ACTIONFLOW: Efficient, Accurate, and Fast Policies with Spatially Symmetric Flow Matching

Niklas Funk[1], Julen Urain[1,2], Joao Carvalho[1], Vignesh Prasad[1], Georgia Chalvatzaki[1,3], and Jan Peters[1,2,3,4]
[1]Computer Science Dept., TU Darmstadt, Germany [2]German Research Center for AI (DFKI), Research Department: SAIROL, Darmstadt, Germany [3]Hessian.AI, Darmstadt, Germany [4]Centre for Cognitive Science, TU Darmstadt
Email: niklas@robot-learning.de

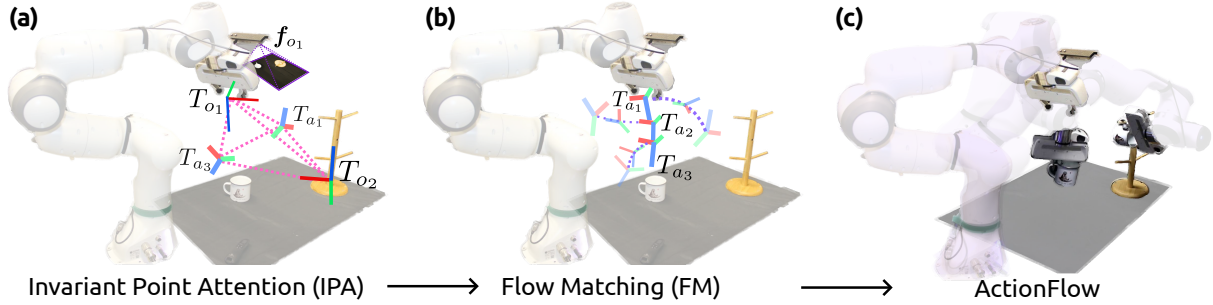

Fig. 1: **Overview of ActionFlow.** ActionFlow represents both observations & actions in one common space and describes every token by its pose T and features $\boldsymbol{f}$. **(a)** The scene consists of two observations ($\boldsymbol{f}_{o1,o2}, T_{o1,o2}$), and the action sequence is initialized by generating random samples, i.e.,($\boldsymbol{f}_{a1\ldots a3}, T_{a1\ldots a3}$). Based on the scene representation, an Invariant Point Attention (IPA)-based transformer computes the attention between the different tokens considering their relative SE(3) poses, thereby exploiting the spatial symmetries of the task. **(b)** The output of this model defines a flow in the action space. The flow results in refining the action sequence to obtain local trajectories for fulfilling the task. **(c)** When using the procedure of scene encoding & action sequence generation through Flow Matching iteratively, i.e., as a policy, we can generate highly accurate, SE(3) equivariant action sequences at low inference times.

***Abstract*—Most robotic tasks require a proper understanding of the scene's 3D geometry. Despite the impressive results of deep generative models in complex manipulation tasks, the lack of a representation that encodes the intricate spatial relations between observations and actions usually leads to small spatial generalization capabilities, requiring large amounts of demonstrations. To tackle this problem, we introduce a novel policy class, ActionFlow. ActionFlow integrates spatial symmetry inductive biases while generating expressive action sequences. On the representation level, ActionFlow introduces an SE(3) Invariant Transformer architecture, which enables informed spatial reasoning based on the relative SE(3) poses between observations and actions. For action generation, ActionFlow leverages Flow Matching, a novel, state-of-the-art deep generative model that has been shown to generate high-quality samples with fast inference, an essential property for feedback control. In combination, ActionFlow policies exhibit strong spatial and locality biases and SE(3) equivariant action generation. Our experiments demonstrate the effectiveness of ActionFlow and its two main components on several simulated and real-world robotic manipulation tasks and confirm that we can obtain efficient, accurate, and fast policies with spatially symmetric flow matching. Project website: https://flowbasedpolicies.github.io/**

I. INTRODUCTION

Recently, we have observed impressive results in using deep generative models for solving complex manipulation tasks [48, 8, 3, 46]. However, it is well known that models that naively integrate observations and actions usually require copious amounts of demonstrations to solve the task properly. In this direction, there has been a collection of research that explored how to exploit the spatial relations between observations and actions [45, 12, 13, 40, 33, 18] to learn more sample efficient policies. **Equivariant** policies generalize the policy's behavior under global scene translations or rotations [41, 12, 45, 29, 16, 11, 34, 38]. If the observations are rotated or translated, the generated actions will be equally transformed, thereby adding an effective inductive bias.

Herein, we are not only interested in equivariant policies but also in **local spatial relations** [4, 9, 12]. Consider, for example, the task of picking & hanging a mug (cf. Fig. 1). When the robot is approaching to pick up the mug, it should be capable of reasoning based on the relative poses between its own pose, the mug, and its next actions. But when hanging the mug, the robot should also focus on the relative poses between mug & hanger. Thus, equipping the policy with the ability to reason based on the relative poses between the different observations and actions, i.e., their spatial relations, might be essential to learning policies efficiently. How can we integrate all these desiderata and still learn **dexterous, fast, and expressive policies from demonstrations**?

Inspired by recent successes in the protein folding community [21, 42, 43, 20, 17], in which SE(3) symmetric models are integrated with highly-expressive deep generative models, we introduce **ActionFlow**, a novel policy class for robotics, suitable for learning dexterous manipulation skills while integrating geometric notions for sample efficient learning. In essence, ActionFlow is composed of two main elements: **(1)** a state-of-the-art [10] highly-expressive deep generative model (Flow Matching) [24, 6, 5] that has been shown capable of generating high-quality samples within very small inference times, and, **(2)** an SE(3) Invariant Transformer network [21, 43] that frames a *relative positional encoding* [21, 26] based on

the tokens' relative SE(3) poses (Fig. 1). Combining those components results in several interesting properties that make ActionFlow an appealing candidate for learning robot policies and, in particular, robotic manipulation from demonstrations:

Fast and accurate action sequence generation. Given the underlying Flow Matching generative model, we can generate precise action trajectories with low inference times [25, 10].

SE(3) equivariant action generation. ActionFlow inherently preserves the tasks' spatial structure and naturally adapts the actions. Given translated or rotated observations, the actions are equally transformed, thereby providing SE(3) equivariance. While ActionFlow's underlying transformer network is invariant, we achieve SE(3) equivariance by applying the flow matching updates w.r.t. the actions' local frame [21, 43].

Relative Pose Aware Attention. The SE(3) Invariant Transformer allows the actions to attend to the different observation tokens based on their relative poses. This allows finding correlations based on spatial relative information and enhances the generalization to scenes where objects are arranged differently.

In summary, our main contributions are: **(a)** we investigate Flow Matching for fast and precise robotic action generation, **(b)** we introduce an SE(3) Invariant Transformer architecture for geometry-aware robot learning. Our experiments in simulated and real robot environments underline the effectiveness of both components and showcase that their combination, i.e., our proposed ActionFlow, yields accurate and fast manipulation policies while showcasing sample efficiency.

II. ACTIONFLOW

ActionFlow represents observations $\boldsymbol{O} : (\boldsymbol{T}_o, \boldsymbol{F}_o)$ and actions $\boldsymbol{A} : (\boldsymbol{T}_a, \boldsymbol{F}_a)$ with a sequence of poses $\boldsymbol{T}{=}(T^1, \dots, T^N)$ and features $\boldsymbol{F}{=}(\boldsymbol{f}^1, \dots, \boldsymbol{f}^N)$ (cf. Fig. 1). The pose $T{=}(\boldsymbol{r}, \boldsymbol{p}){\in}SE(3)$ consists of a rotation matrix $\boldsymbol{r}{\in}SO(3)$ and a 3D position vector $\boldsymbol{p}{\in}\mathbb{R}^3$. The associated feature $\boldsymbol{f}{\in}\mathbb{R}^d$ represents semantic information related to the respective pose. This representation is pretty flexible. For observations, a pose might represent a camera location while the features represent the (encoded) image observation (cf. Fig. 1(a)). Alternatively, a pose might represent an object's location while its features represent semantic information describing it (color, shape, ...). For actions, the pose represents the desired target pose, while the features might represent when this target pose should be reached. Given this representation, we aim to learn a policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{T}_a|\boldsymbol{O})$ that generates action pose sequences $\boldsymbol{T}_a$ given an observation $\boldsymbol{O}$. The desiderata for this policy is to be fast, accurate, and expressive, while being sample efficient by capturing the spatial relations between observations & actions, ultimately resulting in SE(3) equivariant action generation. To this end, ActionFlow is built on two elements: a Flow Matching based generative model for generating action poses and a transformer model that represents the relative positional encoding based on the tokens' relative SE(3) poses.

A. Flow Matching on the Action Space

Flow Matching permits the learning of expressive generative models with fast inference [25, 10]. Similarly to diffusion models [37, 15], sample generation is done by iteratively calling a learned model. Herein, we apply Flow Matching to generate action pose sequences. We consider an action space represented with a sequence of N SE(3) poses $\boldsymbol{T}_a{=}(T_a^1, \dots, T_a^N){\in}SE(3)^N$. Thus, similarly to [42], we adapt Flow Matching to the Lie Group SE(3). For a background on Flow Matching, see App. A. Without loss of generality, we derive the solution for a single SE(3) action pose $T{\in}SE(3)$by adapting a common Flow Matching method (Rectified Linear Flow [25, 10, 6]) to the Lie Group SE(3). We define a decoupled flow between the rotation and the translation, allowing us to represent the distribution path and the vector fields independently.

SE(3)-Rectified Linear Flow. Rectified Linear Flow proposes representing the data point conditioned flow $\boldsymbol{\phi}_t(\boldsymbol{a}|\boldsymbol{a}_1)$ with a straight line from a noisy sample $\boldsymbol{a}_0{\sim}\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ at $t{=}0$ to the datapoint $\boldsymbol{a}_1{\in}\mathcal{D}$ at $t{=}1$. In our case, we move an initially randomly sampled action pose T_0 towards an action pose from the dataset T_1 by defining a straight line path for the translation $\boldsymbol{p}$ and the rotation $\boldsymbol{r}$. The flow $T_t = \boldsymbol{\phi}_t(T_0|T_1)$ that moves a noisy initial sample $(\boldsymbol{p}_0, \boldsymbol{r}_0)$ to a pose sampled from the dataset, i.e., $(\boldsymbol{p}_1, \boldsymbol{r}_1) \sim \mathcal{D}$, is represented by

$$\text{Translation} \quad \boldsymbol{p}_t = \boldsymbol{\phi}_t(\boldsymbol{p}_0|\boldsymbol{p}_1) = t\boldsymbol{p}_1 + (1-t)\boldsymbol{p}_0 \tag{1}$$

$$\text{Rotation} \quad \boldsymbol{r}_t = \boldsymbol{\phi}_t(\boldsymbol{r}_0|\boldsymbol{r}_1) = \boldsymbol{r}_0 \text{Exp}\left(t\text{Log}(\boldsymbol{r}_0^{-1}\boldsymbol{r}_1)\right), \tag{2}$$

with Log and Exp, the logarithmic map and the exponential map for the SO(3) manifold [35]. Equation (2) represents a path through the geodesic on SO(3) from $\boldsymbol{r}_0$ to $\boldsymbol{r}_1$.

Given the flow is decoupled, the vector field $\boldsymbol{u}_t = \mathrm{d}\boldsymbol{\phi}_t/\mathrm{d}t$ is also decoupled. In particular, the translation & rotation velocity (i.e., $\dot{\boldsymbol{p}}_t{\in}\mathbb{R}^3$ & $\dot{\boldsymbol{r}}_t{\in}\mathbb{R}^3$) equate to $\dot{\boldsymbol{p}}_t{=}\boldsymbol{r}_t^\intercal(\boldsymbol{p}_t - \boldsymbol{p}_1)/(1-t)$ & $\dot{\boldsymbol{r}}_t = (\text{Log}(\boldsymbol{r}_t^{-1}\boldsymbol{r}_1))/(1-t)$. Notice that even if rotations are represented in $\boldsymbol{r} \in SO(3)$, the velocity vector for the rotations $\dot{\boldsymbol{r}}_t \in \mathbb{R}^3$ is a 3D vector (axis-angle representation) represented in the tangent space centered around $\boldsymbol{r}_t$. Also, the translation velocity is premultiplied with the transpose of the current rotation $\boldsymbol{r}_t^\intercal$ as we aim to represent the velocity in the action frame.

Training. Our parameterized model $(\boldsymbol{v}_{\boldsymbol{p}}, \boldsymbol{v}_{\boldsymbol{r}}) = \boldsymbol{v}_{\boldsymbol{\theta}}(T, \boldsymbol{O}, t)$ outputs both a translation vector $\boldsymbol{v}_{\boldsymbol{p}} \in \mathbb{R}^3$ and a rotation vector $\boldsymbol{v}_{\boldsymbol{r}} \in \mathbb{R}^3$. Given a dataset $\mathcal{D} : \{T^i, \boldsymbol{O}^i\}_{i=0}^I$, the training objective is to minimize the mean-squared error between the model outputs $(\boldsymbol{v}_{\boldsymbol{p}}, \boldsymbol{v}_{\boldsymbol{r}})$ and the velocity vectors $\boldsymbol{u}_t = (\dot{\boldsymbol{p}}_t, \dot{\boldsymbol{r}}_t)$.

Sampling in SE(3). To generate an action pose $T{=}(\boldsymbol{p}, \boldsymbol{r})$, we sample a rotation $\boldsymbol{r}_0{\sim}\mathcal{U}(SO(3))$ and translation $\boldsymbol{p}_0{\sim}\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and iteratively run Euler-discretization for K inference steps, i.e., $\boldsymbol{p}_{k+1}{=}\boldsymbol{p}_k{+}\boldsymbol{r}_k\boldsymbol{v}_{\boldsymbol{\theta}}(T_k, \boldsymbol{O}, t)\Delta t$ & $\boldsymbol{r}_{k+1}{=}\boldsymbol{r}_k\text{Exp}(\Delta t\boldsymbol{v}_{\boldsymbol{\theta}}(T_k, \boldsymbol{O}, t))$, with $\Delta t = 1/K$ and $t = k\Delta t$.

B. SE(3) Invariant Transformer

ActionFlow's network architecture is an SE(3) Invariant Transformer (cf. Fig. 2). The main element of the proposed architecture is a geometry-aware attention layer known as Invariant Point Attention (IPA) [21, 43]. The IPA layer augments the queries, keys, and values of classical attention [39] with a set of 3D points that are generated in the local frames of

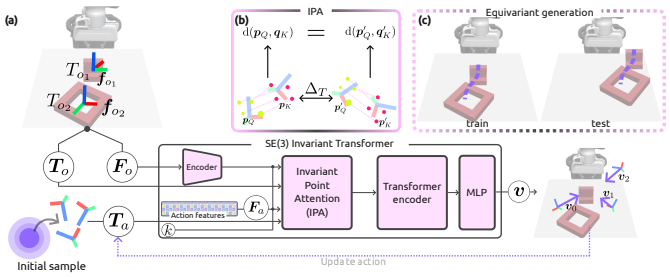

Fig. 2: **Spatial Symmetries in ActionFlow**. **(a)**, A visual representation of the SE(3) Invariant Transformer. Given a set of observations of the scene $\boldsymbol{F_o}$, a set of poses $\boldsymbol{T}_o$ and candidate actions $\boldsymbol{T}_a$, the SE(3) Invariant Transformer predicts a vector $\boldsymbol{v}$ to update the actions (cf. Section II-A). To refine the actions, we iteratively call our model K times. **(b)** IPA augments the classical attention with points $\boldsymbol{p}_Q$ and $\boldsymbol{p}_K$ generated in the local frames of the query and key poses. The layer is designed to generate the same output under global SE(3) transformations $\Delta_T \in SE(3)$. **(c)** ActionFlow is SE(3) equivariant. If we apply a transformation over the observation poses, the generated actions will be equally transformed.

the query $\boldsymbol{T}_Q$ and key $\boldsymbol{T}_K$ poses. The layer is designed such that the output is invariant to global rotations and translations (cf. Fig. 2 (b)). *This relative attention makes the network both invariant and object-centric.* If we apply a transformation $\Delta_T \in SE(3)$ over both observation poses $\boldsymbol{T}'_o{=}\Delta_T\boldsymbol{T}_o$ and action poses $\boldsymbol{T}'_a{=}\Delta_T\boldsymbol{T}_a$, the network generates the same output. Additionally, given the IPA layer, the network can reason considering all the relative poses between all the entities in the scene. We hypothesize that given the invariant and object-centric nature of the network, we can achieve more data-efficient policies.

Network Architecture. Given the observation $\boldsymbol{O}{=}(\boldsymbol{T}_o, \boldsymbol{F}_o)$ and a candidate action sequence $\boldsymbol{T}_a{\in}SE(3)^N$ of length N, the SE(3) Invariant Transformer outputs vectors $\boldsymbol{v}{\in}\mathbb{R}^{6\times N}$ that predict the direction in which the actions $\boldsymbol{T}_a$ should be updated. The network architecture is inspired by protein folding network FrameDiff [43, 42]. Given a set of poses $\boldsymbol{T}$ and features $\boldsymbol{F}$, the network first applies an IPA layer to capture the spatial relative attention between the different entities, followed by a transformer encoder to find higher-order interactions. We use a small linear layer to map the transformer output to the vector $\boldsymbol{v}$. Notice that prior to the IPA, we employ an observation encoder that maps all observation features into a common latent feature space. The action features $\boldsymbol{F}_A$ are given by a learnable parameter vector.

Action Generation starts with a randomly sampled action sequence $\boldsymbol{T}_a$ and iteratively updates the actions by calling the SE(3) invariant transformer K times. Given the invariant network and the action updates within their current local frame, the resulting policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{T}_a|\boldsymbol{T}_o, \boldsymbol{F}_o)$ is equivariant. If we apply a transformation over $\boldsymbol{T}_o$, the generated actions $\boldsymbol{T}_a$ are guaranteed to be equally transformed (see Fig. 2 (c)).

III. Experimental Results

The experiments are split into two parts. First, we evaluate the impact of the SE(3) Invariant Transformer, particularly whether IPA helps for learning policies in a data-efficient manner. Second, we evaluate ActionFlow for real robot manipulation. Additionally, in App. B, we explore the performance of Flow Matching to generate high-quality samples within few inference steps.

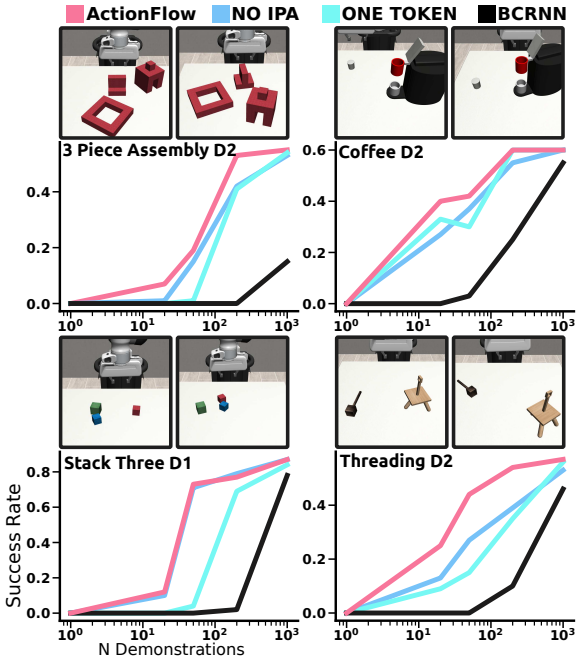

Fig. 3: Success rate of models trained on different number of demonstrations $(20, 50, 200, 1000)$ on a subset of Mimicgen environments [28]. We report the average success rate on 50 test environments for the last 10 checkpoints of a single seed. The top row shows two randomly sampled initial configurations for each task.

A. SE(3) Invariant Transformer Evaluation: Multi-Token Observations and Invariant Point Attention

We evaluate the proposed SE(3) Invariant Transformer (cf. Section II-B) w.r.t. two design choices:
(**1**) Does a **multi-token representation**, in which each object is treated as a single token, enhance policy performance?
(**2**) Does the **IPA** layer, which allows computing the relative poses of all the tokens among each other, help in finding informative features to improve policy performance?

Dataset & Evaluation Environment. The experiments are conducted in a subset of Mimicgen environments [28]. The datasets consist of 1000 synthetically generated demonstrations, given 10 original demonstrations.

Models. We consider three variations of ActionFlow for evaluation: (**A.1**) We eliminate the IPA layer and represent all observations as a single token. (**A.2**) We eliminate the IPA layer but keep each object as an independent token. (**A.3**) The original ActionFlow as introduced in Section II-B. All the models consider 5 inference steps. Additionally, we consider as baseline (**B**) a RNN-GMM model as introduced in [27, 28].

Results. We train the models with different amounts of demonstrations $(20, 50, 200, 1000)$ and evaluate their performance in 50 randomly sampled test environments. The results in Fig. 3 reveal that the original ActionFlow consistently outperforms the variations and baselines across the tasks,

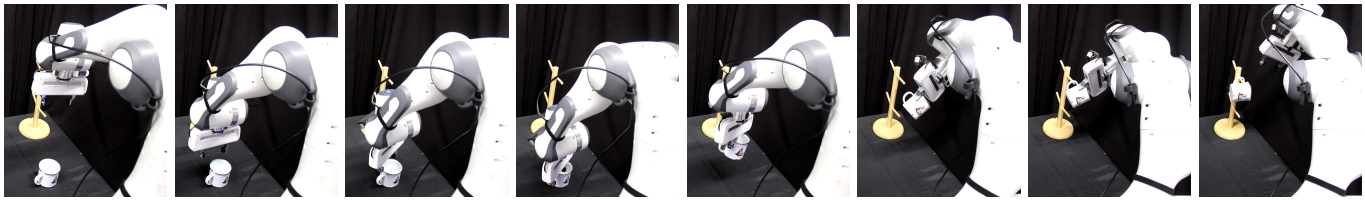

Fig. 4: ActionFlow policy rollout on the mug hanging task, in which the hanger is placed in a test configuration (cf. Fig. 5).

Initialization	ActionFlow Success Rate
Train	9/10
Test	8/10

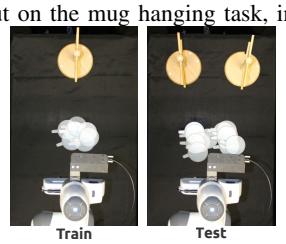

Fig. 5: **Left:** Success Rates on train and test configurations. **Right:** Visualization of the initialization configurations for training and testing evaluations.

specifically in low demonstration regimes. This indicates that IPA is beneficial for learning policies in a sample-efficient way, while with large datasets, all models appear to converge to similar performances. We also observe that representing the observations with multiple tokens shows performance benefits compared to representing the whole observation as a single token. Finally, all ActionFlow variations outperform the baseline in different data percentages. We hypothesize that this performance increase could be directly related to the expressivity of Flow Matching with respect to GMM.

B. Real Robot Experiments

We evaluate ActionFlow's equivariance on the real robotic task of placing a mug onto a hanger. Another experiment evaluating ActionFlow's accuracy is provided in App. C.

Setup. The experimental platform consists of a 7DoF Franka Panda manipulator with a RealSense D435 mounted at its end-effector (cf. Fig. 1). We employ a token for the hanger's pose and the robot's end-effector $(T, \boldsymbol{f})$. The end-effector is described by its pose T and the features $\boldsymbol{f}$ contain the encoded RGB images (using a ResNet18 [14]) and the gripper's opening width.

Results. For policy training, we collect 200 demonstrations using variations as shown in Fig. 5 - right. Notably, the demonstrations only include slight variations of the mug poses, while the hanger always stays in the same pose. The results are presented in Fig. 5. The table's first row reveals that the ActionFlow policy achieves high success rates of 90% upon evaluating in similar scenarios as those encountered during training. We only observe one failure in which the mug is not grasped properly. Importantly, our policies run online in real-time as action generation only takes $0.03\,\mathrm{s}$ on an NVIDIA RTX 3090 GPU. We also evaluate the policy in previously unseen test scenarios (cf. Fig. 5), where the hanger is moved to either side of the table. The results show that our policy can still handle these novel test scenarios well, achieving 80% success. Fig. 4 shows a policy rollout in one of the testing scenarios. These high success rates, despite the previously unseen scenarios, demonstrate the equivariance property of our proposed ActionFlow, which inherently can handle these translated scene instances.

IV. Related Work

Spatial Symmetries on Robot Learning. A large line of research [45, 12, 18, 22, 34, 38, 13, 31] explored designing methods that exploit the spatial symmetries between observations and actions to alleviate the data requirements of learning from demonstrations. A set of works have proposed *projecting the action to the visual space* [45, 32, 13, 23, 40] representing the actions in the pixel space. Closer to our approach, a set of works propose *representing both observations and actions in the 3D space* [30, 12, 31, 33, 41, 34, 38, 44, 22]. In particular, [12, 22] propose using spatial relative attention to allow the policy to reason based on the relative 3D positions between observations and actions. Our work extends the attention to SE(3) poses allowing the agent to reason based on both relative position and orientation.

Flow Matching for Decision Making. Despite the recent emergence of flow matching methods [24, 25, 5] for learning deep generative models, there has been a wide set of fields in which they have been applied from image generation [10] to protein backbone generation [42, 1]. In the context of decision making, Zheng et al. [47] introduce guided flow matching to condition the flow-based models on arbitrary contexts. Then, they apply the conditioned flow-based models in Offline RL setups, similarly to [19]. Concurrent to this work, Braun et al. [2] introduce Riemannian Flow Matching Policies that propose learning a flow-based model to generate trajectories in arbitrary Riemannian manifolds. Their work showed promising results on the LASA dataset. In this work, we focus on robotic manipulation and the combination of Flow Matching with an SE(3) Invariant Transformer architecture, enabling SE(3) equivariant action generation with successful application to real robot experiments on challenging manipulation tasks.

V. Conclusion

This paper presented ActionFlow, a new policy class for robot learning from demonstrations. On the representation level, ActionFlow consists of an SE(3) Invariant Transformer equipped with geometry-aware Invariant Point Attention. Actions are generated using Flow Matching, a new generative model capable of obtaining high-quality samples with low inference times. The resulting policies are fast, represent both actions and observations in one common space, and yield SE(3) equivariant action generation. Our experiments underline the effectiveness of ActionFlow's individual components and demonstrate its capabilities for solving real robotic manipulation tasks.

References

[1] Avishek Joey Bose, Tara Akhound-Sadegh, Kilian Fatras, Guillaume Huguet, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael Bronstein, and Alexander Tong. Se (3)-stochastic flow matching for protein backbone generation. *arXiv preprint arXiv:2310.02391*, 2023.

[2] Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. *arXiv preprint arXiv:2403.10672*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[4] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent service robotics*, 9:1–29, 2016.

[5] Ricky TQ Chen and Yaron Lipman. Riemannian flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

[6] Ricky TQ Chen, Meta FAIR, and Yaron Lipman. Flow matching on general geometries.

[7] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[8] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[9] Christian RG Dreher, Mirko Wächter, and Tamim Asfour. Learning object-action relations from bimanual human demonstration using graph networks. *IEEE Robotics and Automation Letters*, 5(1):187–194, 2019.

[10] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024.

[11] Chongkai Gao, Zhengrong Xue, Shuying Deng, Tianhai Liang, Siqi Yang, Lin Shao, and Huazhe Xu. Riemann: Near real-time se (3)-equivariant robot manipulation without point cloud segmentation. *arXiv preprint arXiv:2403.19460*, 2024.

[12] Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: Infinite resolution action detection transformer for robotic manipulation. *arXiv preprint arXiv:2306.17817*, 2023.

[13] Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[16] Haojie Huang, Dian Wang, Arsh Tangri, Robin Walters, and Robert Platt. Leveraging symmetries in pick and place. *The International Journal of Robotics Research*, page 02783649231225775, 2024.

[17] Guillaume Huguet, James Vuckovic, Kilian Fatras, Eric Thibodeau-Laufer, Pablo Lemos, Riashat Islam, Cheng-Hao Liu, Jarrid Rector-Brooks, Tara Akhound-Sadegh, Michael Bronstein, et al. Sequence-augmented se (3)-flow matching for conditional protein backbone generation. *arXiv preprint arXiv:2405.20313*, 2024.

[18] Stephen James, Kentaro Wada, Tristan Laidlow, and Andrew J Davison. Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13739–13748, 2022.

[19] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[20] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles. *arXiv preprint arXiv:2402.04845*, 2024.

[21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[22] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[23] Yen-Chen Lin, Pete Florence, Andy Zeng, Jonathan T Barron, Yilun Du, Wei-Chiu Ma, Anthony Simeonov, Alberto Rodriguez Garcia, and Phillip Isola. Mira: Mental imagery for robotic affordances. In *Conference on Robot Learning*, pages 1916–1927. PMLR, 2023.

[24] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[25] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[26] Antoine Liutkus, Ondřej Cıfka, Shih-Lun Wu, Umut Simsekli, Yi-Hsuan Yang, and Gael Richard. Relative positional encoding for transformers with linear complexity.

In *International Conference on Machine Learning*, pages 7067–7079. PMLR, 2021.

[27] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, pages 1678–1690. PMLR, 2022.

[28] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

[29] Hyunwoo Ryu, Jiwoo Kim, Junwoo Chang, Hyun Seok Ahn, Joohwan Seo, Taehan Kim, Jongeun Choi, and Roberto Horowitz. Diffusion-edfs: Bi-equivariant denoising generative modeling on se (3) for visual robotic manipulation. *arXiv preprint arXiv:2309.02685*, 2023.

[30] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *arXiv preprint arXiv:2210.05663*, 2022.

[31] William Shen, Ge Yang, Alan Yu, Jansen Wong, Leslie Pack Kaelbling, and Phillip Isola. Distilled feature fields enable few-shot language-guided manipulation. *arXiv preprint arXiv:2308.07931*, 2023.

[32] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pages 894–906. PMLR, 2022.

[33] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[34] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[35] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[36] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=St1giarCHLP.

[37] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[38] Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5923–5930. IEEE, 2023.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[40] Vitalis Vosylius, Younggyo Seo, Jafar Uruç, and Stephen James. Render and diffuse: Aligning image and action spaces for diffusion-based behaviour cloning. *arXiv preprint arXiv:2405.18196*, 2024.

[41] Jingyun Yang, Congyue Deng, Jimmy Wu, Rika Antonova, Leonidas Guibas, and Jeannette Bohg. Equivact: Sim (3)-equivariant visuomotor policies beyond rigid object manipulation. *arXiv preprint arXiv:2310.16050*, 2023.

[42] Jason Yim, Andrew Campbell, Andrew YK Foong, Michael Gastegger, José Jiménez-Luna, Sarah Lewis, Victor Garcia Satorras, Bastiaan S Veeling, Regina Barzilay, Tommi Jaakkola, et al. Fast protein backbone generation with se (3) flow matching. *arXiv preprint arXiv:2310.05297*, 2023.

[43] Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

[44] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[45] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747. PMLR, 2021.

[46] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[47] Qinqing Zheng, Matt Le, Neta Shaul, Yaron Lipman, Aditya Grover, and Ricky TQ Chen. Guided flows for generative modeling and decision making. *arXiv preprint arXiv:2311.13443*, 2023.

[48] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1199–1210. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/zhu23a.html.

APPENDIX A
BACKGROUND - FLOW MATCHING

Let us consider a data point $\boldsymbol{a} \in \mathbb{R}^d$ and a probability path $\rho_t(\boldsymbol{a})$ that connects a noise distribution $\rho_0(\boldsymbol{a})$ at $t{=}0$ to the data distribution $\rho_1(\boldsymbol{a})$ at time $t{=}1$ with its associated flow $\boldsymbol{a}_t = \boldsymbol{\phi}_t(\boldsymbol{a}_0)$, which defines the motion for the particle $\boldsymbol{a}_0$, Flow Matching [24] proposes learning Continuous Normalizing Flows (CNF) [7] by regressing the vector field $\boldsymbol{u}_t(\boldsymbol{a}) = \mathrm{d}\boldsymbol{\phi}(\boldsymbol{a})/\mathrm{d}t$ with a parametric one $\boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{a}, t)$. Unfortunately, in general, there is no closed-form solution for $\boldsymbol{u}_t$ that generates ρ_t, making the direct flow matching untractable.
Instead, Conditional Flow Matching (CFM) proposes an efficient approach to learn CNF by regressing a conditioned vector field $\boldsymbol{u}_t(\boldsymbol{a}|\boldsymbol{z})$ that generates the probability path $\rho_t(\boldsymbol{a}|\boldsymbol{z})$

$$\mathcal{L}_{\text{CFM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\rho_t(\boldsymbol{a}|\boldsymbol{z}),\rho_{\mathcal{D}}(\boldsymbol{z})} \left\| \boldsymbol{v}_{\boldsymbol{\theta}}(\boldsymbol{a}, t) - \boldsymbol{u}_t(\boldsymbol{a}|\boldsymbol{z}) \right\|^2, \quad (3)$$

with $\rho_{\mathcal{D}}(\boldsymbol{z})$ being the data distribution. As shown in [24], $\boldsymbol{v}_{\boldsymbol{\theta}}$ recovers the marginalized conditioned vector field $\boldsymbol{u}_t = \int_{\boldsymbol{z}} \boldsymbol{u}_t(\boldsymbol{a}|\boldsymbol{z})\rho_t(\boldsymbol{a}|\boldsymbol{z})\rho_{\mathcal{D}}(\boldsymbol{z})/\rho_t(\boldsymbol{a})\mathrm{d}\boldsymbol{z}$ that generates the marginalized distribution path $\rho_t = \int_{\boldsymbol{z}} \rho_t(\boldsymbol{a}|\boldsymbol{z})\rho_{\mathcal{D}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$. Then, the problem boils down to designing a conditioned vector field $\boldsymbol{u}_t(\boldsymbol{a}|\boldsymbol{z})$ that moves a randomly sampled point at time $t = 0$ to the datapoint $\boldsymbol{z}$ at time $t = 1$.

APPENDIX B
FAST AND ACCURATE ACTION SEQUENCE GENERATION IN SIMULATION

We compare Flow Matching against Diffusion Policy [8] for action sequence generation in the simulated Robomimic tasks [27]. For the comparison, we replace the diffusion process with flow matching to estimate the action distribution conditioned on the current observations. To ensure a fair comparison, for both methods, we use the transformer architecture from [8] to model the (observation-conditioned) vector field in Equation (3) or the denoising network. Moreover, we consider Flow Matching in the Euclidean space and use the same hyperparameters from the Diffusion Policy in Flow Matching. Both policies are trained with for 4000 epochs with $K = 100$ time steps. Checkpoints are evaluated every 200 episodes. For testing, we pick the best-performing checkpoint during training and report the average success rate from policy rollouts starting from 50 different initial configurations (from the test set) across 3 training seeds. During inference, it is desirable to use fewer steps than during training since it enables higher-frequency policies. In Flow Matching, reduced inference time steps are obtained by interpolating the training time steps, while for Diffusion Policy we use DDIM [36] for faster sampling.

Results. Fig. 6 shows the results. We depict how the success rate varies with the number of inference steps $\{2, 5, 10, 20, 100\}$. As both methods use the same underlying transformer network, their inference times are similar, and the resulting frequencies for the tested number of steps are $\{100, 33, 20, 9, 2\}$ Hz, respectively. The results show that for most environments and number of inference steps, Flow Matching and diffusion perform almost equally. However, Flow Matching results surpass those from Diffusion Policy for very small inference steps. This effect is most noticeable in the Tool Hang task, which requires the policy to produce very accurate actions. These experiments show that Flow Matching can achieve good success rates using only 2 inference steps (allowing inference at $100\,\mathrm{Hz}$) while Diffusion Policy results degrade.

APPENDIX C
EVALUATING ACCURATE TRAJECTORY GENERATION ON THE REAL ROBOT

Lightbulb Mounting - Evaluating Accurate Action Sequence Generation. This task of mounting a lightbulb (cf. Fig. 7) consists of two main phases: first, the lightbulb and its two pins have to be precisely inserted into the socket. Second, the bulb has to be rotated to tighten it and turn it on. For this task, we collected 100 demonstrations using our teleoperation interface based on OptiTrack.
We consider two baselines. The first one consists of replaying 10 randomly selected demonstrations. The second one is replaying 10 randomly selected demonstrations with an additional offset of $1.5\,\mathrm{mm}$ sampled in a random direction. The results in Fig. 7 show that replaying 10 randomly selected demonstrations results in 8 out of 10 successful executions. Adding a small perturbation of $1.5\,\mathrm{mm}$ reduces the number of successes to only 5. These findings underline that this task requires high accuracy and precise trajectory execution, as slight perturbations significantly reduce the success rate. Despite the tight tolerances and initializing in perturbed states, our learned ActionFlow policy, which is only conditioned on the input of the RGB camera, achieves a success rate of 80%. Two successful policy rollouts are presented in Fig. 7. This experiment underlines that ActionFlow is capable of generating highly accurate trajectories suitable for solving delicate manipulation tasks.

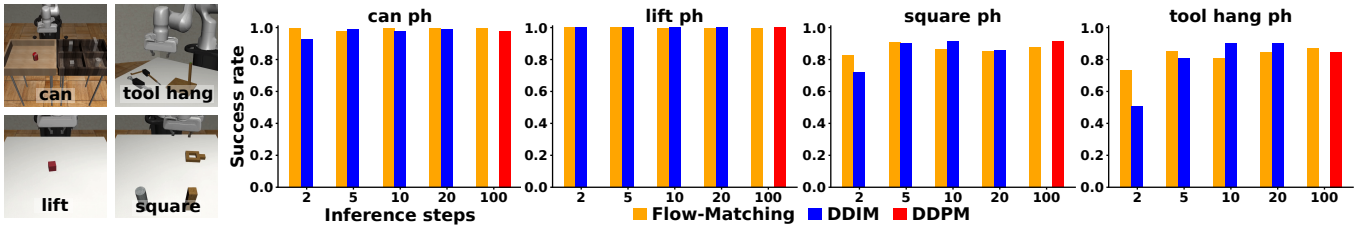

Fig. 6: **Results in Robomimic tasks with state-based observations**. Success rate evaluation on Robomimic tasks with state-based observations averaged over 3 seeds and 50 environments initializations. We evaluate flow-matching and diffusion policy [8] (using DDIM) with different inference steps. For 100 inference steps, the same as training time steps, the diffusion policy is run using DDPM.

Method	**Success Rate**
Replay Demonstrations	8/10
Perturbed Demonstrations Replay	5/10
ActionFlow (Ours)	8/10

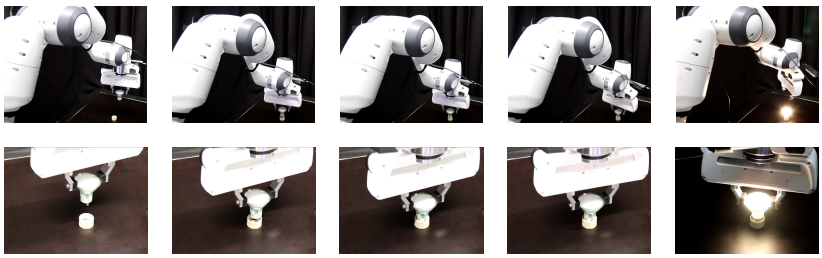

Fig. 7: Real robot light bulb experiments. We report the performance of our model and two baselines, i.e., replaying 10 randomly selected demonstrations and replaying the demonstrations with a position offset in one randomly selected direction of $1.5\,\mathrm{mm}$. The table and the pictures from successful rollouts of our learned action flow policies on the right illustrate that our proposed method can generate highly accurate trajectories.