Track2Act: Predicting Point Tracks from Internet Videos enables Generalizable Robot Manipulation

anonymous

Abstract: We seek to learn a generalizable goal-conditioned policy that enables *diverse* robot manipulation — interacting with unseen objects in novel scenes without test-time adaptation. While typical approaches rely on a large amount of demonstration data for such generalization, we propose an approach that leverages web videos to predict plausible interaction plans and learns a task-agnostic transformation to obtain robot actions in the real world. Our framework, *Track2Act* predicts tracks of how points in an image should move in future time-steps based on a goal, and can be trained with diverse videos on the web including those of humans and robots manipulating everyday objects. We use these 2D track predictions to infer a sequence of rigid transforms of the object to be manipulated, and obtain robot end-effector poses that can be executed in an open-loop manner. We then refine this open-loop plan by predicting residual actions through a closed loop policy trained with a few embodiment-specific demonstrations. We show that this approach of combining scalably learned track prediction with a residual policy requiring minimal in-domain robot-specific data enables diverse generalizable robot manipulation, and present a wide array of real-world robot manipulation results across unseen tasks, objects, and scenes.

Keywords: web videos, diverse manipulation

Figure 1: Glimpse of the diverse robot manipulation capabilities across physical office and kitchen scenes enabled by *Track2Act*. We learn to predict point tracks from web videos for learning interaction plans that can be used for inferring robot actions in unseen scenarios. This enables a *common* goal-conditioned policy to perform everyday tasks like closing microwaves, pulling out drawers, flipping open toasters, pouring from jars etc. Columns show first/last images of policy rollouts.

1 Introduction

Robots that can be reliably deployed out-of-the-box in new scenarios have the potential for helping humans in everyday tasks. To realize this vision of generalizable robot manipulation, it is crucial

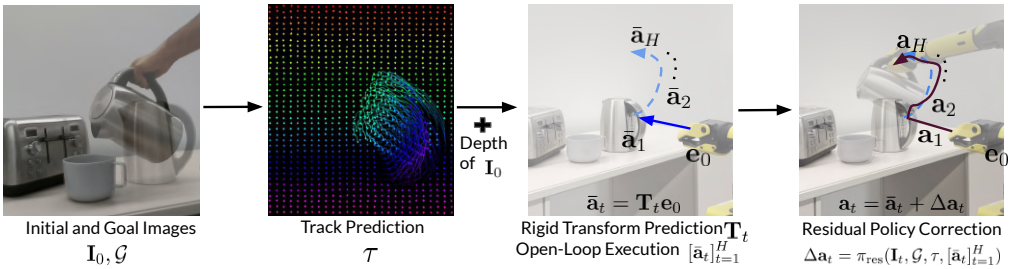

Figure 2: Illustration of the *Track2Act* pipeline for learning track prediction from web video datasets, inferring rigid transforms of objects based on the predicted tracks in a robot's environment, and fine-tuning with a residual policy learned with limited robot data. This approach allows us to learn a single goal-conditioned policy for diverse (unseen) tasks.

to develop *zero-shot* execution capabilities i.e. being able to execute a task out-of-the-box without requiring any test-time training through demonstrations or self-practice before solving a specified task. This is an important desiderata for the system to be repeatedly usable without any downtime, and safe to work alongside humans without performing any exploratory actions. We pursue the goal of developing such *zero-shot* robot manipulation systems that can perform a broad set of everyday tasks. In addition to being deployable *zero-shot*, to be widely accessible, we aim to make the robot manipulators generalizable to diverse offices, and kitchens in the real world.

Developing zero-shot manipulation capabilities has been attempted by prior works, through behavior cloning on robot interaction datasets [1, 2, 3, 4]. While this approach is in-principle scalable with data, collecting diverse real-world robot interaction data is challenging due to operational constraints. Indeed, recent works that have attempted to scale robot datasets, including cross-robot and cross-domain datasets [5, 6, 1] still suffer from task diversity issues and are mostly restricted to lab-like structured scenarios. Instead of learning a single-policy that can be zero-shot deployed, some recent works aimed at in-the-wild deployment have adopted the method of test-time training [7, 8]. They require either a video of a human performing the task [7] followed by online exploration, or a demo through a robot end-effector held by a human [8]. These approaches are not very convenient for diverse deployments because they require a human to solve the task first, and several hours after that for the robot to learn how to solve that exact task in the exact scene. Thus, such approaches are not *zero-shot* deployable for new tasks in new scenes.

Our insight to develop an in-the-wild manipulation strategy that is also zero-shot deployable is to factorize a manipulation policy into an *interaction-plan* that can leverage diverse large-scale video sources on the web of humans and robots manipulating everyday objects and a residual policy that requires a small amount of embodiment-specific robot interaction data. Such a factorized structure is inspired by prior works (e.g. [9]), however, unlike hand-object masks in [9], we instantiate this *interaction-plan* in an embodiment agnostic-manner by predicting how points in an image of the initial scene move in future frames. This choice of an *interaction-plan* is more expressive compared to hand-object masks adopted by Bharadhwaj et al. [9], as it directly captures point correspondences across time, while at the same time being easier to compute than full RGB frames [10]. Given an initial image of the scene, a goal image defining the task to be performed, and a random set of points in the initial image, we define the interaction plan to be a 2D trajectory of the locations of the points in future frames, such that the goal is achieved. Importantly, we can train this model purely from the abundantly available human and robot videos on the web without any data specific to the deployment robot embodiment, by using off-the-shelf point-tracking approaches [11] for generating the ground-truth point trajectories. For deployment in a robot's environment, we can convert the 2D interaction-plan to a sequence of 3D end-effector poses, by having a depth image of the initial scene as an additional input and solving an optimization problem to obtain rigid transforms of the object being manipulated. Finally, with a small amount of embodiment-specific robot interaction data for

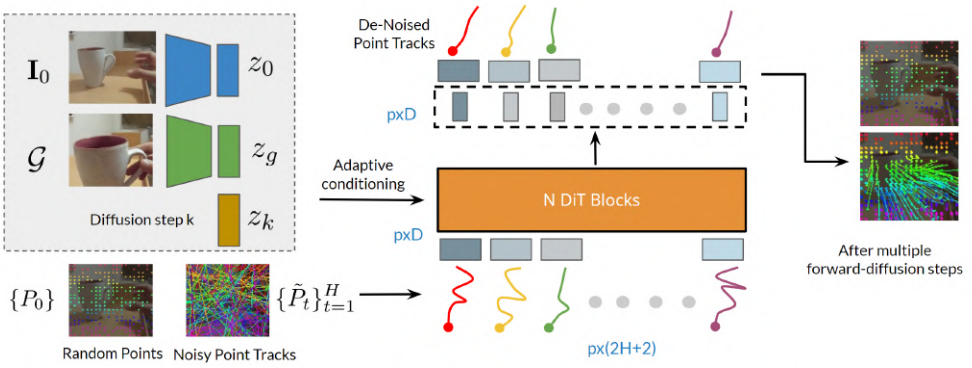

Figure 3: Architecture of the Diffusion Transformer $\mathcal{V}_\theta$ for denoising track predictions given initial image $\mathbf{I_0}$, goal $\mathcal{G}$, and an initial set of p points P_0.

different tasks ($\sim$ 400 trajectories overall), we can learn a goal-conditioned residual policy that corrects for errors in the predicted plan at each time-step and allows for closed-loop deployment.

In summary, we develop *Track2Act* with the following contributions:

- We develop a framework for predicting embodiment-agnostic *interaction-plans* in the form of point tracks from diverse web videos.
- We show how the interaction-plan prediction model can be used for obtaining 3D rigid transforms in a robot's environment for zero-shot manipulation without using any robot data or online exploration.
- Given a few ($\sim$ 400) embodiment-specific task demonstrations, we show how to learn a goal-conditioned residual policy that can correct for errors in the predicted plan at each time-step. The interaction-plan prediction model combined with the residual policy correction can then be used for zero-shot closed-loop deployment for new tasks in new scenes.

Our real-world robot manipulation results with a Spot robot (highlighted in Fig. 1) show broad generalization across diverse tasks involving unseen objects in unseen scenes, and demonstrate the potential for leveraging easily available passive videos on the web for learning embodiment-agnostic interaction plans. This is significant as it enables zero-shot robot manipulation with a *common* goal-conditioned policy, that generalizes to unseen tasks without requiring collection of large scale in-domain manipulation datasets.

2 Approach

We aim to develop a zero-shot robot manipulation system, *Track2Act* that can scalably leverage diverse video data for generalizable real-world manipulation. Our key insight (Fig. 2) is to have a factorized policy for 1) learning embodiment-agnostic *interactions plans* of how points in an image of a scene should move in subsequent time-steps to realize a specified goal, followed by 2) inferring robot actions based on the interaction plan through a residual policy. We show how *Track2Act* allows us to generalize to diverse scenarios involving unseen tasks and objects, since the prediction model by virtue of being trained on web data generalizes well to new scenes, and the residual policy has a much simpler task of correcting the robot actions derived from the interaction plan.

2.1 Overview and Setup

Given a scene specified by an RGB image $\mathbf{I_0}$ and a goal image $\mathcal{G}$ denoting what task should be performed, we want to have a robot manipulator execute actions $\mathbf{a}_{1:H}$ in the scene to achieve the desired goal. To achieve this in unseen scenarios, we leverage web video data by learning a model $\tau = \mathcal{V}_\theta(\mathbf{I_0}, \mathcal{G}, P_0)$ to predict future locations (tracks) of p random points P_0 in the initial image.

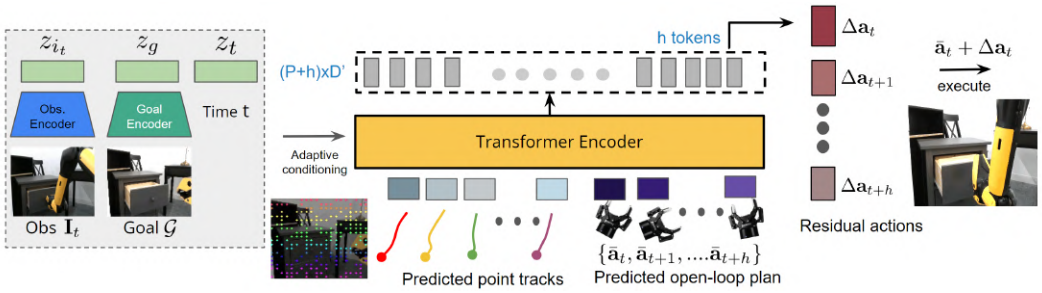

Figure 4: Architecture of the residual policy of *Track2Act* that predicts corrections at each time-step over the predicted open-loop plan, and enables closed-loop deployment.

Given a depth image for the initial frame, we leverage a subset of the predicted tracks τ_{obj} (corresponding to moving points) to infer rigid-transforms of the object being manipulated $[\mathbf{T}_t]_{t=1}^{H}$ and show that these allow obtaining an open-loop plan in the form of robot end-effector poses $[\bar{\mathbf{a}}_t]_{t=1}^{H}$. Finally, we consider training a closed-loop residual policy $\pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=1}^{H})$ that corrects the open-loop action sequence $[\bar{\mathbf{a}}_t]_{t=1}^{H}$ by predicting residual actions at each timestep $\Delta\mathbf{a}_t$, such that the executed action sequence is $[\mathbf{a}_t = \bar{\mathbf{a}}_t + \Delta\mathbf{a}_t]_{t=1}^{H}$. In the subsequent sections, we explain the architecture and algorithm design for each of the three stages in our approach.

2.2 Point Track Prediction from Web Videos

For *Track2Act*, we instantiate track prediction as a denoising process through a DiT based diffusion model [12]. Let $\mathbf{I_0}$ denote the first frame of a video, and $\mathcal{G}$ denote the goal, which we consider to be the last frame of the video. For longer videos, we obtain multiple video clips of 4-5 seconds each for training. Let there be p points in the initial frame to be tracked, such that P_0 denotes the set of those points and let H be the prediction horizon. $[P_t]_{t=1}^{H}$ denotes the future locations of those points in the subsequent time-steps that we want to predict. In the forward diffusion process, all the points P_t are corrupted by incrementally adding noise ϵ_k (k denotes the diffusion time-step), to obtain $\tilde{P}_t$, and converging to a unit Gaussian distribution $N(\mathbf{0}, \mathbf{I})$. New samples can be generated by reversing the forward diffusion process, by going from Gaussian noise back to the space of point locations. To solve the reverse diffusion process, we need to train a noise predictor $\mathcal{V}_\theta(\mathbf{I_0}, \mathcal{G}, P_0, k)$. We design a DiT Transformer based architecture [12] for $\mathcal{V}_\theta$ illustrated visually in Fig. 3. Different from the original DiT model, we condition on embeddings of initial (z_0) and goal (z_g) images in addition to that of the diffusion step (z_k). The input to the Transformer in each batch is a sequence of p tokens corresponding the number of points specified for tracking. The initial P_0 points are not noisy, as is the convention in training conditional diffusion models on time-series data.

We train the prediction model with web videos by considering variable number of initial points p that need to be tracked. For flexible modeling, the locations of the p points are also randomized, such that at test-time any set of points in the initial image can be specified. We do not make any assumptions on objects to be tracked or camera motions in the videos, and do not curate the training videos in any way apart from ensuring they are of 4-5 second duration. If the goal image is such that multiple objects have moved from the initial scene, or the camera has moved, the track prediction model will predict different groups of motions for different objects and also predict motions of background points to account for camera motion. However, for robot experiments, we consider only a single object to be manipulated at a time, which is indeed the case with diverse real-world tasks.

2.3 Inferring Coarse Manipulator Trajectory from Interaction Plan

Given an image of a scene in a robot's environment $\mathbf{I}_0$, a goal $\mathcal{G}$, and a random set of points P_0 in the initial image, we can use the trained track prediction model to obtain future 2D locations of these points $\hat{P}_t$. As we consider scenarios with only a single object being manipulated under a fixed

camera, only a subset of the points have a large predicted motion. We identify these p points and denote their predicted trajectories as $\tau_{\text{obj}} = [\{(x_t^i, y_t^i)\}_{i=1}^p]_{t=1}^H$. We consider the robot to be equipped with an RGBD camera, so we also have depth for the points P_0 in the first frame. Let us denote these 3D points as $P_0^{\text{3D}} = \{(x_0^i, y_0^i, z_0^i)\}_{i=1}^p$.

We seek to infer a (smooth trajectory of) per-time rigid transforms $\mathbf{T}_t$ of the object to be manipulated at time t relative to first frame, given 3D points in the first frame P_0^{3D}, predicted 2D trajectory of points on the object τ_{obj}, and the camera intrinsic matrix $\mathbf{K}$. As described in Algorithm 1, these can be obtained by ensuring that the projection of the transformed 3D points i.e. $\mathbf{K}\mathbf{T}_t P_0$ matches the predicted 2D tracks $\{(x_t^i, y_t^i)\}_{i=1}^p$ as closely as possible at each time-step t. Let $\mathbf{K}\mathbf{T}_t P_0 \simeq \{(u_t^i, v_t^i, 1)\}_{i=1}^p$. So the 2D projection of the i^{th} point at time t is (u_t^i, v_t^i). Alternatively, we have the same coordinate for the point from the predicted track i.e. (x_t^i, y_t^i). In order to determine the rigid transforms $\mathbf{T}_t$, we can solve the optimization problem in line 6 of Algorithm 1 (e.g. with PnP solvers). This optimization is not under-constrained because the same 3D rigid transform $\mathbf{T}_i$ must explain the 2D motions of several points P_0 in the initial scene. Note that the obtained rigid transforms are *embodiment-agnostic* and describe how the object should move in the scene.

Now, to actually manipulate the object in the scene, we need to bring the robot end-effector[1] near the object, and optionally execute a grasp to hold on to the object, followed by transforming the end-effector based on the predicted rigid transforms $[\mathbf{T}_t]_{t=1}^H$. For the first step, we use a heuristic such that given initial end-effector pose $\mathbf{e}_0$ we define the first transform $\mathbf{T}_0$ to be such that the end-effector moves to the center of the 3D points $\{(x_t^i, y_t^i)\}_{i=0}^p$ with the same orientation as $\mathbf{e}_0$. After moving the end-effector to this pose $\mathbf{e}_1$ we execute a grasp to hold the object. We obtain subsequent end-effector poses (open-loop action trajectory) by applying the rigid transforms $\bar{\mathbf{a}}_t = \mathbf{T}_t \mathbf{e}_1$.

2.4 Closed-loop Manipulation with Residual Policy Correction

The open-loop execution of the predicted 3D end-effector transforms described in the last section $[\bar{\mathbf{a}}_t]_{t=1}^H$ might fail due to small errors in the prediction. In addition, since the approach does not use any embodiment-specific data, it does not have accurate information for reasoning about contact with objects and might suffer from failures like being unable to grasp the object, in spite of executing the rest of the predicted trajectory correctly. To remedy this, we propose learning a residual policy $\pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=1}^H)$ shown in Fig. 4 to correct the predicted end-effector poses in each time-step. So the end-effector pose at time t is

$$\hat{\mathbf{a}}_t = \bar{\mathbf{a}}_t + \Delta\mathbf{a}_t \;\; ; \quad \text{where} \;\; \Delta\mathbf{a}_t = \pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=1}^H) \tag{1}$$

Instead of predicting just a single residual action $\Delta\mathbf{a}_t$ we predict residuals h steps in the future $\Delta\mathbf{a}_{t:t+h}$ and during deployment execute just the first action. This multi-step prediction has been shown to mitigate compounding errors in behavior-cloning based training [13, 1]. We can learn the residual policy with a small amount of robot demonstrations ($\sim$ 400 trajectories overall) of representative tasks through behavior cloning. The data for each trajectory consists of observation-action pairs of the form $[(\mathbf{I}_t, \mathbf{a}_t)]_{t=1}^H$. Here, $\mathbf{I}_t$ denotes images observed from the robot's camera and $\mathbf{a}_t$ denotes actions in the form of end-effector poses.

Crucially, since the aim of this policy is to learn only small corrections to the predicted waypoints $[\bar{\mathbf{a}}_t]_{t=1}^H$, we do not need to learn this policy with data from the exact scenarios that the system will be deployed in and the prediction model is expected to generalize to unseen scenarios by virtue of diverse training. The rationale is that having some embodiment-specific demonstration data in a few scenarios will help correct for the open-loop predictions from web-only data. For evaluation, we consider different levels of generalization with unseen object instances and completely unseen objects in unseen scenes.

3 Experiment Setup

We focus our experiments for *Track2Act* on in-the-wild manipulation scenarios where a mobile manipulator needs to manipulate objects in different living rooms, offices, and kitchens based on

[1] by end-effector we mean the part of the robot that interacts with an object

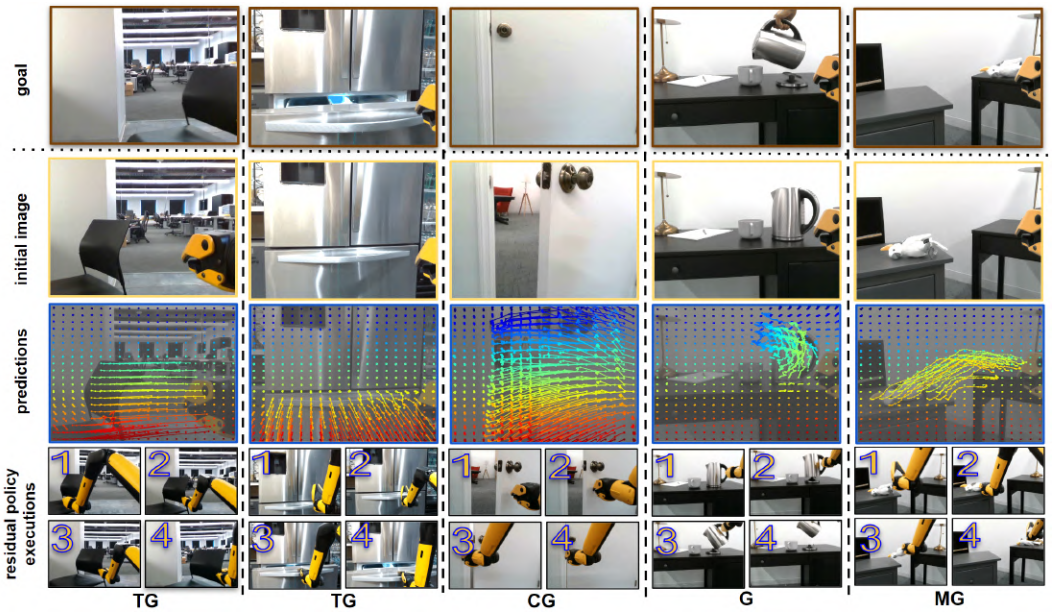

Figure 5: We show visualizations of point track predictions for different tasks, followed by closed-loop execution with the residual policy. We can see that the predictions are plausible and the robot execution successfully realizes the predictions to complete the respective tasks specified by the goal images. The bottom row shows the generalization level for each task, defined in section 3.1.

specified goals. For all the robot experiments, we use a Boston Dynamics Spot robot equipped with a manipulator (hand) and a front facing Intel RealSense camera [14]. We manipulate the arm through end-effector control based on the outputs of our policy.

3.1 Evaluation Details

As is the convention in goal-conditioned robot learning, we perform evaluations by quantifying success rate, where a successful trajectory is defined to be one where the final pose of the object in the scene to be manipulated is identical to the pose of the object in the goal image. We categorize results based on different levels of generalization, the definitions of which are inspired by prior works [1, 9, 2, 3]: Mild Generalization (**MG**): unseen configurations of seen object instances in seen scenes; organic scene variations like lighting and background changes. Standard Generalization (**G**): unseen object instances in seen/unseen scenes. Combinatorial Generalization (**CG**): unseen activity-object type combinations in seen/unseen scenes. Type Generalization (**TG**): completely unseen object types, or completely unseen activities, in unseen scenes

3.2 Baselines and Comparisons

We perform several comparisons with baselines and ablation studies for goal-conditioned robot manipulation. For baselines, we use the same embodiment-specific demonstrations as *Track2Act*, the goal-conditioned policy that predicts residuals over open-loop actions at each time-step (Algorithm 2). *Goal-Conditioned BC* is a baseline for multi-task policy learning, similar to prior works [2, 3, 1]. *Affordance-Conditioned BC* is the approach from [15] that conditions the policy on predicted affordances in the initial image. *Video-Conditioned BC* based on [16, 10, 17] first predicts RGB video and then does tracking on top of it. *Hand-Object Mask Conditioned BC* from [9] conditions the policy on a predicted interaction plan consisting of masks of hands and objects. *Track2Act(Open Loop)* is the approach for track prediction followed by open-loop execution as described in Algorithm 1. This does not use any embodiment-specific data for training. To understand the benefit of predicting residuals over actions as opposed to predicting complete actions, we

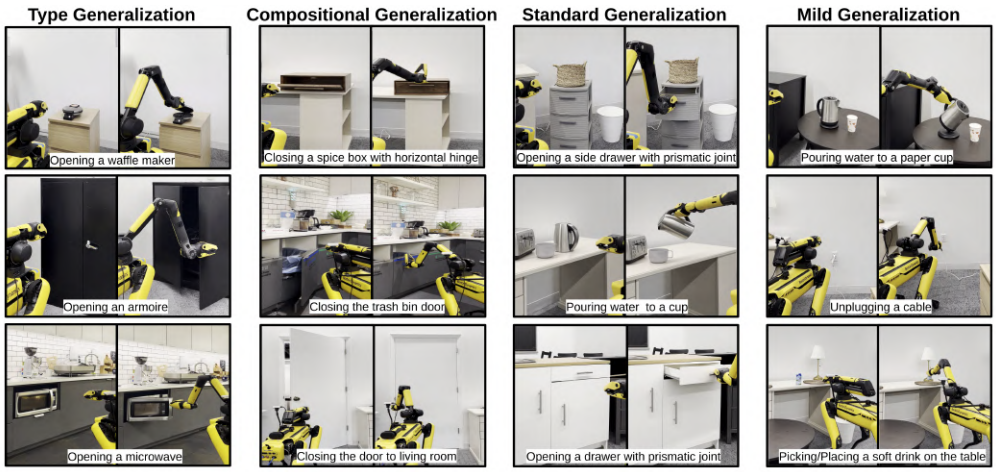

Figure 6: Qualitative results showing robot executions (from a third person view) with the residual policy for different tasks with respect to the generalization levels defined in section 3.1. We show the first and last images of a rollout.

compare with an ablated variant *Track2Act(actions; not residuals)* that predicts actions $\hat{\mathbf{a}}_t$ directly without predicting residuals $\Delta\mathbf{a}_t$ and not relying on an open-loop plan as input.

3.3 Training Data

For training the track prediction model, we leverage diverse passive videos available on the web that are not collected by us. Specifically, we use human video clips from EpicKitchens [18] (clipping videos to ensure they are of 4-5 seconds duration), human videos on YouTube sourced in SmthSmthv2 [19], and large-scale robot videos released in RT1 data [2] and BridgeData [20]. To obtain ground-truth tracks for training the prediction model, we run Co-Tracker [11] on the resulting 400,000 video clips. Note that the robot datasets (RT1 and Bridge) are on completely different robots and scenarios than the robot we use for experiments (Spot). For training the residual policy, the embodiment-specific data we collect consists of $\sim$ 400 trajectories obtained by tele-operating the Spot, for solving 10 tasks of manipulating everyday objects like doors, drawers, bottles, jugs. Note that this embodiment-specific data we collect is 3-4 orders of magnitude less than that what related works [2, 3, 1] require for policy learning.

4 Results

We visualize results of point track predictions using the trained prediction model in a robot's environment, followed by residual policy executions based on the predictions. In Fig. 5 we show the track predictions overlayed on the initial image, corresponding to the goal image shown in the top row. The bottom row shows robot execution with the first frame (1), two intermediate frames (2,3) and the last frame (4) of a rollout. We can see that the predicted tracks correspond to points on the object moving in a way that satisfies the goal, and the policy is able to manipulate the object to the desired goal configuration. Since the camera doesn't move between the initial and goal images, we can see that background (non object) pixels remain stationary in the predictions, which is useful for accurate prediction of rigid transforms of the object.

In Table 1 we show comparisons for robot manipulation experiments, respectively for each level of generalization. We evaluate each approach for 20 rollouts in each level, across a total of 25 tasks in 5 different physical kitchen, office, and living room locations. We first note that our residual policy outperforms our approach for directly executing an open-loop plan based on predicted rigid transforms. This suggests that the residual policy is able to correct for inaccuracies in the open-loop

Table 1: Evaluation of goal-conditioned robot manipulation experiments, per the protocol described in section 3.1. The numbers denote success rate averaged over 20 rollouts for different tasks within each generalization axis (Higher is better). Detailed list of tasks are in the Supplementary pdf. Refer to Fig. 6 for visualizations of some task rollouts corresponding to each of the generalization axes.

	MG	G	CG	TG
Behavior Cloning (BC)	60%	20%	0%	0%
Affordance-Conditioned	65%	30%	10%	5%
Video-Conditioned	60%	25%	0%	0%
Hand-Object Mask-Conditioned	70%	40%	25%	20%
***Track2Act*(Open-Loop)**	35%	25%	30%	25%
***Track2Act*(Ablation; actions not residuals)**	70%	45%	30%	30%
Track2Act	70%	60%	55%	40%

plan by virtue of leveraging some embodiment-specific data that helps in performing accurate grasps on objects and recovering from potential failures during a trajectory.

We observe that for mild generalization (MG), the goal-conditioned BC baseline has slightly lower success rate compared to our residual policy, and significantly lower (or zero) success rates for standard (G), compositional (CG), and type (TG) generalization. This suggests the benefit of leveraging web video data for learning interaction plans that helps our approach generalize effectively. Finally, compared to baselines that also leverage web data like affordance-conditioned BC, video-conditioned BC, and hand-object mask-conditioned BC, we observe significant gains from our approach in the higher levels of generalization (CG and TG). This suggests that predicting static affordances without reasoning about motion trajectories, hallucinating RGB videos that suffer from incorrect generations and produce implausible artifacts in the scene, or predicting 2D masks of hands and objects without reasoning about correspondences are insufficient cues for effectively leveraging web videos. Compared to these, our interaction plan learned through track prediction provides sufficient cues for solving unseen manipulation tasks by virtue of allowing inference of 3D rigid transforms, and the residual policy helps correct for errors in the predictions.

4.1 Analysis of Failures

Here we discuss the failures displayed by *Track2Act*. For the open-loop plan based on residual transforms, the main failure modes we observe are inability to grasp the object at the right location, and inability to recover from intermediate failures. The residual policy corrects for these behaviors by virtue of leveraging some embodiment-specific data, and thus has higher success rates. We note that the success rate for higher levels of generalization CG and TG) is still not very high since these are very challenging settings and the residual policy sometimes fails by incorrectly grasping the object, getting stuck during the execution by trying to execute a non-feasible motion, or by executing a trajectory that does not conform with the goal image specified.

5 Discussion and Conclusion

In this paper, we developed *Track2Act* - a framework for generalizable zero-shot robot manipulation by leveraging large-scale web video data to learn embodiment agnostic plans of how objects should be manipulated in a scene to satisfy a goal. We combined this with a small amount of embodiment-specific data to learn residual corrections over the predicted plans through a closed-loop policy. Our real world manipulation results across a range of diverse tasks with varying levels of generalization demonstrate the potential of scalably leveraging web data to predict plans for object manipulation. While our framework allows for strong generalization to unseen tasks in-the-wild, the tasks are still of short-horizon and involve manipulating a single object in the scene. It would be an interesting direction of future work to extend our framework for tackling long-horizon tasks that involve successive manipulations of multiple objects in the scene.

References

[1] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. *ICRA*, 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[3] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *CoRL*, 2023.

[4] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. 2022.

[5] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[6] H.-S. Fang, H. Fang, Z. Tang, J. Liu, J. Wang, H. Zhu, and C. Lu. Rh20t: A robotic dataset for learning diverse skills in one-shot. *arXiv preprint arXiv:2307.00595*, 2023.

[7] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. *RSS*, 2022.

[8] N. M. Mahi Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv e-prints*, pages arXiv–2311, 2023.

[9] H. Bharadhwaj, A. Gupta, V. Kumar, and S. Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans. *ICRA*, 2024.

[10] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *NeurIPS*, 2024.

[11] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023.

[12] W. Peebles and S. Xie. Scalable diffusion models with transformers. *ICCV*, 2023.

[13] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[14] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik. Intel realsense stereoscopic depth cameras. *CVPR*, 2017.

[15] S. Bahl, R. Mendonca, L. Chen, U. Jain, and D. Pathak. Affordances from human videos as a versatile representation for robotics. In *CVPR*, 2023.

[16] T.-J. Fu, L. Yu, N. Zhang, C.-Y. Fu, J.-C. Su, W. Y. Wang, and S. Bell. Tell me what happened: Unifying text-guided video completion via multimodal masked video generation. *CVPR*, 2023.

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

[18] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. *ECCV*, 2018.

[19] R. Goyal, S. Ebrahimi Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. *CVPR*, 2017.

[20] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. 2023.

[21] P. Das, C. Xu, R. F. Doell, and J. J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. *CVPR*, 2013.

[22] F. De la Torre, J. Hodgins, A. Bargteil, X. Martin, J. Macey, A. Collado, and P. Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. 2009.

[23] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. *CVPR*, 2022.

[24] D. Shan, J. Geng, M. Shu, and D. Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020.

[25] Y. Li, M. Liu, and J. M. Rehg. In the eye of beholder: Joint learning of gaze and actions in first person video. *ECCV*, 2018.

[26] C. Zimmermann and T. Brox. Learning to estimate 3d hand pose from single rgb images. In *CVPR*, 2017.

[27] U. Iqbal, P. Molchanov, T. Breuel Juergen Gall, and J. Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *ECCV*, 2018.

[28] A. Spurr, J. Song, S. Park, and O. Hilliges. Cross-modal deep variational hand pose estimation. In *CVPR*, 2018.

[29] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan. 3d hand shape and pose estimation from a single rgb image. In *CVPR*, 2019.

[30] S. Baek, K. I. Kim, and T.-K. Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *CVPR*, 2019.

[31] A. Boukhayma, R. d. Bem, and P. H. Torr. 3d hand shape and pose from images in the wild. In *CVPR*, 2019.

[32] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019.

[33] D. Kulon, R. A. Guler, I. Kokkinos, M. M. Bronstein, and S. Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *CVPR*, 2020.

[34] S. Liu, H. Jiang, J. Xu, S. Liu, and X. Wang. Semi-supervised 3d hand-object poses estimation with interactions in time. In *CVPR*, 2021.

[35] Y. Rong, T. Shiratori, and H. Joo. Frankmocap: Fast monocular 3d hand and body motion capture by regression and integration. *arXiv preprint arXiv:2008.08324*, 2020.

[36] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. *ICCV*, 2017.

[37] M. Rad and V. Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *ICCV*, 2017.

[38] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv*, 2018.

[39] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6d object pose estimation. *CVPR*, 2019.

[40] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. *CVPR*, 2020.

[41] T. Nagarajan, C. Feichtenhofer, and K. Grauman. Grounded human-object interaction hotspots from video. *CVPR*, 2019.

[42] S. Liu, S. Tripathi, S. Majumdar, and X. Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. *CVPR*, 2022.

[43] M. Goyal, S. Modi, R. Goyal, and S. Gupta. Human hands as probes for interactive object understanding. *CVPR*, 2022.

[44] K. Mo, L. J. Guibas, M. Mukadam, A. Gupta, and S. Tulsiani. Where2act: From pixels to actions for articulated 3d objects. *CVPR*, 2020.

[45] S. Brahmbhatt, A. Handa, J. Hays, and D. Fox. Contactgrasp: Functional multi-finger grasp synthesis from contact. *arXiv*, 2019.

[46] W. Yan, D. Hafner, S. James, and P. Abbeel. Temporally consistent transformers for video generation. *arXiv preprint arXiv:2210.02396*, 2022.

[47] A. Gupta, L. Yu, K. Sohn, X. Gu, M. Hahn, L. Fei-Fei, I. Essa, L. Jiang, and J. Lezama. Photorealistic video generation with diffusion models. *arXiv preprint arXiv:2312.06662*, 2023.

[48] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

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

[50] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. *CVPR*, 2008.

[51] C. Doersch, A. Gupta, L. Markeeva, A. Recasens, L. Smaira, Y. Aytar, J. Carreira, A. Zisserman, and Y. Yang. Tap-vid: A benchmark for tracking any point in a video. *NeurIPS*, 2022.

[52] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[53] S. Young, D. Gandhi, S. Tulsiani, A. Gupta, P. Abbeel, and L. Pinto. Visual imitation made easy. *CoRL*, 2020.

[54] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. *CoRL*, 2018.

[55] H. Bharadhwaj, A. Gupta, and S. Tulsiani. Visual affordance prediction for guiding robot exploration. *ICRA*, 2023.

[56] D. Seita, Y. Wang, S. J. Shetty, E. Y. Li, Z. Erickson, and D. Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. *CoRL*, 2023.

[57] C. Pan, B. Okorn, H. Zhang, B. Eisner, and D. Held. Tax-pose: Task-specific cross-pose estimation for robot manipulation. *CoRL*, 2023.

[58] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017.

[59] Z. Qin, K. Fang, Y. Zhu, L. Fei-Fei, and S. Savarese. Keto: Learning keypoint representations for tool manipulation. *ICRA*, 2020.

[60] M. Vecerik, C. Doersch, Y. Yang, T. Davchev, Y. Aytar, G. Zhou, R. Hadsell, L. Agapito, and J. Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023.

[61] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

[62] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[63] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *ICLR*, 2023.

[64] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, P. Abbeel, J. Malik, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *arXiv preprint arXiv:2303.18240*, 2023.

[65] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022.

[66] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[67] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[68] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[69] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[70] K. Shaw, S. Bahl, and D. Pathak. Videodex: Learning dexterity from internet videos. *CoRL*, 2023.

[71] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. *CoRL*, 2023.

[72] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv*, 2019.

[73] 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. *IROS*, 2021.

[74] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, F. Yu, D. Tao, and A. Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Appendix

6 Related Works

Understanding interactions from videos. Several computer vision methods have investigated deciphering interactions between hands and objects across various daily activities through curation of large-scale video datasets [19, 21, 22, 23, 24, 18, 25], hand pose estimation [26, 27, 28, 29, 30, 31, 32, 33, 34, 35], object pose estimation [36, 37, 38, 39, 40], interaction hotspot/grasp prediction [41, 42, 43, 44, 45]. Some approaches have investigated *generating* videos given a description of the task, and often conditioned on a scene [46, 10, 47, 48]. Other works have attempted understanding generic videos by identifying visual correspondences between frames [49, 50]. Recent approaches have made significant advances on this problem by developing general-purpose video tracking systems that can track points specified in a frame, across other frames in the video [51, 11]. Our track prediction model is inspired by these developments, and is based on leveraging the video tracking approaches to generate ground-truth tracks from web videos, and training a model to *predict* future points tracks of arbitrary points given an initial image and a goal.

Learning Visual Representations for Manipulation. Visual imitation is a promising technique for generalizable robot manipulation [52, 53, 54, 55]. Recent works that have scaled this approach for learning large-scale models for manipulation require extremely high number of expert robot trajectories, often demanding years for collection [2, 3, 1], and still suffer from limited generalization to unseen scenarios for novel objects. Going beyond image observations, prior works have also investigated structured representations like point-clouds [56, 57, 58] and keypoints [59] for manipulation, but are restricted to tasks in structured table-top scenarios. Some of these that predict action in the form of flow-based representations [56, 49] require 3D datasets of robot interactions (often from simulation) which constrain them from generalizable real-world deployments. More recently, Vecerik et al. [60] use point tracking for visual servoing, and the setup requires structured multi-stage definitions of the task and is limited to only minor test-time variations compared to training data. Concurrent work [61] that improves upon [60] by predicting future tracks of points in the current image can learn a policy by combining in-domain human videos with in-domain robot videos. However, the framework is not directly amenable for leveraging web videos because the policy relies on per-step image observations for track prediction. Compared to this, and developed independently, we learn to predict trajectories of arbitrary points from web videos given just an initial image and a goal. We show how we can use these predicted tracks to infer rigid transforms of objects for open-loop execution, and further improve the open-loop plan by predicting residuals over the actions, for closed-loop deployment. This enables much diverse robot manipulation behaviors with a single model, that generalizes to unseen novel objects and scenes in-the-wild.

Leveraging Non-Robot Datasets for Manipulation. One common way of using data beyond robot interactions for efficient learning is to pre-train the visual representations which serve as backbones for the policy models [62, 63, 64, 65, 66] with passive human videos [23, 67] and image data [68]. However, these methods still crucially rely on a lot of in-domain robot data or deployment-time training, and are restricted to learning task-specific policies. Some works that do not require deployment-time training, go beyond visual representations and use curated data of human videos to leverage human hand motion information [69, 70] for learning task-specific policies (instead of a single model across generic tasks). Others that train a single policy across tasks require large in-domain perfectly aligned human-robot data [71, 72, 73] and are not capable of leveraging web data. Towards learning structure more directly related to manipulation from web videos, some works try to predict visual affordances in the form of where to interact in an image, and local information of how to interact [44, 43, 15, 42]. While these could serve as good initializations for a robotic policy, they are not sufficient on their own for accomplishing tasks, and so are typically used in conjunction with online learning, requiring several hours of deployment-time training and robot data [7, 15]. Others learn to predict masks of hand and objects in the scene [9] for conditional behavior cloning and are unable to leverage information of accurate object state changes that is usually ambiguous with a mask. Our work differs from these in terms of predicting an approximate motion of how objects in

Algorithm 1 Predicting Rigid Transforms from Point Tracks

procedure RIGID TRANSFORMS($\tau, \mathbf{I_0}, \mathcal{G}, P_0^{\text{3D}}, \mathbf{K}, H$)
  $\{\{(x_t^i, y_t^i)\}_{i=1}^p\}_{t=1}^H = \tau_{obj} = \texttt{filter}(\tau)$ ▷ Filter moving point tracks
  Unknown rigid transforms $[\mathbf{T}_t]_{t=1}^H$ ▷ $\mathbf{T}_t$ has dimension 3x4
  Run RANSAC on τ_{obj} to filter outliers ▷ optional
  for $t \leftarrow 1$ to H **do**
    $\mathbf{T}_t = \arg\min_{\mathbf{T}_t} \sum_i^N (||x_t^i - u_t^i|| + ||y_t^i - v_t^i||)$
    where $(u_t^i, v_t^i, 1) \simeq \mathbf{K}\mathbf{T}_t P_t$ ▷ projections in homogeneous coordinates
  return $\{\mathbf{T}_t = (\mathbf{R}_t, \mathbf{t}_t)\}_{t=1}^T$

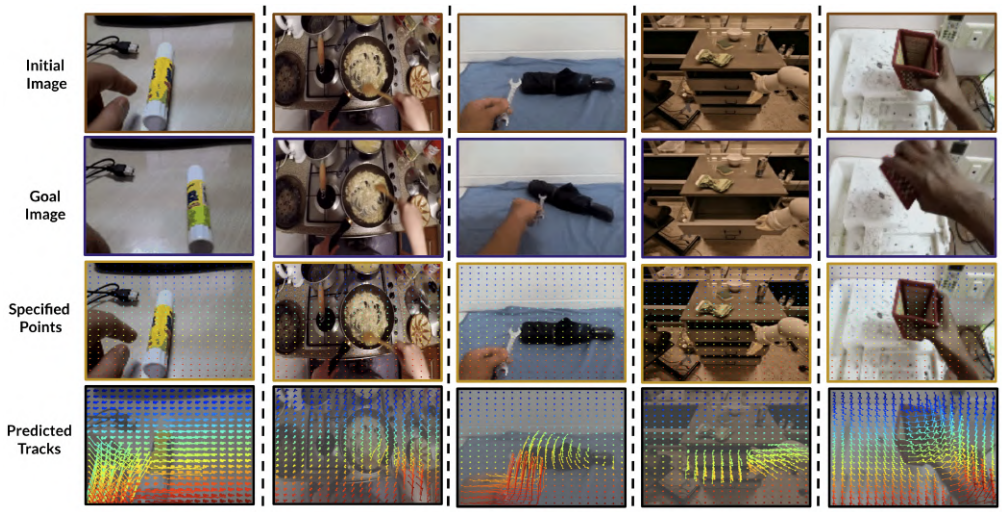

Figure 7: We show qualitative results of the track predictions for *Track2Act* on unseen initial and goal images across diverse datasets. Given specified points on the initial image we predict future tracks of these points, corresponding to the goal image. We can see that the predictions are plausible and correspond to manipulating the object(s) in the scene.

the scene move in the future through point tracks for the entire trajectory and is *zero-shot* in terms of not requiring any deployment-time training.

6.1 Robot Experiment Details

We perform all robot manipulation experiments with a Boston Dynamics Spot Robot, operated through end-effector control. The robot is a quadruped with an arm attached to its base. We connect a front-facing Intel Realsense camera to the base such that it always moves with the robot, and it static with respect to the base. The end-effector of the arm is a two-fingered gripper. The horizon H of rollouts is 50 steps, and we operate the robot at a frequency of 5 Hz. For the residual policy, at each step we predict actions $h = 4$ time-steps in the future, and execute the first action. We execute the predicted actions on the robot through an Inverse Kinematics (IK) controller. This controller converts the end-effector poses to robot joint actions for appropriately manipulating the arm. We use the IK controller provided by Boston Dynamics for this purpose.

6.2 Point Track Prediction Results

Evaluation detials. For quantitative evaluation of the track prediction model, we adopt a modification of the metric developed by prior works [11, 51], δ_t^x. For evaluation videos, we consider the output of Co-Tracker [11] to be the ground-truth and compare the difference with respect to the predictions, based on the δ_t^x metric. We define δ_t^x to be the fraction of points that are within a threshold pixel distance of x of their ground truth in a time-step t. We report the *area under the curve* Δ

Table 2: Evaluation of track prediction performance on held-out videos from different datasets on the web. EpicKitchens [18] and SmthSmthv2 [19] are datsets of human videos, and BridgeData [20] and RT1 data [2] are datasets of robot videos. Note that we train a *single* model that we evaluate on these different datasets. The metric Δ is defined in section 3.1. Higher is better and the range is from 0 to 1.

	EpicKitchens [18]	**SmthSmthv2** [19]	**BridgeData** [20]	**RT1 Data** [2]
Flow [74]	0.21	0.27	0.42	0.38
Video [16]	0.30	0.17	-	-
Track2Act	0.67	0.70	0.77	0.75

with δ_t^x by varying x from 1 to $N = 10$ and taking the average across the prediction horizon H i.e. $\Delta = (\sum_{t=1}^{H} \sum_{x=1}^{N} \delta_t^x)/H$. Hence, Δ can vary from 0 to 1 with higher being better.

We perform comparisons with two baselines that have the same inputs as our track prediction model, i.e. an initial image, a goal image and points specified on the initial image, and the same output type i.e. point tracks in between the initial and goal images. We compare with a flow-based baseline that directly predicts flow between the initial and goal images, and then performs a per-timestep interpolation of the flow vectors [74]. The second baseline performs video-infilling given initial and goal images [16], and then uses Co-Tracker [11] to obtain tracks on the generated video.

We now look at some qualitative results of the track prediction model in different unseen scenes. In Fig. 7 we show visualization of track predictions on unseen initial and goal images across diverse datasets. We choose points on a grid in the initial frame, as shown in the third row. The prediction model is conditioned on the initial image, the goal image, and the set of points in the initial image whose future tracks are to be predicted. We can see that the predictions (shown in the fourth row) are plausible and correspond to manipulating the objects in the scene as described by the respective goal images. We can also see that when multiple entities (e.g. human and object or robot and object) or the camera moves between the initial and goal images, there are different sets of point tracks predicting the respective motions.

In Table 2 we perform evaluations for track prediction by comparing with the flow-based [74] and video-based [16] baselines. We can see that both the baselines have much lower accuracy compared to our approach of predicting point tracks. This is because flow is too coarse to capture large non-linear state changes in between the initial and goal images. Whereas, predicting an RGB video followed by tracking suffers due to issues of implausible generation because video generation is a much more complex task than predicting the tracks of a set of points where the details about appearance, texture etc. are abstracted out. For reference, not predicting any movement for any point at all time-steps scores 0.03, 0.05, 0.36, 0.28. This suggests the benefit of directly predicting future point tracks as done by our approach if the aim is to capture motion of objects in the scene between an initial and a goal image.

6.3 Track Prediction Model details

We instantiate track prediction as a denoising process through a DiT based diffusion model [12]. Let $\mathbf{I_0}$ denote the first frame of a video, and $\mathcal{G}$ denote the goal, which we consider to be the last frame of the video. For longer videos, we obtain multiple video clips of 4-5 seconds each for training. Let there be p points in the initial frame to be tracked, such that P_0 denotes the set of those points and let H be the prediction horizon. $[P_t]_{t=1}^{H}$ denotes the future locations of those points in the subsequent time-steps that we want to predict. In the forward diffusion process, all the points P_t are corrupted by incrementally adding noise ϵ_k (k denotes the diffusion time-step), to obtain $\tilde{P}_t$, and converging to a unit Gaussian distribution $N(\mathbf{0}, \mathbf{I})$. New samples can be generated by reversing the forward diffusion process, by going from Gaussian noise back to the space of point locations. To solve the reverse diffusion process, we need to train a noise predictor $\mathcal{V}_\theta(\mathbf{I_0}, \mathcal{G}, P_0, k)$. We design a DiT Transformer based architecture [12] for $\mathcal{V}_\theta$ illustrated visually in Fig. 3. Different from the original DiT model, we condition on embeddings of initial (z_0) and goal (z_g) images in addition to

Algorithm 2 Closed-Loop Deployment with Residual Policy Correction

procedure RESIDUAL CORRECTION($\mathcal{V}_\theta(\cdot), \mathbf{I_0}, \mathcal{G}, P_0, P_0^{\text{3D}}, \mathbf{K}, h, \mathbf{e}_0, \pi_{\text{res}}$)
  $\tau = \mathcal{V}_\theta(\mathbf{I_0}, \mathcal{G}, P_0)$ ▷ Predict future point tracks
  $[\mathbf{T}_t]_{t=1}^{H}$ = RIGID TRANSFORMS($\tau, \mathbf{I_0}, \mathcal{G}, P_0^{\text{3D}}, \mathbf{K}, h$)
  for $t \leftarrow 0$ to H **do**
    $\bar{\mathbf{a}}_t = \mathbf{T}_t \mathbf{e}_0$ ▷ Infer Open-Loop Plan
  for $t \leftarrow 1$ to H **do**
    $\Delta \mathbf{a}_{t:t+h} = \pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=t}^{t+h})$ ▷ Predict residual correction
    $\hat{\mathbf{a}}_t = \bar{\mathbf{a}}_t + \Delta \mathbf{a}_t$ ▷ Corrected action at time t
    Execute action $\hat{\mathbf{a}}_t$ with the robot ▷ with IK and gripper action
    if t == H **then**
      Gripper Open; Reset end-effector to home

that of the diffusion step (z_k). The input to the Transformer in each batch is a sequence of p tokens corresponding the number of points specified for tracking. The initial P_0 points are not noisy, as is the convention in training conditional diffusion models on time-series data. We train the prediction model with web videos by considering variable number of initial points p that need to be tracked. We vary p from 200 to 400. For flexible modeling, the locations of the p points are also randomized, such that at test-time any set of points in the initial image can be specified. We do not make any assumptions on objects to be tracked or camera motions in the videos, and do not curate the training videos in any way apart from ensuring they are of 4-5 second duration.

The model has 24 DiT blocks, with a hidden size of 1024, and 16 heads. The ResNet18 embeddings of initial image and goal image have dimensions 512. The condition to each DiT block consists of the sum of initial image embedding, goal image embedding, and diffusion time-step embedding through adaptive modulation (adaLN) layers. The adaptive modulation layers and final MLP layers are zero-initialized, and the rest are Xavier uniform initialized. We use Adam optimizer with default Adam betas = (0.9,0.999) and a constant learning rate of 1e-4 for experiments. The rest of the architecture and training details are similar to DiT [12].

6.4 Residual Policy Model details

To correct the predicted open-loop plan, with a small amount of embodimen-specific data, we propose learning a residual policy $\pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=1}^{H})$ shown in Fig. 4 to correct the predicted end-effector poses in each time-step. So the end-effector pose at time t is $\hat{\mathbf{a}}_t = \bar{\mathbf{a}}_t + \Delta \mathbf{a}_t$; where $\Delta \mathbf{a}_t = \pi_{\text{res}}(\mathbf{I}_t, \mathcal{G}, \tau, [\bar{\mathbf{a}}_t]_{t=1}^{H})$ Instead of predicting just a single residual action $\Delta \mathbf{a}_t$ we predict residuals h steps in the future $\Delta \mathbf{a}_{t:t+h}$ and during deployment execute just the first action. This multi-step prediction has been shown to mitigate compounding errors in behavior-cloning based training [13, 1]. We can learn the residual policy with a small amount of robot demonstrations ($\sim$ 400 trajectories overall) of representative tasks through behavior cloning. The data for each trajectory consists of observation-action pairs of the form $[(\mathbf{I}_t, \mathbf{a}_t)]_{t=1}^{H}$. Here, $\mathbf{I}_t$ denotes images observed from the robot's camera and $\mathbf{a}_t$ denotes actions in the form of end-effector poses.

The residual policy model is a Transformer based on the DiT architecture. The model has 12 DiT blocks, with a hidden size of 512, and 8 heads. The ResNet18 embeddings of initial image and goal image have dimensions 512. The condition to each DiT block consists of the sum of current image embedding, goal image embedding, and emebedding of the current time-step t through adaptive modulation (adaLN) layers. The adaptive modulation layers and final MLP layers are zero-initialized, and the rest are Xavier uniform initialized. We use Adam optimizer with default Adam betas = (0.9,0.999) and a constant learning rate of 1e-4 for experiments. The input to the model consists of the predicted tracks of p points in the initial image (we keep $p = 400$ to ensure a dense grid in the initial image of dimensions 256x256x3) and the predicted open-loop plan with h steps from $t : t + h$. So there are $p + h$ input tokens. We read off the final h tokens corresponding to the updated open-loop plan for these h steps and after a final MLP layer, output actions for h steps . We will release all code and models upon acceptance.

6.5 Training Data for Track Prediction

We use four different web data sources for training the track prediction model - videos from Something-Something-v2 [19], Epic-Kitchens [18], RT1 data [2], and BridgeData [20]. Something-Something-v2 contains short YouTube videos of people doing everyday activities. We consider videos from this dataset as is, and choose the first frame as the initial image and the last frame as goal image. Epic-Kitchens contains ego-centric videos of humans in different locations performing diverse tasks in kitchens. Since these videos are long ($\geq$ 20 min each), we choose clips of duration 4-5 seconds by cutting the long videos, and choosing clips where a human hand is visible in the scene (so as to have clips where an object is being manipulated, instead of a person just moving around). RT1 Data and Bridge Data are large-scale robot datasets that contains rollouts of two different types of robots being tele-operated for different tasks. For these datasets, we consider the first and last images to be the first and last frames of a rollout, and each rollout to be a separate video.

In total we obtain around 400,000 videos clips from the above sources, choose a dense grid of 400 points on the first frame and we run Co-Tracker [11] on these clips, for obtaining the ground-truth intermediate tracks of points. Our prediction model is conditioned on the first and last frames for each video, and the task of predicting the tracks of random points on the initial frame is supervised by the tracks we obtain from Co-Tracker (ground-truth).

6.6 Training Data for Residual Policy

For training the residual policy we collected tele-operated demonstrations with the Spot robot by controlling it with a joystick across 10 tasks in 3 physical locations. These scenarios correspond to only a subset of the diverse tasks, objects, and scenes we consider for evaluation . Concretely, the evaluation scenarios with same tasks as the collected data correspond to the *mild generalization* (MG) category. Rest of the generalization axes corresponding to unseen instances and categories are described in detail in section 3.1.

The training data consists of 400 teleoperated trajectories, each consisting of H (observation,action) pairs ($H = 50$). The data for each trajectory consists of observation-action pairs of the form $[(\mathbf{I}_t, \mathbf{a}_t)]_{t=1}^{H}$. Here, $\mathbf{I}_t$ denotes images observed from the robot's camera and $\mathbf{a}_t$ denotes actions in the form of end-effector poses. This data is collected at the same frequency of 5 Hz that we deploy the policy for eventual evaluations. Note that this embodiment-specific data we collect is 3-4 orders of magnitude less than that what related works [2, 3, 1] require for policy learning. This is a major advantage of our framework as it precludes the need to spend years on real-world data collection, while achieving generalization to more diverse scenarios by virtue of leveraging *passive* web videos for track prediction.

6.7 Details on baselines

We perform several comparisons with baselines and ablation studies for goal-conditioned robot manipulation. For baselines, we use the same embodiment-specific demonstrations as *Ours*, the goal-conditioned policy that predicts residuals over open-loop actions at each time-step (Algorithm 2).

- *Goal-Conditioned BC* is a baseline for multi-task policy learning, similar to prior works [2, 3, 1]. This is trained with the same data we use for training our residual policy, and is conditioned on goal image, similar to our residual policy.
- *Affordance-Conditioned BC* is the approach from [15] that conditions the policy on predicted affordances in the initial image. These affordances capture what is *plausible* to be manipulated in the scene, and so are different from our time-series predictions of point tracks. We directly adopt the affordance model from [15] that was trained on web data, and use the same embodiment-specific data as our residual policy for training through conditional behavior cloning.

- *Video-Conditioned BC* based on [16, 10, 17] first predicts RGB video and then does tracking on top of it. We adopt the video prediction model from [16] (without language conditioning) trained on web data, and use the same embodiment-specific data as our residual policy for training through conditional behavior cloning.
- *Hand-Object Mask Conditioned BC* from [9] conditions the policy on a predicted interaction plan consisting of masks of hands and objects. We use the hand-object plan prediction model from [9], and use the same embodiment-specific data as our residual policy for training through conditional behavior cloning. Note that this baseline is slightly different from the translation model in [9] because we do not collect paired human-robot demonstrations unlike [9] and so the policy is conditioned on predicted hand-object plans as opposed to ground-truth plans unlike [9].

Comparison to *Goal-Conditioned BC* helps understand the potential benefits of leveraging web data for generalizable manipulation, and comparisons to *Affordance-Conditioned BC*, *Video-Conditioned BC*, *Hand-Object Mask Conditioned BC* help understand the potential of predicting point tracks from web videos, compared to other ways of using web data for prediction geared towards manipulation.

Ours (Open Loop) is the approach for track prediction followed by open-loop execution as described in Algorithm 1. This does not use any embodiment-specific data for training. To understand the benefit of predicting residuals over actions as opposed to predicting complete actions, we compare with an ablated variant *Ours (actions; not residuals)* that predicts actions $\hat{\mathbf{a}}_t$ directly without predicting residuals $\Delta\mathbf{a}_t$ and not relying on an open-loop plan as input.

6.8 Qualitative Results for baselines

We provide qualitative comparisons of the baselines with our approach, in the figures below. For detailed qualitative video results of our approach, please refer to the website.

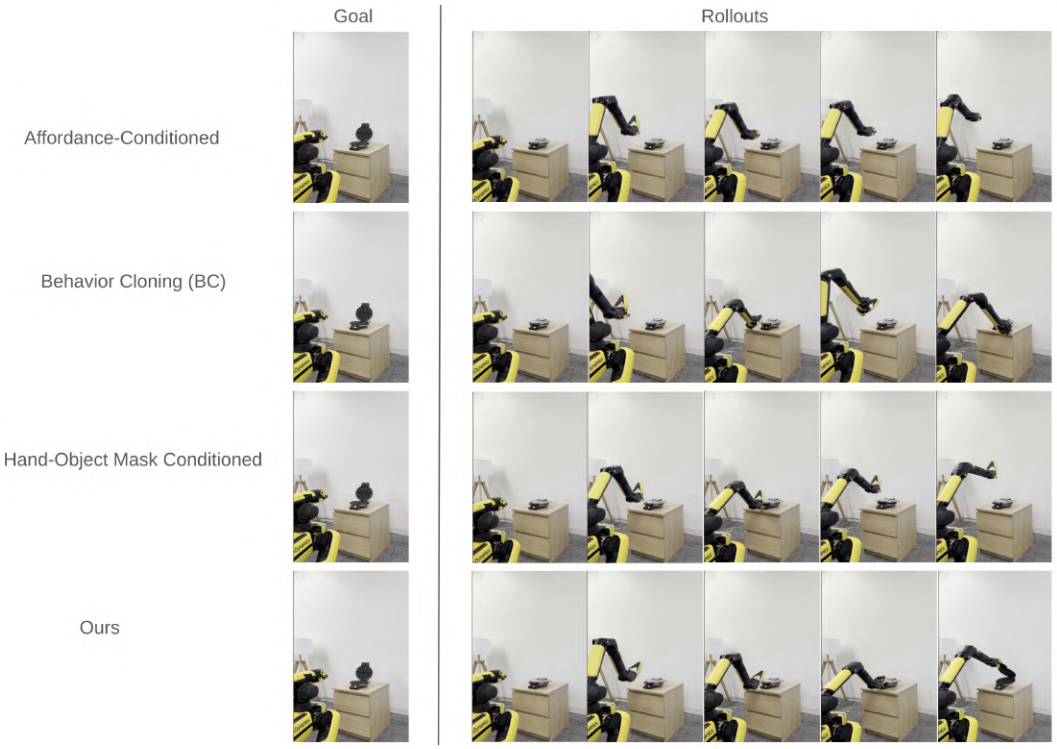

Figure 8: Type Generalization (TG). We show rollouts from baselines for the same goal. The views are from a third person camera.

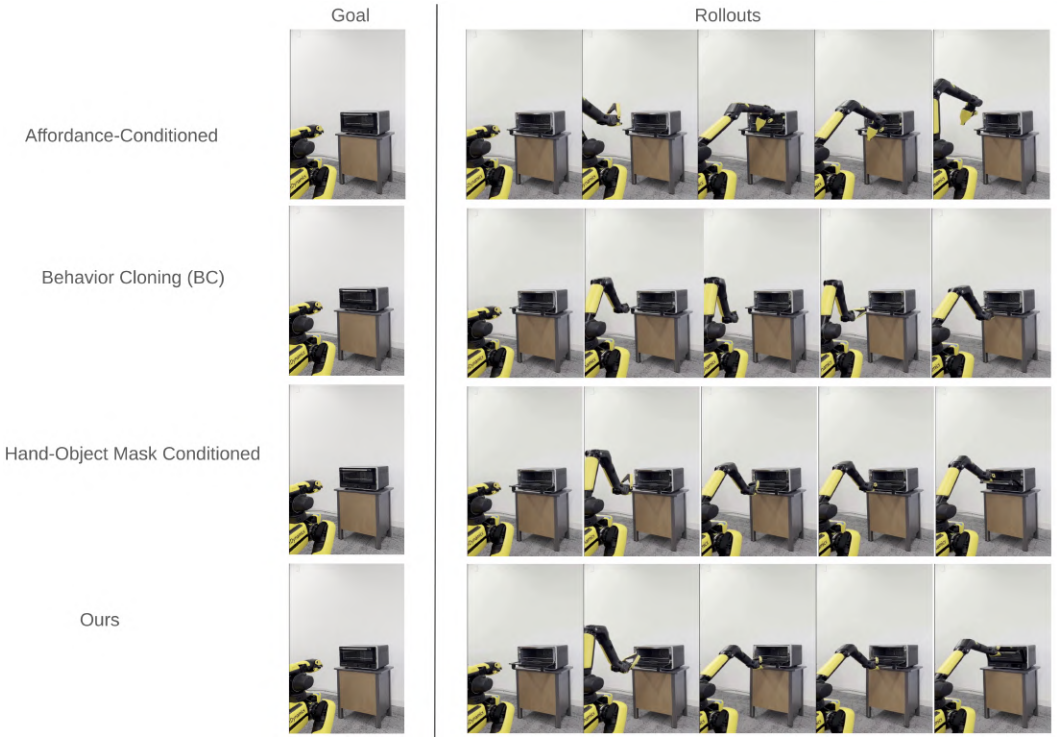

Figure 9: Compositional Generalization (CG). We show rollouts from baselines for the same goal. The views are from a third person camera.

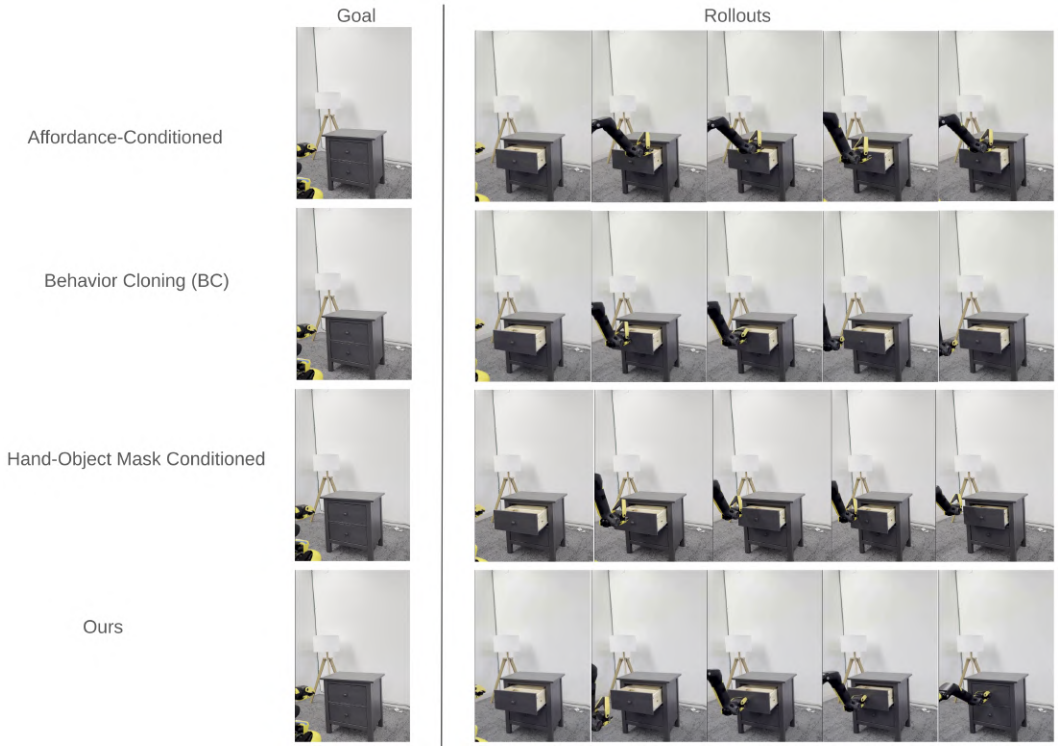

Figure 10: Standard Generalization (G). We show rollouts from baselines for the same goal. The views are from a third person camera.

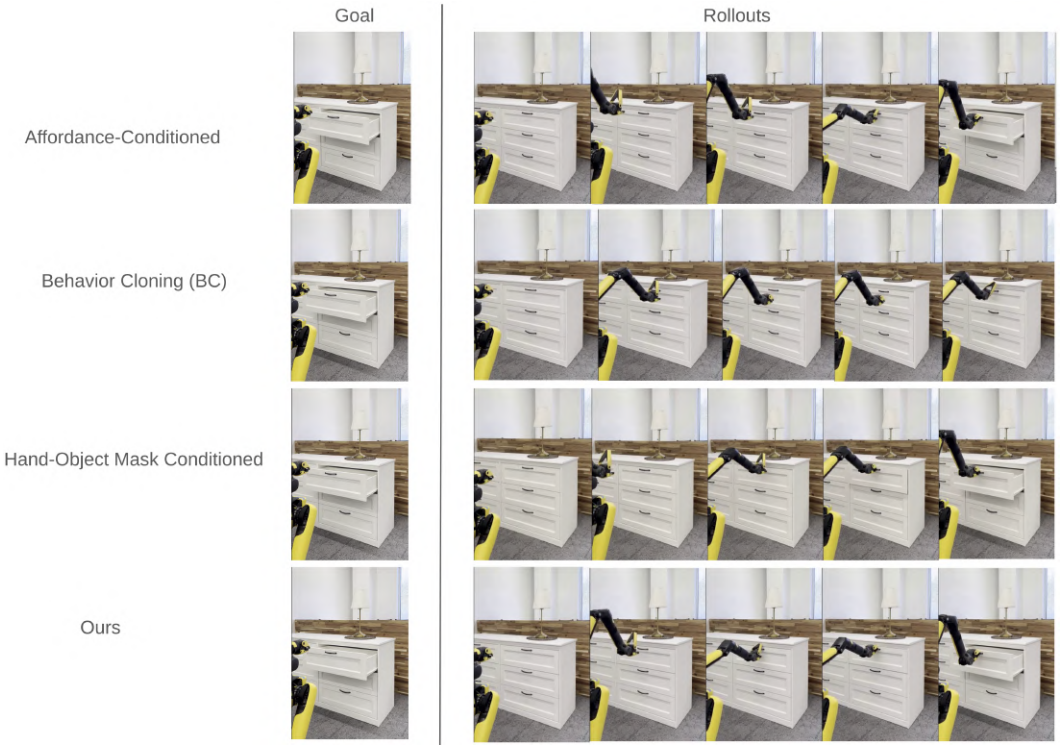

Figure 11: Mild Generalization (MG). We show rollouts from baselines for the same goal. The views are from a third person camera.

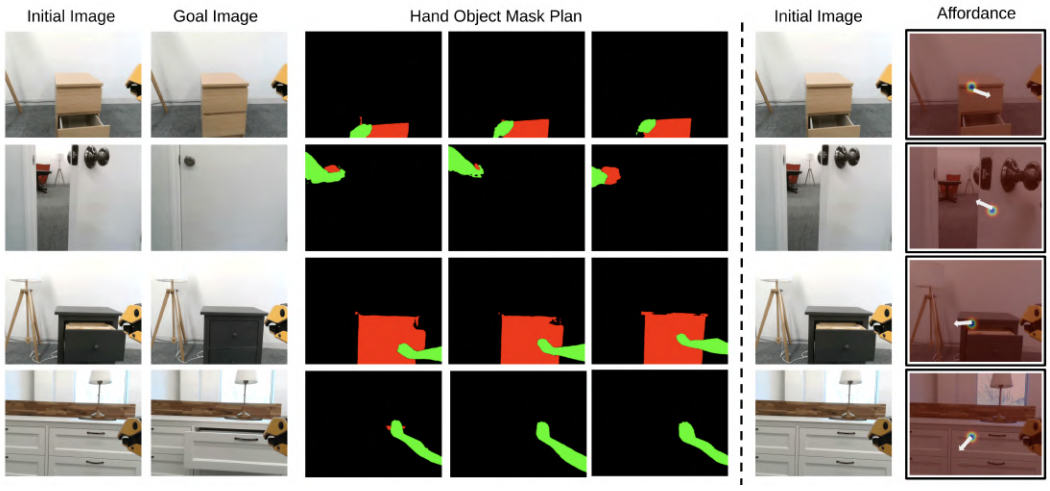

Figure 12: We show visualizations of predictions from the Hand-Object Mask Prediction and Affordance Prediction baselines, on different initial and goal images in the robot's environment.