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

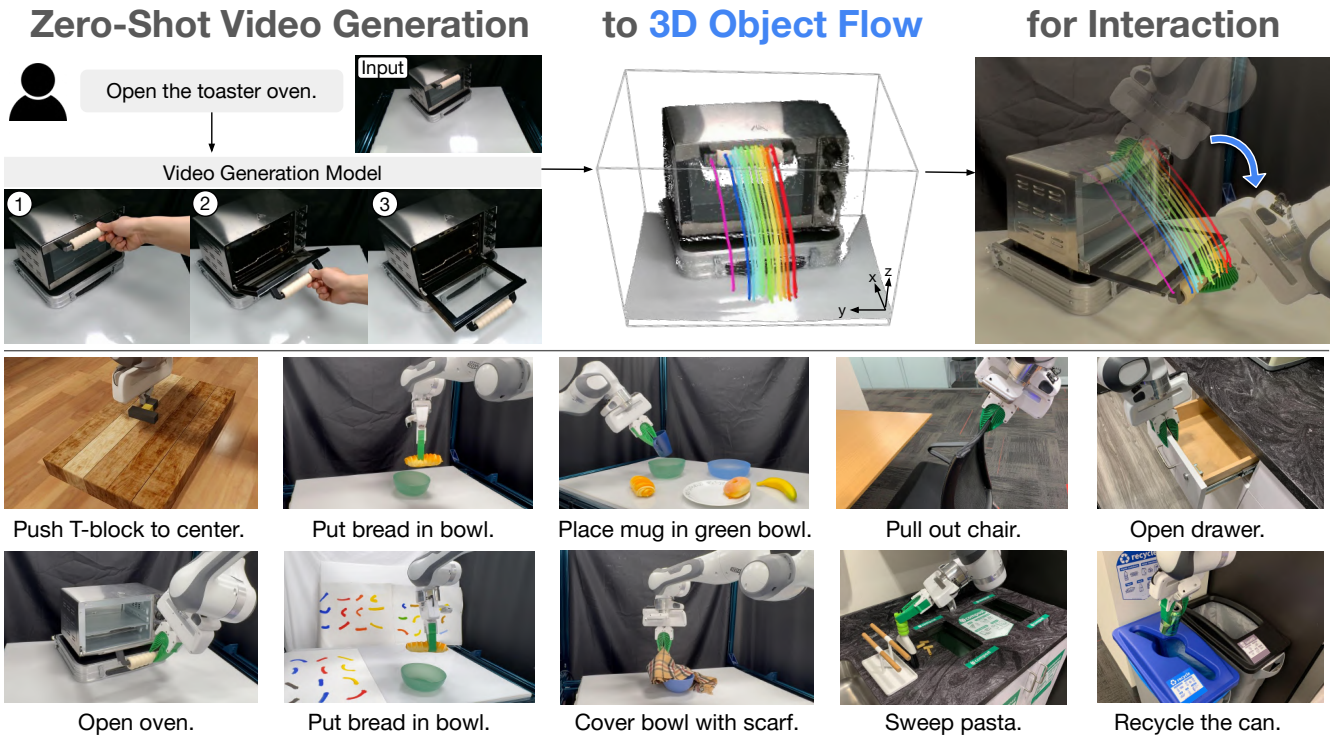

Fig. 1: **Dream2Flow** leverages off-the-shelf video generation models to produce videos of the task being performed in the same scene of the robot. Dream2Flow then extracts a 3D object flow from the motion in the video, allowing for downstream planning and execution with a robot across a wide variety 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 benefit from visual world models that predict how an environment evolves under interaction. Recent generative video models can synthesize plausible physical interactions from an unseen image and an open-ended task instruction [1], offering rich priors for novel tasks in unseen environments. Despite their promise, it remains unclear what role such models should serve in a robot manipulation system. Most frontier video generators work best with human embodiments where supervision is more abundant, but this poses an embodiment gap for robot control.

We address this challenge by extracting actionable signals from visual predictions, which will then be enacted by a robot. Our method, **Dream2Flow**, uses **3D object flow** as an intermediate interface between video generation and robot control. Rather than mimic human motion, we reconstruct and track the task-relevant object motion in 3D. The problem

*Equal Advising. Correspondence: Wenlong Huang

becomes object trajectory tracking: the robot follows the generated object flow while respecting embodiment-specific constraints. This approach separates what should happen in the scene from how a robot executes it, and supports both motion planning and sensorimotor policies.

Using off-the-shelf models and tools, we demonstrate an autonomous pipeline that 1) generates a text-conditioned video of plausible interactions [1], 2) reconstructs 3D object flow via depth estimation and point tracking [2–4], and 3) synthesizes actions with trajectory optimization or reinforcement learning. Since 3D object flow captures task-relevant state changes, it enables manipulation of rigid, articulated, deformable, and granular objects without task-specific demonstrations.

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

Manipulation tasks have been specified through symbolic goals and constraints [5–7]. Learning-based systems specify tasks often through language and perception, mapping instructions to actions via language-conditioned visuomotor policies and vision–language–action models [8–11]. Object-centric alternatives rely on descriptors or keypoints to capture task-relevant structure [12]. Recently, foundation models enable higher-level interfaces that compile intent into actionable specifications via code [13], 3D value maps akin to potential fields [14], keypoint relations [15], or affordance maps [16].

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 [17, 18]. In scene-centric formulations, policies parameterize or condition on motion in 2D or 3D to decide actions [19–26]. In object-centric formulations, desired object motion is specified independently of embodiment and then converted into actions via policy inference, planning, and optimization [27–33]. Our approach follows the this path by reconstructing 3D object flow from language-conditioned generations and tracking it under embodiment constraints, complementing other flow-conditioned policy representations [19, 21, 27, 29, 30].

C. Video Models for Robotics

Recent work increasingly integrates video models across robotic tasks as auxiliary training objectives [34–39], reward models [40–42], policies [43, 44], or as a simulator for the environments [45, 46]. Notably, predictive modeling in robotics can leverage video frame prediction as a form of dynamics model [44, 47–51]. Another notable direction is that video generation can directly provide new training data for robot learning by imagining new trajectories in the form of videos for imitation learning [52, 53].

III. Method

Herein, we introduce the problem formulation of Dream2Flow in Sec. III-A. Leveraging 3D object flow as an interface, we then describe how to extract it from video generations in Sec. III-B and how to infer actions from it in Sec. III-C.

A. Problem Formulation

Given a task instruction ℓ, an initial RGB-D observation (I_0, D_0), and 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 follows an object motion inferred from a generated video. We make no assumption about a specific action parameterization: $\mathcal{U}$ may denote 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 $\{Z_t\}_{t=1}^T$. Given a binary mask M of the task-relevant object, we lift masked image points with $Z_{1:T}$ and Π to obtain an object-centric 3D trajectory $P_{1:T} \in \mathbb{R}^{T\times n\times 3}$, which we call the 3D object flow.

Action Inference with 3D Object Flow. We represent the state as points on the task-relevant object and robot proprioception, $x_t = (x_t^{\text{obj}}, r_t)$. 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 solve:

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

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

B. Extracting 3D Object Flows from Videos

Video Generation: Given instruction ℓ and the initial RGB image I_0, Dream2Flow uses an off-the-shelf image-to-video model to generate a video $\{V_t\}_{t=1}^T$ of the task being performed. We exclude the robot from the initial frame and text prompt because current video models often produce less plausible fine-grained robot interactions, which in turn degrades the reconstructed trajectories (Appendix C).

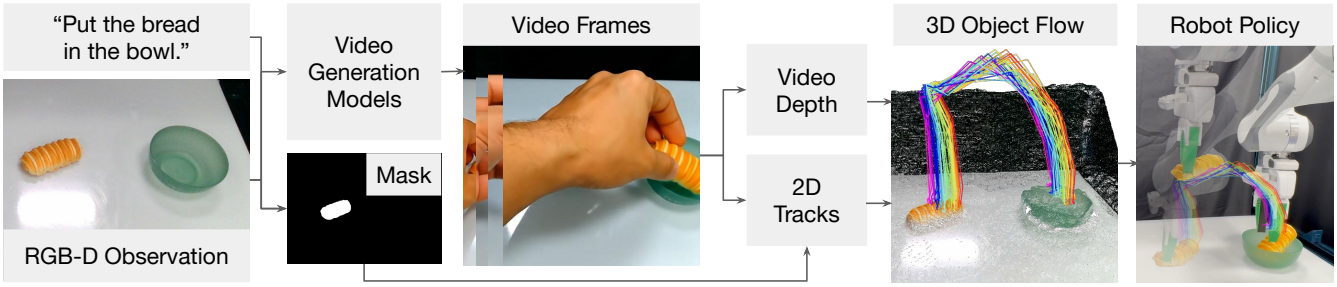

Fig. 2: **An overview of Dream2Flow.** Given a task instruction and an initial RGB-D observation, an image-to-video model synthesizes video frames conditioned on the instruction. We additionally obtain object masks, video depth, and point tracking from vision foundation models, which are used to reconstruct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

Video Depth Estimation: We estimate per-frame depth $\{\tilde{Z}_t\}_{t=1}^{T}$ with SpatialTrackerV2 [4, 54]. To resolve monocular scale-shift ambiguity, we align the first frame to the robot depth D_0 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 task-relevant object using Grounding DINO [55] and SAM 2 [2]. From the mask at $t{=}1$, we sample n pixels and track them with CoTracker3 [3] to obtain 2D trajectories and visibilities. Visible points are then lifted to 3D using the calibrated depths and camera intrinsics/extrinsics, yielding $P_{1:T}$ in the robot frame.

C. Action Inference with 3D Object Flow

Simulated Push-T Domain. For Push-T, Dream2Flow uses a push primitive parameterized by start position, direction, and distance. We learn a particle-based forward dynamics model which takes in as input the feature-augmented scene points, consisting of positions, RGB colors, surface normals, and push parameters. We proceed to use random-shooting to sample r push skill parameters and then select the candidate with the lowest predicted flow-tracking cost. The target points for the cost are selected from the flow L timesteps ahead of the timestep with the closest points to the current observation $t^\star$. Further details are in Appendix E.

Real-World Domain. We use absolute end-effector poses as the action space and a rigid-grasp dynamics model. We first proceed to grasp the desired part on the relevant object, and then use the dynamics model and point-flow following objective to move the end-effector such that the grasped part motion is similar to the video. Candidate grasps come from AnyGrasp [56], and we choose the one closest to the thumb detected in the generated video via HaMeR [57], which indicates the intended interaction point, such as a handle. The rigid-grasp dynamics model assumes that grasped points move with the end-effector while non-grasped points remain fixed, allowing us to produce an end-effector trajectory with PyRoki [58] using flow-tracking, smoothness, and reachability costs (Appendix H).

Simulated Door Opening Domain. For Door Opening, we use reinforcement learning to learn a sensimotor policy which moves the object according to the 3D object flow with SAC [59]. 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 manner. The reward function involves a contact term and 3D object flow progress term (details in Appendix J).

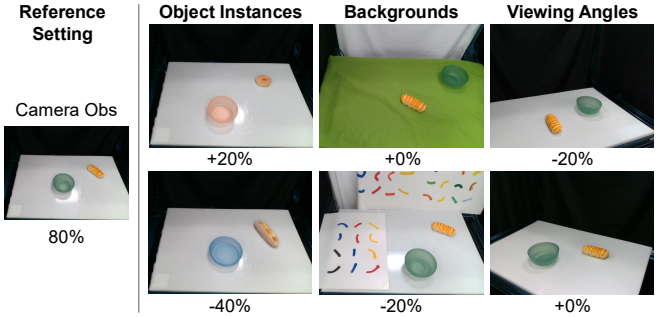

Fig. 3: **Robustness evaluations.** Relative performance across instance, background, and task variations, showing Dream2Flow remains robust under various different settings.

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?

We evaluate Dream2Flow on simulated and real manipulation tasks spanning rigid, articulated, deformable, and granular objects, including Push-T, Bread in Bowl, Oven Opening, Bowl Covering, and Door Opening as seen in Figures 1 and 4.

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

For the simulated Push-T environment, we use Wan2.1 [60] with a goal image prompt and evaluate 10 initial states with 10 seeds each for 100 total trials. Six generated videos exhibited severe T-block morphing, which corrupted tracking and execution. In the real world, we use Veo 3 [61] and run 10 trials per task across the Bread in Bowl, Oven Opening, and Bowl Covering tasks as shown in Table I.

Task	AVDC	RIGVID	Dream2Flow
Push-T	-	-	**52/100**
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.

For evaluating certain axes of generalization, we run five trials each across variations in object instance, background, and viewpoint as in Fig. 3. Performance remains similar to the original setting except on the large-bread variation. We additionally observe that different language prompts on the same scene can lead to performing different tasks in Appendix C. We additionally show case studies demonstrating completion of in-the-wild tasks in Fig. 1 and Appendix I.

Across 60 real-world trials, failures mainly stem from video artifacts, tracking loss under occlusion or rotation, and failed executions (Fig. 5). Among video failures, common modes are object morphing and hallucination, while execution failures mostly occur in the Cover Bowl task involving a deformable object.

B. How does Dream2Flow perform compared to alternative interfaces?

For the three real-world tasks, we compare against AVDC [62], which uses dense optical flow and depth to estimate a sequence of rigid object transforms from the generated video as well as RIGVID [63] which tracks rigid object poses. For deformable settings, we adapt RIGVID by solving for rigid transforms between initial and visible 3D points.

Table I shows that Dream2Flow outperforms AVDC and RIGVID. AVDC can track bread reasonably well, but the optical flow could not capture the motion of the oven door. AVDC and RIGVID become brittle when visible points are sparse or occluded, making rigid transform estimation (and consequently execution) noisy. In contrast, Dream2Flow moves smoothly between points of high visibility since it does not perform rigid transform estimation.

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

We also use 3D object flow as an RL reward. SAC [59] policies trained with the hand-crafted object state reward and with the 3D object flow based reward achieve comparable success across a Franka Panda, a floating base Spot, and a GR1 over 100 random door positions (Table II). Figure 4 illustrates that different embodiments discover distinct yet effective strategies, with Spot using base motion for reachability and GR1 relying more on palm-finger contact for stability.

V. Conclusion

We presented Dream2Flow, which converts text-conditioned video generations into executable robot behavior by reconstructing and tracking 3D object flow. By separating task-relevant object motion from embodiment-specific control, the method supports planning and policy learning across simulated and real tasks involving rigid, articulated, deformable, and granular objects. Results show stronger performance than rigid-trajectory baselines and highlight current limits from video artifacts, tracking loss, and grasp selection. Overall, 3D object flow provides a practical bridge from open-ended video generation to robot control.

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.

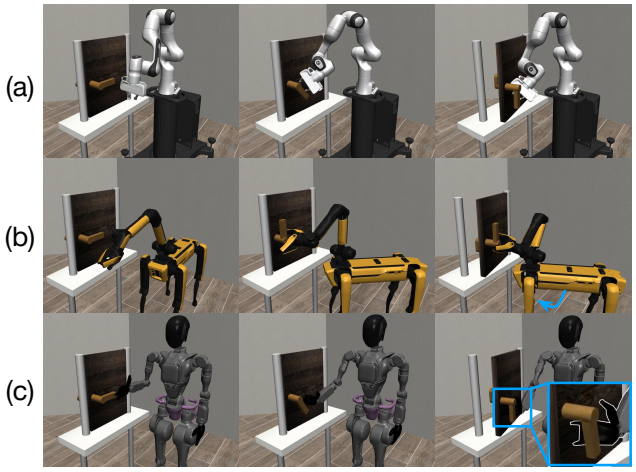

Fig. 4: **Rollouts from policies trained using 3D object flow as a reward.** Different embodiments such as the (a) Panda, (b) Spot, or (c) GR1 use different strategies to open the door. The Spot is able move its base for better reachability while the GR1 uses the area between its fingers and palm to pull for better stability.

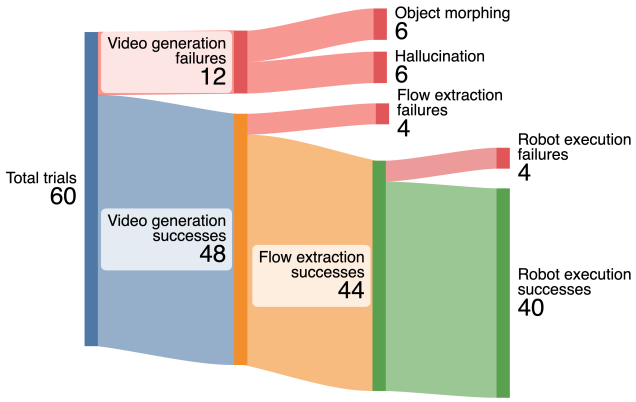

Fig. 5: **Failure breakdown on real-robot experiments.** Common causes include video artifacts (object morphing, object hallucination), tracking errors, and grasp selection mismatches.

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] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[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] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

[28] 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

[29] 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.

[30] 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.

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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.

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

[68] 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. Acknowledgement

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.

B. Tasks

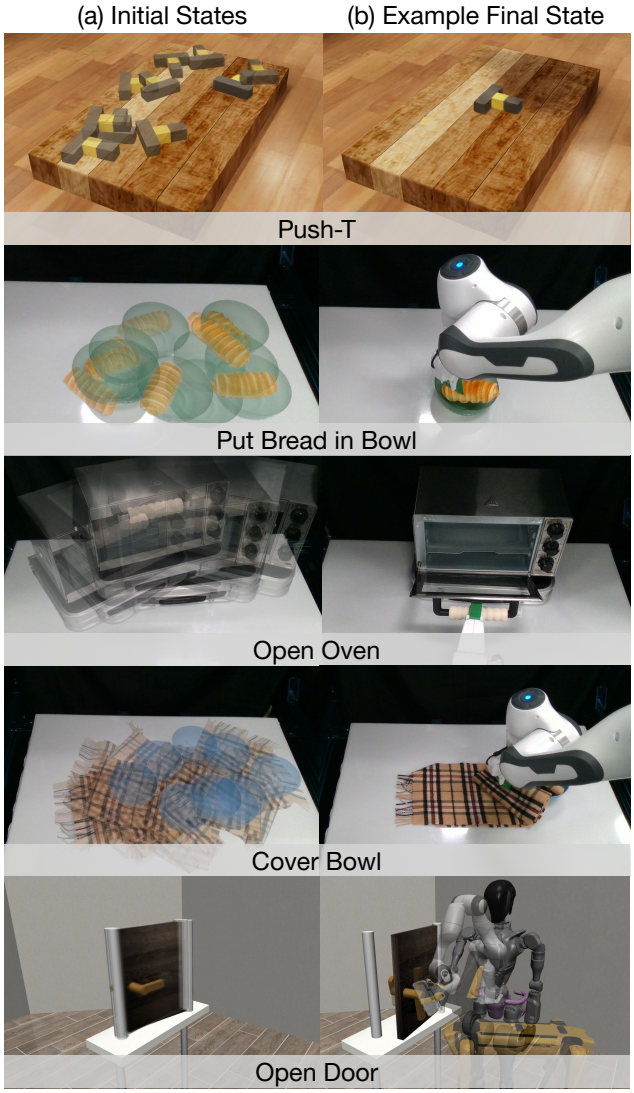

Fig. 6: **Evaluation Tasks.** (a) The initial states of each task used in the evaluation trials. (b) For one of the initial states, the corresponding final state after the robot performs the task. For Push-T only, the desired final state is the same for all initial states.

For evaluation, we consider several tasks involving different types of objects and manipulation strategies (Fig. 6).

1) Push-T: We develop a simulated Push-T task in OmniGibson [64], 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 [65], 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.

C. 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. 7: **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 7. 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.

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.

D. 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

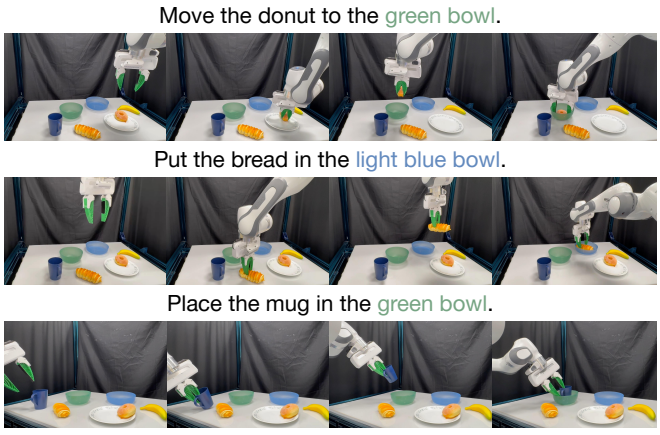

Fig. 8: **Multiple tasks in the same scene.** With different language goals, Dream2Flow adapts object-flow targets to produce distinct behaviors in the same environment.

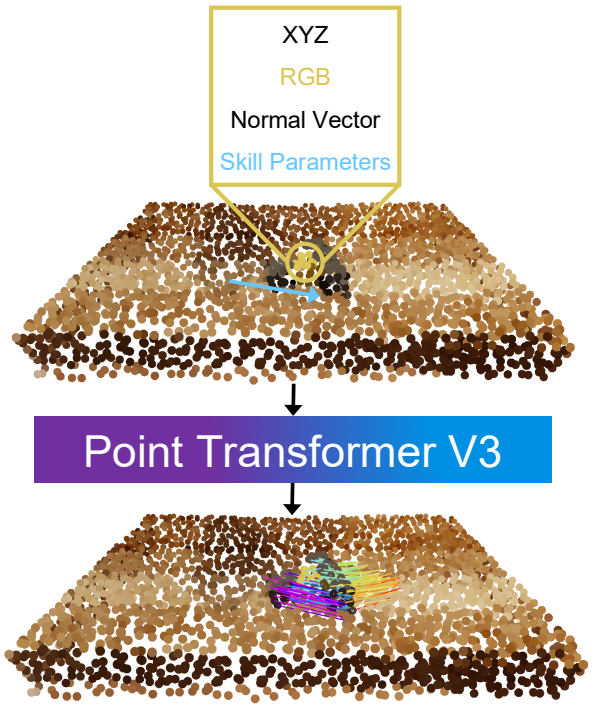

Fig. 9: **Particle dynamics model.** The particle dynamics model primarily composed of a Point Transformer V3 backbone used in the Push-T task takes as input a set of feature-augmented particles and outputs delta position predictions for all particles in the scene.

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 [66] 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.

E. 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.

F. Grasp Selection

Dream2Flow uses AnyGrasp [56] 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 [57] 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.

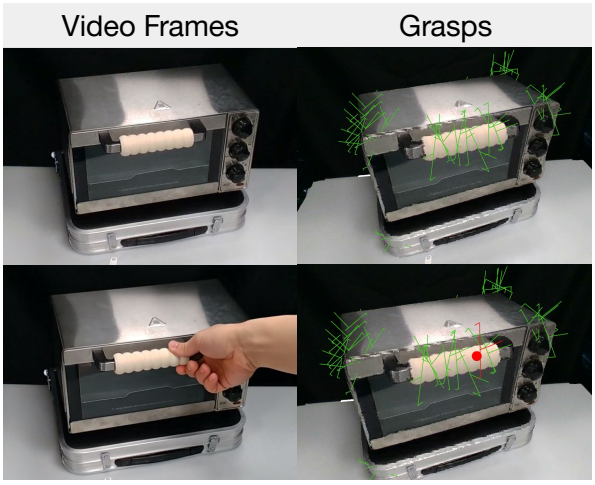

Fig. 10: **Grasp selection with thumb position.** Dream2Flow selects a grasp closest to the position of the thumb in the video, if such a grasp exists within 2cm. The red sphere indicates the thumb position predicted by HaMer, with the red grasp being the selected one.

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 [57] 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).

G. 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 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).

H. 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,

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.

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 [67] to get target joint angles and then a joint impedance controller from Deoxys [68] commands the Franka to reach those positions.

I. 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.

J. Open Door Reinforcement Learning Details

For the Open Door task, we utilize Soft Actor-Critic (SAC) [59] 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 III. 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 III: **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.

K. Effect of Video Generation Model

Video Generation Model	Push-T	Open Oven
Wan2.1 [60]	**52/100**	2/10
Kling 2.1	31/100	4/10
Veo 3 [61]	-	**8/10**

TABLE IV: **Effect of video generator.** Veo 3 performs best on the real Open Oven task, while Wan 2.1 performs better on the simulated Push-T domain.

To evaluate how different video generation models affect downstream performance, we run Dream2Flow on the simulated Push-T task and the real Open Oven task using Wan2.1 [60], Kling 2.1, and Veo 3 [61]. The results are shown in Table IV. There are no Veo 3 results for Push-T because, at the time of evaluation, Veo 3 did not support prompting with a goal image.

For Push-T, Kling 2.1 more frequently produced videos with substantial object morphing, which corrupted tracking and reduced task success. For Open Oven, Wan2.1 often introduced noticeable camera motion despite the still-camera prompt, violating the assumptions of the downstream depth pipeline. Both Kling 2.1 and Wan2.1 also occasionally generated incorrect articulation directions, such as rotating the oven door about the wrong axis, which further hurt performance relative to Veo 3.

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

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

L. Effect of Dynamics Model

For Push-T, we compare the particle-based dynamics model used for planning against two simpler alternatives. The first is a learned pose-space model that takes the T-block pose and push skill parameters as input and predicts the pose change after the push, trained on the same data as the particle model. The second is a heuristic model that translates the T-block in the push direction without modeling rotation.

Table V shows that the particle representation is important for this task. Although all three models receive the same 3D object-flow guidance, the pose and heuristic models do not capture the rotational and contact-dependent effects needed to solve Push-T reliably, leading to much lower success rates.

M. Video Generation Failures

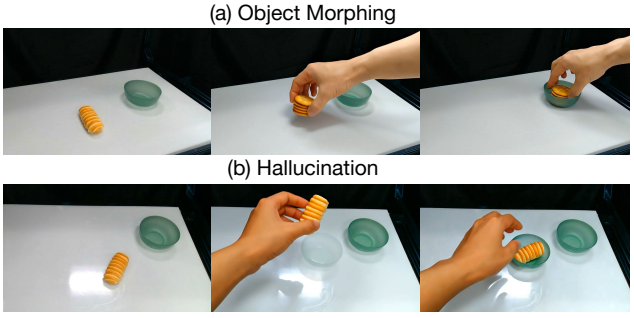

Fig. 12: **Video Generation Failure Examples.** (a) The bread undergoes object morphing, where it turns into a stack of crackers, causing the downstream tracking to fail. (b) Another green bowl appears due to model hallucination, causing a downstream execution failure where the bread is dropped onto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

N. 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 [4] or full 4D representations.