Dream2Flow: Bridging Video Generation and Open-World Manipulation with 3D Object Flow

Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei*, Ruohan Zhang*
Stanford University

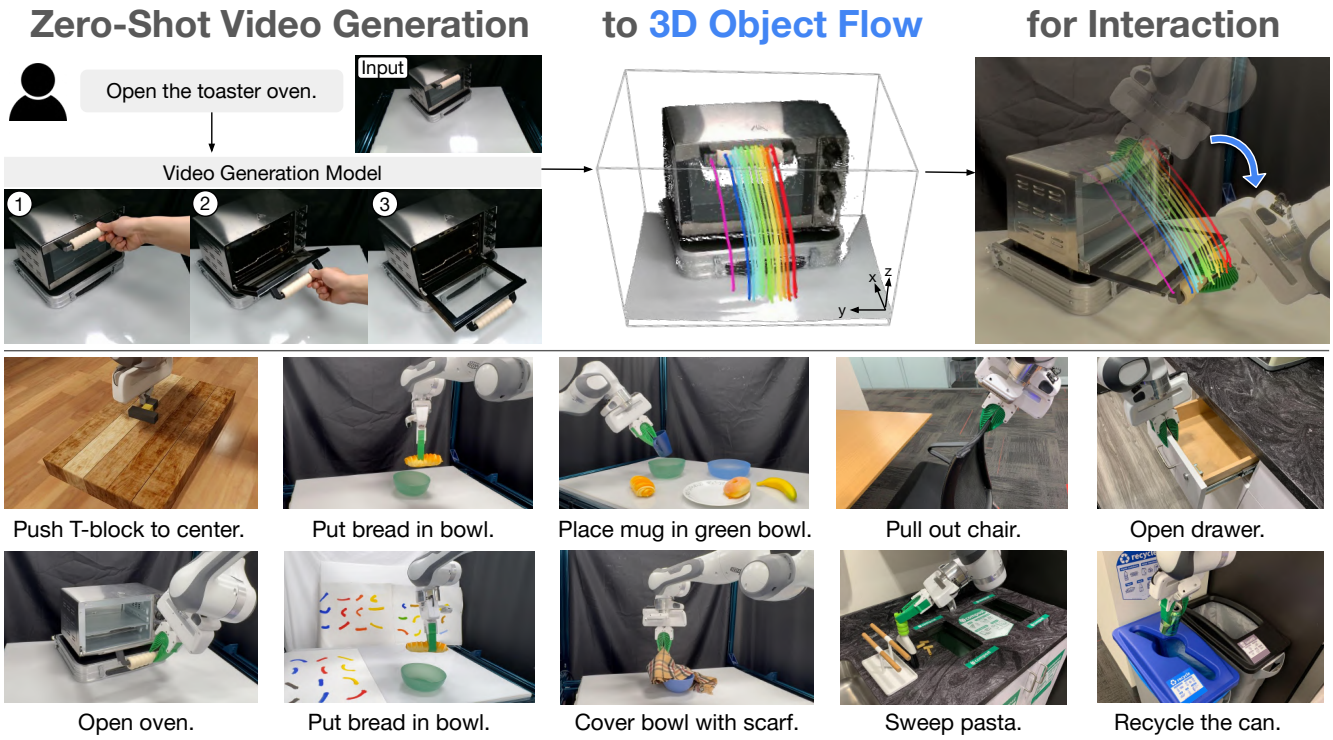

Fig. 1: **Dream2Flow** leverages off-the-shelf video generation models to produce videos of the task being performed in the same scene of the robot. Dream2Flow then extracts a 3D object flow from the motion in the video, allowing for downstream planning and execution with a robot across a wide variety of tasks.

***Abstract*— Generative video modeling has emerged as a compelling tool to zero-shot reason about plausible physical interactions for open-world manipulation. Yet, it remains a challenge to translate such human-led motions into the low-level actions demanded by robotic systems. We observe that given an initial image and task instruction, these models excel at synthesizing sensible object motions. Thus, we introduce Dream2Flow, a framework that bridges video generation and robotic control through 3D object flow as an intermediate representation. Our method reconstructs 3D object motions from generated videos and formulates manipulation as object trajectory tracking. By separating the state changes from the actuators that realize those changes, Dream2Flow overcomes the embodiment gap and enables zero-shot guidance from pre-trained video models to manipulate objects of diverse categories—including rigid, articulated, deformable, and granular. Through trajectory optimization or reinforcement learning, Dream2Flow converts reconstructed 3D object flow into executable low-level commands without task-specific demonstrations. Simulation and real-world experiments highlight 3D object flow as a general and scalable interface for adapting video generation models to open-world robotic manipulation. Videos, visualizations, and appendix are available at https://dream2flow.github.io/.**

I. Introduction

Robotic manipulation in the open world could greatly benefit from visual world models that predict how an environment would evolve given an agent's interactions. Recent advances in generative video modeling have produced systems capable of zero-shot synthesizing minute-long, high-fidelity clips of physical interactions in pixel space, conditioned on an unseen initial image and an open-ended task instruction [1]. Such video models implicitly capture intuitive physics and rich priors of object properties and interactions, making them compelling for open-world manipulation settings where a robot is tasked to complete novel tasks in unseen environments through partial observations.

Despite their promise, it remains unclear what role such models should serve in a robot manipulation system. Most frontier video generators produce the best interaction clips with a human embodiment. This reflects where supervision is most abundant as human interactions are significantly more

*Equal Advising. Correspondence: Wenlong Huang

broadly documented than those of robots. But this becomes a challenge for using them in robotic manipulation due to the embodiment gap and hence the different action spaces.

We propose extracting actionable signals from their visual predictions of human interactions, which will then be enacted by a robot. This essentially separates the state changes in the real world from the actuators that realize those changes. Our proposed method, **Dream2Flow**, employs **3D object flow** as an intermediate interface that bridges high-level video simulation with low-level robot actions. Dream2Flow works because, despite occasional visual artifacts, state-of-the-art video generation models often predict physically plausible object motions that align with the task intent in open-world manipulation tasks. Then, given generated videos, instead of trying to directly mimic the human motions for completing a given task, we focus on reconstructing and reproducing the object flows in 3D.

The problem is thus reduced to object trajectory tracking: the robot's job is to manipulate the object to closely follow the generated flow that the video model imagined. This approach cleanly separates what needs to happen (i.e., state changes in an environment) from how a particular embodiment achieves it with respect to its kinematic and dynamic constraints (i.e., actions). Importantly, it seamlessly interfaces with both motion planners and sensorimotor policies—the extracted object motion in 3D serves as the tracking goal for trajectory optimization or a reinforcement learning policy, which then yields a sequence of low-level robot joint commands.

Leveraging off-the-shelf models and tools, we demonstrate an autonomous pipeline that 1) generates a text-conditioned video of plausible interaction [1], 2) obtains 3D object flow by performing depth estimation and point tracking using vision foundation models [2, 3], and 3) synthesizes robot actions that realize this flow using trajectory optimization and reinforcement learning. Notably, this design enjoys a number of desirable benefits in manipulation tasks. By first leveraging video models—pre-trained on large corpora of human activities—to interpret and ground open-ended language commands in visual predictions, our system inherits a scalable mechanism for task specification. These predictions are then distilled into reconstructed 3D object flow, a representation that naturally captures diverse object interactions spanning rigid, articulated, deformable, and granular objects. Together, this synergy enables an end-to-end pipeline that performs open-world manipulation directly from visual perception and language, without task-specific data or training.

In summary, our key contributions are:

- We propose 3D object flow as an interface for adapting off-the-shelf video generation models for open-world manipulation by formulating it as an object trajectory tracking problem.
- We demonstrate its effectiveness by implementing the approach in both simulated and real domains, which performs diverse tasks given only RGB-D observations and language instructions in a zero-shot manner.
- We examine the properties of 3D object flow by comparing it with alternative intermediate representations and by studying its key design choices as well as generalization properties.

II. Related Works

A. Task Specification in Manipulation

Specifying tasks for the wide range of manipulation problems spans symbolic, learning-based, outcome-driven, and object-centric interfaces. Classical approaches encode goals and constraints with symbolic formalisms such as PDDL and temporal logics, or optimize cost-augmented formulations [4–6]. Learning-based systems specify tasks often through language and perception, mapping instructions to actions via language-conditioned visuomotor policies and vision–language–action models [7–11]. Outcome-based specification sets goals by example observations, e.g., image goals with goal-conditioned policies [12–17], and some works incorporate force targets [18]. Object-centric alternatives rely on descriptors or keypoints to capture task-relevant structure [19]. Recently, foundation models enable higher-level interfaces that compile intent into actionable specifications via code [20], 3D value maps akin to potential fields [21], keypoint relations [22], or affordance maps [23].

B. 2D/3D Flow in Robotics

Dense motion fields—optical flow, point tracks, and 3D scene/object flow—provide an embodiment-agnostic, mid-level interface for manipulation [24, 25]. In scene-centric formulations, policies parameterize or condition on motion in 2D or 3D to decide actions, with point/keypoint interfaces unifying perception and action and with action-flow improving precision [26–33]. In object-centric formulations, desired object motion is specified independently of embodiment and then converted into actions via policy inference, planning, and optimization [34–40]. When actions are absent, retargeting with predicted tracks and dense correspondences offers a practical bridge across embodiments [41–43]. Advances in perception and tracking such as RGB-D motion-based segmentation, 3D scene flow in point clouds, zero-shot monocular and ToF-based scene flow, rigid-motion learning, refractive flow for transparent objects, deformable and articulated reconstruction, and structured scene representations [44–54] make the interface reliable. Flow-derived supervision further supports learning, and video generation supplies plausible visual rollouts for planning and imitation [55–60]. Our approach follows the object-centric path by reconstructing 3D object flow from language-conditioned generations and tracking it under embodiment constraints, complementing other flow-conditioned policy representations [26, 28, 34, 36, 37].

C. Video Models for Robotics

Recent work increasingly integrates video models across robotic tasks in various ways [61, 62]. They can serve as auxiliary training objectives [63–68], as reward models [69–71], as policies [72, 73], or as a simulator for the environments [74, 75]. Notably, predictive modeling in robotics can

leverage video frame prediction as a form of world model. By simulating future visual observations, these models can enable visual planning and manipulation by anticipating how the environment will evolve. For example, a video generative model can simulate long-horizon task outcomes, effectively acting as a visual planner [76]. Video models in this way can also serve as as dynamics models [73, 77–81]. Another notable direction is that video generation can directly provide new training data for robot learning. Several recent works devise frameworks to imagine new trajectories in the form of videos and use them to train or finetune policies, particularly for imitation learning [82, 83].

III. Method

Herein, we introduce the problem formulation of Dream2Flow in Sec. III-A. Leveraging 3D object flow as an interface, we subsequently discuss how to extract 3D object flow from video generations in Sec. III-B and how to plan actions with 3D object flow for manipulation in Sec. III-C.

A. Problem Formulation

Given a task instruction ℓ, an initial RGB-D observation ($I_0 \in \mathbb{R}^{H\times W\times 3}$, $D_0 \in \mathbb{R}^{H\times W}$), and a known camera projection Π (intrinsics and extrinsics to the robot frame), our goal is to output an action sequence $u_{0:H-1} \in \mathcal{U}^H$ that accomplishes the task by following an object motion inferred from a generated video. We make no assumption about a specific action parameterization: $\mathcal{U}$ may represent motion primitives, end-effector poses, or low-level controls.

Extracting 3D Object Flow. From (I_0, ℓ), an image-to-video model produces frames $\{V_t\}_{t=1}^T$, and a video-depth estimator provides a per-frame depth sequence $\{Z_t\}_{t=1}^T$. Using a binary mask M of the task-relevant object, and the projection Π, we lift masked image points with $Z_{1:T}$ to obtain an object-centric 3D trajectory $P_{1:T} \in \mathbb{R}^{T\times n\times 3}$ in the robot frame; we refer to $P_{1:T}$ as the 3D object flow.

Action Inference with 3D Object Flow. We represent state as the task-relevant object and the robot: $x_t = (x_t^{\text{obj}}, r_t)$, where $x_t^{\text{obj}} \in \mathbb{R}^{n\times 3}$ are object points and r_t denotes the robot state. Let f be a dynamics model and $\hat{x}_{t+1} = f(\hat{x}_t, u_t)$ with $\hat{x}_0 = x_0$. At each planning step t, we use a time-aligned target $\tilde{P}_t \in \mathbb{R}^{n\times 3}$ derived from the video object flow (e.g., via uniform time-warping or nearest-shape matching). We formulate action inference as an optimization problem:

$$\begin{aligned}\min_{\{u_t\in\mathcal{U}\}} \quad & \sum_{t=0}^{H-1} \lambda_{\text{task}}\big(\hat{x}_t^{\text{obj}}, \tilde{P}_t\big) + \lambda_{\text{control}}(\hat{x}_t, u_t) \\ \text{s.t.} \quad & \hat{x}_{t+1} = f(\hat{x}_t, u_t), \quad \hat{x}_0 = x_0, \\ & \lambda_{\text{task}}\big(\hat{x}_t^{\text{obj}}, \tilde{P}_t\big) = \sum_{i=1}^{n} \big\|\hat{x}_t^{\text{obj}}[i] - \tilde{P}_t[i]\big\|_2^2,\end{aligned}$$

Section III-C instantiates $\mathcal{U}$ and f for different domains.

B. Extracting 3D Object Flows from Videos

Video Generation: Given the task language instruction ℓ and an RGB image of the workspace without the robot visible $I_0 \in \mathbb{R}^{H\times W\times 3}$, Dream2Flow uses an off-the-shelf image-to-video generation model to produce an RGB video $\{V_t\}_{t=1}^T$ with $V_t \in \mathbb{R}^{H\times W\times 3}$, showing the task being performed. We do not include the robot in the initial frame or mention the robot in the text prompt, as we empirically find that current image-to-video generation models not specifically finetuned on robotics data tend to produce physically implausible fine-grained interactions, and consequently have worse object trajectories (for details see Appendix A).

Video Depth Estimation: Given the generated video, Dream2Flow leverages SpatialTrackerV2 [84, 85] to estimate per-frame depth $\{\tilde{Z}_t\}_{t=1}^T$, $\tilde{Z}_t \in \mathbb{R}^{H\times W}$. Due to the scale-shift ambiguity of monocular video, we compute global $(s^\star, b^\star)$ by aligning the first frame to the initial depth D_0 from the robot, and obtain calibrated depths $Z_t = s^\star \tilde{Z}_t + b^\star$.

3D Object Flow Extraction: 3D object flow aims to produce 3D trajectories $P_{1:T} \in \mathbb{R}^{T\times n\times 3}$ with visibilities $V \in \{0,1\}^{T\times n}$ for the task-relevant object. We first localize the relevant object using Grounding DINO [86] to produce a bounding box from (I_0, ℓ), and then use the box to prompt SAM 2 [2] for a binary mask. From the masked region at t=1, we sample n pixels and track them across the video with CoTracker3 [3] to obtain 2D trajectories c_i^t and visibilities v_i^t. Visible points are lifted to 3D using the calibrated depths and camera intrinsics/extrinsics, producing $P_{1:T}$ as in Sec. III-A.

C. Action Inference with 3D Object Flow

Simulated Push-T Domain. For tasks involving non-prehensile manipulation, such as the Push-T task, Dream2Flow uses a push skill primitive parameterized with the start push position on a flat table (c_x, c_y), a unit push direction $(\Delta c_x, \Delta c_y)$, and the distance of the push d. In this setting, we learn a forward dynamics model that takes as input a feature-augmented particles $\tilde{x}_t \in \mathbb{R}^{N\times 14}$ of the whole scene and produces $\Delta\hat{x}_{t+1}$, the delta of positions of each point for the next timestep. The features associated with each point consist of the position, RGB color, and normal vector as determined from camera observations along with the push parameters (as further elaborated in Appendix B).

To optimize the 3D object flow following cost with the learned dynamics model, we use random-shooting, where r push skill parameters are randomly sampled such that all pushes will make contact with the object of interest at different points and directions. Then, we select the push skill parameter out of r ones that has the least cost according to the predicted point positions from the dynamics model. For determining which timestep from the video should be used in the cost function, we find the timestep $t^\star$ where the points of the relevant object in the trajectory are closest to the current observed points with further details in Appendix C.

Real-World Domain. We use absolute end-effector poses as the action space and a rigid-grasp dynamics model for the real robot domain. With this combination, we first proceed to grasp the desired part from the relevant object, and then use the dynamics model and point-flow following objective to move the end-effector such that the grasped part moves

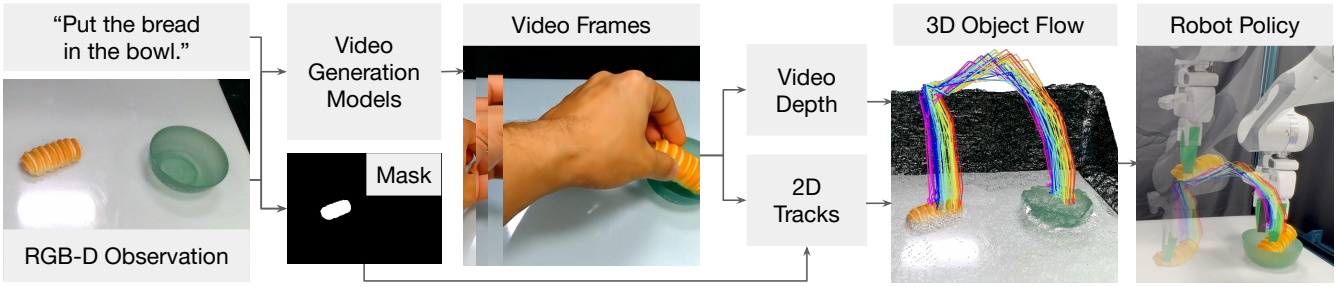

Fig. 2: **An overview of Dream2Flow.** Given a task instruction and an initial RGB-D observation, an image-to-video model synthesizes video frames conditioned on the instruction. We additionally obtain object masks, video depth, and point tracking from vision foundation models, which are used to reconstruct 3D object flow. Finally, a robot policy generates executable actions that track the 3D object flow using trajectory optimization or reinforcement learning.

in a fashion similar to the video. We use AnyGrasp [87] to propose candidate grasps on the object of interest, but these grasps may not lie exactly on the desired part. We select the grasp closest to the thumb as detected from the video by HaMer [88], as we observe that in generated videos, the hand tends to interact with the relevant part of an object, such as the handle.

The rigid-grasp dynamics model assumes that the grasped part is rigid; consequently, the prediction of point positions to the next timestep can be described by a sequence of rigid transformations for the grasped subset, while non-grasped points remain unchanged. To produce a trajectory of end-effector poses, Dream2Flow directly optimizes the objective using PyRoki [89], incorporating pose smoothness and reachability costs as λ_{control} in addition to the 3D object-flow following cost λ_{task} as in Appendix F.

Simulated Door Opening Domain. For the Door Opening task, we use reinforcement learning to learn a sensimotor policy which moves the object according to the 3D object flow with SAC [90]. This approach can be viewed as using the simulator as a dynamics model for compiling the optimization process prescribed in Eq. III-A into a parametric policy in an offline fashion, using 3D object flow as the reward function. Depending on the embodiment used, the action space consists of delta end-effector poses and delta joint angles for the gripper or dexterous hand. The reward function used consists of one term which encourages the end-effector to move toward the current mean object particle position along with another term for encouraging matching the 3D object flow, $\frac{t^\star}{t_{\text{end}}}$, where $t^\star$ is the timestep with the closest particle positions to the 3D object flow and t_{end} is the last timestep of the object motion (further details in Appendix H). By training policies with such an object-centric reward, we observe different strategies emerge to accomplish the same object motion across embodiments, including quadruped manipulators, humanoids with dexterous hands, and fixed-base arms with parallel grippers.

IV. Experiments

We seek to answer the following research questions through our experiments: $\mathcal{Q}\mathbf{1}$: What are properties of 3D object flow when used as an interface to bridge videos and robot control? $\mathcal{Q}\mathbf{2}$: How does Dream2Flow perform compared to alternative interfaces? $\mathcal{Q}\mathbf{3}$: How effective is 3D object flow as a reward for learning sensimotor policies? $\mathcal{Q}\mathbf{4}$: How does the choice of video model affect Dream2Flow in simulation and in real-world tasks? $\mathcal{Q}\mathbf{5}$: How does the choice of dynamics model affect the performance of Dream2Flow?

A. Tasks

For evaluation, we consider several tasks involving different types of objects and manipulation strategies (Fig. 3).

1) Push-T: We develop a simulated Push-T task in OmniGibson [91], where a T-shaped block is placed with a random position and yaw angle on the wooden platform. It is a success if the T-block ends up within 2cm translation and 15 degrees rotation from the goal position, where the T-shaped block is in the center of the board facing forwards.

2) Put Bread in Bowl: The Put Bread in Bowl task consists of a fake piece of bread and a green bowl placed randomly on the workspace. A trial is a success if the bread is inside the bowl at the end.

3) Open Oven: In the Open Oven task, the toaster oven is randomly placed in a semi-circular arc with the orientation towards the base of the robot. A trial is considered a success if the opening angle is at least 60 degrees.

4) Cover Bowl: The Cover Bowl task starts with a folded scarf placed at a random position and orientation on the workspace, with a blue bowl placed next to it. The trial is a success if the scarf is covering at least 25% of the top of the bowl after the robot finishes execution.

5) Open Door: In the Open Door task from Robosuite [92], a door is placed at random positions and orientations on top of a table. Rotating the handle and pulling the door open by at least 17° without timing out is a success.

B. Properties of 3D Object Flow as a Video-Control Interface

We run Dream2Flow on the simulated Push-T task with Wan2.1 [93] as the video generation model. For this task only, we allow prompting with a goal image of the T-block in the center, similar to the example final state in Fig. 3, as this task requires accurate movement. We consider 10 different initial states with the same final state. We then proceed to run 10 trials for each initial configuration on different seeds

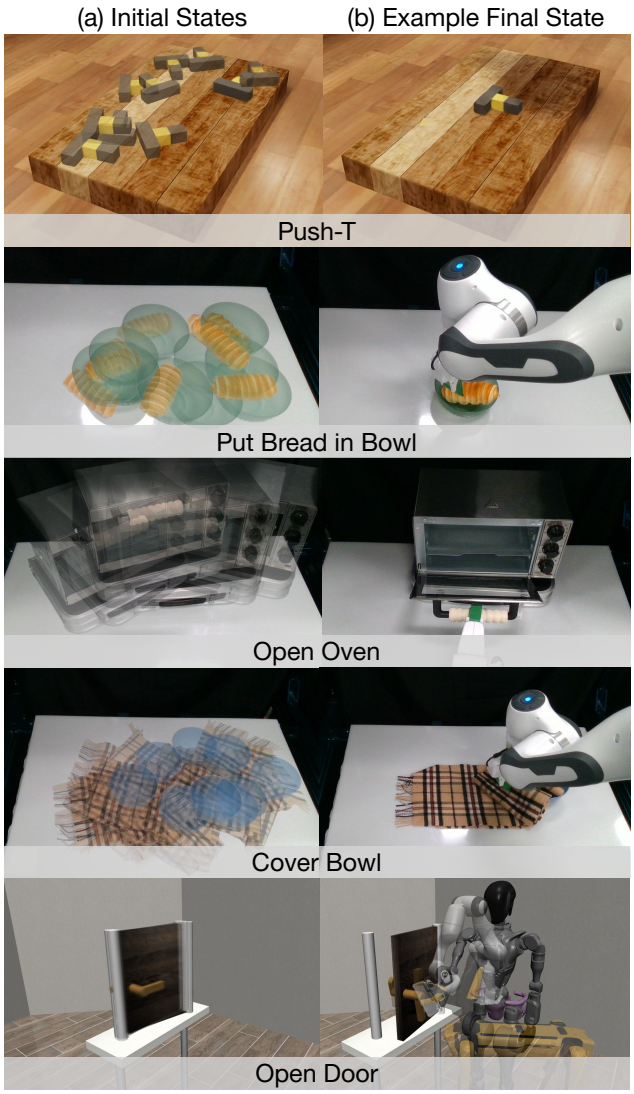

Fig. 3: **Evaluation Tasks.** (a) The initial states of each task used in the evaluation trials. (b) For one of the initial states, the corresponding final state after the robot performs the task. For Push-T only, the desired final state is the same for all initial states.

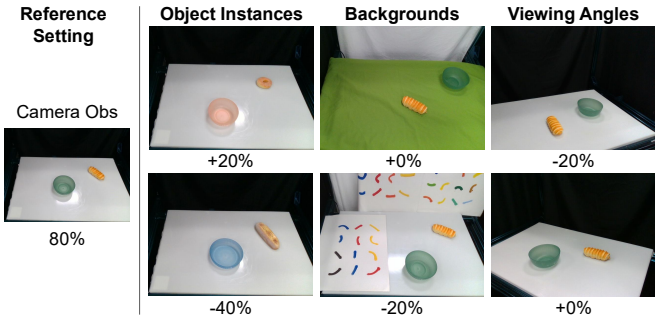

Fig. 4: **Robustness evaluations.** Relative performance across instance, background, and task variations, showing Dream2Flow remains robust under various different settings.

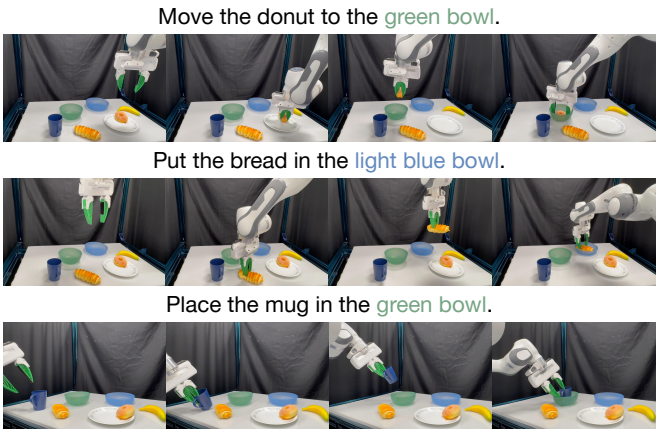

Fig. 5: **Multiple tasks in the same scene.** With different language goals, Dream2Flow adapts object-flow targets to produce distinct behaviors in the same environment.

to properly evaluate performance due to the stochastic nature of random shooting, yielding 100 total trials. The result appears in Table IV. We note that in this task, 6 generated videos included substantial morphing of the T-block, which consequently ruined the tracking and downstream execution.

We evaluate Dream2Flow's ability to perform manipulation with different types of objects with Veo 3 [94] as the video generation model in the real world for 10 trials per task, and report the results in Table I.

To assess Dream2Flow's generalization to different object instances, backgrounds, and viewing angles, we conduct an additional five trials each for six different scenarios, as shown in Fig. 4. Compared to the reference setting, with exception of putting a large piece of bread, there is not a significant drop off in performance, suggesting that Dream2Flow inherits some generalization through the use of video generation models. We additionally show that Dream2Flow can perform different tasks from the same scene in Fig. 5 due to the ability of video generation models to follow different task instructions given the same input image. We have additional case studies demonstrating that 3D object flow can be used for downstream manipulation in in-the-wild settings for tasks such as pulling a chair, opening a drawer, sweeping pasta, and recycling a can, as shown in Fig. 1 and Appendix G.

For all 60 trials of Dream2Flow executed in the real world, we provide a breakdown of failures in Fig. 7. In the 12 video generation failures, for half the time, the generated video either morphs an object in an implausible way or hallucinates new objects, causing tracking to unreasonably fail or making the robot move an object to an incorrect 3D location. The four flow extraction failures occurred because of severe rotations or objects temporarily going out of view of the camera, leading to tracks with no visibility in the end. The four robot execution failures occurred in the Bowl Covering task, where the robot either did not grasp at the correct point or did not move enough.

Task	AVDC	RIGVID	Dream2Flow
Bread in Bowl	7/10	6/10	**8/10**
Open Oven	0/10	6/10	**8/10**
Cover Bowl	2/10	1/10	**3/10**

TABLE I: **Comparisons of intermediate representations on real robot.** Dream2Flow outperforms AVDC and RIGVID across three tasks by following 3D object flow rather than rigid transforms alone.

C. How does Dream2Flow perform compared to alternative interfaces?

For the three tasks in the real world, we consider the following alternative interfaces related to extracting object trajectories from videos:

1) AVDC: AVDC [42] leverages generated videos by computing dense optical flow between frames to track points on a rigid object. Then, using the initial depth and the point correspondences, it solves for a sequence of rigid transforms of the relevant object, allowing for trajectory playback after a grasp has occurred. In our implementation, we utilize the video depth corresponding to the tracks from the optical flow, and optimize for rigid transforms relative to the initial frame, as we empirically found that to be less noisy than the original optimization procedure.

2) RIGVID: RIGVID [58] uses 6D object pose tracking to generate a rigid object trajectory from a generated video. Since it is ill-defined to have such a pose for deformable objects, we do not use a 6D pose tracker, but instead adapt their approach by solving for a rigid pose transformation between the initial 3D points and visible 3D points from our 3D Object Flow computation in the same manner as AVDC.

We present the results in Table I. While AVDC does a reasonable job at tracking the bread, the dense optical flow does not keep up with the motion of the oven, resulting in insufficient motion. Under certain circumstances, there are only a few visible points for RIGVID and AVDC, making transform estimation noisy. Attempting to follow the noisy transform trajectories leads to execution failures or optimization instability. Dream2Flow is less affected by this issue, because there is typically not a heavy cost when most of the points are occluded, allowing the planned end-effector poses to smoothly move between areas of high point visibility. The Cover Bowl task remains a challenge for AVDC and RIGVID, as in addition to video generation and tracking failures, the transform estimates are incorrect since the points are now on a deformable object.

D. How effective is 3D object flow as a reward for learning sensimotor policies?

The 3D object flow extracted by Dream2Flow can be used in an RL reward for training policies across different embodiments. We evaluate SAC [90] policies trained using handcrafted object state reward from Robosuite and 3D object flow rewards on a Franka Panda, a Spot with floating base, and a GR1 with the right arm only across 100 random door positions, and report the results in Table II with corresponding episode visualizations in Figure 6. The policies trained with the object state and 3D object flow reward have comparable performance across all the embodiments. The learned strategies are different between the embodiments, as the Spot is able to move its base for better reachability and kinematic range, and the GR1 uses the area between the fingers and palm for pulling the door as opposed to individual fingers for better stability.

Reward Type	Franka	Spot	GR1
Object State	99/100	99/100	**96/100**
3D Object Flow	**100/100**	**100/100**	94/100

TABLE II: **Comparison of policies trained using different rewards.** The policies trained using the 3D object flow reward perform comparably to those trained with the object state reward across different embodiments.

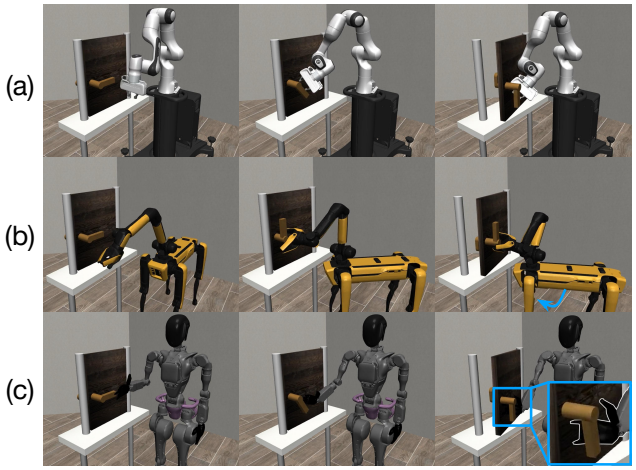

Fig. 6: **Rollouts from policies trained using 3D object flow as a reward.** Different embodiments such as the (a) Panda, (b) Spot, or (c) GR1 use different strategies to open the door. The Spot is able move its base for better reachability while the GR1 uses the area between its fingers and palm to pull for better stability.

E. How does the choice of video model affect Dream2Flow in simulation and in real-world tasks?

Video Generation Model	Push-T	Open Oven
Wan2.1 [93]	**52/100**	2/10
Kling 2.1	31/100	4/10
Veo 3 [94]	-	**8/10**

TABLE III: **Effect of video generator.** Veo 3 excels on real-world domains such as "Open Oven", while Wan 2.1 performs better on simulated domains such as "Push-T".

To evaluate how well different video generation models capture physically plausible object trajectories, we run Dream2Flow on the simulated Push-T task and the real Open Oven task using three models: Wan2.1 [93], Kling 2.1, and Veo 3 [94]. The results are shown in Table III. Note that

Dynamics Model Type	Success Rate
Pose	12/100
Heuristic	17/100
Particle	**52/100**

TABLE IV: **Dynamics model ablation.** Particle dynamics substantially outperform pose and heuristic models, highlighting the importance of per-point predictions.

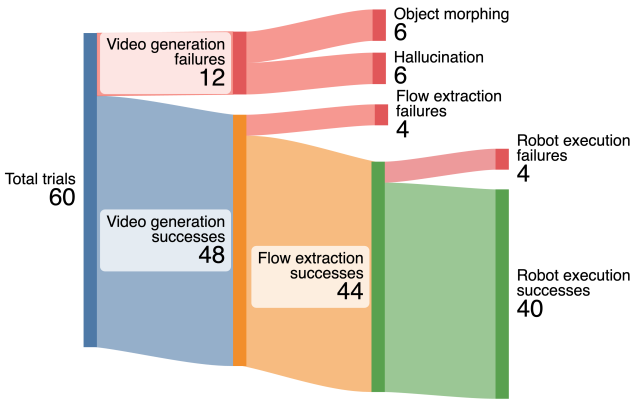

Fig. 7: **Failure breakdown on real-robot experiments.** Common causes include video artifacts (object morphing, object hallucination), tracking errors, and grasp selection mismatches.

there are no results for Veo 3 on Push-T because at the time of evaluation, Veo 3 did not support prompting with a goal image.

For the Push-T task, Kling 2.1 had more videos that had substantial morphing, throwing off the tracking, resulting in more downstream failures. For the Open Oven task, Wan2.1 tends to produce more videos with substantial camera motion, which violates the still camera assumption. Additionally, both Kling 2.1 and Wan2.1 produce videos where the direction of articulation is incorrect, such as revolving around the wrong axis, leading to far more failures than Veo 3.

F. How does the choice of dynamics model affect the performance of Dream2Flow?

For the Push-T task, we compare the effect of different dynamics models used for planning. In addition to the particle-based dynamics model, we consider another learned dynamics model which takes in the block pose and push skill parameters and predicts the delta pose of the T-block, trained on the same data as the particle based dynamics model, as well as a heuristic dynamics model, which translates the points of the T-block in the direction and amount of the push without any rotation. We present the results in Table IV, and find that the particle representation for this dynamics model is crucial to ensure success, as despite having the same 3D object flow guidance, the pose and heuristic based dynamics models could not sufficiently account for the rotation needed.

V. Conclusion

We presented Dream2Flow, a simple, general interface that turns text-conditioned video predictions into executable robotic actions by reconstructing and tracking 3D object flow. This decouples what should happen in the world (task-relevant object motion and state change) from how a particular embodiment realizes it under kinematic, dynamic, and morphology constraints. Built entirely from off-the-shelf video generation and perception tools, Dream2Flow solves open-world manipulation tasks in simulation and on real robots across rigid, articulated, deformable, and granular objects using only RGB-D observations and language, without task-specific demonstrations. Experiments show consistent gains over trajectory baselines derived from dense optical flow or rigid pose transforms, robustness to variations in instances, backgrounds, and viewpoints, and the importance of both the upstream video model and downstream dynamics choice (with particle dynamics proving most reliable). A failure analysis highlights current bottlenecks, such as video artifacts (morphing, hallucinations), occlusion-induced tracking dropouts, and grasp selection mismatches as concrete directions for improvement. Overall, our results indicate that 3D object flow is a scalable bridge from open-ended video generation to robot control in unstructured environments.

Acknowledgment

This work is in part supported by the Stanford Institute for Human-Centered AI (HAI), the Schmidt Futures Senior Fellows grant, ONR MURI N00014-21-1-2801, ONR MURI N00014-22-1-2740, ONR MURI N00014-24-1-2748, and NSF RI #2338203.

References

[1] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman *et al.*, "Video generation models as world simulators," *OpenAI Blog*, vol. 1, no. 8, p. 1, 2024.

[2] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: https://arxiv.org/abs/2408.00714

[3] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht, "CoTracker3: Simpler and better point tracking by pseudo-labelling real videos," *arxiv*, 2024.

[4] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual review of control, robotics, and autonomous systems*, vol. 4, no. 1, pp. 265–293, 2021.

[5] Z. Zhao, S. Cheng, Y. Ding, Z. Zhou, S. Zhang, D. Xu, and Y. Zhao, "A survey of optimization-based task and motion planning: From classical to learning approaches," *IEEE/ASME Transactions on Mechatronics*, 2024.

[6] M. Toussaint, "Logic-geometric programming: An optimization-based approach to combined task and motion planning." in *IJCAI*, 2015, pp. 1930–1936.

[7] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on robot learning*. PMLR, 2022, pp. 894–906.

[8] ——, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[9] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "pi0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[11] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong *et al.*, "Openvla: An open-source vision-language-action model," in *Conference on Robot Learning*. PMLR, 2025, pp. 2679–2713.

[12] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Conference on robot learning*. Pmlr, 2020, pp. 1113–1132.

[13] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv preprint arXiv:1812.00568*, 2018.

[14] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine, "Zero-shot robotic manipulation with pretrained image-editing diffusion models," *arXiv preprint arXiv:2310.10639*, 2023.

[15] A. Xie, A. Singh, S. Levine, and C. Finn, "Few-shot goal inference for visuomotor learning and planning," in *Conference on Robot Learning*. PMLR, 2018, pp. 40–52.

[16] P. Sharma, D. Pathak, and A. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak, "Discovering and achieving goals via world models," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 379–24 391, 2021.

[18] A. Adeniji, Z. Chen, V. Liu, V. Pattabiraman, R. Bhirangi, S. Haldar, P. Abbeel, and L. Pinto, "Feel the force: Contact-driven learning from humans," *arXiv preprint arXiv:2506.01944*, 2025.

[19] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann, "Neural descriptor fields: Se (3)-equivariant object representations for manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6394–6400.

[20] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[21] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *Proceedings of Machine Learning Research*, vol. 229, 2023.

[22] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2025, pp. 4573–4602.

[23] Y. Tang, W. Huang, Y. Wang, C. Li, R. Yuan, R. Zhang, J. Wu, and L. Fei-Fei, "Uad: Unsupervised affordance distillation for generalization in robotic manipulation," *arXiv preprint arXiv:2506.09284*, 2025.

[24] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song, "Flow as the cross-domain manipulation interface," in *Conference on Robot Learning*. PMLR, 2025, pp. 2475–2499.

[25] C. Yuan, C. Wen, T. Zhang, and Y. Gao, "General flow as foundation affordance for scalable robot learning," in *Conference on Robot Learning*. PMLR, 2025, pp. 1541–1566.

[26] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held, "Fabricflownet: Bimanual cloth manipulation with a flow-based policy," *arXiv preprint arXiv:2111.05623*, 2021.

[27] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," *arXiv preprint arXiv:2202.00732*, 2022.

[28] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held, "Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds," *arXiv preprint arXiv:2211.09006*, 2022.

[29] T. Chen, Y. Mu, Z. Liang, Z. Chen, S. Peng, Q. Chen, M. Xu, R. Hu, H. Zhang, X. Li, and P. Luo, "G3flow: Generative 3d semantic flow for pose-aware and generalizable object manipulation," *arXiv preprint arXiv:2411.18369*, 2024.

[30] S. Wang, J. You, Y. Hu, J. Li, and Y. Gao, "Skil: Semantic keypoint imitation learning for generalizable data-efficient manipulation," in *Robotics: Science and Systems (RSS)*, 2025.

[31] S. Haldar and L. Pinto, "Point policy: Unifying observations and actions with key points for robot manipulation," *arXiv preprint arXiv:2502.20391*, 2025.

[32] S. Guo, X. Liang, J. Lin, Y. Zhuang, L. Lin, and X. Liang, "Actionsink: Toward precise robot manipulation with dynamic integration of action flow," *arXiv preprint arXiv:2508.03218*, 2025.

[33] Y. Yang, Z. Cai, Y. Tian, J. Zeng, and J. Pang, "Gripper keypose and object pointflow as interfaces for bimanual robotic manipulation," *arXiv preprint arXiv:2504.17784*, 2025.

[34] B. Eisner, H. Zhang, and D. Held, "Flowbot3d: Learning 3d articulation flow to manipulate articulated objects," in *Robotics: Science and Systems (RSS)*, 2022.

[35] H. Zhang, B. Eisner, and D. Held, "Flowbot++: Learning generalized articulated objects manipulation via articulation projection," 2024. [Online]. Available: https://arxiv.org/abs/2306.12893

[36] C. Gao, H. Zhang, Z. Xu, Z. Cai, and L. Shao, "Flip: Flow-centric generative planning as general-purpose manipulation world model," *arXiv preprint arXiv:2412.08261*, 2024.

[37] J. Guo, X. Ma, Y. Wang, M. Yang, H. Liu, and Q. Li, "Flowdreamer: A rgb-d world model with flow-based motion representations for robot manipulation," *arXiv preprint arXiv:2505.10075*, 2025.

[38] H. Zhi, P. Chen, S. Zhou, Y. Dong, Q. Wu, L. Han, and M. Tan, "3dflowaction: Learning cross-embodiment manipulation from 3d flow world model," *arXiv preprint arXiv:2506.06199*, 2025.

[39] Y. He and Q. Nie, "Manitrend: Bridging future generation and action prediction with 3d flow for robotic manipulation," *arXiv preprint arXiv:2502.10028*, 2025.

[40] Z.-H. Yin, S. Yang, and P. Abbeel, "Object-centric 3d motion field for robot learning from human videos," *arXiv preprint arXiv:2506.04227*, 2025.

[41] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani, "Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation," in *European Conference on Computer Vision*. Springer, 2024, pp. 306–324.

[42] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum, "Learning to act from actionless videos through dense correspondences," *arXiv preprint arXiv:2310.08576*, 2023.

[43] Y. Chen, P. Li, Y. Huang, J. Yang, K. Chen, and L. Wang, "Ec-flow: Enabling versatile robotic manipulation from action-unlabeled videos via embodiment-centric flow," *arXiv preprint arXiv:2507.06224*, 2025.

[44] L. Shao, P. Shah, V. Dwaracherla, and J. Bohg, "Motion-based object segmentation based on dense rgb-d scene flow," *arXiv preprint arXiv:1804.05195*, 2018.

[45] X. Liu, C. R. Qi, and L. J. Guibas, "Flownet3d: Learning scene flow in 3d point clouds," *arXiv preprint arXiv:1806.01411*, 2018.

[46] "Zero-shot monocular scene flow estimation in the wild," CVPR 2025, CVF Open Access, 2025.

[47] "Zeromsf: Zero-shot monocular scene flow in the wild," NVIDIA Research Project Page, 2025.

[48] J. Sander, G. Caroleo, A. Albini, and P. Maiolino, "Estimating scene flow in robot surroundings with distributed miniaturized time-of-flight sensors," *arXiv preprint arXiv:2504.02439*, 2025.

[49] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks," *arXiv preprint arXiv:1606.02378*, 2016.

[50] T. Tang, J. Liu, J. Zhang, H. Fu, W. Xu, and C. Lu, "Rftrans: Leveraging refractive flow of transparent objects for surface normal estimation and manipulation," *arXiv preprint arXiv:2311.12398*, 2023.

[51] B. P. Duisterhof, Z. Mandi, Y. Yao, J.-W. Liu, J. Seidenschwarz, M. Z. Shou, D. Ramanan, S. Song, S. Birchfield, B. Wen, and J. Ichnowski, "Deformgs: Scene flow in highly deformable scenes for deformable object manipulation," *arXiv preprint arXiv:2312.00583*, 2024.

[52] J. Kerr, C. M. Kim, M. Wu, B. Yi, Q. Wang, K. Goldberg, and A. Kanazawa, "Robot see robot do: Imitating articulated object manipulation with monocular 4d reconstruction," *arXiv preprint arXiv:2409.18121*, 2024.

[53] H. Jiang, B. Huang, R. Wu, Z. Li, S. Garg, H. Nayyeri, S. Wang, and Y. Li, "Roboexp: Action-conditioned scene graph via interactive exploration for robotic manipulation," *arXiv preprint arXiv:2402.15487*, 2024.

[54] O. Shorinwa, J. Tucker, A. Smith, A. Swann, T. Chen, R. Firoozi, M. Kennedy III, and M. Schwager, "Splat-mover: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting," *arXiv preprint arXiv:2405.04378*, 2024.

[55] I. Guzey, Y. Dai, G. Savva, R. Bhirangi, and L. Pinto, "Bridging the human to robot dexterity gap through object-oriented rewards," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 3344–3351.

[56] K. Yu, S. Zhang, H. Soora, F. Huang, H. Huang, P. Tokekar, and R. Gao, "Genflowrl: Shaping rewards with generative object-centric flow in visual reinforcement learning," *arXiv preprint arXiv:2508.11049*, 2025.

[57] H. Bharadhwaj, D. Dwibedi, A. Gupta, S. Tulsiani, C. Doersch, T. Xiao, D. Shah, F. Xia *et al.*, "Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation," *arXiv preprint arXiv:2409.16283*, 2024.

[58] S. Patel, S. Mohan, H. Mai, U. Jain, S. Lazebnik, and Y. Li, "Robotic manipulation by imitating generated videos without physical demonstrations," *arXiv preprint arXiv:2507.00990*, 2025.

[59] S. Lee, Y. Jung, I. Chun, Y.-C. Lee, Z. Cai, H. Huang, A. Talreja, T. D. Dao, Y. Liang, J.-B. Huang, and F. Huang, "Tracegen: World modeling in 3d trace space enables learning from cross-embodiment videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[60] H. Li, L. Sun, Y. Hu, D. Ta, J. Barry, G. Konidaris, and J. Fu, "Novaflow: Zero-shot manipulation via actionable flow from generated videos," *arXiv preprint arXiv:2510.08568*, 2025.

[61] S. Yang, J. Walker, J. Parker-Holder, Y. Du, J. Bruce, A. Barreto, P. Abbeel, and D. Schuurmans, "Video as the new language for real-world decision making," *arXiv preprint arXiv:2402.17139*, 2024.

[62] R. McCarthy, D. C. Tan, D. Schmidt, F. Acero, N. Herr, Y. Du, T. G. Thuruthel, and Z. Li, "Towards generalist robot learning from internet video: A survey," *Journal of Artificial Intelligence Research*, vol. 83, 2025.

[63] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong, "Unleashing large-scale video generative pre-training for visual robot manipulation," *arXiv preprint arXiv:2312.13139*, 2023.

[64] Y. Seo, K. Lee, S. L. James, and P. Abbeel, "Reinforcement learning with action-free pre-training from videos," in *International Conference on Machine Learning*. PMLR, 2022, pp. 19 561–19 579.

[65] J. Wu, H. Ma, C. Deng, and M. Long, "Pre-training contextualized world models with in-the-wild videos for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 39 719–39 743, 2023.

[66] J. Yang, B. Liu, J. Fu, B. Pan, G. Wu, and L. Wang, "Spatiotemporal predictive pre-training for robotic motor control," *arXiv preprint arXiv:2403.05304*, 2024.

[67] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen, "Video prediction policy: A generalist robot policy with predictive visual representations," *arXiv preprint arXiv:2412.14803*, 2024.

[68] S. Li, Y. Gao, D. Sadigh, and S. Song, "Unified video action model," *arXiv preprint arXiv:2503.00200*, 2025.

[69] T. Huang, G. Jiang, Y. Ze, and H. Xu, "Diffusion reward: Learning rewards via conditional video diffusion," in *European Conference on Computer Vision*. Springer, 2024, pp. 478–495.

[70] A. Escontrela, A. Adeniji, W. Yan, A. Jain, X. B. Peng, K. Goldberg, Y. Lee, D. Hafner, and P. Abbeel, "Video prediction models as rewards for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 68 760–68 783, 2023.

[71] A. S. Chen, S. Nair, and C. Finn, "Learning generalizable robotic reward functions from" in-the-wild" human videos," *arXiv preprint arXiv:2103.16817*, 2021.

[72] A. Ajay, S. Han, Y. Du, S. Li, A. Gupta, T. Jaakkola, J. Tenenbaum, L. Kaelbling, A. Srivastava, and P. Agrawal, "Compositional foundation models for hierarchical planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 22 304–22 325, 2023.

[73] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel, "Learning universal policies via text-guided video generation," *Advances in neural information processing systems*, vol. 36, pp. 9156–9172, 2023.

[74] D. Valevski, Y. Leviathan, M. Arar, and S. Fruchter, "Diffusion models are real-time game engines," *arXiv preprint arXiv:2408.14837*, 2024.

[75] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps *et al.*, "Genie: Generative interactive environments," in *Forty-first International Conference on Machine Learning*, 2024.

[76] Y. Luo and Y. Du, "Grounding video models to actions through goal conditioned exploration," *arXiv preprint arXiv:2411.07223*, 2024.

[77] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," *arXiv preprint arXiv:2310.06114*, vol. 1, no. 2, p. 6, 2023.

[78] S. Zhou, Y. Du, J. Chen, Y. Li, D.-Y. Yeung, and C. Gan, "Robodreamer: Learning compositional world models for robot imagination," *arXiv preprint arXiv:2404.12377*, 2024.

[79] O. Rybkin, K. Pertsch, K. G. Derpanis, K. Daniilidis, and A. Jaegle, "Learning what you can do before doing anything," *arXiv preprint arXiv:1806.09655*, 2018.

[80] R. Mendonca, S. Bahl, and D. Pathak, "Structured world models from human videos," *arXiv preprint arXiv:2308.10901*, 2023.

[81] H. Che, X. He, Q. Liu, C. Jin, and H. Chen, "Gamegen-x: Interactive open-world game video generation," *arXiv preprint arXiv:2411.00769*, 2024.

[82] J. Jang, S. Ye, Z. Lin, J. Xiang, J. Bjorck, Y. Fang, F. Hu, S. Huang, K. Kundalia, Y.-C. Lin *et al.*, "Dreamgen: Unlocking generalization in robot learning through neural trajectories," *arXiv e-prints*, pp. arXiv–2505, 2025.

[83] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick, "Dreamitate: Real-world visuomotor policy learning via video generation," *arXiv preprint arXiv:2406.16862*, 2024.

[84] Y. Xiao, J. Wang, N. Xue, N. Karaev, I. Makarov, B. Kang, X. Zhu, H. Bao, Y. Shen, and X. Zhou, "Spatialtrackerv2: 3d point tracking made easy," in *ICCV*, 2025.

[85] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, "Depth anything v2," *Advances in Neural Information Processing Systems*, vol. 37, pp. 21 875–21 911, 2024.

[86] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[87] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics (T-RO)*, 2023.

[88] G. Pavlakos, D. Shan, I. Radosavovic, A. Kanazawa, D. Fouhey, and J. Malik, "Reconstructing hands in 3D with transformers," in *CVPR*, 2024.

[89] C. M. Kim*, B. Yi*, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, "Pyroki: A modular toolkit for robot kinematic optimization," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025.

[90] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870.

[91] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, and et al., "Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation," *arXiv preprint arXiv:2403.09227*, 2024.

[92] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[93] T. Wan, A. Wang, B. Ai, B. Wen, C. Mao, and et al., "Wan: Open and advanced large-scale video generative models," *arXiv preprint arXiv:2503.20314*, 2025.

[94] Google DeepMind, "Veo-3 Technical Report," 2025. [Online]. Available: https://storage.googleapis.com/deepmind-media/veo/Veo-3-Tech-Report.pdf

[95] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler, faster, stronger," in *CVPR*, 2024.

[96] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[97] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *arXiv preprint arXiv:2210.11339*, 2022.

APPENDIX

A. Video Generation Prompts

For all real world tasks, the prompt to each video generation model consists of the initial RGB observation from one camera, as well as a language instruction of the form:

Real World Task Language Prompt

<TASK> by one hand. The camera holds a still pose, not zooming in or out.

Values of `<TASK>`:

- *Put Bread in Bowl:* The bread is grabbed and placed into the green bowl
- *Open Oven:* The toaster oven is opened
- *Cover Bowl:* The scarf is lifted by a corner and directly dragged over the blue bowl in one smooth motion without flinging the cloth or any dynamic / fast motions
- *Pull Out Chair:* The chair is pulled out straight from under the table to the right by grabbing the middle
- *Open Drawer:* The partially opened drawer is opened all the way out
- *Sweep Pasta:* The brush with a green handle moves left to right to push the pasta into the compost bin
- *Recycle Can:* The can is grabbed and dropped into the recycling bin

For the robustness evaluations, the word "bread" is replaced with "donut" or "long piece of bread" for the different object instances, but otherwise the prompt remains the same.

Since Dream2Flow leverages the position of a hand in the video to help select the grasp on the object, the "by one hand" phrase must be included. Additionally, "the camera holds a still pose" must be included to increase the probability that the generated videos do not have substantial camera motion, as the depth estimation pipeline assumes a still camera.

Push-T Task Language Prompt

The T-shaped block slides and rotates smoothly to the center of the wooden platform.

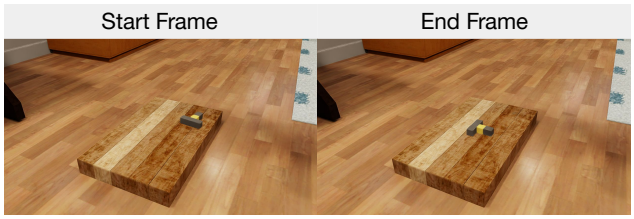

Fig. 8: **Example Push-T image prompt.** To generate sufficiently accurate motions for the Push-T task, the visual component of the prompt includes both a start and end frame.

For Push-T, since the task success requires accurate positions and to provide a better hint as to where the T block should go, we also provide a goal image of the T-block, as shown in Figure 8. At the time of evaluation, Veo 3 did not have the ability to take in an end frame, and hence was not considered for Push-T experiments.

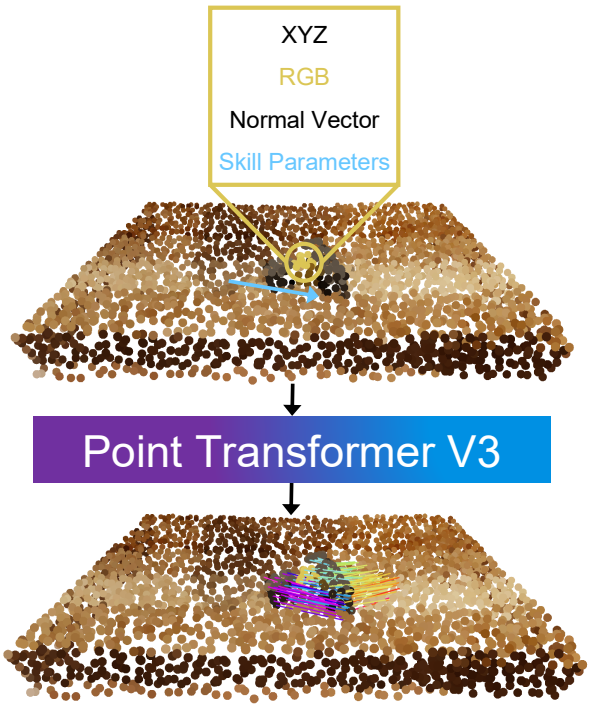

Fig. 9: **Particle dynamics model.** The particle dynamics model primarily composed of a Point Transformer V3 backbone used in the Push-T task takes as input a set of feature-augmented particles and outputs delta position predictions for all particles in the scene.

Open Door Task Language Prompt

The door opens all the way by itself to the right. The camera holds a still pose, not zooming in or out.

The Open Door task prompt does not include a hand to perform the action as it is in a simulated environment.

All prompts discussed in this section are identical for each video generation model evaluated. For Kling 2.1, a relevance value of 0.7 and a negative prompt of "fast motion, morphing, camera motion" are added.

B. Particle Dynamics Model

The particle dynamics model used in the Push-T task takes as input feature-augmented particles $\tilde{x}_t \in \mathbb{R}^{N\times 14}$, consisting of the position, RGB value, normal vector, and push parameters and produces $\Delta\hat{x}_{t+1}$, the delta of positions of each point for the next timestep, as shown in Figure 9. The architecture of the particle dynamics model is a small Point Transformer V3 [95] backbone surrounded by MLPs to project between the desired input and output sizes.

To get the input set of particles from the scene, there are 4 virtual cameras around the workspace, whose RGB-D observations are combined into a single point cloud. Then, points outside of the bounds of the wooden platform that the T-shaped block is on are discarded. Finally, this point

cloud is voxel downsampled by randomly keeping only 1 particle per each 1.5cm sided cube. The same push parameter is appended to each particle's features.

To train the particle dynamics model, we collect 500 transitions of random pushing actions. In this setting, we track particle positions before and after the push by using all objects' poses from the simulator for efficiency, but in practice this can also be tracked with CoTrackerV3 [3] and depth.

C. Push-T Planning Details

To optimize the particle trajectory following cost with the learned particle based dynamics model, Dream2Flow replans after every single push with random shooting until the T-block is within the specified tolerance to the goal or a maximum number of pushes have been executed. In this setting, at each replanning step, r push skill parameters are randomly sampled such that all pushes will make contact with the object of interest at different points and directions. Then, Dream2Flow selects the push skill parameter out of those r ones that has the least cost according to the predicted particle positions from the dynamics model with respect to a subgoal of particle positions (may not necessarily be the final goal position). While the T-block is being pushed, Dream2Flow uses the online version of CoTrackerV3 [3] to track its motion, and once the push is completed, the tracked points are lifted into 3D, forming the updated particles of the T-block. We find that while parts of the T-block may be occluded during a pushing action, CoTrackerV3 [3] tends to recover visibility of previously occluded points when the gripper lifts up.

Since Dream2Flow tracks particles for the T-block while the particle dynamics model takes in downsampled particles of the entire scene, Dream2Flow performs nearest neighbor matching to find correspondences between the currently tracked particles and the feature-augmented particles which are the input to the dynamics model. With these correspondences, and the predicted delta positions of these corresponding particles, Dream2Flow computes the predicted particle positions of the T-block as $\hat{x}_{t+1} = \hat{x}_t + \Delta\hat{x}_{t+1}$.

For determining which timestep from the video should be used in the cost function as a subgoal, Dream2Flow first finds the timestep $t^\star$ where the tracked particles are closest to the particles in the 3D object flow, as a single push can encompass the motion of many timesteps. Dream2Flow then chooses the particles from the timestep $\min(t^\star + L, t_{\text{end}})$ as the next subgoal for planning, where L is a fixed look ahead time amount, and t_{end} is the final timestep of the video. In our experiments, we used $L = 20$. We choose to use intermediate subgoals to be the target for random shooting as opposed to only the final particle positions of the T-block, as for cases involving substantial rotation, only having the final positions can result in the T-block have small translation error but large rotation error, eventually lead to timeouts.

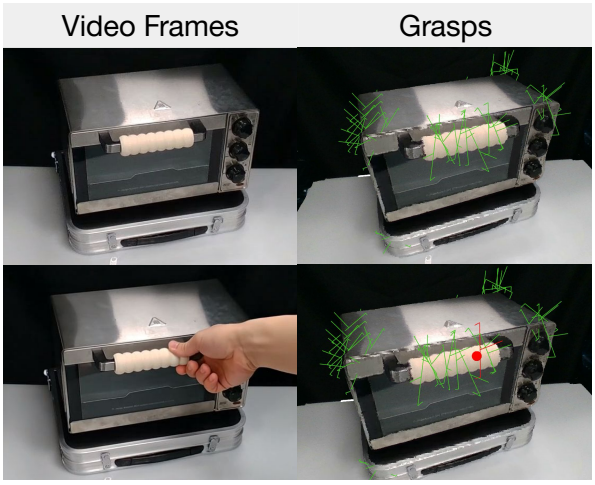

Fig. 10: **Grasp selection with thumb position.** Dream2Flow selects a grasp closest to the position of the thumb in the video, if such a grasp exists within 2cm. The red sphere indicates the thumb position predicted by HaMer, with the red grasp being the selected one.

D. Grasp Selection

Dream2Flow uses AnyGrasp [87] to propose a set of up to 40 top-down grasps after applying the mask to the object of interest. Up to 20 grasps come from a point cloud in the coordinate frame of the camera, and another 20 grasps come from transforming those points into the coordinate frame of a virtual camera looking straight down in the center of the workspace. We added this addition virtual camera so that some more vertical grasps could be proposed.

While for rigid objects consisting of one part, such as a piece of bread, it is typically fine to grasp at any stable position, for articulated objects, the part that moves needs to be grasped instead. To perform grasp selection in such cases, we exploit video generation models' ability to synthesize plausible hand-object interactions, typically where the hand first grabs the relevant part of the object and then proceeds to move it. Dream2Flow uses HaMer [88] to detect the position of the hand, and in particular the thumb. If the predicted thumb position comes within 2cm of a proposed grasp, then such a grasp at the earliest timestep is chosen. An example of grasp selection with this method is shown in Figure 10.

It is possible that there are no such grasps detected, either because the grasp planner proposes grasps that are not near the hand in the video or the detections from HaMer [88] are not accurate enough. In such cases, Dream2Flow defaults to a heuristic of selecting the closest grasp to the points on the movable part of the rigid object (obtaining points on the movable part is described in the next section).

E. Movable Part Flow Filtering for Rigid-Grasp Dynamics Model

Since the rigid-grasp dynamics model assumes that points that are grasped move with the end-effector, it is necessary to identify what points are part of the movable object. To determine this, we employ the heuristic that individual flows

Fig. 11: **In-the-Wild Task Rollouts.** The Franka successfully pulls out a chair, opens a partially opened drawer, sweeps pasta into the compost bin, and recycles an aluminum can.

that move at least 1 pixel on average per timestep are part of the movable flow. We empirically chose the 1 pixel threshold, as we noticed that points tracked on the stationary parts of articulated objects such as the oven had low averages of how many pixels they moved.

For the 2D tracks associated with the movable part, it may be possible that for certain intermediate frames they are 1 pixel off, making the individual tracks belong to the background or floor when lifted into 3D, causing part of the 3D object flow to be dramatically incorrect, leading to downstream execution failures (such as trying to push the toaster oven's door in the direction of the hinge). To remedy this issue, we utilize SAM 2 [2] with positive point prompts in the initial frame for the movable part and negative point prompts for the non-movable part, and use the part mask over each consecutive frame to constrain the 2D flow (if any tracked point is outside of the mask, it is considered to be invalid).

F. Real World Planning Details

The optimization problem with a rigid-grasp assumption in the real world follows the same formulation as in Sec. III-A, where we optimize robot joint angles $\mathbf{q} \in \mathbb{R}^7$ to minimize:

$$\sum_{t=0}^{H-1} \lambda_{\text{task}}\left(\hat{x}_t^{\text{obj}}, \tilde{P}_t\right) + \lambda_{\text{control}}(\hat{x}_t, u_t) \tag{1}$$

The control cost $\lambda_{\text{control}}(\hat{x}_t, u_t)$ is expanded into three components:

$$\lambda_{\text{control}}(\hat{x}_t, u_t) = w_r \mathcal{C}_r(\mathbf{q}_t) + w_s \mathcal{C}_s(\mathbf{q}_t, \mathbf{q}_{t-1}) + w_m \mathcal{C}_m(\mathbf{q}_t) \tag{2}$$

where:

- $\mathcal{C}_r(\mathbf{q}_t)$: Reachability cost penalizing joint configurations outside the robot's workspace
- $\mathcal{C}_s(\mathbf{q}_t, \mathbf{q}_{t-1})$: Pose smoothness cost measuring the difference between consecutive end-effector poses after running forward kinematics
- $\mathcal{C}_m(\mathbf{q}_t)$: Manipulability cost encouraging configurations with good manipulability

The weight parameters are set to $w_r = 100$, $w_s = 1$, and $w_m = 0.01$ to encourage the robot to make smooth motions within its joint limits. The task cost λ_{task} uses a weight of $w_f = 10$ and follows the same formulation as in Sec. III-A. The optimized end-effector poses after running forward kinematics are then fit with a B-spline and sampled such that each sampled pose is at least 1cm away from the previous and next pose and/or has a rotation difference of at least 20 degrees. To execute this planned trajectory, we use the IK solver from PyBullet [96] to get target joint angles and then a joint impedance controller from Deoxys [97] commands the Franka to reach those positions.

G. In-the-Wild Tasks

For In-the-Wild tasks, we consider:

1) Pull Out Chair: In the Pull Out Chair task, a black rolling chair is placed underneath a table. A trial is considered a success if the chair is moved at least 5cm horizontally from its initial position. The reason for the limited motion requirement is that in certain configurations, it is difficult for the Franka robot to push the chair.

2) Open Drawer: The Open Drawer task involves opening a partially opened drawer out to at least 90% of its full possible extension.

3) Sweep Pasta: In the Sweep Pasta task, there is a brush with a green handle placed against a wooden support structure, along with four pieces of dried pasta next to a compost bin. The objective is for the robot to grasp the handle of the brush and use the brush to push the pasta into the compost bin. If all pieces of pasta are inside of the compost bin, it is considered a success.

4) Recycle Can: In the Recycle Can task, an aluminum can is placed in between a recycling and trash bin while upright. A trial is successful if the can is inside of the recycling bin.

See Fig. 11 for rollouts for each of these in-the-wild tasks.

H. Open Door Reinforcement Learning Details

For the Open Door task, we utilize Soft Actor-Critic (SAC) [90] to train sensorimotor policies across three different embodiments: a Franka Panda, a Spot robot, and a GR1 humanoid arm. The policies are trained to follow the 3D object flow extracted from generated videos, which serves as a reward signal.

1) Hyperparameters: The hyperparameters used for SAC training are detailed in Table V. These values were consistent across all reward types and embodiments with exception of the GR1, which uses 10000 training iterations due to the larger action space.

Hyperparameter	Value
Learning rate	3×10^{-4}
Discount factor (γ)	0.99
Batch size	256
Buffer size	10^6
Target update rate (τ)	0.005
Target network update freq	1
Hidden layers	2
Hidden units per layer	256
Training iterations	5000
Episode horizon	500

TABLE V: **SAC Hyperparameters for Open Door Task.**

2) Reward Functions: We compare policies trained with two different reward formulations: a handcrafted object state reward and a 3D object flow reward.

Object State Reward: The handcrafted reward R_{state} consists of a reaching component r_{reach} and a handle rotation component r_{rot}:

$$r_{\text{reach}} = 0.25 \cdot (1 - \tanh(10 \cdot d_{\text{gripper, handle}})) \tag{3}$$

$$r_{\text{rot}} = \text{clip}\left(0.25 \cdot \frac{|\theta_{\text{handle}}|}{0.5\pi}, -0.25, 0.25\right) \tag{4}$$

where $d_{\text{gripper, handle}}$ is the L2 distance between the robot's end-effector and the door handle, and θ_{handle} is the rotation angle of the handle. The total reward is $r_{\text{reach}} + r_{\text{rot}}$, unless the door hinge angle $\theta_{\text{hinge}} > 0.3$ radians, in which case a completion reward of 1.0 is returned.

3D Object Flow Reward: The 3D object flow reward R_{flow} leverages the reference trajectory $P_{1:T}$ extracted from the video. It consists of a particle tracking term r_{particle} and an end-effector alignment term r_{ee}:

$$r_{\text{particle}} = 0.75 \cdot \frac{t^\star}{t_{\text{end}}} \tag{5}$$

where $t^\star$ is the index of the closest timestep in the reference trajectory $P_{1:T}$ based on the current object particle positions $\hat{x}_t^{\text{obj}}$ in the door frame:

$$t^\star = \underset{t \in \{1 \dots t_{\text{end}}\}}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \|\hat{x}_t^{\text{obj}}[i] - P_t[i]\|_2 \tag{6}$$

The end-effector term r_{ee} encourages the robot to stay near the object particles:

$$r_{\text{ee}} = 0.25 \cdot (1 - \tanh(10 \cdot \|ee_{\text{door}} - \bar{x}\|_2)) \tag{7}$$

where ee_{door} is the end-effector position in the door body frame and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} \hat{x}_t^{\text{obj}}[i]$ is the mean position of the object particles. The total reward is $R_{\text{flow}} = r_{\text{particle}} + r_{\text{ee}}$.

Instead of tracking the mean particle position with CoTrackerV3, we proceed to use a simpler approach of transforming the initial particles as the door angle changes for better computation efficiency. We thus consider the door joint angle to be a part of the state x_t.

I. Video Generation Failures

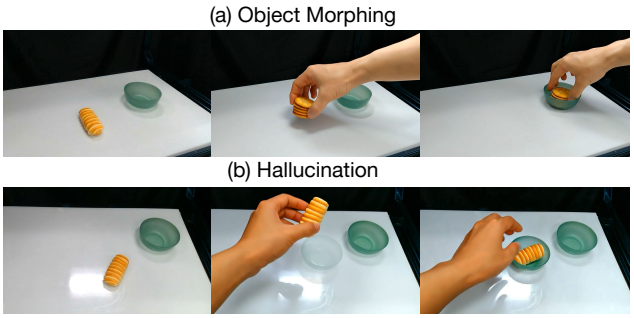

Fig. 12: **Video Generation Failure Examples.** (a) The bread undergoes object morphing, where it turns into a stack of crackers, causing the downstream tracking to fail. (b) Another green bowl appears due to model hallucination, causing a downstream execution failure where the bread is dropped onto the surface of the workspace instead of the original green bowl.

From the generated videos, we observe that there are two common failure modes: morphing and hallucination. Object morphing occurs when an existing object in the scene dramatically changes shape to something else, such as another object or an object with significantly different geometric properties than what it should be. Hallucination occurs when a new object (previously non-existent) appears in the scene. We show examples of morphing and hallucination for the Put Bread task in Fig. 12.

J. Limitations

Dream2Flow has several limitations. First, it relies on a rigid-grasp assumption for real world manipulation, limiting the types of tasks that can be performed. While this work shows that a particle dynamics model can be used for other types of tasks such as non-prehensile pushing, training and scaling a particle dynamics model for the real world is non-trivial and can be considered for future work. Another limitation is that the total processing time to get 3D object flow depending on the video generation model is between 3 and 11 minutes, which limits its usability, with the main bottleneck being the video generation. Since Dream2Flow relies upon one angle in a generated video, it cannot handle heavy occlusions gracefully, such as when the human hand covers the majority of a small object. Future work may consider methods which can deal with such occlusions better, such as 3D point trackers [84] or full 4D representations.