	color_easy	285.7	213.1	263.6	299.3	322.0	448.8	0.89	0.87	0.76
	color_hard	110.1	78.0	105.2	210.9	294.2	332.0	0.95	0.94	0.88
	video_easy	153.9	315.7	232.5	302.5	441.8	537.8	0.96	0.88	0.86
walker_stand	train	837.3	866.1	872.8	866.6	813.3	899.6	0.60	0.64	0.60
	color_easy	487.5	417.9	652.3	741.0	663.7	844.6	0.85	0.78	0.73
	color_hard	296.6	236.6	405.6	539.7	571.9	742.0	0.95	0.94	0.86
	video_easy	460.5	456.0	491.8	759.2	616.8	811.2	0.78	0.90	0.75
finger_spin	train	720.2	696.4	713.9	698.6	663.0	683.6	0.44	0.58	0.54
	color_easy	303.1	253.4	260.6	419.2	388.1	375.7	0.55	0.57	0.58
	color_hard	109.0	87.9	93.5	281.8	241.4	253.9	0.67	0.65	0.70
	video_easy	229.7	185.8	138.0	285.1	244.0	253.6	0.74	0.77	0.76

Table 5: Per-seed DMC-GB2 returns for GFP, mean over 120 paired episodes per cell. Fire is the mean firing fraction of the gate for the corrected arm.

		base			+ASVC			Fire		
Task	Mode	s0	s1	s2	s0	s1	s2	s0	s1	s2
walker_walk	train	905.0	934.1	938.3	904.3	932.0	924.1	0.56	0.54	0.56
	color_easy	742.1	740.2	789.8	753.4	855.6	804.5	0.57	0.62	0.57
	color_hard	473.7	430.5	520.4	671.2	738.5	731.2	0.59	0.67	0.56
	video_easy	416.4	570.5	512.4	582.7	715.3	677.1	0.64	0.59	0.55
walker_stand	train	951.2	969.5	974.2	965.1	960.5	974.9	0.44	0.54	0.58
	color_easy	939.8	936.9	965.8	929.0	940.6	968.7	0.64	0.59	0.66
	color_hard	820.4	700.0	879.5	925.1	924.4	953.8	0.67	0.67	0.79
	video_easy	663.3	830.9	761.1	910.5	932.9	945.5	0.71	0.54	0.83
finger_spin	train	780.0	770.0	760.3	770.8	765.7	754.0	0.65	0.33	0.54
	color_easy	756.3	697.5	686.7	758.4	705.7	702.4	0.55	0.47	0.45
	color_hard	635.2	485.1	499.6	700.2	556.3	578.6	0.50	0.50	0.40
	video_easy	418.9	410.1	305.3	435.0	411.4	369.4	0.54	0.56	0.57

A.4.4 PER-SEED RESULTS

Tables 4, 5, and 6 show the per-seed results of Table 1, evaluated over 120 paired episodes. Under shift, our method improves the base policy on all 3 seeds in 23 of 27 settings. On the clean mode, ASVC slightly degrades returns on most seeds, with the largest drop on ReinFlow walker_walk.

Table 6: Per-seed DMC-GB2 returns for Diffusion Policy, mean over 120 paired episodes per cell. Fire is the mean firing fraction of the gate for the corrected arm.

		base			+ASVC			Fire		
Task	Mode	s0	s1	s2	s0	s1	s2	s0	s1	s2
walker_walk	train	833.0	878.9	910.5	815.3	892.6	891.6	0.50	0.60	0.55
	color_easy	595.6	650.6	683.9	688.6	837.2	824.0	0.59	0.60	0.53
	color_hard	490.7	556.4	571.9	601.1	744.8	747.9	0.72	0.68	0.64
	video_easy	457.4	570.9	503.4	594.0	799.3	678.3	0.58	0.58	0.50
walker_stand	train	946.2	914.0	958.4	938.5	914.4	953.4	0.62	0.73	0.64
	color_easy	924.2	820.2	923.3	918.4	889.9	943.8	0.70	0.58	0.49
	color_hard	838.2	768.4	905.8	916.6	877.1	930.6	0.68	0.64	0.46
	video_easy	823.2	776.0	864.9	927.9	866.6	937.2	0.62	0.60	0.50
finger_spin	train	746.7	748.5	747.7	737.7	738.8	741.3	0.48	0.46	0.44
	color_easy	708.3	716.1	677.0	697.6	693.8	684.2	0.54	0.53	0.50
	color_hard	633.0	643.8	564.2	665.5	641.7	620.0	0.59	0.57	0.56
	video_easy	359.8	365.5	343.9	430.6	421.5	424.0	0.68	0.63	0.66

A.5 Robot Experiments

A.5.1 Robot Setup

Our robot experiment uses a stationary Trossen WidowX AI robot with two arms. For the cube stacking experiment, our policy exclusively controls the right arm using 7 joints: six arm joints and one gripper joint. Observations come from four cameras at 15 frames per second. We use an adapted Diffusion Policy with a rectified-flow action head, which re-plans at every control step and averages action chunks using temporal ensembling. Training data includes 100 recorded expert demonstrations of the stacking task where the red cube position and orientation is fixed while the blue cube position and orientation varies.

Evaluation covers the source scene and two shift conditions: added distractors and room lighting change. For the distractor shift, office supplies were added to the robot desk to obstruct the cameras' views, although the supplies were not placed in a way that would obstruct the robot from performing the task. For the lighting shift, the room lights were dimmed and an external light source was placed to cast shadows. The lights were dimmed enough so that the cameras could still see the task being performed. For each shift, we place the blue cube at 10 fixed start locations and run 3 episodes with ASVC and 3 episodes without ASVC. An episode succeeds when the blue cube rests on the red cube with the gripper released. Figures 42 to 44 show start locations of each shift from the overhead camera.

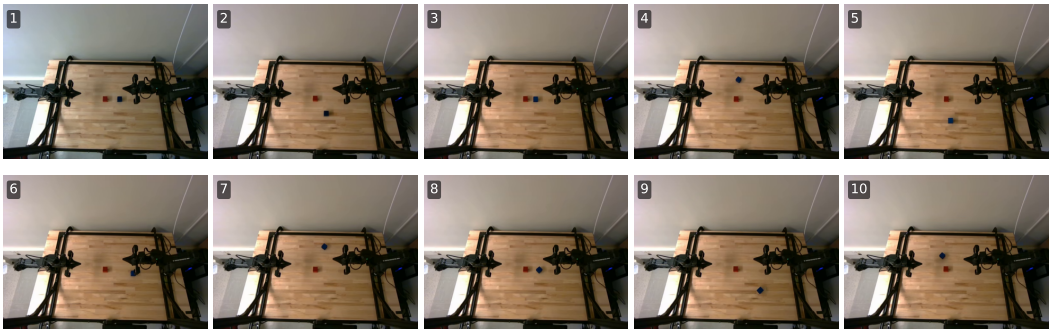

Figure 42: Overhead camera showing start locations of the blue cube under the source scene.

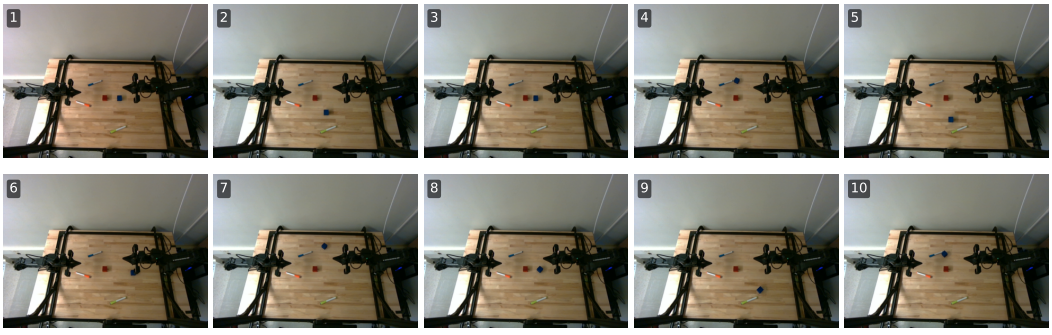

Figure 43: Overhead camera showing start locations of the blue cube under added distractors.

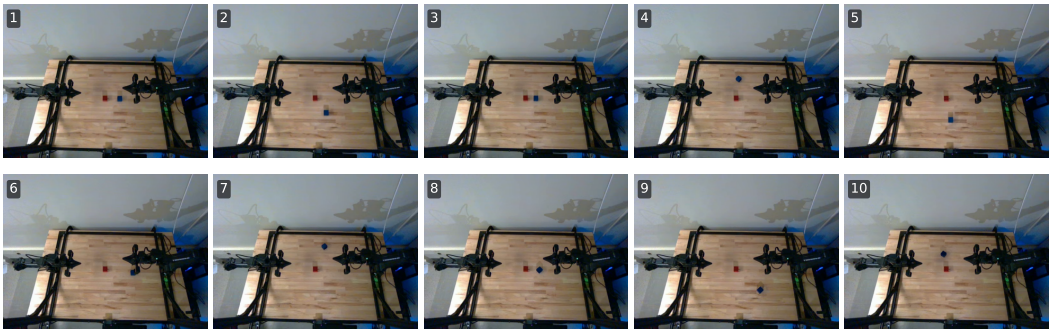

Figure 44: Overhead camera showing start locations of the blue cube under the room lighting change.

A.5.2 TRAINING SAMPLES

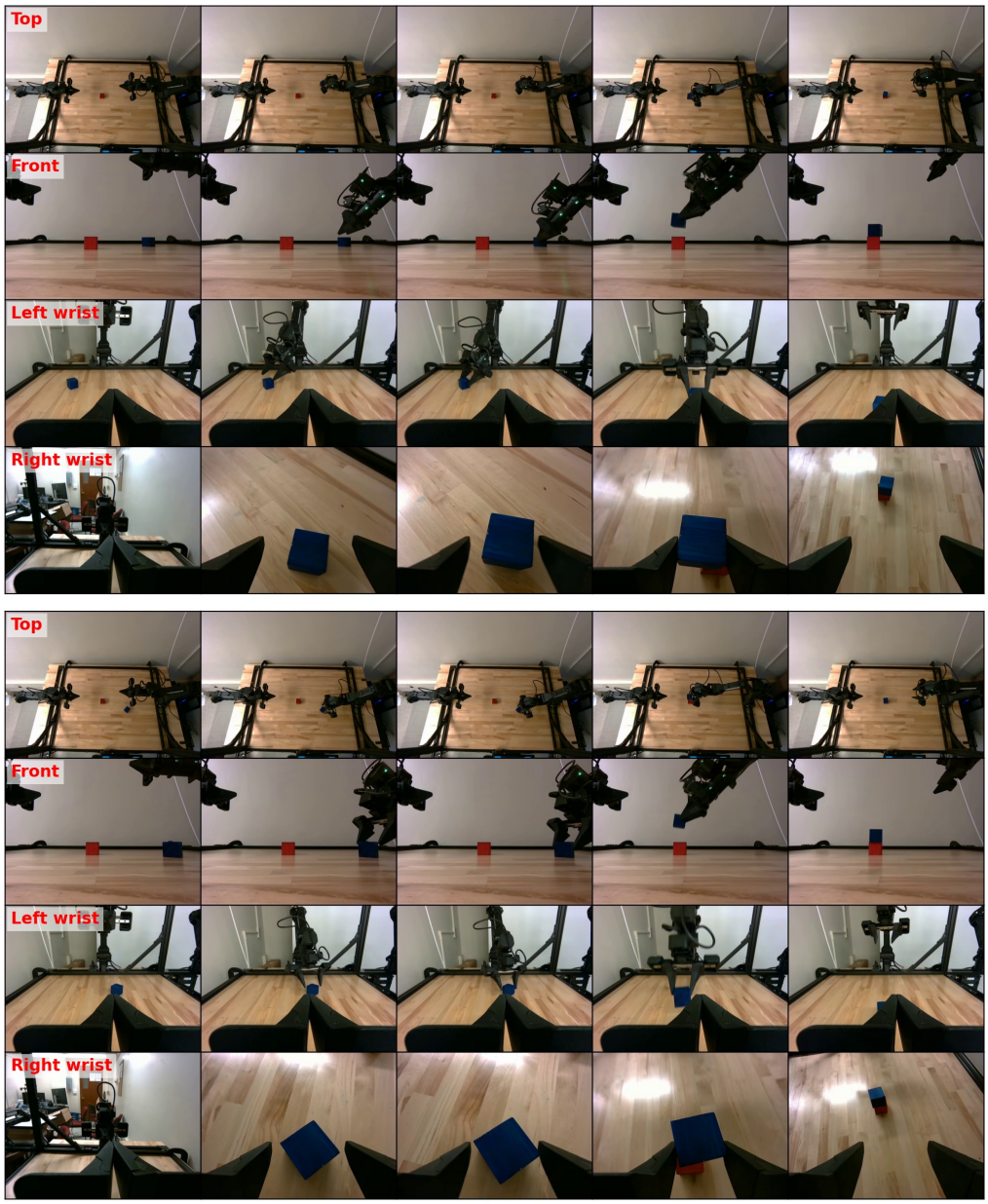

Figure 45: Two robot training demonstrations over the four cameras.

A.5.3 SOURCE CONDITION

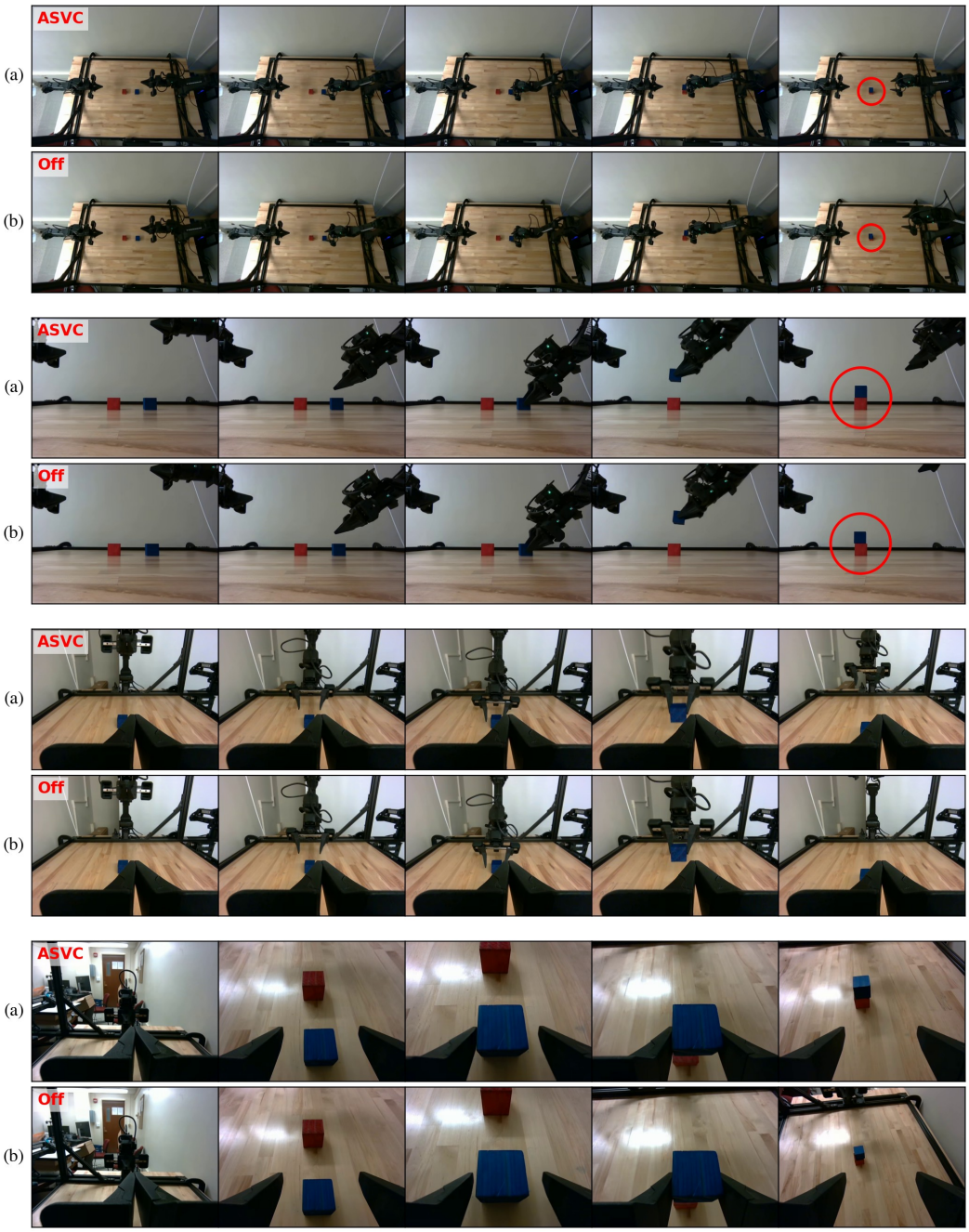

Figure 46: Rollouts on the source condition.

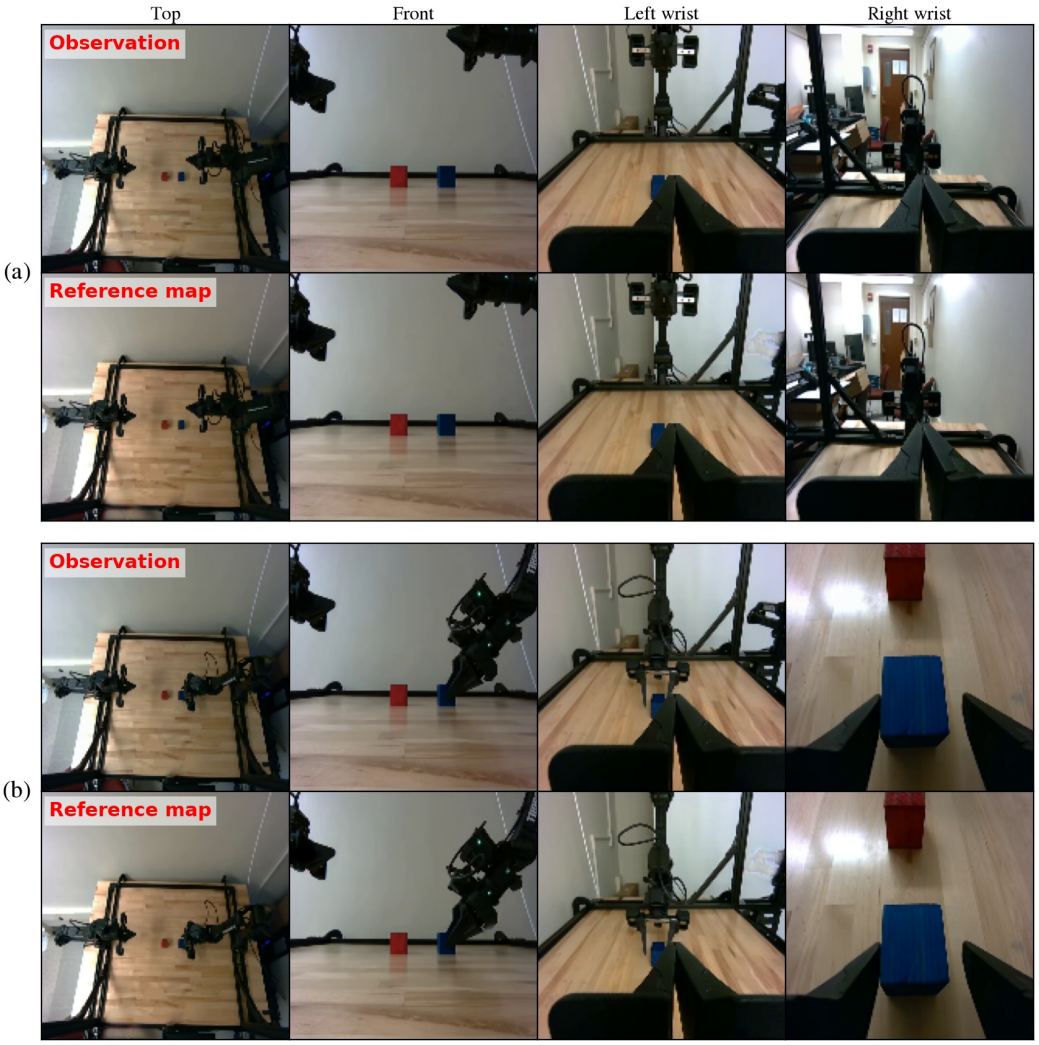

Figure 47: Robot reference map samples under the source scene.

A.5.4 DISTRACTOR CONDITION

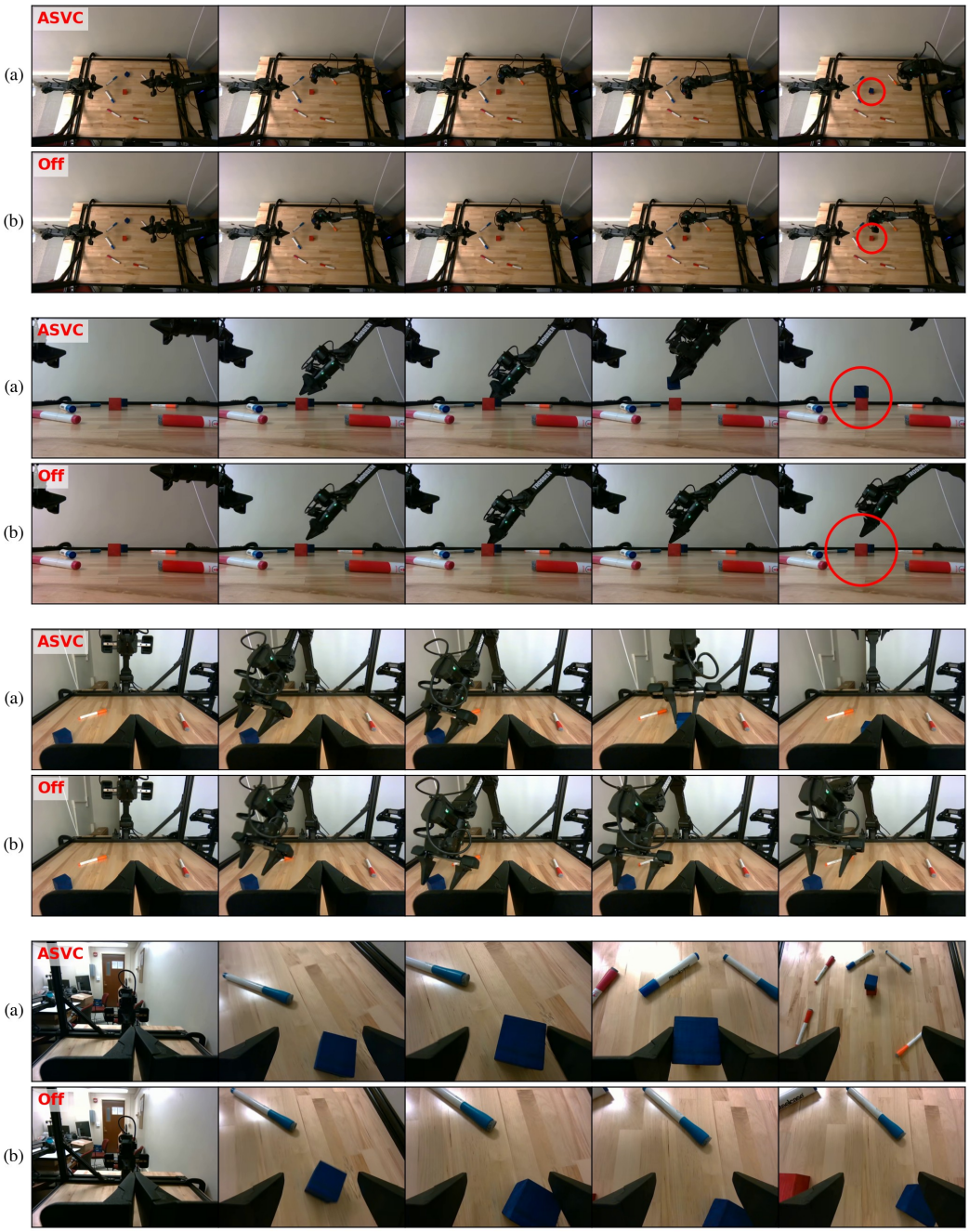

Figure 48: Rollouts under distractor objects.

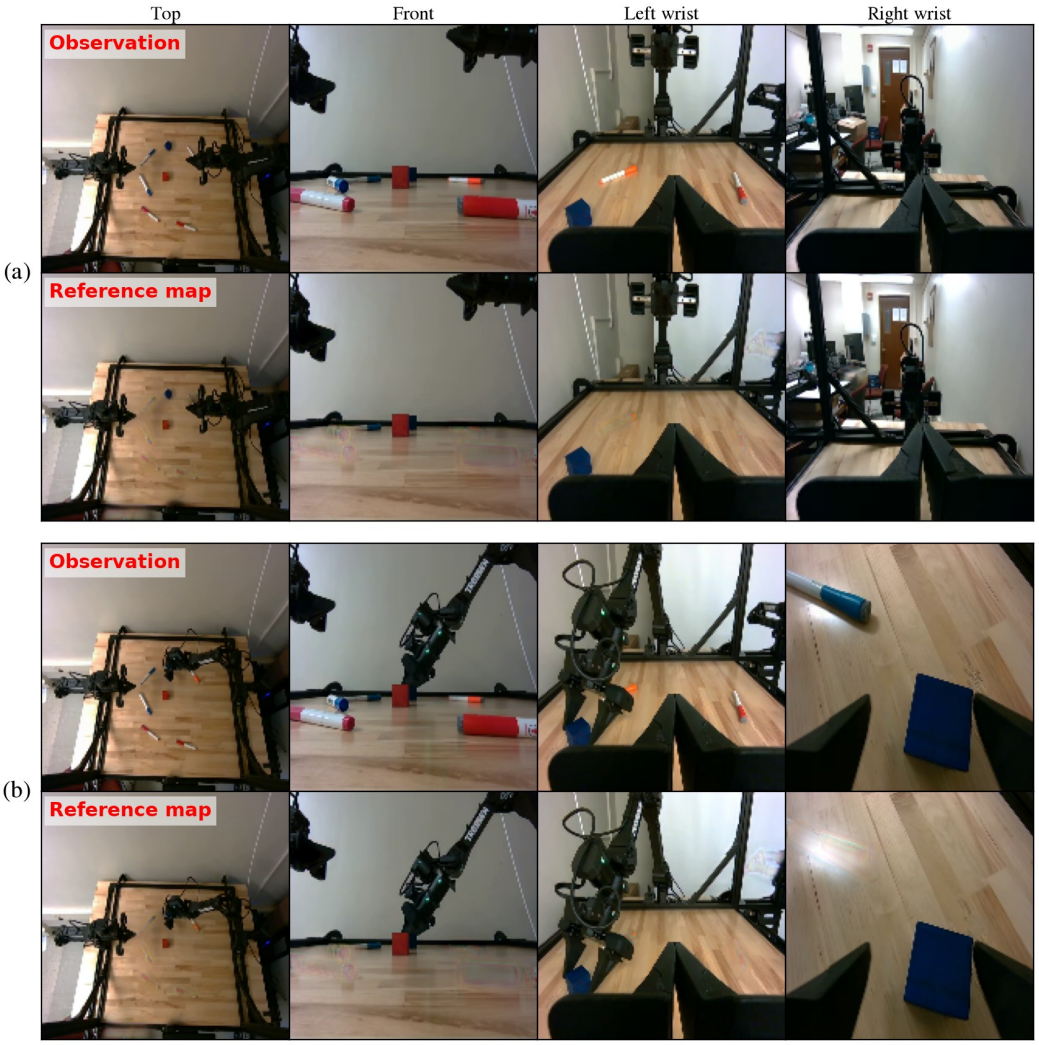

Figure 49: Robot reference map samples under added distractors.

A.5.5 LIGHTING CONDITION

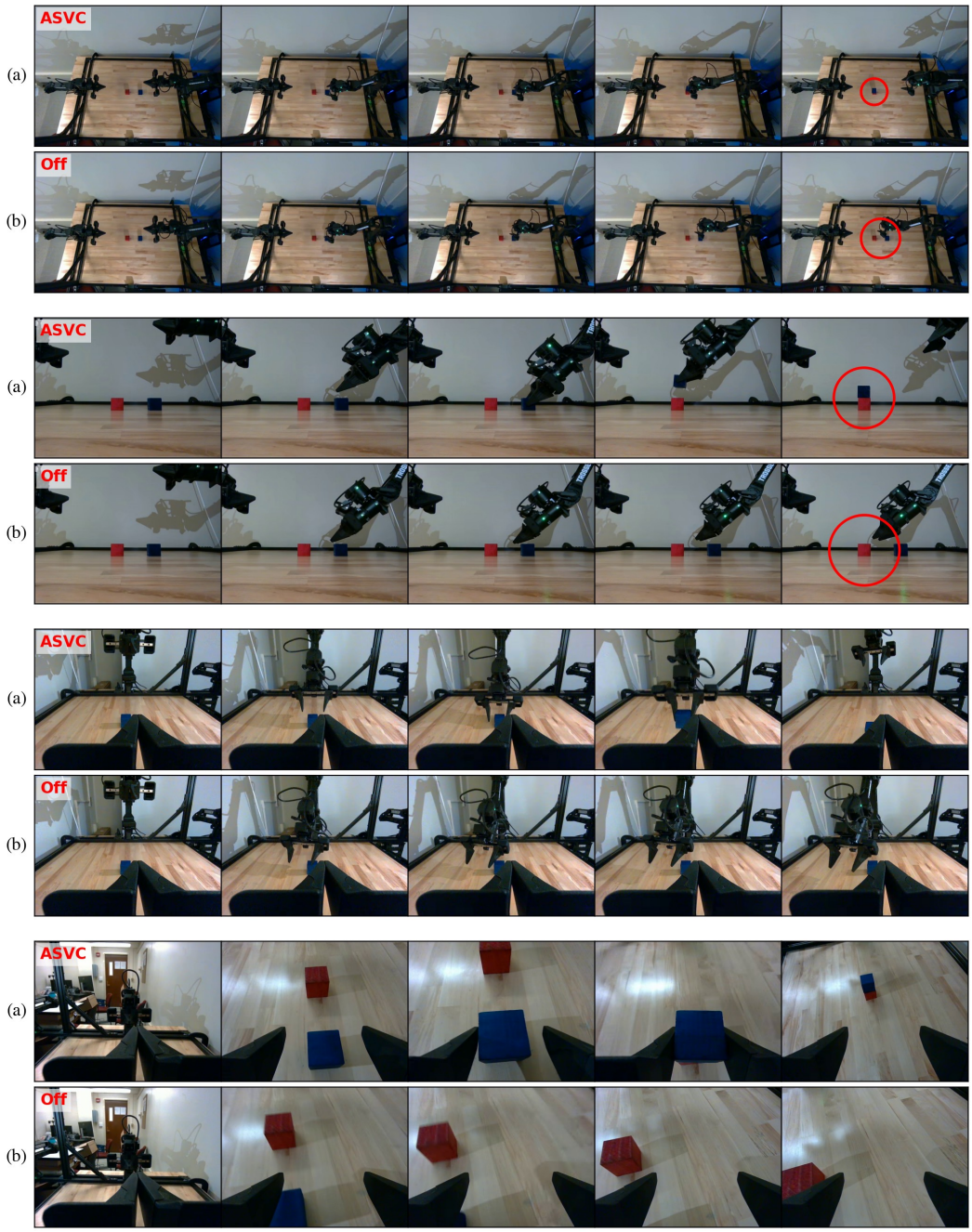

Figure 50: Rollouts under the lighting change.

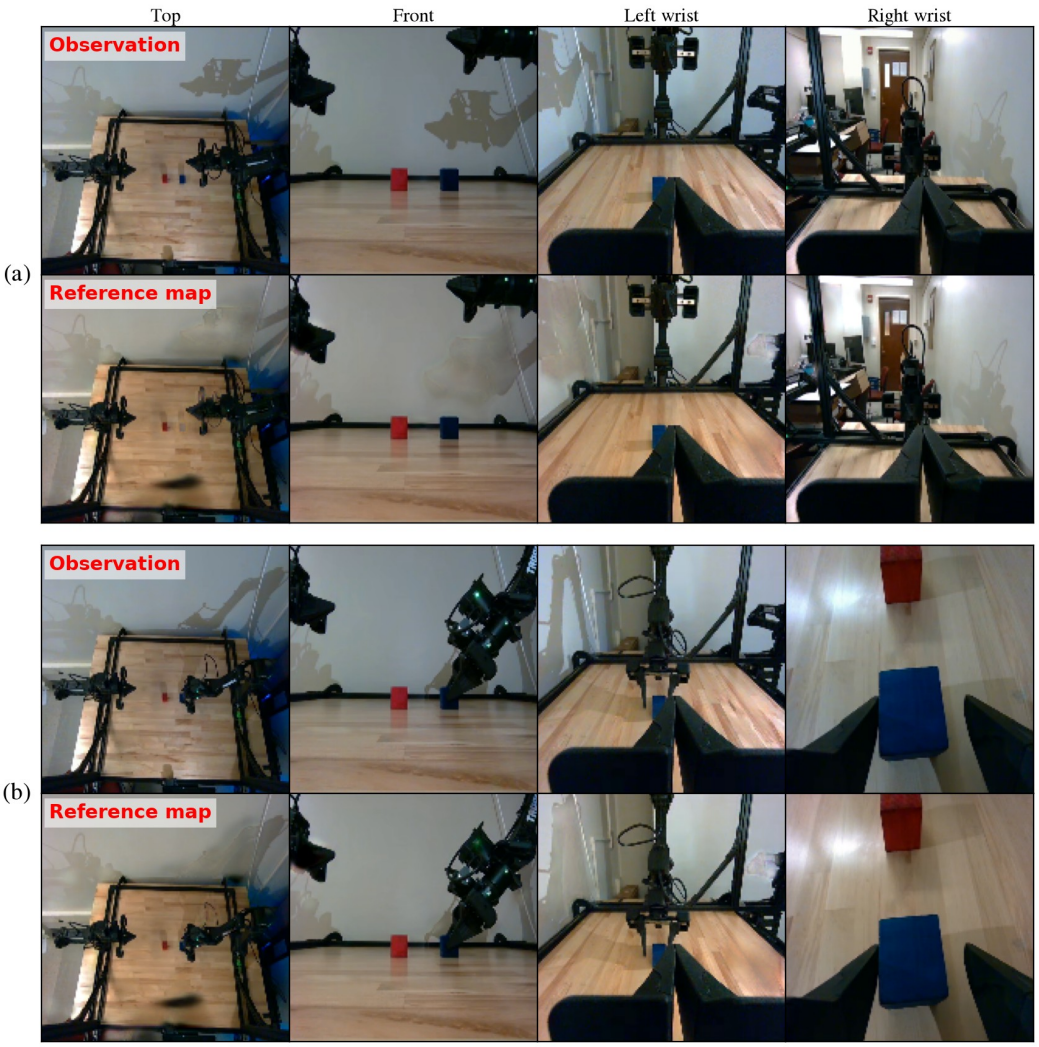

Figure 51: Robot reference map samples under the room lighting change.

A.5.6 AI DISCLOSURE

In this work, we used generative AI tools for assisting in the writing of proofs, providing feedback on research methods and experiments, and implementing methods using code generation. We have not used generative AI tools for generating synthetic datasets, assist in the development of theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, or interpreting results. Additionally, we used generative AI tools for creating or editing software code, brainstorming, sourcing/searching for information, identifying relevant literature, editing a research paper to improve readability, and proposing a title or keywords for a research paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.