Appendix

6.1 Video results

Please refer to the supplementary website https://homangab.github.io/track2act/ for detailed qualitative results of our framework including robot video evaluations.

6.2 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.

6.3 Track Prediction Model details

We instantiate track prediction as a denoising process through a DiT based diffusion model [68]. 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 [68] 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. 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 [68].

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 [69, 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 [12], Epic-Kitchens [17], RT1 data [2], and BridgeData [74]. 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 4.1.

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 [67] 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 [67] 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 [72, 10, 73] first predicts RGB video and then does tracking on top of it. We adopt the video prediction model from [72] (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.

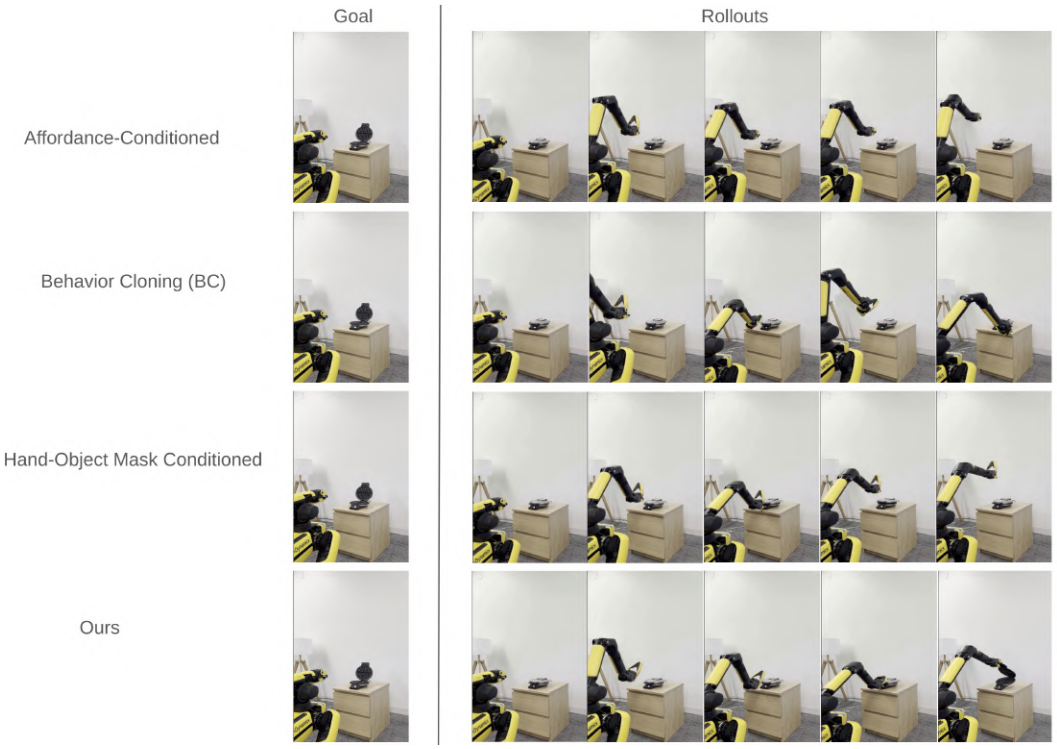

Figure 8: Type Generalization (TG). We show rollouts from baselines for the same goal. The views are from a third person camera.

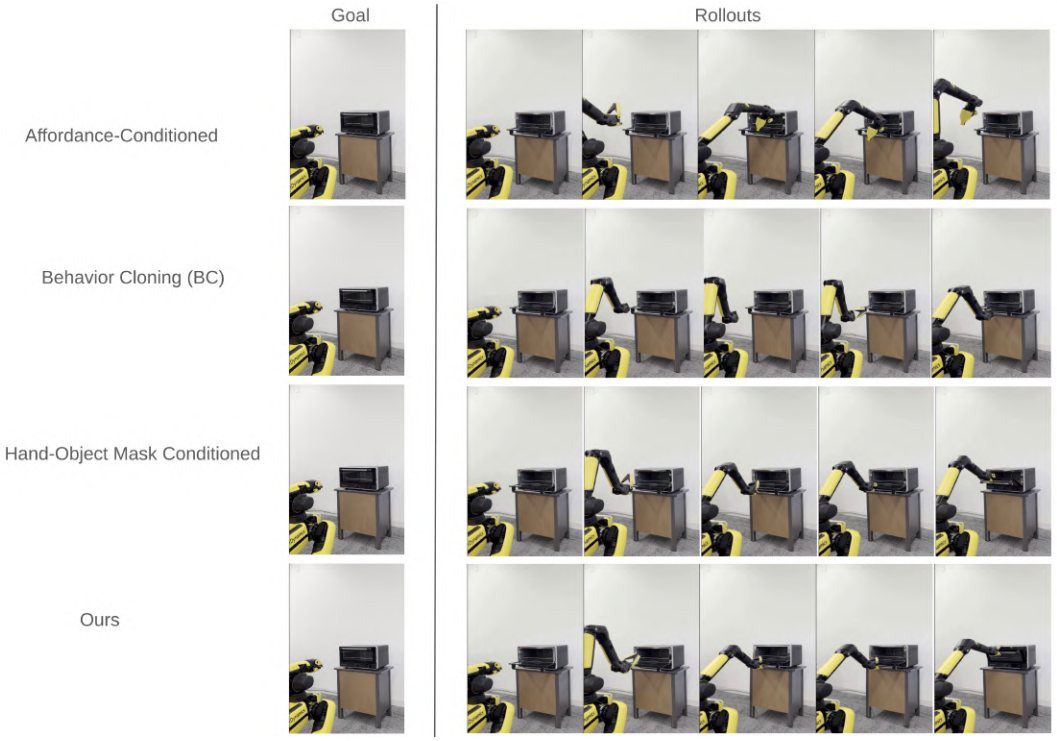

Figure 9: Compositional Generalization (CG). We show rollouts from baselines for the same goal. The views are from a third person camera.

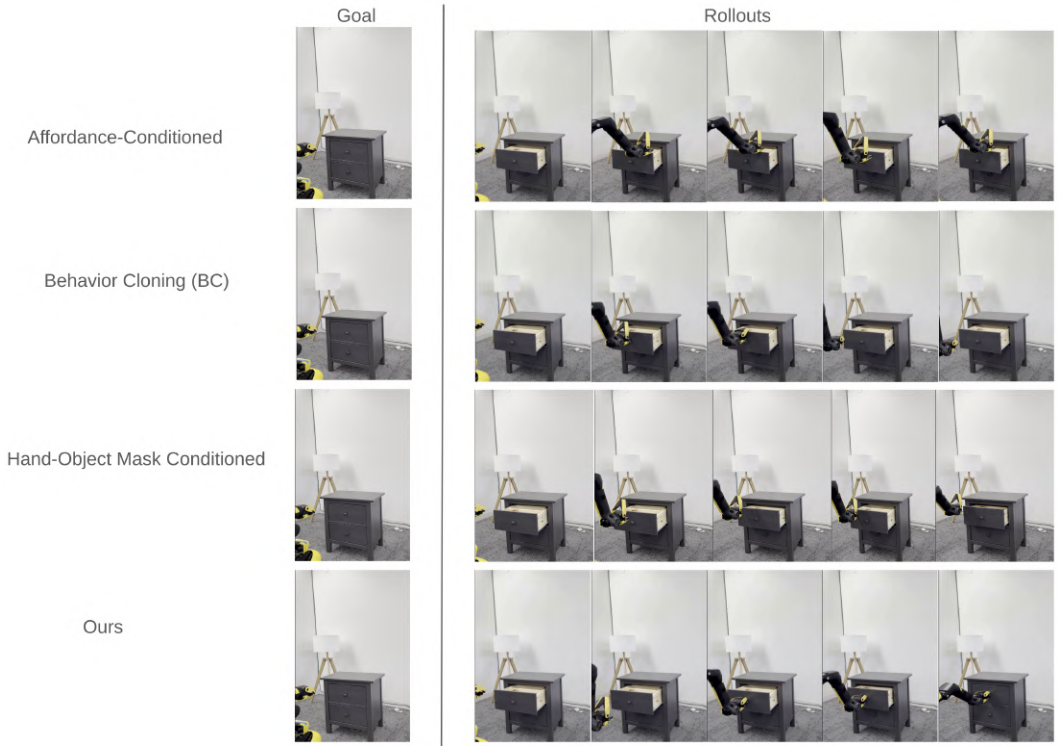

Figure 10: Standard Generalization (G). We show rollouts from baselines for the same goal. The views are from a third person camera.

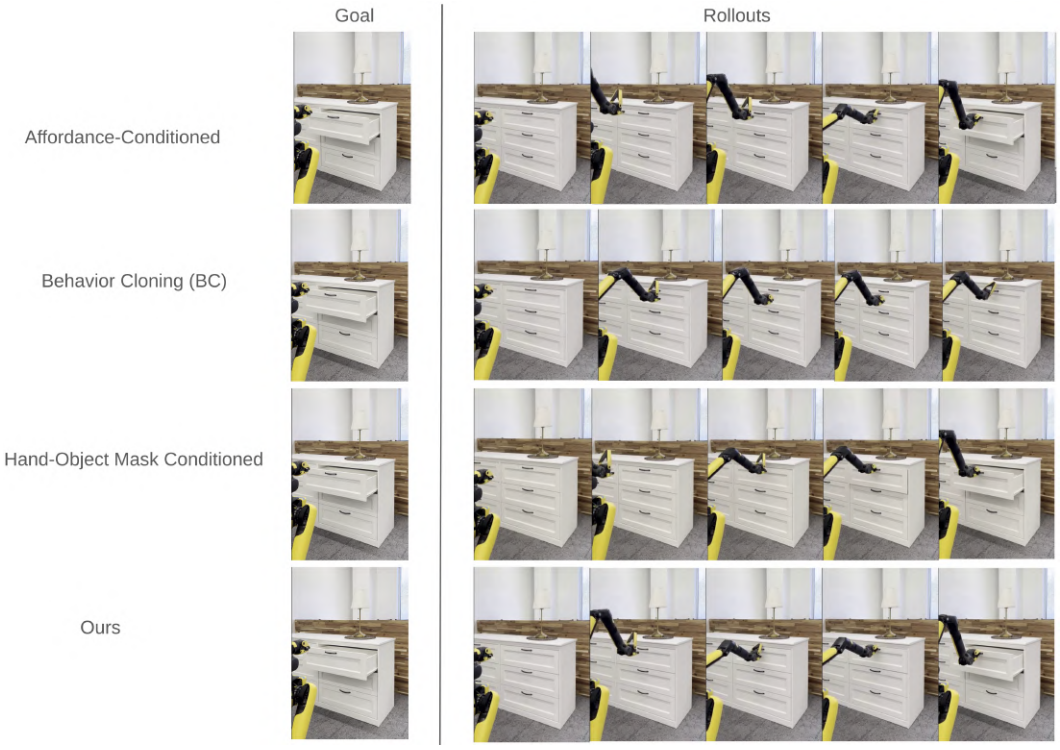

Figure 11: Mild Generalization (MG). We show rollouts from baselines for the same goal. The views are from a third person camera.

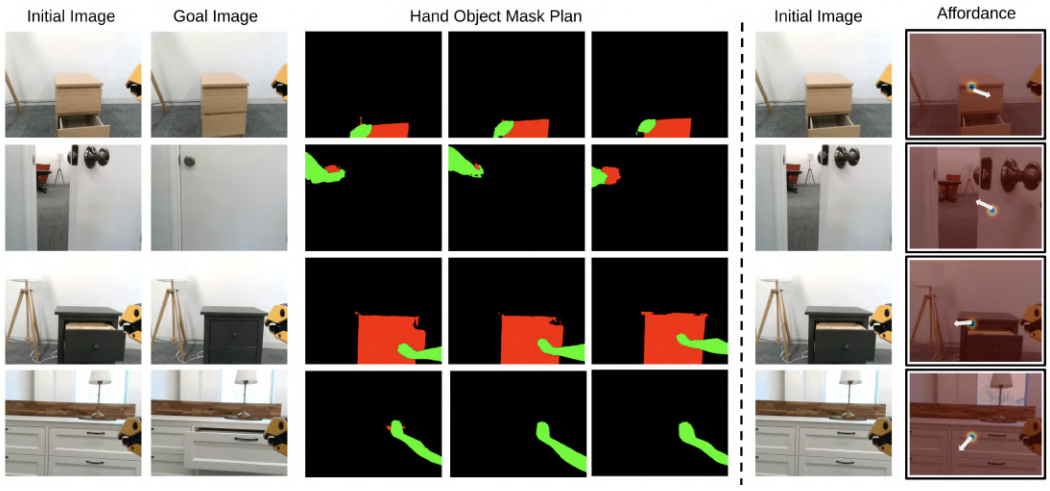

Figure 12: We show visualizations of predictions from the Hand-Object Mask Prediction and Affordance Prediction baselines, on different initial and goal images in the robot's environment.