Do You Know Where Your Camera Is? View-Invariant Policy Learning with Camera Conditioning

Tianchong Jiang[1], Jingtian Ji[1], Xiangshan Tan[1], Jiading Fang[2,*], Anand Bhattad[3], Vitor Guizilini[4,†], Matthew R. Walter[1,†]

***Abstract*— We study view-invariant imitation learning by explicitly conditioning policies on camera extrinsics. Using Plücker embeddings of per-pixel rays, we show that conditioning on extrinsics significantly improves generalization across viewpoints for standard behavior cloning policies, including ACT, Diffusion Policy, and SmolVLA. To evaluate policy robustness under realistic viewpoint shifts, we introduce six manipulation tasks in robosuite and ManiSkill that pair "*fixed*" and "*randomized*" scene variants, decoupling background cues from camera pose. Our analysis reveals that policies without extrinsics often infer camera pose using visual cues from static backgrounds in *fixed* scenes. This shortcut collapses when workspace geometry or camera placement shifts. Conditioning on extrinsics restores performance and yields robust RGB-only control without depth. We release the tasks, demonstrations, and code to facilitate reproducibility and future research. Code and project materials are available at ripl.github.io/know_your_camera.**

I. Introduction

Significant recent attention has been devoted to imitation learning as a means of enabling robots to perform diverse and complex manipulation tasks [1, 2, 3, 4]. These policies usually directly operate on RGB images and are trained with fixed third-person viewpoints. However, when deployed in the real world, it is often not possible to position the cameras to exactly match their training pose due to environmental constraints. Consequently, many such models fail when testing on different viewpoints. The problem is exacerbated when the robot's embodiment changes, which inherently involves diverse camera configurations and viewpoints, making viewpoint invariance essential for cross-embodiment transfer [5].

Concurrently, accurate camera geometry is increasingly accessible. Such information can be obtained directly from the datasets [6] used for training, or estimated through classic methods such as hand-eye calibration [7], visual-SLAM and structure-from-motion methods [8, 9, 10], or modern learning-based methods that handle sparse or dynamic views [11, 12]. Neglecting this readily available geometric information is a significant missed opportunity for improving robot learning.

To achieve viewpoint invariance, a policy must be aware of the camera's geometry relative to the robot. This raises a fundamental question: should the end-to-end policy *implicitly* infer camera geometries during training, or should the robot policy model *explicitly* condition on camera geometries when they are readily available?

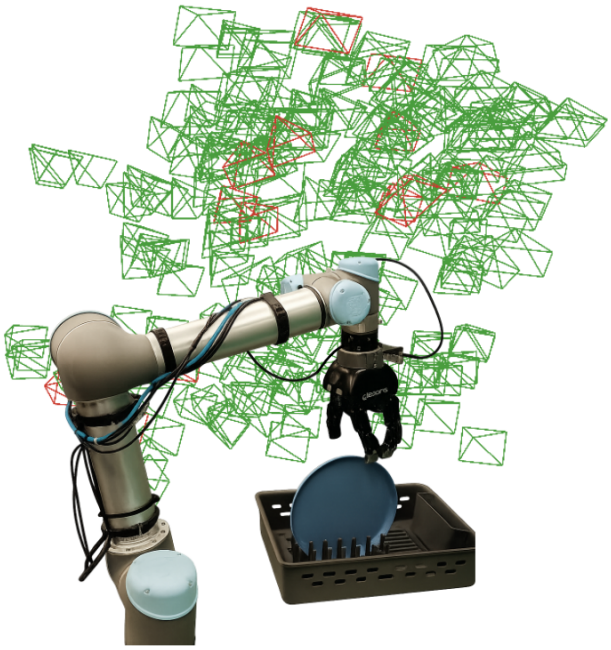


Fig. 1: Visualization of camera poses in the real-robot experiment. Training cameras are visualized in green, and test cameras are visualized in red.

While the architectural simplicity of end-to-end learning is appealing, especially in data-rich domains like language modeling, robotics is notoriously data-scarce. Expecting a single model to simultaneously estimate camera pose while also learning complex manipulation policies poses a serious trade-off between learnability and data efficiency. Therefore, we investigate the benefits of *explicit* camera conditioning. Assuming access to camera geometry, we study its effect on the robustness of robot policies under viewpoint variations.

More specifically, our contributions are the following:

- **A Novel Camera Conditioning Method.** We propose an approach to effectively condition common imitation learning policies on explicit camera geometries with per-pixel Plücker embeddings.
- **Comprehensive Empirical Analysis.** We show the benefits of explicit camera conditioning for policy generalization to view variations, and identify key factors

[1]Toyota Technological Institute at Chicago (TTIC), Chicago, IL, USA. tianchongj@ttic.edu
[2]Waymo, USA. *Work completed while at TTIC.
[3]Johns Hopkins University.
[4]Toyota Research Institute (TRI), USA.
† Equal advising

affecting performance, including action space, random cropping, early vs. late conditioning, the effect of the pretrained visual encoder, and the scaling law with respect to the number of camera views.
- **New Benchmarks for Viewpoint Generalization.** We introduce benchmark tasks—three in robosuite and three in ManiSkill—targeting challenging viewpoint generalization, and the corresponding demonstrations.

II. Related Work

We review imitation learning, methods for view-invariant policies, and approaches to encoding camera geometry.

A. Imitation Learning

Imitation learning offers a direct and sample-efficient approach to teaching robots complex manipulation skills by learning from expert demonstrations. To mitigate compounding errors from single-step policies, Zhao et al. [13] introduced Action Chunking with Transformers (ACT), where policies predict entire action sequences rather than single actions, effectively reducing the task horizon. A key challenge is modeling the multi-modal nature of human demonstrations, as a task can often be solved in multiple valid ways. To address this multi-modality, methods like ACT, Diffusion Policy (DP) [14], and Behavioral Transformers (BeT) [15] learn distributions over actions rather than regress to the mean. A recent trend is to scale policies using large, diverse datasets. Models like RT-1 [16], Octo [1], OpenVLA [3], Pi-0 [4], LBM [17], and SmolVLA [18] demonstrate that training a single, high-capacity model on large, heterogeneous datasets results in improved generalization to novel variations of seen tasks and policies that can be efficiently fine-tuned for new settings.

B. Learning View-Invariant Robot Policies

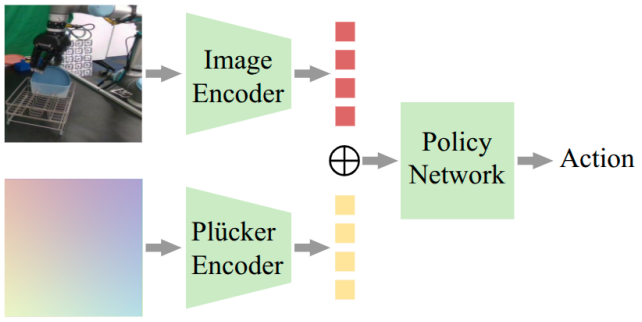
Achieving policy invariance to camera viewpoint is a long-standing goal in robot learning. Existing methods can be broadly categorized by their input modality and whether they learn invariance implicitly or explicitly.

a) 3D and Depth-Based Methods: When depth is available, RGB-D camera feeds can be lifted into explicit 3D representations that are naturally robust to camera pose [19]. One line of work transforms sensor data into voxel grids or learned feature fields [20, 21, 22]. Another approach synthesizes canonical views for the policy, decoupling perception from the physical camera placement [23]. Others explore inherently view-invariant SE(3)-equivariant architectures [24, 25] or operate directly on point clouds [26, 27].

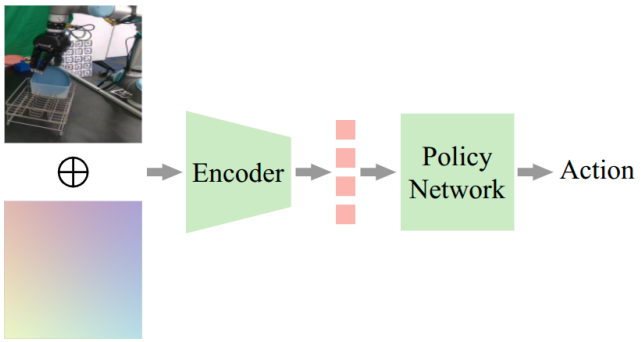
b) RGB-Only Methods: While effective, the 3D methods above rely on high-quality depth sensors, which remain expensive and less ubiquitous than standard cameras. We specifically focus on RGB-based policies because RGB-only input arguably represents the minimal sensing modality for general-purpose robots. Furthermore, vast quantities of internet data and most large-scale behavior cloning datasets are predominantly RGB-based [16, 28]. Within RGB-only methods, a popular strategy for improving robustness is to implicitly learn invariance by randomizing camera viewpoints during data collection or training. This approach has been adopted for several real-world robotics datasets [6, 29, 30, 31, 32]. Similarly, synthetic data augmentation techniques re-render scenes from novel camera extrinsics using techniques like novel view synthesis [33, 34] or 3D Gaussian Splatting [35]. However, this strategy raises a fundamental question: if the robot is barely visible under heavy viewpoint randomization, how can a policy implicitly determine the camera's pose relative to its own action frame?

Another line of implicit methods focuses on learning view-robust representations through self-supervision. These approaches use cross-view correspondence [28, 36], contrastive learning [37], or test-time adaptation [38] to produce features that are invariant to viewpoint shifts without explicit geometric information.

In contrast to these implicit approaches, our work investigates the benefits of explicitly conditioning policies on camera geometry. By providing camera intrinsics and extrinsics directly to the model, we decouple the difficult task of pose inference from the primary goal of learning a manipulation policy, aiming for a more direct and data-efficient path to view invariance.



(a) Policy with a pretrained encoder



(b) Policy without a pretrained encoder

Fig. 2: We propose two ways by which to encode Plücker ray-maps for policies (a) with and (b) without a pretrained encoder. The $\oplus$ sign indicates channel-wise concatenation.

C. Encoding Camera Geometries

When camera geometry is known, various methods exist to encode it as an explicit input to a model. The dominant

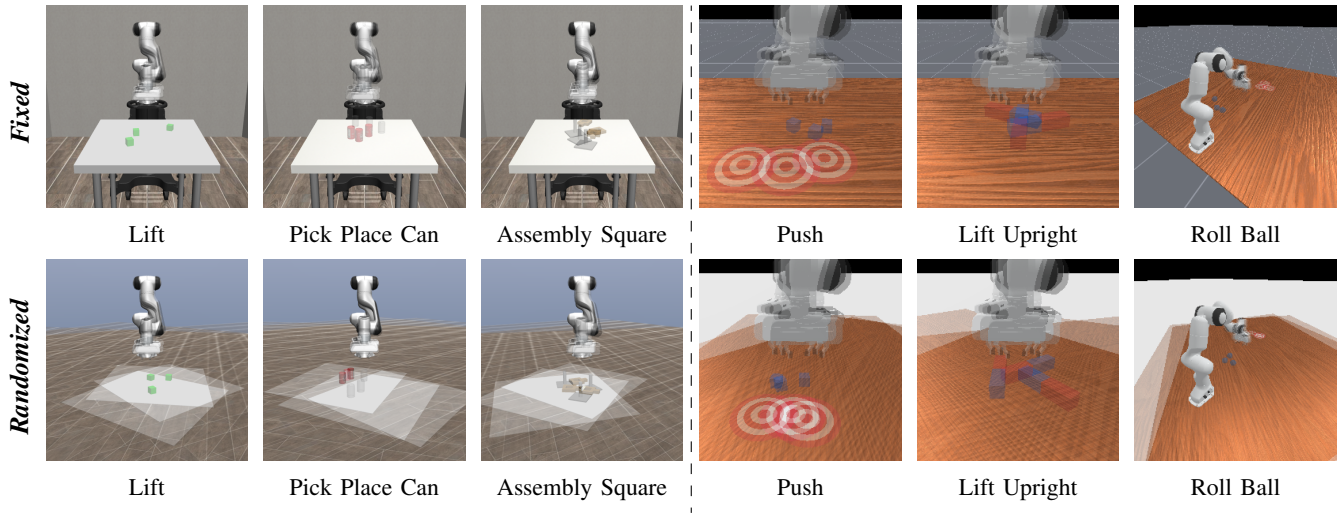


Fig. 3: We introduce six new task variants, three in robosuite and three in ManiSkill. The top row is the *fixed* variants and the bottom row is the *randomized* variants. The left three are in robosuite, and the right three are in ManiSkill. Each sub-figure overlays three images with different initialization seeds to illustrate variations in the environments.

approach in multi-view transformers is to use ray-maps, which are pixel-aligned embeddings that provide geometric information at the token level. These ray-maps typically concatenate the 6D ray origin and direction for each pixel, derived from camera intrinsics and extrinsics [39, 40]. A common variant uses 6D Plücker coordinates, which represent directed lines in space and offer the advantage of being invariant to the choice of origin along the ray [41].

To the best of our knowledge, our work is the first to introduce such camera embedding directly into RGB-based policies for robot control.

III. Approach

A. Plücker Embedding as Camera Conditioning

To provide the robot policy model with detailed information about camera geometries that is compatible with the pixel input space, we follow previous works from the 3D-vision community that condition the model with only input-level inductive biases for 3D reconstruction [11, 42, 43]. We use a per-pixel *ray-map*, where each entry is a six-dimensional Plücker coordinate $r = (d, m) \in \mathbb{R}^6$ representing the ray that projects to that pixel [11, 44, 45]. A Plücker ray consists of a unit direction vector $d \in \mathbb{R}^3$ and a moment vector $m = p \times d$, where p is any point on the ray (e.g., the camera center). This representation is homogeneous and satisfies the bilinear constraint $d^\top m = 0$, capturing oriented 3D lines in a uniform, continuous form. Given a camera with intrinsic matrix K and extrinsics (i.e., pose) (R, t), we compute the world-frame ray direction associated with pixel (u, v) as

$$d_w = \frac{R^\top K^{-1} \tilde{u}}{\|R^\top K^{-1} \tilde{u}\|}, \quad \text{where } \tilde{u} = [u, v, 1]^\top, \tag{1}$$

and obtain the camera center as $C = -R^\top t$. The full Plücker ray is then $r = (d_w, C \times d_w)$. This defines a fixed mapping from each pixel to a ray in 3D space, independent of the scene content.

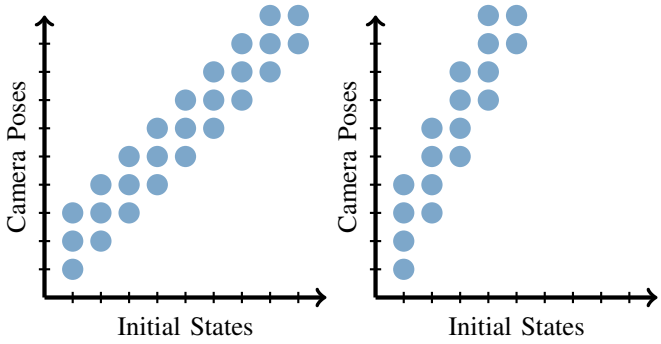
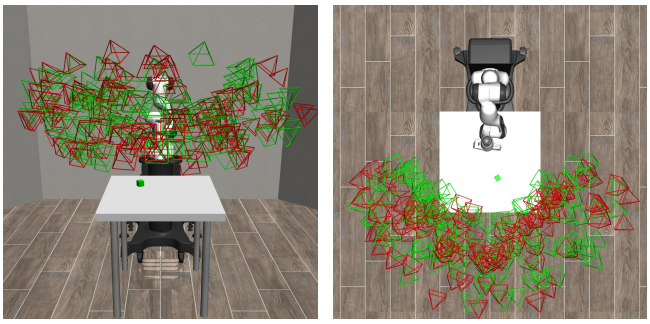


Fig. 4: Visualization of changes of two factors in data collection: camera pose and initial state of environment. On the left, $n = 3$ and $m = 1$. On the right, $n = 4$ and $m = 2$.

A challenge when integrating geometric conditioning is compatibility with policies that use pretrained vision encoders. We propose two ways to encode the $H \times W \times 6$ ray-map (Fig. 2), depending on whether the policy includes a pretrained visual encoder. For policies that do not include a pretrained encoder (e.g., Diffusion Policy [14]), we concatenate the Plücker map channel-wise with the image (Fig. 2(b)). We modify the first layer of the image encoder such that it takes a 9-channel rather than 3-channel input and leave the rest of the policy network unchanged. For policies with pretrained encoders, we employ a small convolutional network to encode the Plücker map to the same dimension as the image latent (Fig. 2(a)). We then concatenate the output with the image latent channel-wise.

B. Benchmarks and Task Setup

We create three new variants of tasks in robosuite—Lift, Pick Place Can, Assembly Square—and three new variants of tasks in ManiSkill—Push, Lift Upright, Roll Ball—where we



Front View Top View

Fig. 5: Visualization of camera poses. The training camera poses are in green and the test camera poses are in red.

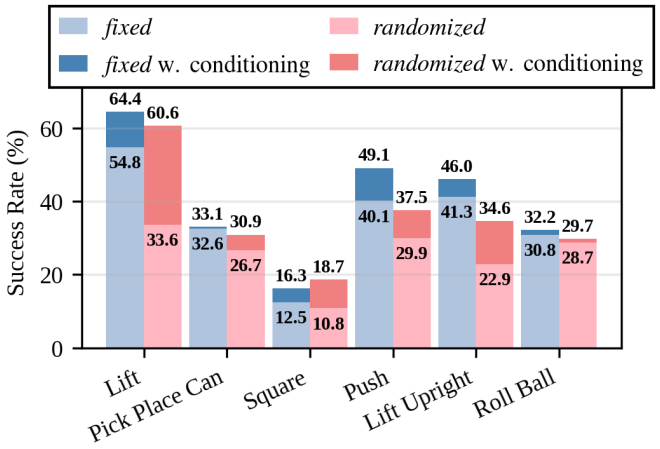


Fig. 6: Success rates of policies for *fixed* and *randomized* variants, with and without Plücker conditioning.

diversify the camera poses. We introduce two variants of each task as visualized in Figure 3. In the *fixed* variant, the robot is fixed relative to the table and floor. In the *randomized* variant, we randomize (i) the robot's position and orientation on the table; and (ii) the table position and orientation relative to the floor. The *randomized* variant emulates practical settings in which policies cannot exploit visual cues from the table or floor to (implicitly) estimate the pose of the camera relative to the robot.

For our experiments, we intentionally exclude the wrist camera, as manipulation tasks often require information from third-person views that the wrist camera alone cannot provide. Doing so allows us to isolate the effects of camera conditioning on these third-person views and, in turn, policy performance.

Notice that when randomizing the camera poses during data collection, there are two factors being randomized: the camera pose and the initial state of the environment. For the model to efficiently generalize compositionally, we follow Gao et al. [46] and collect our data in a "stair"-shaped manner (Fig. 4): each demonstration episode is recorded with n cameras, episodes i and $i+1$ share $n-m$ camera poses, and episode $i+1$ involves m new camera poses. For Can and Square, due to the difficulty of the task, we use $n=4$ and $m=2$. For the real-robot experiments and the rest of the simulated experiments, we use $n=3$ and $m=1$.

The camera poses are visualized in Figure 5. For all tasks, we randomized the azimuth in a 180 degree range and elevation from 30 degrees to 60 degrees. We also randomized the 3D point that the cameras point to in a small 3D box.

C. Evaluation Protocol

We follow the evaluation protocol of Chi et al. [14]: for each policy we train with three torch random seeds $(0, 1, 2)$, take the last ten checkpoints from each run, and evaluate every checkpoint on 50 distinct initial position–camera pose pairs, yielding $3 \times 10 \times 50 = 1500$ rollouts per setting. For SmolVLA, due to compute constraints, we only run seed 0, which yields 500 rollouts per setting. To ensure fairness, we fix the environment random seeds so that the initial conditions and stochastic environment noise are identical across checkpoints, torch seeds, and policies within each task setting. All the simulated experiments, including ablation studies, follow this evaluation protocol.

IV. Results

We evaluate the robustness of behavior cloning policies to viewpoint variations and the benefits of pose conditioning through both simulation and real-world experiments. Simulation results are averaged across 1500 rollouts for ACT and Diffusion Policy and 500 rollouts for SmolVLA.

Unless otherwise specified, the results in this section involve the *randomized* variants, the *Delta End-Effector Pose* action space, and random cropping.

A. Simulated Experiments

1) Camera Conditioning Increases Success Rate Across Models: We train ACT [13], Diffusion Policy [14], and SmolVLA [18] on the six tasks in simulation with and without camera conditioning. See Section IV-A.8 for details. We see in Table I that adding camera pose conditioning increases success rates for all models in all tasks.

2) Fixed vs. Randomized Background Variants: Figure 6 evaluates the benefits of conditioning an ACT policy on camera extrinsics for the *fixed* and *randomized* variants of each task. Camera-pose conditioning is most beneficial in the *randomized* variants, where the randomization removes the policy's shortcut of implicitly inferring the camera-robot transformation. In the *fixed* variants, these cues remain consistent, so the gain is smaller, but still positive.

3) Random Cropping: Random cropping is applied to both the Plücker maps and the images, preserving the one-to-one correspondence between image pixels and Plücker "pixels". Figure 7 shows the success rates of the three robosuite tasks with and without random cropping. Cropping consistently improves performance across all tasks. One interpretation of the results is that cropping essentially adds virtual cameras with different parameters. Without cropping, the Plücker map provides a (nearly) unique fingerprint of each camera pose to which the conditioned policy can overfit and thus generalize more poorly than image-only policies.

TABLE I: Success rates (%) on simulated tasks with and without camera pose conditioning.

Method	Lift	Pick Place Can	Assembly Square	Push	Lift Upright	Roll Ball
ACT	33.6	26.7	10.8	29.9	22.9	28.7
ACT w. conditioning	60.6 (+27.0)	30.9 (+4.2)	18.7 (+7.9)	37.5 (+7.6)	34.6 (+11.7)	29.7 (+1.0)
DP	29.1	23.1	2.0	20.0	9.5	19.9
DP w. conditioning	51.1 (+22.0)	39.3 (+16.2)	2.4 (+0.4)	30.3 (+10.3)	20.7 (+11.2)	23.7 (+3.8)
SmolVLA	19.6	56.0	22.0	39.0	23.6	27.6
SmolVLA w. Conditioning	54.4 (+34.8)	70.0 (+14.0)	26.4 (+4.4)	43.8 (+4.8)	33.4 (+9.8)	30.4 (+2.8)

4) Action Space: We ablate across four action spaces: Absolute End-Effector Pose, Delta End-Effector Pose, Absolute Joint Position, and Delta Joint Position. To ensure fairness, we use the same set of demonstrations, converting them into each action space, and verify by replay that all achieve success rates above 99 percent. Delta actions share the same underlying absolute controllers: Delta End-Effector Pose is derived by differencing adjacent absolute poses and accumulating at test time, while Delta Joint Position is derived analogously from Absolute Joint Position. Thus, any differences between delta and absolute variants reflect the policies' generalization, not the controllers themselves. We see in Figure 8 that the policy performs best when actions take the form of Delta End-Effector Pose and that the performance is noticeably worse for Absolute Joint Position and Absolute End-Effector Pose. Absolute actions depend heavily on the accuracy of the camera-to-action-frame transformation, while relative actions are more tolerant because they primarily require local directional corrections from the current configuration. Nonetheless, the results reveal that camera conditioning increases performance for all four action spaces.

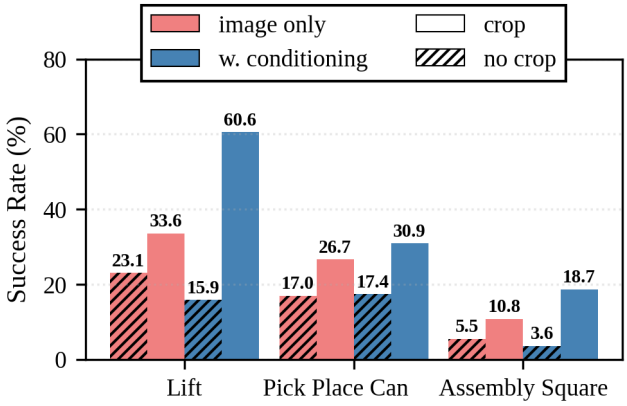


Fig. 7: Ablation of random cropping's effect on task success.

5) Different Ways to Encode Camera Poses: We ablate different approaches to encoding camera poses, as shown in Figure 9. In the *linear projection* method, the extrinsic matrix is linearly projected to a token. In the *early fusion* method, the Plücker map is concatenated with the image pixel-wise before the pretrained encoder. In the *late fusion* method, the Plücker and image latents are concatenated channel-wise after separate encoders, which is the approach we adopt for ACT and SmolVLA in the main experiments. We find that the *linear projection* method provides no observable benefit. In most cases, *early fusion* is less effective than *late fusion*. This may be because concatenating the Plücker map with the image creates inputs that are out of distribution for the pretrained ResNet at the start of training, whereas encoders trained from scratch do not face this issue.

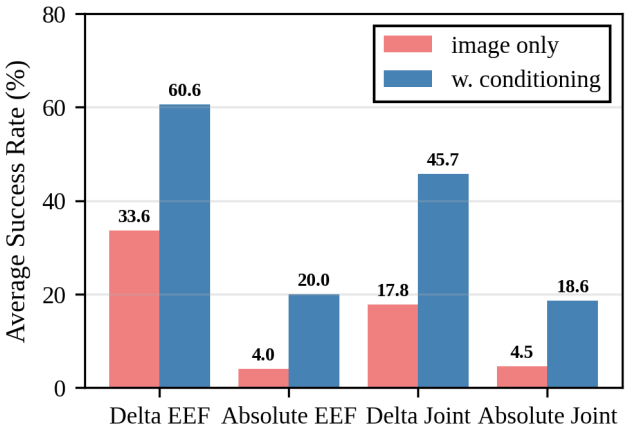


Fig. 8: Success rates for different action spaces.

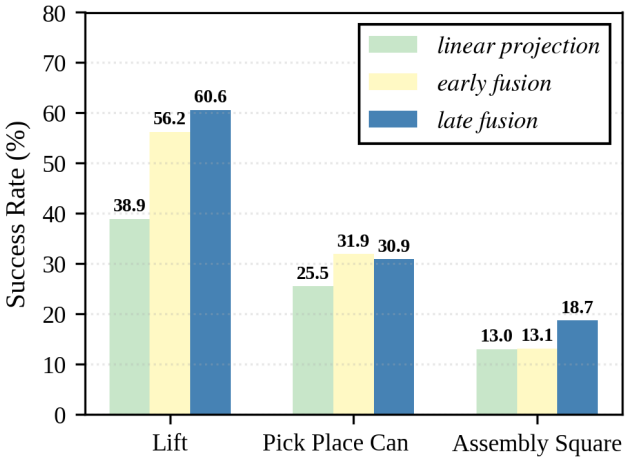


Fig. 9: Ablation of different camera pose encoding methods.

6) The Effect of Pretrained Vision Encoders: We find that pretraining the image encoder (Fig. 10) has little effect on the success rate. We trained ACT with ResNet encoders under

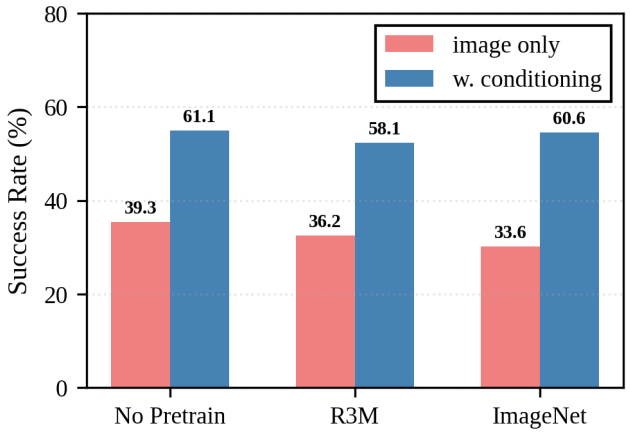


Fig. 10: Ablation on pretraining of image encoders.

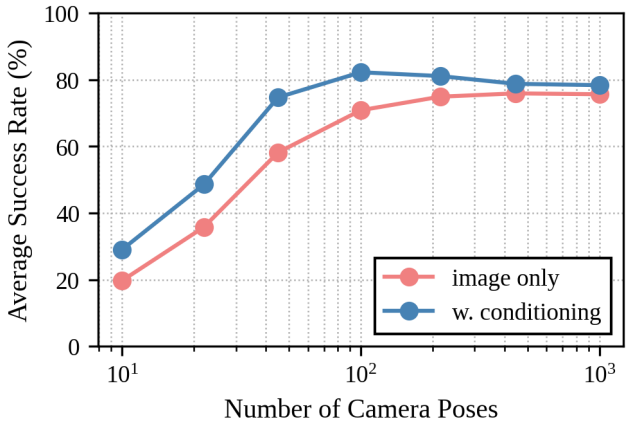


Fig. 11: Experiment on scaling to more camera poses with the Pick task in robosuite.

three settings: no pretraining, R3M weights, and pretraining on ImageNet. Based on these results, we follow the original papers in our experiments: ImageNet-pretrained ResNet for ACT, randomly initialized ResNet for Diffusion Policy, and SigLIP [47] for SmolVLA.

7) Scaling: We evaluate the benefits of conditioning on camera pose when scaling the training data to more views. For these experiments, all demonstrations share the same set of n cameras ($m = 0$). The results are shown in Figure 11. The x-axis shows the number of cameras on a logarithmic scale. We observe that, to achieve the same performance, training without camera pose conditioning requires several times more cameras compared to training with pose conditioning. Moreover, as the number of cameras increases further, conditioning on camera pose still provides a small but consistent improvement.

Since the evaluation involves an average over a wide randomized camera distribution that includes inherently hard or ambiguous viewpoints, success need not reach 100%, even with many training views.

8) Training Details: ACT and SmolVLA are trained for 30k epochs and evaluated every 1k epochs. Diffusion Policy is trained for 80k epochs and evaluated every 2k epochs. The

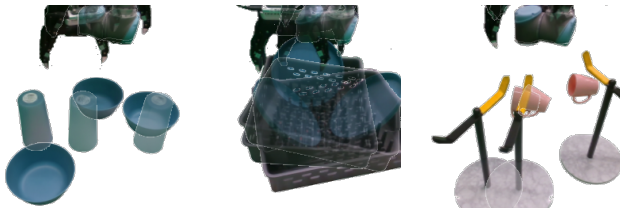


Fig. 12: Real-robot tasks. We overlay three images with different initial states to illustrate variations in the tasks.

number of epochs is selected so that the last 10 evaluation starts well after the policies' success rates have converged. ACT and Diffusion Policy use a constant learning rate schedule. SmolVLA starts from the pretrained weights from LeRobot [18] and we train it with a linear warmup and a cosine decay of learning rate, as we find it is difficult to optimize with a constant learning rate schedule.

B. Real-World Experiments

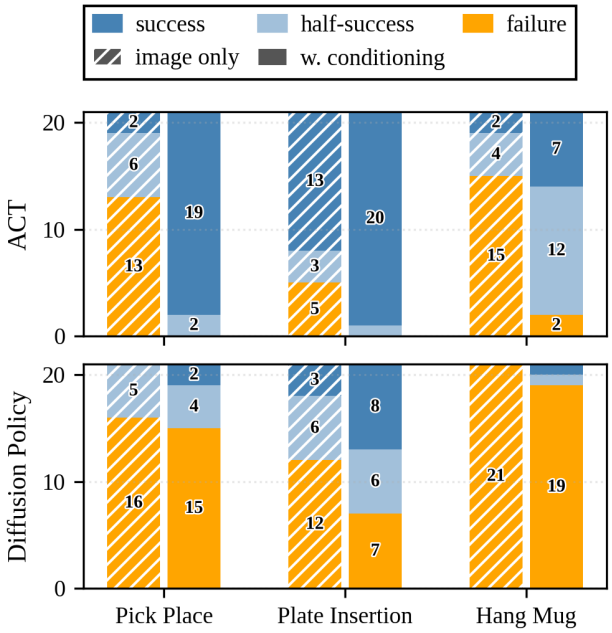


Fig. 13: The performance of (top) ACT and (bottom) Diffusion Policy with and without camera pose conditioning for the three real-world experiments with 21 rollouts per setting.[1]

We conduct real-robot experiments using a UR5 robot arm (Fig. 1) equipped with a Robotiq three-finger gripper on three tasks (Fig. 12): Pick Place, Plate Insertion, and Hang Mug.

For each task, we collect 200 demonstrations using a VR-based teleoperation interface. Three cameras are mounted on articulated camera arms, allowing us to vary their poses. We follow the same randomization strategy as in the simulation experiments: each demonstration is recorded with three cameras ($n = 3$), and after each recording, we change the pose of one camera ($m = 1$). We estimate the pose of each camera using AprilTags [48]. We train ACT for 30k epochs and Diffusion Policy for 60k epochs.

We aim to replicate the *randomized* variant employed in the simulation benchmarks, which is consistent with practical policy deployments. Because the UR5 is heavy and difficult to reposition, we approximate this *randomized* variant by covering the background with green cloth to remove static visual cues.

For evaluation, we test with seven random poses for each of the three cameras. In each setting, we run one rollout using two of the three cameras, resulting in $3 \times 7 = 21$ rollouts per setting. To ensure fairness, we mark the initial positions of objects and reset them consistently across all methods.

Due to the difficulty of the real-robot tasks, we introduce a criterion called half-success. This allows us to provide more nuanced information that is not captured by a simple binary measure of success or failure. We define half-success separately for each task as follows:

- **Pick Place:** Half-success means the robot attempts to lift the cup, but it slips when the grasp is slightly off. Success means lifting the cup and placing it in the bowl.
- **Plate Insertion:** Half-success means the robot successfully picks up the plate. Success requires inserting the plate into the slots of the dish rack.
- **Hang Mug:** Half-success occurs when the mug makes contact with the hanger. Success requires that the mug ends up hanging securely on the hanger.

Figure 13 shows that conditioning on camera poses improves all policies across tasks. Diffusion Policy's failures are dominated by control imprecision rather than viewpoint ambiguity, so conditioning yields smaller gains than for ACT.

V. Conclusion

We studied view-invariant imitation learning by conditioning policies on camera extrinsics using Plücker raymaps. Our experiments show consistent improvements in viewpoint generalization across ACT, Diffusion Policy, and SmolVLA in both simulation and real-robot settings. We also highlighted how policies without extrinsics can exploit static background cues and fail when those cues change, while extrinsic conditioning restores performance.

Despite these encouraging results, our study has its limitations. Our experiments are scoped to offline imitation learning, and we do not study interactive data collection or reinforcement learning. Since the proposed camera-conditioning mechanism is largely orthogonal to the training paradigm, a natural direction is to integrate it with interactive/online methods—e.g., dataset aggregation (DAgger [49]) or reinforcement learning. Moreover, our work mainly focuses on pose conditioning, while many other directions remain unexplored. For example, just as our method enables policies to generalize to novel camera poses, a natural extension would be to support generalization across cameras with different intrinsics.

[1]We observed that the performance of Diffusion Policy is significantly worse than that of ACT. We believe this is partly due to its stochastic nature in precision-demanding tasks. In particular, when multiple trajectories can complete a task, Diffusion Policy often oscillates between them, which can lead to failure.

Looking forward, we believe these contributions will provide both methodological insights and practical benchmarks for advancing viewpoint-robust imitation learning. In particular, our six new robosuite and ManiSkill benchmarks offer a principled way to evaluate robustness, while our findings on action space design and data augmentation highlight key considerations for policy training. Together, these contributions lay the foundation for developing policies that generalize more reliably across the diverse viewpoints encountered in real-world deployment.

References

[1] D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. L. Tan, P. R. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *Proc. Robotics: Science and Systems (RSS)*, 2024.

[2] Open X-Embodiment Collaboration, "Open X-Embodiment: Robotic learning datasets and RT-X models," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6892–6903.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[4] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[5] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi, "XIRL: Cross-embodiment inverse reinforcement learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2022.

[6] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, C. Finn, and S. Levine, "DROID: A large-scale in-the-wild robot manipulation dataset," in *Proc. Robotics: Science and Systems (RSS)*, 2024.

[7] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient three-dimensional robotics hand/eye calibration," *Transactions on Robotics*, vol. 5, no. 3, pp. 345–358, 1989.

[8] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[10] J. Li, H. Wang, M. Z. Irshad, I. Vasiljevic, M. R. Walter, V. C. Guizilini, and G. Shakhnarovich, "FastMap: Revisiting structure from motion through first-order optimization," *arXiv preprint arXiv:2505.04612*, 2025.

[11] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "VGGT: Visual geometry grounded transformer," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[12] J. Huang, Q. Zhou, H. Rabeti, A. Korovko, H. Ling, X. Ren, T. Shen, J. Gao, D. Slepichev, C.-H. Lin *et al.*, "ViPE: Video pose engine for 3D geometric perception," *arXiv preprint arXiv:2508.10934*, 2025.

[13] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-

grained bimanual manipulation with low-cost hardware," in *Proc. Robotics: Science and Systems (RSS)*, 2023.
[14] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int'l J. of Robotics Research*, 2024.
[15] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
[16] G. M. Brohan, N. Brown, E. Luo, K. Pertsch, J. E. Suarez, S. Levine, and K. Hausman, "RT-1: Robotics transformer for real-world control at scale," in *Proc. Robotics: Science and Systems (RSS)*, 2023.
[17] TRI LBM Team, "A careful examination of large behavior models for multitask dexterous manipulation," *arXiv preprint arXiv:2507.05331*, 2025.
[18] M. Shukor, D. Aubakirova, F. Capuano, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadene, "SmolVLA: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.
[19] S. Peri, I. Lee, C. Kim, F. Li, T. Hermans, and S. Lee, "Point cloud models improve visual robustness in robotic learners," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2024.
[20] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-Actor: A multi-task transformer for robotic manipulation," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[21] Y. Ze, G. Yan, Y. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang, "GNFactor: Multi-task real robot learning with generalizable neural feature fields," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[22] J. Gervet, D. F. Fouhey, D. Pathak, and D. Ramanan, "Act3D: 3D feature field transformers for multi-task robotic manipulation," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[23] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y. Chao, and D. Fox, "RVT: Robotic view transformer for 3D object manipulation," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[24] J. Yang, Z. Cao, C. Deng, R. Antonova, S. Song, and J. Bohg, "EquiBot: SIM(3)-equivariant diffusion policy for generalizable and data efficient learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2025.
[25] X. Zhu, Y. Qi, Y. Zhu, R. Walters, and R. Platt, "EquAct: An SE(3)-equivariant multi-task transformer for open-loop robotic manipulation," *arXiv preprint arXiv:2505.21351*, 2025.
[26] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations," in *Proc. Robotics: Science and Systems (RSS)*, 2024.
[27] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3D diffuser actor: Policy diffusion with 3D scene representations," in *Proc. Conf. on Robot Learning (CoRL)*, 2025.
[28] S. Nair, S. Zhu, S. Savarese, C. Finn *et al.*, "R3M: A universal visual representation for robot manipulation," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[29] "RoboSet: A large-scale real-world multi-task dataset for robotic manipulation," https://robopen.github.io/roboset/, 2023, dataset website (see also RoboAgent paper for details).
[30] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu, "RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot," in *Proc. IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2024.
[31] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn, "RoboNet: Large-scale multi-robot learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2020.
[32] H. R. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine, "BridgeData V2: A dataset for robot learning at scale," in *Proc. Conf. on Robot Learning (CoRL)*, 2023.
[33] S. Tian, B. Wulfe, K. Sargent, K. Liu, S. Zakharov, V. Guizilini, and J. Wu, "View-invariant policy learning via zero-shot novel view synthesis," *arXiv preprint arXiv:2409.03685*, 2024.
[34] L. Y. Chen, C. Xu, K. Dharmarajan, R. Cheng, K. Keutzer, M. Tomizuka, Q. Vuong, and K. Goldberg, "RoVi-Aug: Robot and viewpoint augmentation for cross-embodiment robot learning," in *Proc. Conf. on Robot Learning (CoRL)*, 2025.
[35] S. Yang, W. Yu, J. Zeng, J. Lv, K. Ren, C. Lu, D. Lin, and J. Pang, "Novel demonstration generation with gaussian splatting enables robust one-shot manipulation," in *Proc. Robotics: Science and Systems (RSS)*, 2025.
[36] Y. Seo, J. Kim, S. James, K. Lee, J. Shin, and P. Abbeel, "Multi-view masked world models for visual robotic manipulation," in *Proc. Int'l Conf. on Machine Learning (ICML)*, 2023.
[37] S.-W. Lee, X. Kang, B. Yang, and Y.-L. Kuo, "CLASS: Contrastive Learning via Action Sequence Supervision for robot manipulation," *arXiv preprint arXiv:2508.01600*, 2025.
[38] S. Yang, Y. Ze, and H. Xu, "MoVie: Visual Model-based Policy Adaptation for View Generalization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
[39] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole, "CAT3D: Create anything in 3D with multi-view diffusion models," *arXiv preprint arXiv:2405.10314*, 2024.
[40] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Communications of the ACM*, vol. 65, no. 1, 2021.
[41] J. Y. Zhang, A. Lin, M. Kumar, T.-H. Yang, D. Ramanan, and S. Tulsiani, "Cameras as rays: Pose estimation via ray diffusion," *arXiv preprint arXiv:2402.14817*, 2024.
[42] W. Yifan, C. Doersch, R. Arandjelović, J. Carreira, and A. Zisserman, "Input-level inductive biases for 3D reconstruction," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
[43] V. Guizilini, I. Vasiljevic, J. Fang, R. Ambru, G. Shakhnarovich, M. R. Walter, and A. Gaidon, "Depth field networks for generalizable multi-view scene representation," in *Proc. Int'l. Conf. on Computer Vision (ICCV)*, 2022.
[44] D. Lanman, M. Wachs, G. Taubin, and F. Cukierman, "Reconstructing a 3D line from a single catadioptric image," in *Proc. Int'l Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2006.
[45] V. Sitzmann, S. Rezchikov, W. T. Freeman, J. B. Tenenbaum, and F. Durand, "Light field networks: Neural scene representations with single-evaluation rendering," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
[46] J. Gao, A. Xie, T. Xiao, C. Finn, and D. Sadigh, "Efficient data collection for robotic manipulation via compositional generalization," in *Proc. Robotics: Science and Systems (RSS)*, 2024.
[47] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *Proc. European. Conf. on Computer Vision (ECCV)*, 2023.
[48] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS)*, 2016.
[49] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. Int'l Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.