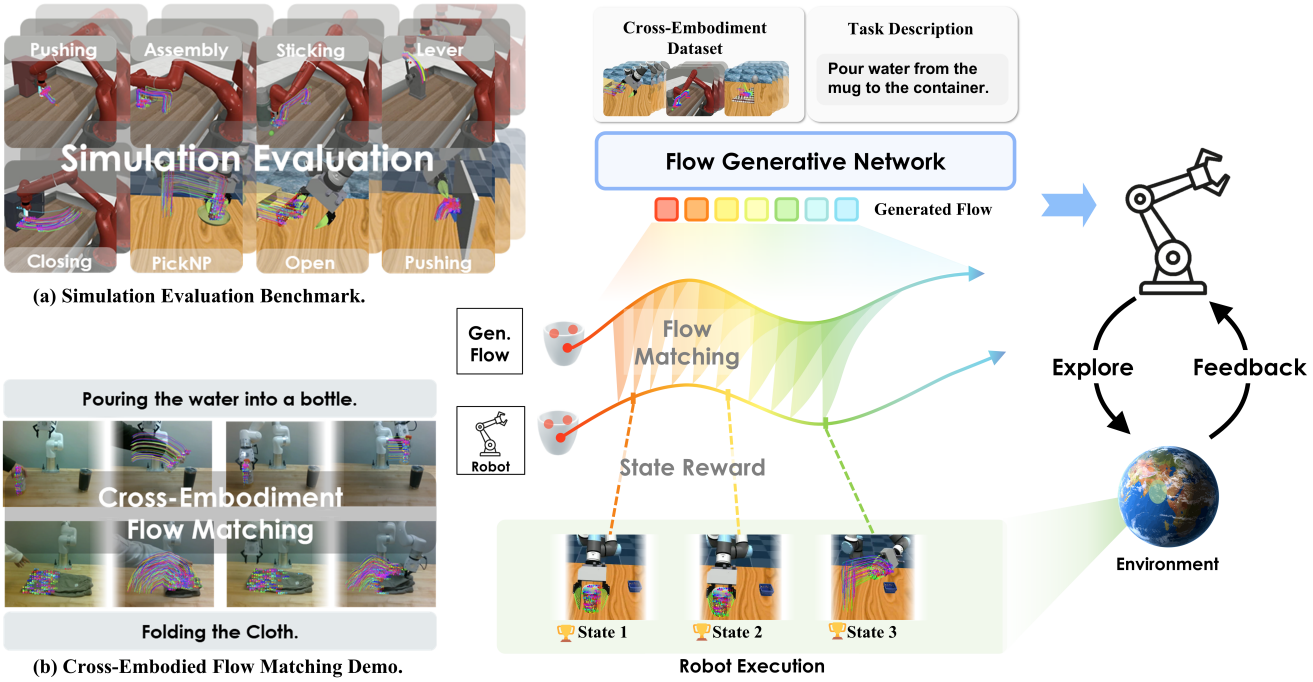
GENFLOWRL: Generative Object-Centric Flow Matching for Reward Shaping in Visual Reinforcement Learning

Kelin Yu[1], Sheng Zhang[1], Harshit Soora[1], Furong Huang[1], Heng Huang[1], Pratap Tokekar[1], Ruohan Gao[1]
[1]University of Maryland, College Park



(a) Simulation Evaluation Benchmark.

(b) Cross-Embodied Flow Matching Demo.

Fig. 1: **Illustration of our GENFLOWRL framework**, which guides visuomotor policy by leveraging generative object-centric flow as task motion prior (Right). In our proposed reward model, dense flow matching between online trajectories and flow prior, synergizing with sparse state-aware reward, facilitates efficient, robust, and generalizable policy learning. Our extensive evaluation includes 10 challenging simulation manipulation tasks (Fig. A) and real-world cross-embodiment reward matchness probing experiments (Fig. B).

***Abstract*— Recent advances have demonstrated the potential of video generation models to guide robot learning by deriving effective robot actions through inverse dynamics. However, these methods heavily depend on the quality of generated data and struggle with fine-grained manipulation due to the lack of environment feedback. While video-based reinforcement learning improves policy robustness, it remains constrained by the artifacts of generated video and the challenges of collecting large-scale in-domain robot datasets for training diffusion models. Motivated by the above, we propose GENFLOWRL, which derives shaped rewards from generated flow trained from cross-embodiment datasets. This enables learning generalizable and robust policies from expert demostrations using low-dimensional, object-centric features. Experiments on 10 manipulation tasks, both in simulation and real-world cross-embodiment evaluations, demonstrate that GENFLOWRL effectively leverages manipulation features extracted from generated object-centric flow, consistently achieving superior performance across diverse and challenging scenarios.**

I. INTRODUCTION

Recent advances in video generation models have demonstrated the potential of enhancing robot learning by deriving effective actions from generated robot videos using inverse dynamics [1–4]. This approach enables learning policies across diverse scenarios with strong generalizability. However, most existing methods rely on open-loop policies that operate solely on the generated data, without actively interacting with the environment. As a result, they are highly dependent on the quality of the generated videos, limiting their robustness. This limitation becomes particularly critical in tasks that handles contact-rich scenarios, where environmental feedback is essential.

Reinforcement Learning (RL), on the other hand, offers a complementary approach by allowing policies to interact with the environment and iteratively improve their robustness [5–7]. Therefore, combining video generation models with RL presents a promising direction [8–11], as pre-trained video generation models can serve as rewards for RL training across diverse settings, thus enhancing generalizability. However, training such models usually require collecting large-scale robotic data, which remains a significant chal-

lenge. Additionally, video generation models often introduce substantial uncertainty, which makes it difficult to produce high-quality videos for shaping rewards in RL training.

To address the aforementioned limitations and to better leverage the strength of both video generation models and RL, we propose to use generated *object-centric flow* [12–15]—the 2D keypoints trajectory of objects—for reward shaping. Object-centric flow has been utilized in robot learning to bridge the embodiment gap between robots and easy-to-collect human hand demonstrations [12, 15]. Compared with robotic videos, it is a much lower-dimensional representation that is easier to generate while retaining more manipulation-related features. Additionally, expert object-centric flow can be directly used for reward shaping without requiring specific representation learning [15]. As shown in Table II, object-centric flow offers a fine-grained representation that can also be used for learning deformable or articulated objects manipulations. These unique properties make it well-suited for both *generation* and *reward shaping*.

Towards this end, we propose GENFLOWRL, a novel framework that integrates object-centric flow generation with reinforcement learning. Our approach harnesses the generalization capability of flow generation models while incorporating environment interactions to overcome the limitations of open-loop policies. By leveraging object-centric flows, GENFLOWRL utilizes easy-to-collect cross-embodiment datasets for training stable generative model, and uses the generated object-centric flows for dense reward shaping. These flow-derived rewards are combined with sparse state-based rewards, providing richer task and environment priors. Finally, the generated flow can be used as a motion prior to improve policy robustness.

Our extensive evaluations on 10 manipulation tasks demonstrate the effectiveness of our flow-derived reward for incentivizing the exploration of expert-like behaviors, outperforming both imitation learning and video-guided RL. Comparisons with other object-centric representations further highlights its unique advantages for fine-grained manipulation tasks. Additionally, our case study using a real robot arm confirms the effectiveness of our flow-derived reward model through human-to-robot transfer.

Our **key contributions** are three-fold: (1) We propose GENFLOWRL, a novel framework that generates object-centric flows from cross-embodiment videos to guide RL policy learning, overcoming the limitations of conventional video-based RL approaches. (2) We comprehensively evaluate various object-centric representations for manipulation policy learning, analyzing their effectiveness and efficiency while uncovering practical insights. (3) Extensive evaluation on 10 challenging fine-grained manipulation tasks, both simulation and real-world cross-embodiment experiments, demonstrate the effectiveness of our method.

II. METHODOLOGY

To enable RL to learn from expert motion priors with more robustness and generalizability, we present GENFLOWRL, a framework that bridges the gap between generative foundation models and RL to tackle diverse manipulation tasks that involve fine-grained and contact-rich interactions. The key innovation of GENFLOWRL is our proposed object-centric flow-derived reward model for policy learning. This reward model uses generated δ-flow representations to mimic expert motion priors from large-scale, cross-embodiment datasets while incentivizing exploration. Additionally, it incorporates real-time feedback to enhance robustness against noise and potential inaccuracies in these priors.

The overall architecture of our method is illustrated in Fig. 2. Our framework is composed of three main components: (1) Task-conditioned object-centric flow generation, which produces object-centric flows conditioned on task descriptions (Sec. II-A). (2) A hybrid reward model guides policy learning through continuous feedback from dense matching with our δ-flow, constructed from the object-centric task flow, while also comprising sparse rewards based on environment and object states (Sec. II-B). (3) Flow-conditioned policy learning, where we use the hybrid reward model to train a generalizable policy that is conditioned on the generated flow (Sec. II-C).

A. Object-Centric Flow Generation

Adapting pre-trained video generative models to predict scene dynamics in the image space often leads to artifacts, physical distortions, and factual inaccuracies that mislead reinforcement learning for robotics tasks [2, 3]. In contrast, object-centric representation avoids these pitfalls thanks to its RL compatibility (see Tab. II). *Object-centric flow* represents the temporal motion of keypoints on a target object relative to the initial frame, abstracting away visual appearances [12]. Motivated to overcome the suboptimality in RL of existing object-centric flow evidenced by our investigation results (Sec. II-B), we propose our object-centric δ-flow representation leveraging video generation priors [12–14], which captures object-centric relative translation and rotation. Method details are in Supp. IV-A.

B. Flow-Derived Reward Model

The keypoints of the generated flow in the previous step offer dense temporal guidance with object-centric spatial features but sometimes contain noisy and misleading cues. Motivated by the above, we first design a more robust flow representation similar to HuDor [15], namely δ-flow, which proves to be efficient and tailored to RL training. Secondly, we develop a hybrid reward model derived from flow, which combines a flow-matching reward—aligning current flow with the generated flow—with a sparse task-based reward. This reward model actively balances exploration and task completion by leveraging expert motion priors from generated flow while following the guidance from sparse reward. As a result, our adaptive reward model enables the robot to generalize to diverse manipulation settings and objects. More details of this system are shown in the Supp. IV-B

C. Policy Design

This section details our robust policy design, which is grounded in our flow-derived reward model. In contrast to existing RL-based methods [10, 15, 16], our policy begins with a low-dimensional generated δ-flow that encodes planned

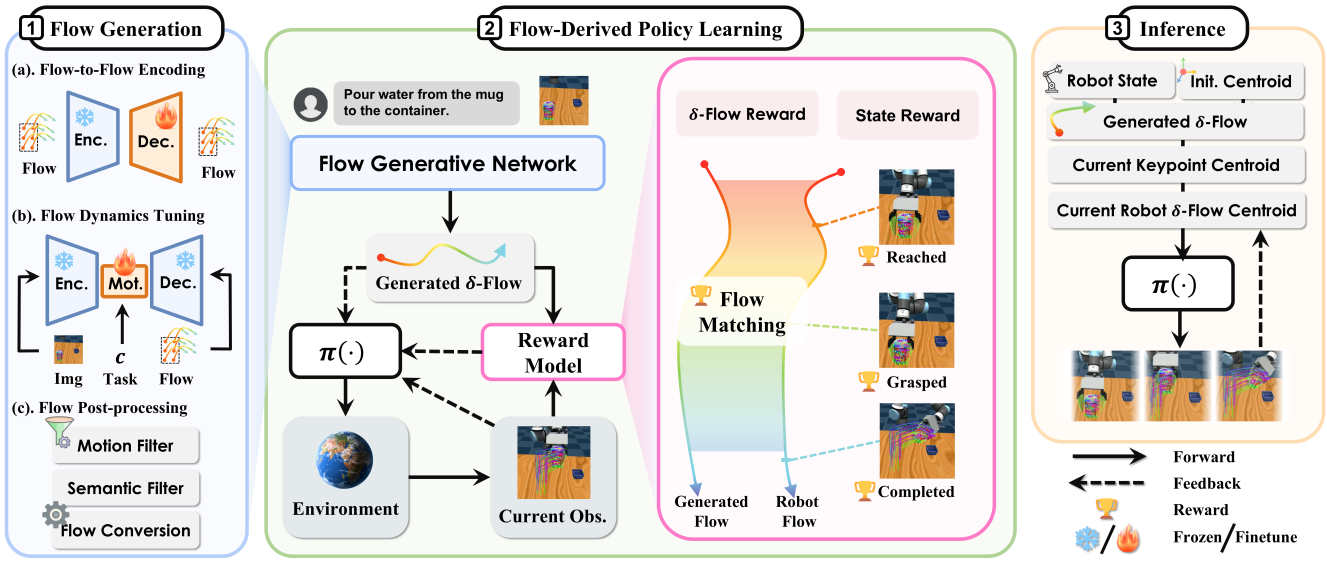


Fig. 2: **Architectural overview of our proposed GenFlowRL framework**, which encompasses our flow generation process (**left**), flow-derived policy learning (**middle**), and inference stage (**right**). In the *object-centric flow generation* process (Sec. II-A), we: (a) Adapt a pre-trained generative model decoder via flow-to-flow reconstruction; (b) Fine-tune the latent motion module on flow generation conditioned on task descriptions and the initial keypoints; (c) Apply motion/semantic filters to refine the generated flow and then convert it into our effective δ-flow (Sec.II-B) representation. In our *flow-derived policy learning* stage, our hybrid reward model (Sec. II-B) guides policy learning through a dense δ-flow matching reward derived from generated flows and a sparse state-aware reward derived from environmental interactions. During *inference*, the policy executes 6D robot actions with conditions: robot state, initial 3D keypoint centroid, future steps of the generated δ-flow, and current observation feedback.

future keypoints complemented with the 3D prior from the initial frame, which helps the policy better comprehend the 3D spatial transformation of objects from the 2D flow transformation. More details are shown in Supp. IV-C

III. Experiments

We evaluate our proposed method on 10 challenging robotic manipulation tasks from two benchmarks. Our experiments are designed to answer the following key questions: (1) Compared to flow-based imitation learning, how robust is RL when using our flow-derived reward model? (Sec. III-A) (2) Compared to video-based reward shaping, how effective is reward shaping with an object-centric representation? (Sec. III-B) (3) How does the choice of object-centric representation impact performance? (Supp. IV-I) In addition to these questions, we also conduct ablation studies on different input variations (Supp. IV-J) and present a case study on reward matching between human hand demonstrations and robot execution in the real world (Sec. III-C). More details about experimental settings are shown in the Supp. IV-E

A. How Robust is RL with Flow-Derived Reward Model?

To answer this question, we evaluate our proposed framework with three baselines on five established robotic manipulation tasks [12]. The comparison results between our method and the baselines are presented in Table I.

Baselines: We compare our framework with two baselines: (1) *Heuristic Policy*: A heuristic action policy that first selects object contact points and applies a pose estimation method to the object 3D flow in the following steps. This method is implemented in General Flow [13], and we provide the ground-truth 3D flow as input. (2) *Im2Flow2Act*: A two-stage flow-based imitation learning method [12], which follows a similar flow generation pipeline but trains task-agnostic flow-based Diffusion Policy using simulated heuristic actions instead of online RL training.

Evaluation: Following existing practice [12], we evaluate in two setups: *Demo-conditioned execution,* which evaluates each flow-based policy conditioned on the oracle flow extracted from expert demos in the dataset and *Language-conditioned execution*, which evaluates each policy conditioned the generated flow from the initial frame and task language descriptions.

Key Findings: The results in Table I demonstrate that our method outperforms all the baselines in these five tasks, especially on challenging *Folding* and *Pivoting*. We can observe that our method clearly outperforms the baselines in both demo-conditioned and language-conditioned setups. For the deformable object manipulation task, *Folding*, the superior robustness of our method can be attributed to two key factors: *Firstly*, our policy is consistent in training and evaluation in that our policy is learned with noisy generated flows. *Second*, in contrast to previous works, our proposed condensed δ-flow representation is superior in reducing the noise of generated flows. For contact-rich manipulation task, *Pivoting*, our approach can leverage interactions with environment to gather a more diverse set of trajectories, thereby ensuring better alignment with the expert motion prior.

B. How Effective is Object-Centric Representation for RL?

To investigate, we compare the training success rate of our framework with four RL-based baselines with different

	Demonstration-Conditioned					Language-Conditioned				
	PickNP.	Pour	Open	Fold	Pivot	PickNP.	Pour	Fold	Fold	Pivot
Heuristic [13]	70	50	30	0	0	/	/	/	/	/
Im2Flow2Act [12]	100	95	95	90	60	90	85	90	35	45
GENFLOWRL (Ours)	**100**	**100**	**100**	**95**	**90**	**95**	**95**	**95**	**80**	**85**

TABLE I: **Performance comparison for reward model ablations** across various methods on 5 challenging simulation tasks under two setups: demonstration-conditioned and language-conditioned execution. Scores reported in success rate.

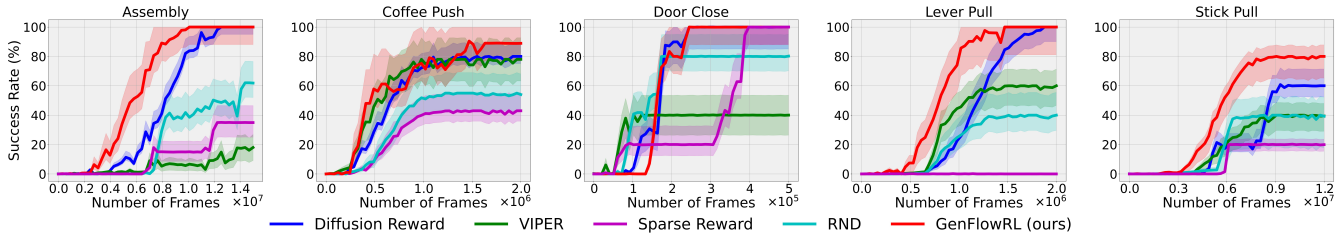


Fig. 3: Comparing with other video-based reward models. The shaded area is the standard deviation for three random seeds.

representations on five robotic manipulation tasks in Meta-World [17]. Comparison results are shown in Fig. 3.

Baselines: We compare our framework with four baselines: (1) *Pure Sparse Reward (PSR)* only uses state-aware sparse reward, which aims to evaluate the effectiveness of reward shaping via a pre-trained generative model. (2) *Diffusion Reward* [10] is the state-of-the-art video-based reward shaping method, which also utilizes diffusion model for video generation with conditional-entropy based reward shaping. (3) *VIPER* [16] uses video generation model, VideoGPT [18], as video prediction model and utilizes log-likelihood of agent observation as reward. (4) *Random Network Distillation (RND)* [19] encourages exploration with a novelty-seeking reward, different from pre-trained generative model based reward shaping.

Key Findings: Compared with non-pretrained methods like PSR and RND, reward shaping with a model pre-trained from expert demonstrations achieves superior performance. While these baselines perform competitively on simpler tasks like *Door Close* and *Coffee Push*, they struggle in more challenging contact-rich manipulation tasks such as *Assembly*. When compared with pre-trained video generation models, the generated object-centric flow shows better training efficacy, indicating that extracting essential manipulation-relevant features from complex video distributions is particularly challenging for intricate tasks. In contrast, our framework leverages δ-flow as a more robust representation with manipulation-centric spatiotemporal features, effectively guiding exploration with expert motion priors.

C. Reward Shaping with Human Hand Demonstration

We perform a case study in this section to show how effectively can our flow-derived reward model perform cross-embodiment human-to-robot transfer. Thus, we set up experiments on two tasks from Im2Flow2Act on a real XArm7 robot arm: *Pouring* and *Folding*. Our goal is to evaluate whether the cross-embodiment object-centric flow from human collected data can be used for real-robot reward shaping. For each task, we collect five expert human hand demonstrations with varying object placements, and then roll out open-loop robot trajectories aligned with each expert motion. In Fig. 4, the visualized results confirm that our flow-derived reward is effectively derived from expert object-centric flow from human demonstrations, and produces a monotonic reward signal, indicating the potential of deploying our policy into the real world with easy-to-collect human hand demonstrations. We present more implementation details and qualitative results in Fig. 14

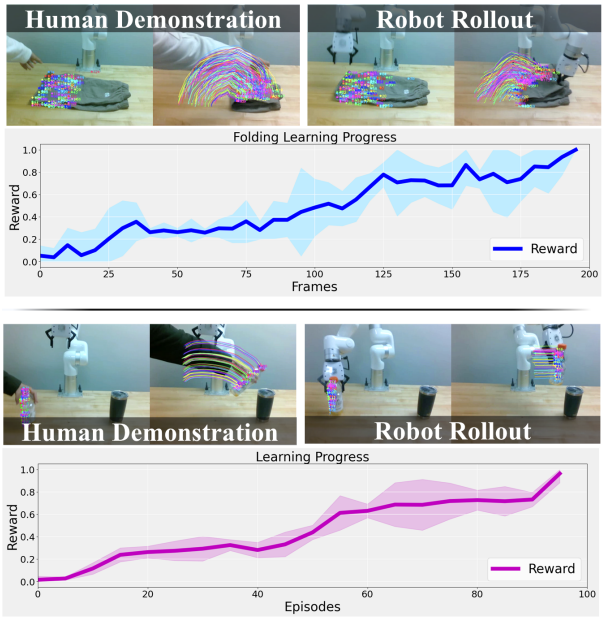


Fig. 4: **Real World Evaluations.** The visualization and reward curve of the real world human to robot flow matching. The shade area represents the standard deviation for five random rollouts.

IV. CONCLUSION

We presented GENFLOWRL, a novel framework that leverages generated flow for reward shaping in visual reinforcement learning, enabling robust and generalizable policy learning. Experiments on 10 challenging manipulation tasks demonstrate the strong performance of our method, outperforming a series of baselines. As future work, we plan to explore whether full 3D flow can overcome the limitations of 2D flow, which may struggle with tasks that involve out-of-plane rotations (*e.g.*, twist-off).

References

[1] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum, "Learning to Act from Actionless Videos through Dense Correspondences," *arXiv:2310.08576*, 2023.

[2] S. Huang *et al.*, "Ardup: Active region video diffusion for universal policies," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8465–8472.

[3] J. Liang *et al.*, "Dreamitate: Real-world visuomotor policy learning via video generation," *arXiv preprint arXiv:2406.16862*, 2024.

[4] B. Wang, N. Sridhar, C. Feng, M. Van der Merwe, *et al.*, "This&that: Language-gesture controlled video generation for robot planning," *arXiv preprint arXiv:2407.05530*, 2024.

[5] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto, "Mastering visual continuous control: Improved data-augmented reinforcement learning," in *International Conference on Learning Representations*, 2022.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, 2018. arXiv: 1801.01290 [cs.LG].

[8] D. Yang, D. Tjia, J. Berg, D. Damen, P. Agrawal, and A. Gupta, "Rank2reward: Learning shaped reward functions from passive video," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 2806–2813.

[9] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[10] T. Huang, G. Jiang, Y. Ze, and H. Xu, "Diffusion reward: Learning rewards via conditional video diffusion," *European Conference on Computer Vision (ECCV)*, 2024.

[11] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg, "Learning by watching: Physical imitation of manipulation skills from human videos," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 7827–7834.

[12] M. Xu *et al.*, *Flow as the cross-domain manipulation interface*, 2024. arXiv: 2407.15208 [cs.RO].

[13] C. Yuan, C. Wen, T. Zhang, and Y. Gao, "General flow as foundation affordance for scalable robot learning," *arXiv preprint arXiv:2401.11439*, 2024.

[14] C. Gao, H. Zhang, Z. Xu, C. Zhehao, and L. Shao, "FLIP: Flow-centric generative planning for general-purpose manipulation tasks," in *The Thirteenth International Conference on Learning Representations*, 2025.

[15] I. Guzey, Y. Dai, G. Savva, R. Bhirangi, and L. Pinto, *Bridging the human to robot dexterity gap through object-oriented rewards*, 2024. arXiv: 2410.23289 [cs.RO].

[16] A. Escontrela *et al.*, "Video prediction models as rewards for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[17] T. Yu *et al.*, *Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning*, 2021. arXiv: 1910.10897 [cs.LG].

[18] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, *Videogpt: Video generation using vq-vae and transformers*, 2021. arXiv: 2104.10157 [cs.CV].

[19] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, *Exploration by random network distillation*, 2018. arXiv: 1810.12894 [cs.LG].

[20] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh, "Flowretrieval: Flow-guided data retrieval for few-shot imitation learning," in *8th Annual Conference on Robot Learning*, 2024.

[21] M. Levy, S. Haldar, L. Pinto, and A. Shirivastava, *P3-po: Prescriptive point priors for visuo-spatial generalization of robot policies*, 2024. arXiv: 2412.06784 [cs.RO].

[22] G. Jiang, Y. Sun, T. Huang, H. Li, Y. Liang, and H. Xu, "Robots pre-train robots: Manipulation-centric robotic representation from large-scale robot datasets," *arXiv preprint arXiv:2410.22325*, 2024.

[23] R. Zheng *et al.*, "Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies," *arXiv preprint arXiv:2412.10345*, 2024.

[24] P. Yu, A. Bhaskar, A. Singh, Z. Mahammad, and P. Tokekar, *Sketch-to-skill: Bootstrapping robot learning with human drawn trajectory sketches*, 2025.

[25] Y. Han, Z. Chen, K. A. Williams, and H. Ravichandar, *Learning prehensile dexterity by imitating and emulating state-only observations*, 2024. arXiv: 2404.05582 [cs.RO].

[26] Y. Chen, C. Wang, Y. Yang, and C. K. Liu, *Object-centric dexterous manipulation from human motion data*, 2024. arXiv: 2411.04005 [cs.RO].

[27] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *8th Annual Conference on Robot Learning*, 2024.

[28] W. Tang *et al.*, *Embodiment-agnostic action planning via object-part scene flow*, 2024. arXiv: 2409.10032 [cs.RO].

[29] S. Liu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European Conference on Computer Vision*, Springer, 2025, pp. 38–55.

[30] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, "CoTracker: It is better to track together," 2023.

[31] Y. Guo *et al.*, *Animatediff: Animate your personalized text-to-image diffusion models without specific tuning*, 2024. arXiv: 2307.04725 [cs.CV].

[32] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," *CoRR*, vol. abs/2012.09841, 2020. arXiv: 2012.09841.

[33] E. J. Hu *et al.*, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, 2022. arXiv: 2112.10752 [cs.CV].

[35] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[36] C. Doersch *et al.*, *Tapir: Tracking any point with per-frame initialization and temporal refinement*, 2023. arXiv: 2306.08637 [cs.CV].

[37] A. Radford *et al.*, *Learning transferable visual models from natural language supervision*, 2021. arXiv: 2103.00020 [cs.CV].

[38] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG].

[39] A. Kirillov *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[40] C.-C. Hsu *et al.*, *Spot: Se(3) pose trajectory diffusion for object-centric manipulation*, 2024. arXiv: 2411.00965 [cs.RO].

[41] S. Patel *et al.*, *A real-to-sim-to-real approach to robotic manipulation with vlm-generated iterative keypoint rewards*, 2025. arXiv: 2502.08643 [cs.RO].

[42] OpenAI *et al.*, *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL].

[43] H. Touvron *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[44] J. Liang *et al.*, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[45] D. Driess *et al.*, "Palm-e: An embodied multimodal language model," in *arXiv preprint arXiv:2303.03378*, 2023.

[46] T. Xie *et al.*, *Text2reward: Reward shaping with language models for reinforcement learning*, 2024. arXiv: 2309.11489 [cs.LG].

[47] Y. J. Ma *et al.*, "Eureka: Human-level reward design via coding large language models," *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[48] Z. Li, K. Yu, S. Cheng, and D. Xu, "LEAGUE++: EMPOWERING CONTINUAL ROBOT LEARNING THROUGH GUIDED SKILL ACQUISITION WITH LARGE LANGUAGE MODELS," in *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

[49] C. Chi *et al.*, *Diffusion policy: Visuomotor policy learning via action diffusion*, 2024. arXiv: 2303.04137 [cs.RO].

[50] J. Fu, K. Luo, and S. Levine, *Learning robust rewards with adversarial inverse reinforcement learning*, 2018. arXiv: 1710.11248 [cs.LG].

[51] J. Ho and S. Ermon, *Generative adversarial imitation learning*, 2016. arXiv: 1606.03476 [cs.LG].

[52] S. Kareer *et al.*, *Egomimic: Scaling imitation learning via egocentric video*, 2024. arXiv: 2410.24221 [cs.RO].

[53] K. Yu, Y. Han, Q. Wang, V. Saxena, D. Xu, and Y. Zhao, "Mimictouch: Leveraging multi-modal human tactile demonstrations for contact-rich manipulation," in *8th Annual Conference on Robot Learning*, 2024.

APPENDIX

A. Object-Centric Flow Generation

Flow Generation Process: Overall, the entire flow generation process is a three-step pipeline: (1) *flow dataset construction*: we convert an offline trajectory video dataset into an object-centric flow dataset by detecting the positions of interactive objects at the initial frame using Grounding-Dino [29], and then uniformly sampling the keypoints in the detected object bounding box utilizing an off-the-shelf tracker [30]. (2) *generative model adaptation*: we adapt the pre-trained diffusion-based video generation architecture, AnimateDiff [31], as our flow generator, conditioned on task descriptions in two fine-tuning stages. First, we utilize the autoencoder of the VQ-GAN [32] and only finetune the decoder of it to better adapt the latent embeddings to the flow; Second, we finetune the LoRA [33] injected into the latent motion module to model the temporal dynamics of object flows. (3) *post-processing of generated flows*: we apply a series of motion filters to ensure the sampled and generated keypoints reside within the object boundaries, making them applicable for reward computation.

B. Flow-Derived Reward Model

δ-Flow Construction: To further reduce the noises in generated object-centric flow (from Sec. II-A), we propose to transform it into the condensed δ-flow representation. It characterizes each timestep information of the original keypoint flow with three primary statistical estimates: the 2D centroid positions of object keypoints $\bar{\mathbf{P}}^t$, between-frame average translation of keypoints $\boldsymbol{\delta}^t_{tr}$, and between-frame average rotary transformation of keypoints $\boldsymbol{\delta}^t_{rot}$ at t-th step. The definitions of these estimates are formulated as:

$$\bar{\mathbf{P}}^{\mathbf{t}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{P}^{\mathbf{t}}_{\mathbf{i}}, \quad \boldsymbol{\delta}^t_{\mathrm{tr}} = \bar{\mathbf{P}}^{\mathbf{t}} - \bar{\mathbf{P}}^{\mathbf{1}}, \tag{1}$$

$$\boldsymbol{\delta}^t_{\mathrm{rot}} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\mathbf{P}^{\mathbf{t}}_{\mathbf{i}} - \bar{\mathbf{P}}^{\mathbf{t}}\right)\odot\left(\mathbf{P}^{\mathbf{1}}_{\mathbf{i}} - \bar{\mathbf{P}}^{\mathbf{1}}\right)\right], \tag{2}$$

where $\mathbf{P}^{\mathbf{t}}_{\mathbf{i}}$ denotes the 2D image coordinates of i-th keypoint on the t-th frame, and both $\boldsymbol{\delta}^t_{\mathrm{tr}}, \boldsymbol{\delta}^t_{\mathrm{rot}}$ are computed with $\bar{\mathbf{P}}^{\mathbf{1}}$, the centroid of the 1^{st} frame. For simplicity, we summarize the object motion at t-th step as $\mathcal{T}^t = (\boldsymbol{\delta}^t_{\mathrm{tr}}; \boldsymbol{\delta}^t_{\mathrm{rot}})$, which we refer to as the δ-flow. Finally, we differentiate between two types of object motion: the robot execution or observational δ-flow, denoted as $\mathcal{T}^t_R$, and the generated δ-flow, as $\mathcal{T}^t_G$.

The δ-flow representation provides a compact yet informative characterization of object 2D motion/dynamics. With its Monte Carlo strategy, it effectively reduces the negative impact of unreliable noisy keypoints predicted by the flow generation module. This enhances the robustness and reliability of reward computation, ultimately improving the efficiency of RL training (evidenced in Sec. III-B).

δ-Flow Matching as Dense Reward: The generated δ-flow, despite its imperfections, can readily serve as an effective continuous prior for object motion, guiding policy learning for coarse movements, which helps to overcome the sparse reward issue. Thus, we aim to design a dense reward with a proposed δ-flow matching strategy, *i.e.*, by quantifying the alignment between the observation flow induced by the online policy and the generated flow prior. *Specifically*, we consider the t-th timestep of generated and robot flows, $\mathcal{T}^t_G$ and $\mathcal{T}^t_R$, as samples drawn from their respective underlying probabilistic distributions, $P_{\mathcal{T}_R}$ and $P_{\mathcal{T}_G}$, which captures the inherent uncertainty present in each stochastic δ-flow trajectory. *Furthermore*, we quantify the alignment between these flows with a distributional distance functional $\mathcal{D}$ that maps two distributions to a scalar. Intuitively, the t-th step of the δ-flow matching reward, R^t_δ, should be inversely proportional to the distributional distance:

$$R^t_\delta \propto -\mathcal{D}\Big(P_{\mathcal{T}^t_R}(\cdot|\mathbf{x}_{1:t}, \mathbf{P}^{\mathbf{1}}, c), P_{\mathcal{T}^t_G}(\cdot|\mathbf{x}_1, \mathbf{P}^1, \phi, c)\Big) \tag{3}$$

where $\mathbf{x}_{1:t}$ denotes the observations up to t timestep. $P_{\mathcal{T}^t_R}$ is assumed to be a Gaussian distribution of the δ-flow with its true mean approximated by $\mathcal{T}^t_R$ using sampled keypoints on objects; while $P_{\mathcal{T}^t_G}$ is also assumed to be a Gaussian modeled by a flow generative model parameterized by ϕ and conditioned on task c.

Our objective is to align two flows by maximizing the per-timestep reward, equivalently, minimizing the distance between the two δ-flow distributions. In practice, for simplicity, we assume tied variance and use the Kullback–Leibler divergence (KLD) as our distance metric $\mathcal{D}$. Consider that the KLD between two Gaussian with tied variance is L_2-norm of mean, under these conditions, the normalized flow-matching reward at the t-th step is formulated as follows:

$$R^t_\delta = 1 - \mathrm{clip}\left(\frac{\left(\mathcal{T}^t_R - \mathcal{T}^{t'}_G\right)^2}{C}, 0, 1\right) \tag{4}$$

where C is a scaling parameter to control the variance of approximation. In practice, to improve the stability of reinforcement learning training, we clip and normalize the δ-flow reward to the interval $[0, 1]$.

Overall Reward Model: In addition to tracking the expert motion prior in the generated δ flow, we propose an object *state-aware reward* derived from the environmental feedback to reshape the sparse reward that is only grounded on task completion. This state-aware reward not only accelerates RL training but also provides the policy task-specific information, making it capable to accomplish the tasks. By incorporating δ-flow and state-aware rewards, our entire reward model is designed as:

$$R^t = \begin{cases} \alpha\cdot(1-\tanh(\tau\cdot d_{\text{grip}})), & \text{if } d_{\text{grip}} > 0, \\ \alpha, & \text{if finish subgoal}, \\ \alpha + \beta\cdot R^t_\delta, & \text{After subgoal}, \\ 1.0, & \text{if completed}. \end{cases} \tag{5}$$

where d_{grip} is the distance between the gripper to the object. In practice, we set $\alpha = 0.25, \beta = 0.75$ and the temperature $\tau = 10$. This task-agnostic design of our reward signal ensures its broad generalizability across diverse tasks.

Representation	RL Compatibility			Geo. Complexity	
	Low-Dim.	Cross-Emb.	Reward.	Deform.	Artic.
Video [10, 16]				✓	✓
Gripper Keypoints [20, 21]	✓		✓	✓	✓
Active Region [2, 22]				✓	✓
Trace [23, 24]	✓			✓	✓
Object Pose [25, 26]	✓	✓	✓		
Object Keypoints [16, 27]	✓	✓	✓		✓
Object Parts [28]		✓			✓
Object Flows [12–14]	✓	✓	✓	✓	✓

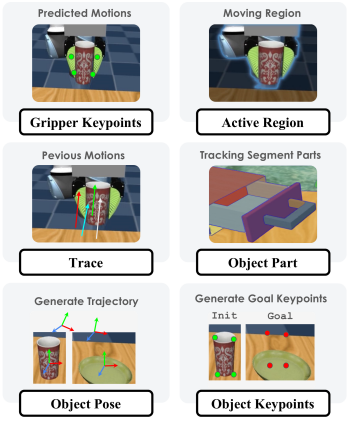


TABLE II: **General comparisons among widely-adopted manipulation-centric representations for Robot Learning**, evaluated on their compatibility with RL and scalability to object geometric complexity. Three key dimensions are considered for *RL compatibility*: (1) the dimensionality of the raw observational representation (Low-Dimensionality); (2) generalizability across different embodiments (Cross-Embodiment); and (3) supporting reward shaping from raw observations (Rewardability). Two key aspects are considered for *geometric complexity*: (1) representability of deformable objects (Deformable); and (2) representability of articulated objects (Articulated). From the comparison, we observe that object-centric flow offers the highest flexibility and representational capacity and being well-suited for RL.

C. Policy Design

Specifically, We define the input space of the policy as: (1) the current robot state $\mathbf{s}_t$, (2) the current centroid positions of object keypoints $\bar{\mathbf{P}}^t$, (3) the current observational object δ-flow during robot execution $\mathcal{T}_R^t$, (4) the k-step lookahead generated keypoint centroid $\bar{\mathbf{P}}_G^{t+1:t+k}$, (5) the k-step lookahead generated δ-flow $\mathcal{T}_G^{t+1:t+k}$, and (6) the 3D centroid positions at the 1^{st} frame $\bar{\mathbf{P}}_{3\text{d}}^1$. Our policy is formulated as:

$$\mathbf{a}_t = \pi\Big(\mathbf{s}_t, \bar{\mathbf{P}}^t, \mathcal{T}_R^t, \bar{\mathbf{P}}_G^{t+1:t+k}, \mathcal{T}_G^{t+1:t+k}, \bar{\mathbf{P}}_{3\text{d}}^1\Big) \quad (6)$$

where the output action $\mathbf{a}_t$ is a 6D pose displacement used for RL exploration and policy learning, which is then transformed by the inverse kinematics (IK) module into joint commands for the robot. During policy training, we optimize the policy by maximizing our flow-derived reward, leveraging the DrQv2 [5] algorithm and replay experience accumulated from interactions.

D. Cross-Embodiment and Cross-Domain Data Collection

To train the flow generation model, we collect a dataset contains $12k$ trajectories three different embodiments in ten different tasks from two different task domains. We aim to utilize different embodiments for highlighting that object-centric flow can be trained with large scale of diverse training data.

In the setting of Im2Flow2Act [12], we utilize their sphere robot dataset as the first kind of cross-embodiment data. Four tasks, which are *PickNPlace*, *Pouring*, *Opening*, and *Folding*, are used for data collection. Also shown in Im2Flow2Act [12], this kind of data are proposed to emulate the cross-embodiment human data in the real world. Out of those tasks, we design a new contact-rich manipulation task *Pivoting* with the UR5 robot for collecting data. To collect robot data, we place the robot in different initial positions

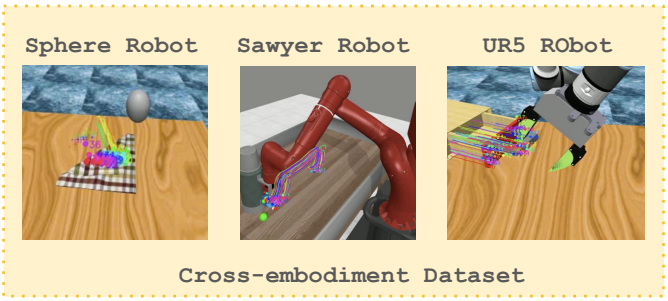


Fig. 5: **Visualization of cross-embodiment data.**

and use a manipulation script to move it to five different contact points. Then, we apply five different action scripts to enable the robot to stand the peg up and make contact with the wall. For data in the MetaWorld [17], the task setting and robot used for data collection was totally different from the Im2Flow2Act[12]. We choose five different tasks, which are *Assembly*, *Coffee Push*, *Door Close*, *Lever Pull*, and *Stick Pull*. Those data are collected by the Saywer Robot. In this benchmark, we rollout the trained RL model in MetaWorld [17] for data collection. Each task contain 1200 trajectories, where we have 12k training data in total. The visualization of each embodiment are shown in Fig. 5.

E. Implementation and Task Settings

Implementation Details: For our flow generative model (Sec. II-A), we conduct two-stage fine-tuning for the pre-trained StableDiffusion-1.5 [34] on a cross-embodiment dataset with 12k trajectories from ten different tasks across three different embodiments: sphere robot, UR5, and Sawyar. See our Supp. for more dataset and training details.

Tasks Settings: We construct our benchmark from: MetaWorld [17], Im2Flow2Act [12], and our implementation. We sample four tasks—*PickNPlace*, *Pouring*, *Opening*, and *Folding* from Im2Flow2Act. Additionally, we construct a

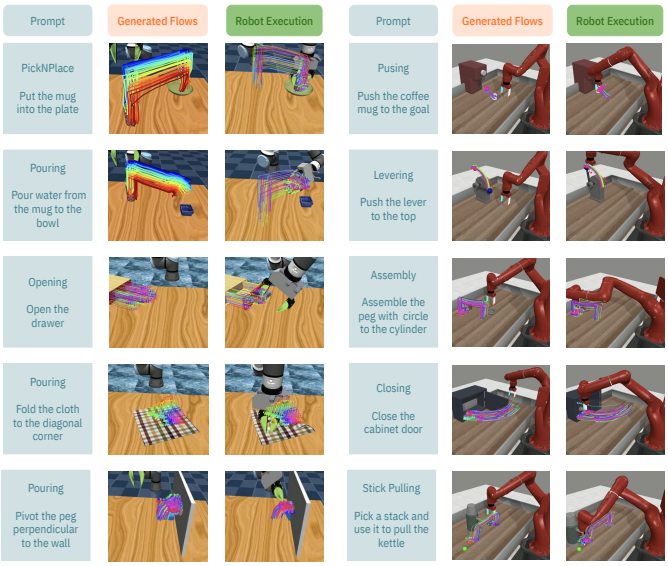


Fig. 6: **Tasks Overview** Overview of the tasks setting and robot execution process. **Left** is the task in Im2Flow2Act [12] and **Right** is the task in MetaWorld [17]. The elements of each row are the input prompt, generated flow, and robot execution, respectively.

novel contact-rich task, *Pivoting*, within its environment. These tasks involve contact-rich and deformable object manipulation and thus are tailored to ablation with flow-based imitation learning (in Sec. III-A). For MetaWorld, we sample five challenging tasks for evaluation—*Assembly*, *Close Door*, *Coffee Push*, *Lever Pull*, and *Stick Pull*. All these tasks are designed based on Mujoco Engine [35]. An UR5e robot and an Sawyar robot are used to collect data and do online exploration during RL training process, respectively. Task details are presented in Fig. 6.

For more details, descriptions of those four tasks from Im2Flow2Act [12], five tasks from MetaWorld [17], and one self-designed task *pivoting* are shown below:

- *PickNPlace:* Pick a mug from one random position to the bowl in another random position.
- *Pouring:* Pick a mug from one random position and pour the water to a bowl in another random position.
- *Opening:* Grasp the handle and open the carbinet.
- *Folding:* Fold the cloth from one corner to another corner.
- *Pivoting:* Contact with the peg without grasping, and pivot it to stand up by interacting with the wall.
- *Coffee Push:* Push the coffee mug to a specific position.
- *Door Close:* Close the door of a carbinet.
- *Assembly:* Pick up a stick and align its square hole with a peg on the table.
- *Lever Pull:* Grasp the lever and pull it up to the up right position.
- *Stick Pull:* Pick up a stick, insert it into a kettle, and pull the kettle to a specific position.

F. Flow Generation Implementation Details

In this section, we aim to share more details about implementation, training, and processing of our flow generation model, which is similar to Im2Flow2Act [12].

G. Implementations

The first step is getting the training dataset. We use Grounding DINO [29] to detect the bounding box of the described object from the initial RGB frame, and then uniformly sample keypoints within the box. To track those keypoints from the video, we apply the SOTA keypoint tracking foundation model [30] for keypoint tracking. Unlike the TAPIR [36] used in previous work [12], CoTracker [30] can track occluded objects in the image, which is extremely important for contact-rich manipulation tasks. Then, we formulate the tracked keypoints as object-centric flow $\mathcal{F}_0 \in \mathbb{R}^{3\times T\times H\times W}$ with temporal representation in T time space. The first two channels represent the pixel coordinates of object keypoints in image space, while the third represents their visibility during the execution.

The generated flow is conditioned on the initial image of the task, the initial keypoints, and the text description. The encoder deisgn for the inputs are also the same as the [12]. The text descriptions are processed into the CIIP [37] to obtain text embeddings. For the initial image, we utilize the CLIP encoder to get the patch embeddings. The initial keypoints are encoded through fixed 2D sinusoidal positional encoding. Finally, those inputs are processed into the denosing process through cross-attention.

With this flow representation, we can leverage the diffusion-based video generation based on AnimateDiff [18] for flow generation. Same as the Im2FLow2Act [12], we encode the object flow into a latent space and train the generative model based on it. Similar to the StableDiffusion [34], we use the auto encoder VA-GAN [32] to encode the flow into low dimentional embeddings. Then, to utilize the low-dimensional latent space, we utilize a two-stage training process. Firstly, we fix the encoder from the AE and finetune the pretrained decoder to better adapt it to the flow images. Then, same as the Im2Flow2Act, we insert the motion module layer into StableDiffusion proposed by Animatediff [31] to model the temporal dynamics for flow generation. The second stage is training the motion module layer from scratch but only insert LoRA (Low-Rank Adaptation) layers [33] into the SD model.

1) Training Details: Training with cross-embodiment data from two different task domains, we process both the image from the tasks in Im2Flow2Act [12] and MetaWorld [17] into resolutions 480×480. For extracting keypoints from the bounding box generated by Grounding Dino [29], we set the spatial and temporal resolution to H = W = 32 and T = 32 for generating flow for 1024 keypoints over 100 steps, which also means that 100 keypoints set can be used for reward shaping in our reward model. To train the model on our cross-embodiment dataset, we firstly train the decoder of VQ-GAN [32] in StableDiffusion [34] for 400 epoches with a learning rate of $5e-5$. Secondly, for training

AnimateDiff, we insert the LoRA [33] with a rank of 128 into the Unet from StableDiffusion and train the motion module layer from scratch with the same hyperparameter shown in Im2Flow2Act [12], which is trained with learning rate of $1e-4$ for 300 epochs using Adamw [38] optimizer with weight deacy $1e-2$ betas $(0.9, 0.999)$, and epsilon $1e-8$.

2) Flow Post Processing: We use motion filter to extract moving keypoints from the object itself. Similar to Im2Flow2Act [12], we use moving filter to remove those static keypoints and use SAM [39] filter to remove keypoints which are not on the object, such as keypoints on the robot or the table.

Moving Filter: Since some of the keypoints selected from bounding box are sampled from the environment, we use the moving filter to extract the moving points from those keypoints. Then, we use moving filter to remove those keypoints whose movement in the image space (480×480) is below a certain threshold. For all the tasks, we select 50 pixels as the threshold for removing those static keypoints. This method can effectively remove those background keypoints.

SAM Filter [39]: Since we also use robot data in our dataset, those keypoints on the robot are also will be counted as moving keypoints. Then, using SAM [39] to do semantic segmentation and remove those moving keypoints on the robot is necessary. We utilize SAM to obtain the segmentation and then iterate through the keypoints, filtering out those whose corresponding segment area exceeds a predefined threshold. To preserve keypoints on objects with rich textures, we set a high threshold of 10,000 across all tasks.

Finally, we randomly sampled 128 points from the selected keypoints for our reward model and policy input.

H. RL Implementation Details

1) Reward and Policy Input with Generated Flow: For both RL training in Im2Flow2Act [12] and MetaWorld [17], we firstly need to generate the initial flow from the first frame for each iteration, which will be used for both policy input and reward shaping. Then, the observation space of the policy input is 128×3, where we sampled 128 keypoints from the object.

To do online flow matching with the generated flow, we also utilize CoTracker [30] for online tracking in the real-robot execution. Using the same motion filters, 128 keypoints will be sampled from the real-time motion. The online keypoint tracking will be generated for each timestep during the robot execution. Then, those keypoints will be processed as δ-flow for reward shaping.

Since our flow generation model will generate 100 frames with keypoints subset, we can use each centroid calculated from the keypoints subset of those 100 frames as reward. For reward calculation and policy input. We limit the $max\_step\_episode$ into number less or equal to 100 steps (respectively for different tasks) and use them for real-time flow matching in reward generation and policy learning.

a) RL Reward Design: Out of the flow-derived reward, our RL training Pipeline also require specific reward design for achieveing the goal.

Reaching Reward: For all the tasks, we need to define similar reaching reward to guide to robot move toward the object. For tasks which required robot to grasp the object, robot needs to open their gripper and reach the grasping position. For the tasks which required robot to push or contact with the object, the robot will be guided to move to contact with a certain area. The reward will be define as: $(1 - \tanh(10.0 \cdot d_{\text{grip}}))$.

Grasping / Contact Reward: We also design sparse reward as a subgoal for guiding robot to accomplish the task. For grasping task, we will set the reward to be 0.25 as the reward. For contact reward, we will set the reward to be 0.25 once the gripper is contact with a certain area of the object.

Goal-conditioned Reward: For all the task, we need to define a goal state to showcase that the robot successfully acchieve the goal. For most of the task, it should be easily, such as those defined task in MetaWorld [17] and some simple task like *PickNPlace*. We can just define the final position for object to be. For some other harder-to-define tasks like pouring, we set up a target pose range as the goal, which is limited to a certain position with certain orientations, where the orientation is sampled from $(\frac{5\pi}{16}\frac{7\pi}{16})$. For opening, the reward will be defined by the opening distance, which is $0.1m$.

b) Training Details: The Training hyperper parameters have been shown in Table. III

TABLE III: Hyperparameters for DrQv2 with Flow-derived Reward.

Hyperparameter	**Value**
Environment	
Action repeat	3 (MetaWorld) 3 (Im2Flow2Act)
Frame stack	1
Rendered Image	480×480
Observation size	128×3
Reward type	Sparse
DrQv2	
Data Augmentation	± 4 RandomShift
Replay buffer capacity	10^6
Discount γ	0.99
n-step returns	3
Seed frames	4000
Exploration steps	2000
Exploration stddev. clip	0.3
Exploration stddev. schedule	Linear(1.0, 0.1, 3×10^6)
Soft update rate	0.01
Optimizer	Adam
Batch size	256
Update frequency	2
Learning rate	10^{-4}

I. How Does the Choice of Object-Centric Representation Affect Performance?

In this section, we aim to identify the most effective object-centric representation for reward shaping. We evaluate the training success rate of our framework using three alternative object-centric representations across three tasks

in Im2Flow2Act [12]. Figure 7 shows a comparison between our framework and the baselines.

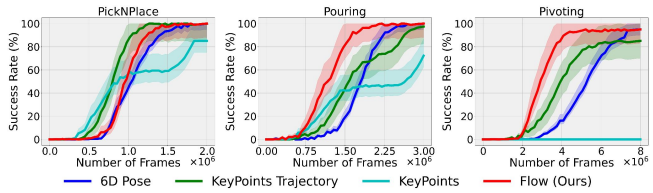


Fig. 7: **Representation Evaluations.** The comparison of RL training results with different object-centric representations in three tasks. The shaded area represents the standard deviation for three random seeds.

Baselines: We compare the object-centric flow with three other representations: `(1)` *6D object poses [25, 26, 40]*, which uses the object's 6D pose trajectory for reward shaping. `(2)` *3D keypoints [27, 41]*, which defines keypoints relative to the object's 6D pose and uses their initial and goal positions for reward shaping. `(3)` *3D keypoints trajectory*, which extends keypoint-based reward shaping by incorporating keypoint trajectories instead of just initial and goal positions. We exclude certain manipulation-centric representations from Table II due to their unsuitability for reward computation or incompatibility with cross-embodiment datasets.

Evaluation: To evaluate different object-centric representations, we use ground-truth demonstrations as policy input. We select *PickNPlace*, *Pouring*, and *Pivoting* as the tasks for evaluation since the other non fine-grained representations cannot handle deformable object manipulation tasks like cloth folding. Notably, the 3D keypoints [27, 41], defined relative to the object's 6D pose, are unsuitable for deformable or articulated object manipulation.

Key Findings: Beyond enabling deformable and articulated objects manipulation, GENFLOWRL achieves competitive or superior performance across all evaluated tasks, particularly excelling in the contact-rich *Pivoting* task.

3D keypoints, which capture only initial and final positions without temporal information, struggle in *Pivoting* as they provide limited expert guidance. In contrast, representations with temporal features, *e.g.*, *6D poses* and *3D keypoint trajectories*, perform better by preserving motion dynamics. While *3D keypoint trajectories* can accelerate early training due to their 3D representation, they often face challenges when used directly as inputs, particularly in tasks involving orientation changes, such as *Pouring* and *Pivoting*. These limitations underscore the advantages of δ-flow and our policy input design (Sec. II-C). Specifically, our δ-flow extracts spatiotemporal manipulation-centric features from low-dimensional representations, making it especially beneficial for complex tasks with more dynamic motions.

J. Ablation Study

We compare the performance of GENFLOWRL with three model variants on three tasks in Im2Flow2Act (*PickNPlace*, *Pouring*, and *Pivoting*): `(1)` *MLP*: Instead of δ-flow, MLP is used to encode tracked keypoints into low dimensional space as the input. `(2)` *w/o 3D Initial Centroids [15]*: Removeing initial 3D centroids from GENFLOWRL. `(3)` *64 keypoints*: Randomly sampling 64 keypoints instead of 128 keypoints. All other settings are the same as GENFLOWRL, differing only in input representations. The results are shown in Fig. 8.

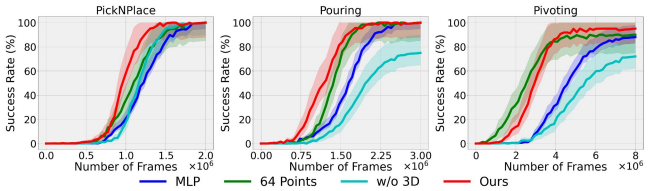


Fig. 8: **Ablation Evaluations.** The comparison of RL training results with ablations of GENFLOWRL in three tasks. The shaded area represents the standard deviation for three random seeds.

These experiments provide three key insights: `(a)` Compared to MLP, utilizing δ-flow and future generated δ-flow as inputs better captures the object's spatiotemporal dynamics, leading to improved performance. `(b)` Utilizing initial 3D keypoints as the input can produce improvement since it is helpful for learning structured 6D actions from 2D keypoint transformations. `(c)` Since we use the processed flow to randomly sample keypoints and compute their centroid, the number of keypoints does not have a significant impact.

K. Ablations on Tracking Module

To further investigate the effects of the tracking module on our GENFLOWRL, we conduct the following ablations.

1) Different Off-the-shelf Tracker: We compare the quality of tracked points from two recent state-of-the-art trackers, *i.e.*, CoTracker V3 [30] and Tapir [36]. As shown in the qualitative results (Fig. 9), CoTracker demonstrates superior temporal consistency, greater robustness to occlusions and contacts, and significantly fewer jittering artifacts. Therefore, we adopt CoTracker V3 [30] to better align with the demands of contact-rich manipulation.

2) Failure Case Analysis: We conduct an additional case study of CoTracker on four challenging real-world tasks: `Pivoting`, `Folding`, `Insertion`, and `Twisting`, featuring high occlusions and dynamic motions. As shwon in Fig. 10, we identify two primary failure modes: `(1)` In the `Insertion` task, when half of the object becomes occluded by the hole, the tracker, lacking prior physical knowledge, misinterprets it as a deformation and shifts the corresponding keypoints to the visible part. `(2)` In the `Twisting` task, the tracker fails to capture the dynamic rotation of round, glossy lids due to the lack of discriminative visual features, a challenge even for human observers.

L. Sensitivity Analysis on Noises in Generated Flows

In real-world scenarios, the flow trajectory predicted by the generator network or the tracker can be biased by noise. To investigate the effect of these noises on policy performance, we simulate real-world noises on our method. Specifically, we build a noise model to cumulatively perturb trajectories and deviate endpoints, using the composition of

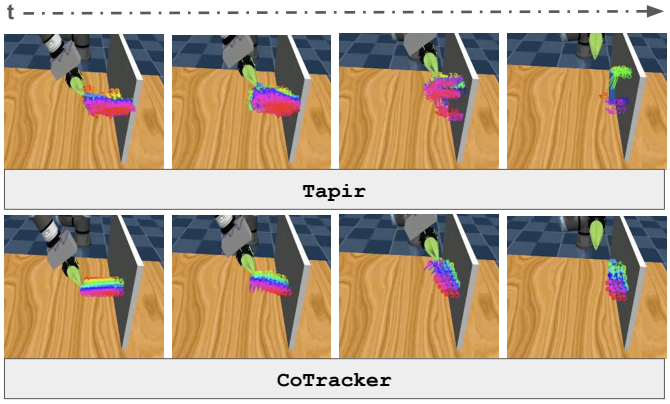


Fig. 9: Visualization of the comparison of trackers.

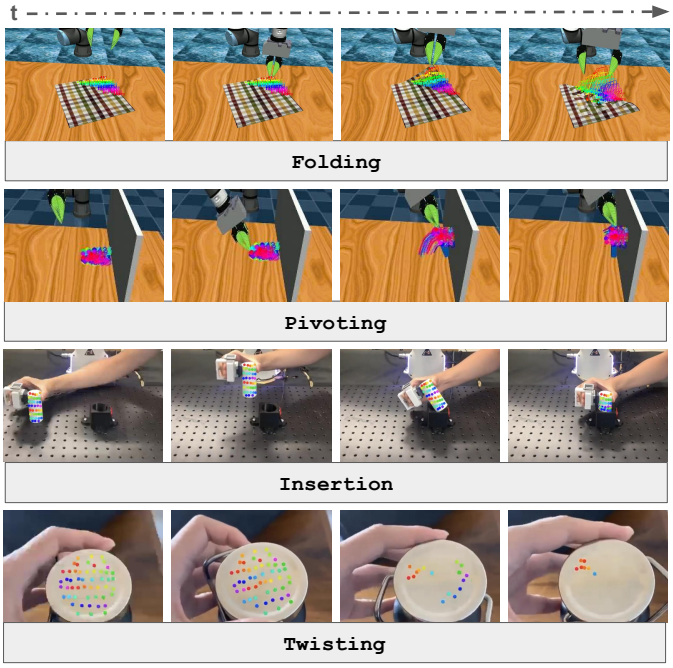


Fig. 10: Visualization of the case study of the CoTracker.

Brownian motion (Gaussian random walk) and Brownian bridge, with separate controllable standard deviations. We set up four types of random noise: small Gaussian (gauss=1x, drift=0x), large Gaussian (gauss=4x, drift=0x), small drift (gauss=2x, drift=1x), and large drift (gauss=2x, drift=2x). A vivid visualization of our noise composite model applied to a Bessel smoothed trajectory is displayed in Fig. 11.

We evaluate the performance of our model after applying this noise model to the generated flow in five challenging tasks. The result is shown in Table IV. Since our model is already trained on generated flows with different magnitudes of noise and due to our task-oriented rewards, it has the robustness to noise in the generated flows. Therefore, our method still achieves a similar performance with trajectory noise, especially for the cases with large Gaussian noises. Furthermore, our method can maintain relatively high performance when the goal position is largely drifted from the ground-truth by ≥ 20 pixels in the tasks with position-sensitive evaluation, *e.g.*, `Folding` and `Pouring` (Fig. 11).

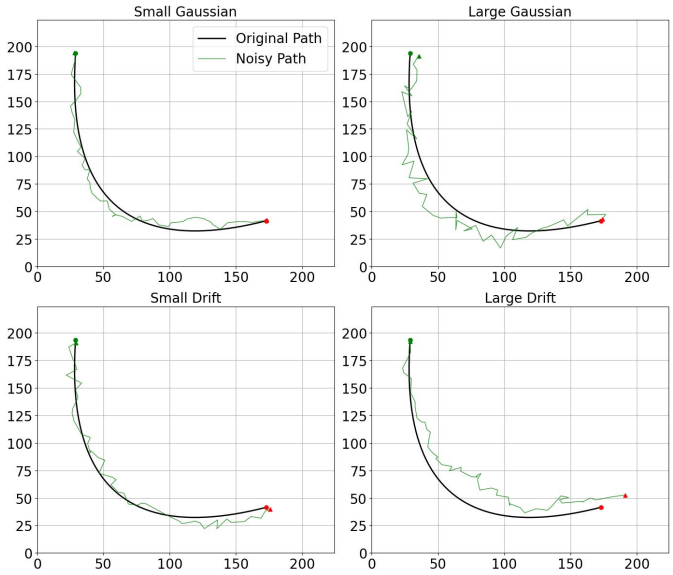


Fig. 11: Visualization of the simulated noised 2D trajectory.

	PickNP.	Pour	Open	Fold	Pivot
GenFlowRL	95	95	95	80	85
+Gauss×1 Drift×0	95	95	90	80	85
+Gauss×4 Drift×0	95	90	90	75	80
+Gauss×2 Drift×1	95	90	90	70	85
+Gauss×2 Drift×2	85	75	85	65	75

TABLE IV: Performance for noise sensitivity analysis on five tasks. Noises are added to flow trajectories for comparisons.

M. Failure Cases Analysis

In this section, we analyze the failure cases of the trained policies. In `Coffee Push` task, generated flows occasionally diverge from the intended path before reaching the goal. In `Stick Pull` task, flow generation often fails after contact with the bottle, leading to premature task termination. These defect examples are visualized in Fig. 12. We argue that the primary limitation stems from the dataset contamination by some defective demonstrations with noisy policies. Besides, occlusions of the tiny object may further lead to tracking difficulties. Therefore, collecting higher-quality training data is essential to mitigate these failure cases.

N. Real World Case Study Details

In this section, we set collect actions from robot actions script with a limit for 200 steps and 100 steps for *Folding* and *Pouring* respectively. The robot control frequency and the CoTracker frequency are both 2.5 Hz. For collecting human demonstrations, the frequency of the CoTracker is 5 Hz, and the final number of steps are the same as the robot demo. Then, we calculate the flow matching reward between them based on it.

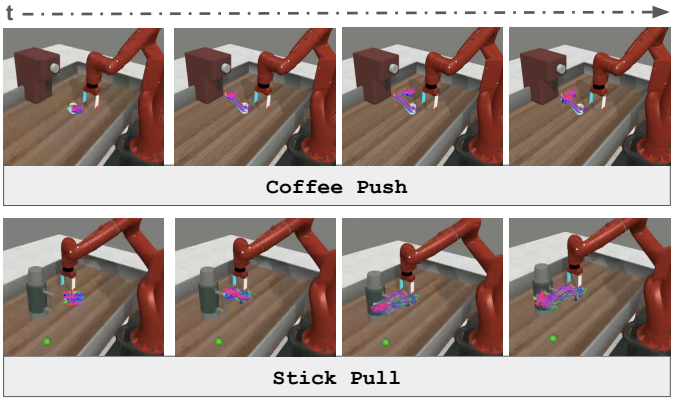


Fig. 12: Visualization of failed generated flows.

marized in Table II.

O. Related Work

Generative Models for Robot Learning. Extensive work has explored the utility of generative models for robot learning, including LLMs [42, 43], VLMs [37, 42], and Diffusion [31]. These methods have been widely used in planning [44, 45], reward generation [46–48], and policy learning [49]. More recently, advancements in foundational video generative models, *e.g.*, Stable Video Diffusion [34], have paved the way for leveraging generated videos for learning universal robot policies [1–4]. However, these methods are often limited by the uncontrollable quality of generated videos and their reliance on open-loop policies. Instead, we integrate generated object-centric flow with RL, enhancing policy robustness through active interaction with the environment.

Video-based Reinforcement Learning. Designing task-specific dense reward in reinforcement learning often requires huge amount of human efforts. To overcome this challenges, inverse RL [50, 51], reward generation framework [46, 47], and video-based RL [8, 10, 11, 16] are proposed. Recent advancements in video-based RL mainly utilize video prediciton models for shaping dense reward [8, 10, 16]. However, these methods often struggle in capturing manipulation-centric features directly from the generated video due to the inherent uncertainty in video generation. Unlike them, our work proposes to use low-dimensional spatiotemporal object-centric flow instead of video, offering a more structured and efficient approach to reward shaping.

Manipulation-Centric Representations. Manipulation-centric representations can be roughly categorized into *gripper-centric representations* such as keypoints [21], trace [23], or active region [2, 22] from the gripper, and *object-centric representations*, such as object keypoints [27, 41], 6D object poses [25, 26, 40], and object parts [28]. Despite their promising results for robot learning, gripper-centric representations rely on real robot demonstrations, making them hard to utilize easy-to-collect human hand demonstrations [12, 52, 53], while most object-centric representations still struggle with deformable and articulated object manipulation due to their limited granularity as sum-

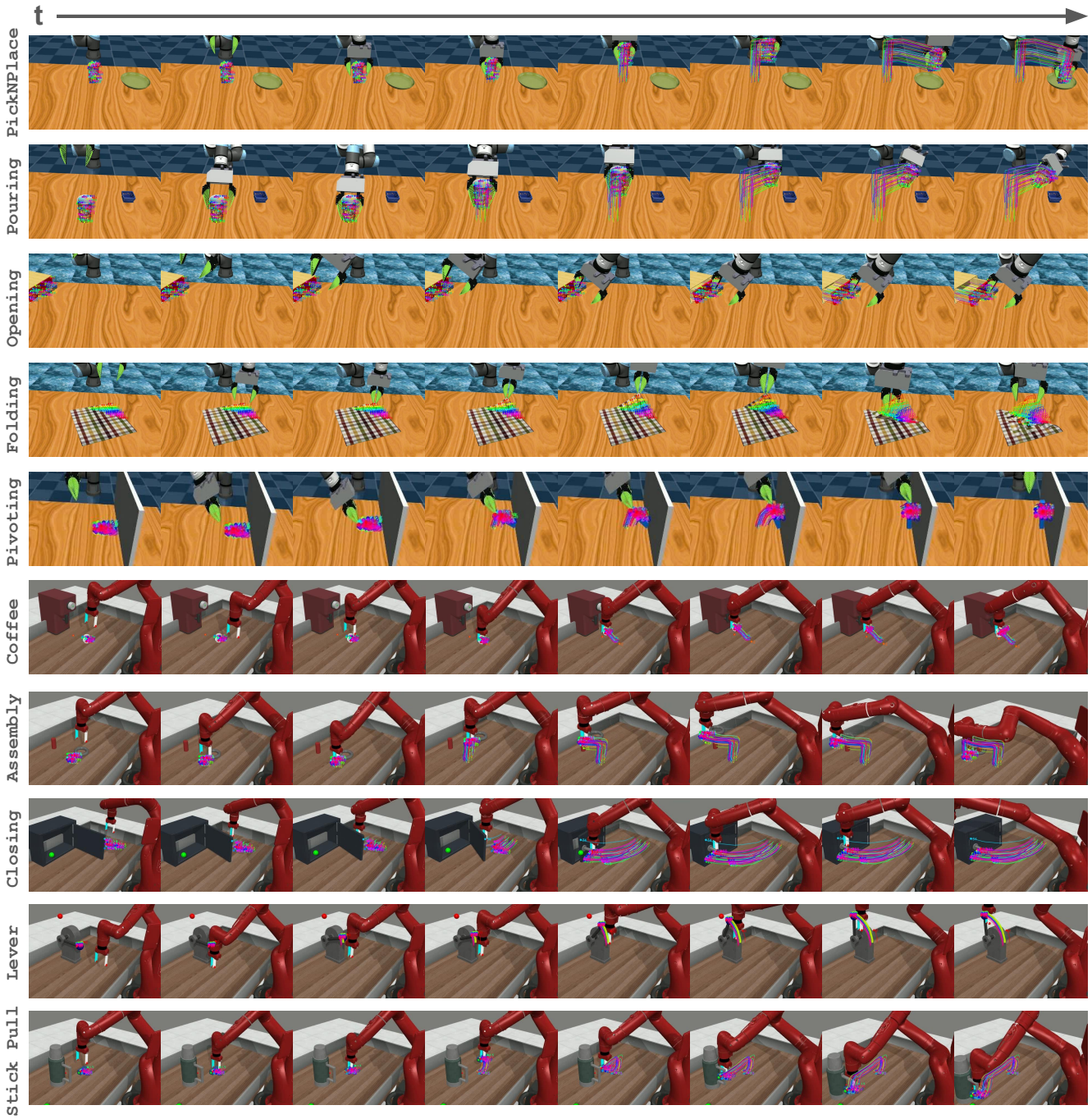


Fig. 13: The qualitative result of the policy rollout in simulatior.

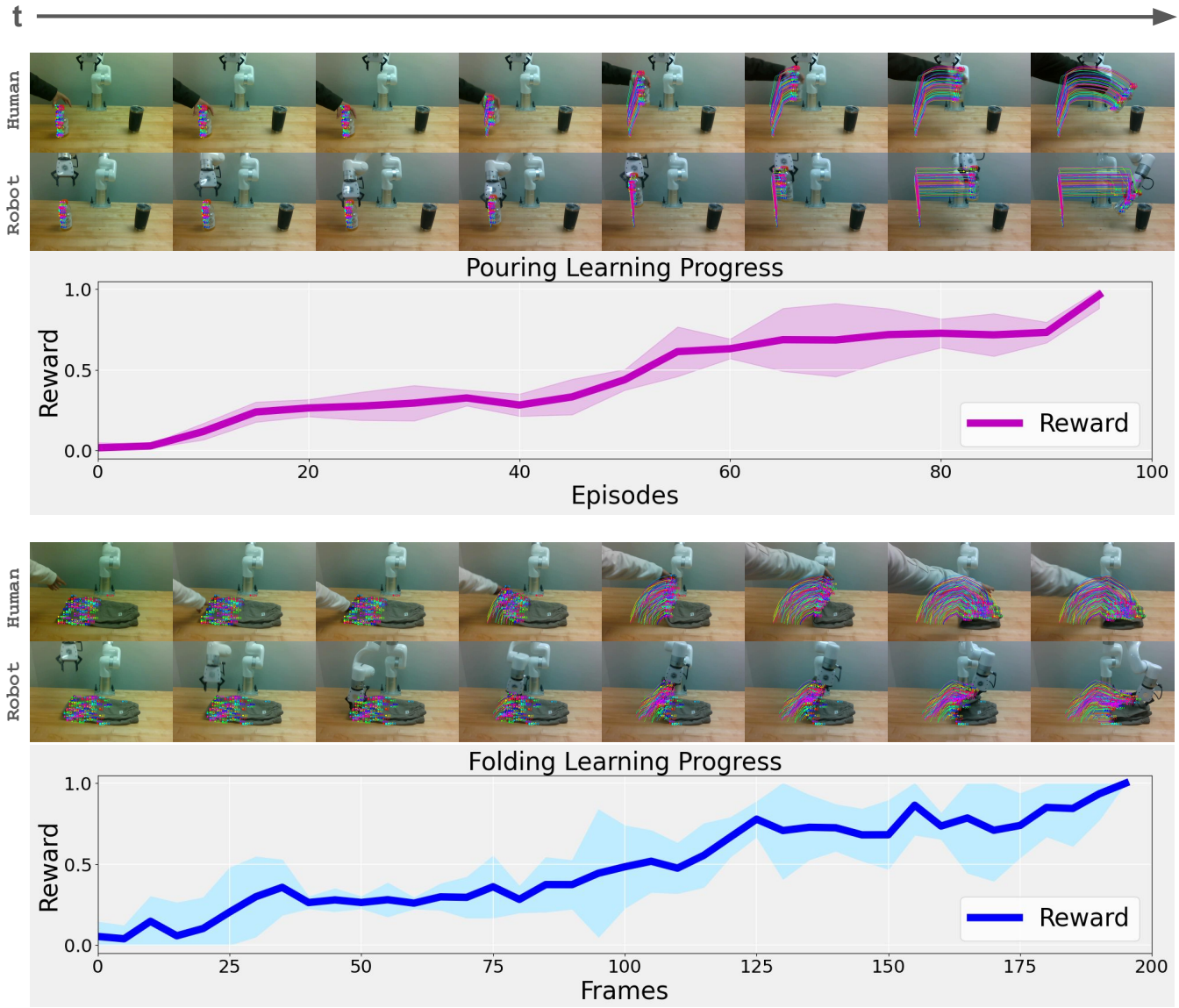


Fig. 14: The qualitative result of the Flow Matching Reward Case Study in the real world.