Spatially Prompted Visual Trajectory Prediction for Egocentric Manipulation

Yifan Li[1]*, **Xinyu Zhou**[1]*, **Yunhao Ge**[2], **Yu Kong**[1]
[1]Michigan State University, [2]NVIDIA Research

Abstract

Robotic manipulation is often specified through language instructions or task identifiers, yet cluttered environments with similar objects are better handled by spatially indicating what to move and where to place it. Addressing the vision-centric challenge of object and goal specification, we present, to the best of our knowledge, the first formalization of *Spatially Prompted Visual Trajectory Prediction* (SP-VTP). This novel setting utilizes initial spatial prompts (like bounding boxes or points) to define task objectives, tasking the model with forecasting future end-effector trajectories from egocentric streams. To study this problem, we collect and annotate *EgoSPT*, a dataset of egocentric spatially prompted manipulation trajectories with first-frame object and target grounding annotations and recovered 3D end-effector motion. SP-VTP is challenging because the task specification is static, while the scene configuration evolves over time. To solve this problem, we propose *SPOT* (*Spatially Prompted Object-Target Policy*), which combines a task encoder for first-frame visual and coordinate spatial prompts, an observation encoder for current visual and history context, and a trajectory generator for future end-effector motion. Experiments under strict scene-level splits show that SPOT improves cross-scene trajectory prediction over non-prompted or single-source prompted baselines. Together, EgoSPT and SPOT establish a new spatial prompting problem SP-VTP, as a simple and scalable task condition for egocentric manipulation.

1 Introduction

Most robot policies rely on language [30, 39, 40, 58, 19] or task identifiers [53, 52], which are expressive but often indirect for specifying manipulation goals. This limitation is pronounced in cluttered scenes with visually similar objects, where "the fork" or a task ID may fail to identify the intended instance or placement region. In such cases, the goal is spatial before it is linguistic: pick *this* object and place it *there*. A point or box prompt on the first egocentric frame provides a direct, low-ambiguity interface while preserving the visual context needed for action.

We formalize this setting as *Spatially Prompted Visual Trajectory Prediction* (SP-VTP). Given first-frame spatial prompts, such as points or boxes indicating the object and target, the model predicts future end-effector (EE) trajectories from streaming egocentric observations. Unlike grounding or tracking, the output is not a mask or box, but a sequence of relative 6D EE motions and gripper states. SP-VTP therefore requires converting sparse visual intent into a temporally extended motion plan.

Some recent hierarchical VLA systems also use spatial planning intermediates for manipulation [22, 49, 45, 54, 51, 47, 48]. For example, HAMSTER predicts coarse 2D image-plane paths from RGB observations and language instructions, which then guide a separate low-level 3D-aware controller [22]. In contrast, SP-VTP does not rely on language instructions and language-conditioned spatial planning intermediates. It only assumes lightweight object–target spatial grounding in the first

*Equal contribution. Correspondence to: liyifa11@msu.edu.

Preprint.

egocentric frame, from which the model predicts future 3D EE trajectory chunks using subsequent egocentric observations. Our focus is therefore spatial prompt-conditioned visual trajectory prediction, rather than language-conditioned spatial planning followed by low-level control. SP-VTP combines four challenges that are typically studied in isolation. First, task specification is static: the object and target are provided only in the first frame. Second, execution is dynamic: the camera moves, the end effector occludes the scene, and the object relocates after grasping. Third, clutter and same-category distractors require persistent instance-level disambiguation. Finally, the same object–target pair can require different motions at different execution phases. Thus, a policy must infer not only what to do, but also where the relevant entities are now and how far the task has progressed.

To study this problem, we introduce *EgoSPT*, an egocentric dataset of spatially prompted manipulation trajectories collected with a modified Universal Manipulation Interface (UMI) [5]. EgoSPT provides first-frame object and target grounding annotations, egocentric visual observations, recovered 3D EE trajectories, and scene/subscene splits for evaluating generalization. The dataset is built around the policy's visual input: models predict from egocentric video, while accurate trajectory and state labels provide supervision. This yields a realistic egocentric prediction setting with reliable motion targets. Building on EgoSPT, we propose *SPOT* (*Spatially Prompted Object–Target Policy*), a policy built on a simple premise: first-frame spatial prompts specify the task, current observations provide the execution context, and future EE motion should be predicted as a coherent trajectory chunk. SPOT represents bounding-box prompts in two complementary ways: visually rendered on the first frame and encoded as coordinate prompt tokens. Object and target tokens attend to first-frame visual features to extract task-specific evidence, which is then fused with current-frame observations and trajectory history. A decoder-style flow-matching head generates future relative trajectory chunks conditioned on this sequence. Because egocentric backgrounds, camera motion, and object layouts are strongly scene-correlated, random episode splits can substantially overestimate performance. We therefore use scene-aware splits, keeping all episodes from the same scene unit within a single partition. We further evaluate predictions with four complementary metrics covering final position, trajectory-level position, 6D rotation, and gripper width. This protocol tests whether a model can use first-frame spatial prompts to predict trajectories in novel scene configurations, rather than memorizing familiar layouts. Our contributions are fourfold:

- We formulate *SP-VTP*, where first-frame spatial prompts specify egocentric manipulation goals and the model predicts future EE trajectories. To our knowledge, this is the *first* setting that frames egocentric manipulation as vision-centric, spatially prompted trajectory prediction.
- We introduce *EgoSPT*, a spatially prompted egocentric manipulation dataset with object–target grounding annotations, egocentric videos, recovered 3D EE trajectories, and scene-aware splits for evaluating generalization.
- We propose *SPOT*, a prompt-centric object–target policy that fuses rendered and coordinate spatial prompts with current observations and trajectory history to generate future visual trajectory chunks.
- We establish a scene-aware evaluation protocol with complementary trajectory metrics to measure cross-scene generalization beyond layout memorization.

2 Related Work

Goal-conditioned Robot Policies for Manipulation. Goal-conditioned manipulation policies aim to generate robot actions from sensor observation, under task specifications such as task identifiers, goal images, or language instructions. In multi-task manipulation, task identifiers provide a compact way to indicate which discrete task a policy should execute [53, 52], but they require task intents to be discretized into fixed labels and lack scene-specific object-target grounding. Goal images specify the desired final visual state [34, 41, 37, 29], but require access to an example of the completed scene and may entangle task intent with irrelevant visual details such as background, or object layout.

Language-conditioned policies provide more flexible interfaces for general robot manipulation [30, 39, 40, 17, 10]. More recently, generalist robot policies, particularly vision-language-action (VLA) models, have demonstrated the scalability of language-conditioned robot control across diverse tasks and embodiments [58, 19, 35, 3]. However, language descriptions can remain ambiguous in cluttered egocentric scenes with multiple visually similar objects and candidate targets. In contrast, our work studies spatial prompts as a lightweight task specification: first-frame points or boxes directly indicate the object to manipulate and the target placement region, and the policy predicts future EE trajectories from subsequent observations.

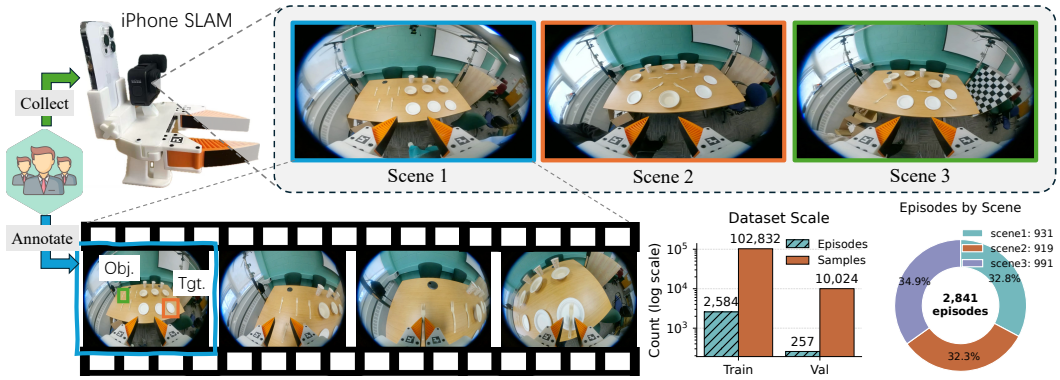

Figure 1: **Illustration of the EgoSPT dataset.** We use a modified UMI device equipped with an iPhone and a GoPro to collect EgoSPT, an egocentric visual trajectory dataset containing five forks and nine targets, including three plates, three bowls, and three cups. EgoSPT covers three scenes designed to evaluate different policy capabilities.

Egocentric Manipulation Datasets. Egocentric manipulation datasets provide visual observations from the viewpoint of the acting agent, that are closely aligned with manipulation actions. Egocentric visual data has been studied in two related but distinct settings. Human-centered egocentric video datasets capture first-person observations from wearable cameras and support the study of human activities [11, 7, 12], 3D hand-object interactions [21, 43, 1], and imitation learning from human videos [18]. Manipulator-centered egocentric demonstration datasets, in contrast, place the camera on or near the manipulation interface, such as a gripper or robotic hand, so the visual stream is more directly aligned with manipulation execution and the action trajectories used for policy learning. Universal Manipulation Interface (UMI) is a representative setup in this direction, introducing a portable hand-held interface for collecting in-the-wild manipulation demonstrations and learning deployable visuomotor policies [5]. Subsequent UMI-style efforts further demonstrate the effectiveness of this action-aligned data-collection paradigm for training manipulation policies in multiple practical settings [14, 24, 13, 57, 26, 38, 44, 28, 50, 33]. Building on a modified UMI setup, EgoSPT collects egocentric manipulation videos with recovered EE trajectories and pairs them with first-frame object and target grounding annotations, turning demonstrations into data for spatially prompted visual trajectory prediction.

Egocentric Visual Trajectory Prediction. Egocentric visual trajectory prediction aims to infer future interaction motion from egocentric visual observations. Prior human-centered egocentric forecasting work studies image-space hand-object interaction prediction, where models forecast future hand motion and contact regions on active objects [27, 15, 31]. Other work extends egocentric forecasting to 3D, predicting action targets in 3D workspaces or future 3D hand trajectories from RGB videos [23, 2, 8]. These methods show that egocentric visual streams contain useful cues for future interaction, but they mainly forecast human-centered motion from observed video context.

Manipulator-centered imitation learning introduces a related but different trajectory prediction setting: future EE trajectory chunks, including pose and gripper state, serve as the action representation for manipulation policies [56, 6]. UMI-style systems use such trajectory chunks to learn deployable visuomotor policies from egocentric demonstrations [5, 14, 33], but the task objective is typically implicit in the demonstration or specified through external instructions. SP-VTP formalizes a spatially prompted version of this problem: the model predicts future relative EE trajectory chunks conditioned on a static first-frame object-target prompt. This distinguishes SP-VTP from both standard egocentric hand forecasting and trajectory-based policy learning without explicit spatial task grounding.

3 EgoSPT

EgoSPT evaluates whether first-frame spatial prompts can be translated into executable egocentric trajectories. As shown in Fig. 1, it contains 2,841 pick-and-place videos with egocentric observations and recovered end-effector (EE) trajectories, collected using a modified UMI device [5, 13]. The

device integrates a GoPro with a 170° fisheye lens for egocentric video capture and an iPhone for 6-DoF EE trajectory tracking. Leveraging the iPhone's visual–inertial SLAM system, the modified UMI provides more accurate trajectory recovery than the original one. Nine trained experts collect and annotate EgoSPT using this device. Each episode requires picking one of five visually similar forks and placing it into one of nine targets, consisting of three cups, three bowls, and three plates.

EgoSPT is organized into three scenes with increasing difficulty. Scene 1 contains structured layouts with 45 object–target combinations and 20 episodes per combination, testing object–target association under clean conditions. Scene 2 uses a cluttered layout with the same 45 combinations and 20 episodes per combination, evaluating robustness to distractors and spatial ambiguity. Scene 3 contains 22 randomly cluttered subscenes, each covering the 45 combinations with one episode per combination, enabling evaluation under diverse and low-data conditions. Together, these scenes form a progressive protocol for measuring both in-distribution performance and cross-scene generalization.

Each video is approximately five seconds long at 30 fps and is downsampled to 10 fps, yielding about 50 trajectory steps per episode. Sliding-window training with horizon $H = 16$ produces roughly 110K trajectory prediction samples. Each episode provides a first frame, current frames, object/target spatial prompts, trajectory history, and future relative trajectory chunks. More statistic results can be found in the Appendix.

The dataset stresses several realistic factors: multiple same-category object instances, three target categories, fisheye egocentric observations, handheld camera motion, and clutter variation. These factors make it a testbed for spatial task conditioning, not merely trajectory regression.

4 SPOT

As shown in Fig. 2, SPOT is a spatial prompt conditioned trajectory policy composed of three modules: a task encoder, an observation encoder, and a trajectory generator. The task input combines visual prompts, obtained by rendering object and target boxes on the first frame, with coordinate prompts, which encode the same boxes as spatial tokens. The observation encoder represents the current egocentric frame and recent trajectory history. Conditioned on the resulting task and observation tokens, the trajectory generator predicts a future EE trajectory chunk.

Problem Formulation. SP-VTP takes a first-frame egocentric image I_0 with an object prompt p_{obj} and a target prompt p_{tgt}, where each prompt is either a point or a box in normalized image coordinates. At timestep t, the policy observes the current egocentric frame I_t and recent trajectory history h_t. The goal is to predict a future trajectory chunk $A_t \in \mathbb{R}^{H\times 10} = \{a_t, \ldots, a_{t+H-1}\}, H = 16$. Each waypoint $a_{t+h} \in \mathbb{R}^{10}$ is parameterized as a 10D vector containing relative translation, 6D rotation, and gripper width. Let $T_t \in SE(3)$ denote the EE pose at timestep t as a homogeneous transformation matrix in the world frame: $T_t \in \mathbb{R}^{4\times 4} = \begin{bmatrix} R_t & \mathbf{p}_t \\ \mathbf{0}^\top & 1 \end{bmatrix}$, where $R_t \in SO(3)$ is the EE orientation and $\mathbf{p}_t \in \mathbb{R}^3$ is its position. The relative pose target is computed in the current EE coordinate frame:

$$\Delta T_{t+h} = T_t^{-1} T_{t+h} = \begin{bmatrix} \Delta R_{t+h} & \Delta \mathbf{p}_{t+h} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad h \in [1, H]. \tag{1}$$

Here, T_t^{-1} transforms future poses from the world frame into the coordinate frame attached to the current EE, so ΔT_{t+h} represents the rigid motion needed to move from the current pose to the future pose. We then convert this relative transform into the 10D waypoint target:

$$a_{t+h} = [\Delta \mathbf{p}_{t+h}, \mathrm{rot6d}(\Delta R_{t+h}), g_{t+h}] \in \mathbb{R}^{10},$$

where $\Delta \mathbf{p}_{t+h} \in \mathbb{R}^3$ is the relative translation, $\mathrm{rot6d}(\Delta R_{t+h}) \in \mathbb{R}^6$ is the 6D representation of the relative rotation, and $g_{t+h} \in \mathbb{R}$ is the gripper width. This representation asks the model to predict where the EE should move next relative to its current state, rather than where it is in a global frame.

Task Encoder. The task encoder processes the first frame I_0 together with the object and target prompts $(p_{\mathrm{obj}}, p_{\mathrm{tgt}})$. In the default setting, p_{obj} and p_{tgt} are bounding boxes. We render these boxes on the first frame to obtain a visually prompted image $\tilde{I}_0$, and also keep their normalized box coordinates as coordinate prompts. A frozen DINOv2 ViT-B/14 [36] backbone first extracts image tokens from $\tilde{I}_0$, which are projected to the policy dimension $D = 768$. We use the patch tokens as

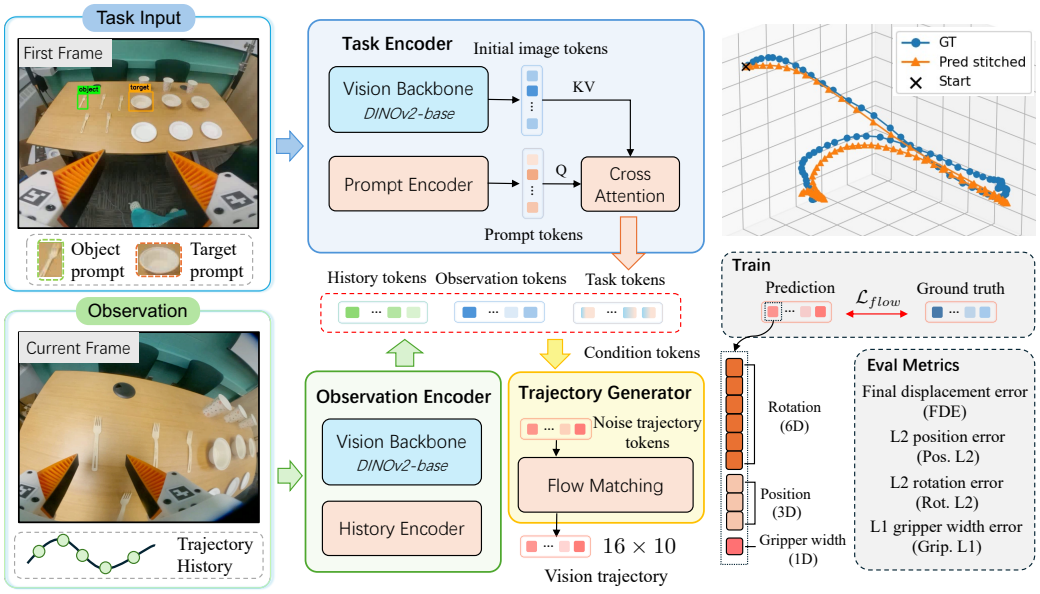

Figure 2: Overview of the proposed SPOT framework. Given a first-frame task input with object and target boxes shown visually on the image and encoded as coordinate prompt tokens, the task encoder extracts visual task tokens and prompt tokens, which are fused through cross-attention to form a spatially prompted task representation. At each timestep, the observation encoder processes the current egocentric frame and motion history, and the trajectory generator predicts future EE pose chunks using a flow-matching objective conditioned on the task and observation tokens. During training, predicted trajectory tokens are supervised by ground-truth trajectories with $\mathcal{L}_{flow}$; during evaluation, performance is measured using Pos. L2, Rot. L2, Grip. L1, and FDE.

visual memory and retain an image summary token, typically the DINOv2 CLS token, to preserve global scene context feature $F_0 \in \mathbb{R}^{(N+1)\times D}$, where N is the number of image tokens.

Coordinate prompts are encoded geometrically. Each 2D prompt coordinate is mapped through Fourier positional features [32, 42] and an MLP (prompt encoder), and a learnable role embedding identifies whether the token belongs to the object or the target. A box prompt contributes two corner tokens for the object and two corner tokens for the target, while a point prompt contributes one object token and one target token. These prompt tokens $Z_{\text{prompt}} \in \mathbb{R}^{2\times D}$ then query the first-frame visual tokens through a Transformer decoder [46]:

$$Z_{\text{task}} = \text{CrossAtt}(Q = Z_{\text{prompt}}, KV = F_0). \tag{2}$$

This prompt-to-image cross-attention lets the object and target prompts actively read task-relevant visual information from the first frame. The resulting task representation $Z_{\text{task}} \in \mathbb{R}^{2\times D}$ contains both the global image summary and the fused object/target prompt tokens, giving the policy a compact representation of what should be manipulated and where it should be placed.

Observation Encoder. The observation encoder represents the execution state at timestep t. The current frame I_t is encoded by the same frozen visual backbone and projection layer used by the task encoder, ensuring that first-frame task tokens and current-frame observation tokens live in a shared visual space. This shared visual encoding makes it easier for the trajectory generator to relate the original task specification to the current egocentric view.

To provide short-term motion context, recent trajectory history tokens $h_t \in \mathbb{R}^{4\times D}$ are encoded by a lightweight MLP (history encoder $E_{\text{hist}}(\cdot)$) and augmented with learnable temporal position embeddings. We set the history length as 4 by default. The history tokens are concatenated with current-frame image tokens $F_t \in \mathbb{R}^{(N+1)\times D}$: $Z_{\text{obs}} \in \mathbb{R}^{(5+N)\times D} = [F_t; E_{\text{hist}}(h_t)]$. If no trajectory history is used, the observation encoder returns only current-frame visual tokens. Otherwise, Z_{obs} jointly describes what the robot currently sees and how the EE has recently moved.

Trajectory Generator. The trajectory generator predicts future trajectory chunks from the encoded task and execution context. It conditions on the concatenated task and observation tokens, $Z_{\text{cond}} = [Z_{\text{task}}; Z_{\text{obs}}]$. In the default visual-prompt plus box-coordinate setting, Z_{task} consists of one image summary token from the prompted first frame and two fused coordinate prompt tokens. For a 224×224 ViT-B/14 input, the current observation provides about 257 image tokens, while the history horizon contributes four trajectory-history tokens. This yields a compact condition sequence capturing task specification, current egocentric context, and recent motion history.

SPOT supports both diffusion [16, 6] and flow-matching [25]trajectory heads, using the same transformer decoder-style architecture in both cases. Noisy or interpolated trajectory tokens serve as the decoder targets, and Z_{cond} serves as the cross-attention memory. Each future waypoint can self-attend to other predicted waypoints and cross-attend to the task, observation, and history tokens.

The default head uses flow matching. Let x_0 denote the normalized ground-truth trajectory chunk and $\epsilon \sim \mathcal{N}(0, I)$ denote Gaussian noise. We sample $t \in [0, 1]$ and construct a straight interpolation path

$$x_t = (1 - t)x_0 + t\epsilon. \tag{3}$$

The target velocity is

$$v^\star(x_t, t) = \epsilon - x_0. \tag{4}$$

The flow head predicts this velocity from interpolated trajectory tokens and the condition memory:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\left[\|v_\theta(x_t, t, Z_{\text{cond}}) - (\epsilon - x_0)\|_2^2\right]. \tag{5}$$

At inference time, SPOT starts from Gaussian noise and integrates the learned velocity field from noise to a trajectory chunk using a small number of Euler steps. Future trajectory chunks are normalized by dataset-level waypoint mean and standard deviation before training. The trajectory head predicts in this normalized space, and predictions are denormalized after sampling. This balances translation, rotation, and gripper dimensions during optimization. During inference, the generated $H \times 10$ trajectory chunk is denormalized and interpreted as relative EE motion.

5 Experiments

5.1 Experiment Configuration

Training and evaluation protocols. All baseline and ablation experiments train on the union of Scene 1, Scene 2, and Scene 3. This setting evaluates whether a single policy can use spatial prompts across structured layouts, cluttered layouts, and diverse subscenes. We report both overall validation metrics and per-scene metrics for Scene 1, Scene 2, and Scene 3.

The default reference configuration uses both visual bounding-box prompts rendered on the first frame and bounding-box coordinate prompt tokens, together with a frozen DINOv2 ViT-B/14 encoder, embedding dimension $D = 768$, cross-attention task fusion, history horizon $K = 4$, and a flow-matching trajectory head. Unless otherwise specified, each experiment changes only one factor from this reference setting. We provide more training and evaluation details in the Appendix.

Evaluation Metrics. We report four validation metrics from the model evaluation pipeline: ① **Final Displacement Error (FDE)** measures the endpoint translation error of the last predictable chunk; ② **Pos. L2** measures the mean L2 error of relative translation over the predicted chunk; ③ **Rot. L2** measures the mean L2 error in the 6D rotation representation; and ④ **Grip. L1** measures the mean absolute error of the gripper width.

5.2 Prompting Baselines

This experiment studies how task specification affects performance. All variants use the same training schedule: NO PROMPT: removes object/target information and replaces prompts with learned null tokens; POINT PROMPT: represents the object and target by their box centers; BBOX PROMPT: uses object and target bounding boxes as coordinate prompt tokens; VISUAL PROMPT: renders object and target boxes on the initial frame without coordinate prompt tokens; and BBOX+VISUAL: is the default SPOT setting, combining rendered bounding boxes with bounding-box coordinate tokens.

Table 1 shows that spatial prompts substantially improve trajectory prediction over the no-prompt baseline. On the overall split, the default BBox + visual input achieves the best endpoint and

Table 1: Prompting baseline results on the overall validation split and each individual scene. Lower is better for all metrics.

Split	Method	FDE	Pos. L2	Rot. L2	Grip. L1
All	No prompt	0.1739	0.0912	0.2021	0.01098
	Point prompt	0.1186	0.0736	0.1948	**0.01067**
	BBox prompt	0.1176	0.0706	0.2031	0.01173
	Visual prompt	0.1175	0.0701	**0.1865**	0.01095
	BBox + visual	**0.1147**	**0.0699**	0.1965	0.01133
Scene 1	No prompt	0.1433	0.0762	0.1623	**0.00630**
	Point prompt	0.1082	0.0674	0.1609	0.00676
	BBox prompt	0.1044	0.0644	0.1635	0.00674
	Visual prompt	**0.1012**	**0.0615**	**0.1504**	0.00665
	BBox + visual	0.1123	0.0700	0.1588	0.00693
Scene 2	No prompt	0.2057	0.1014	0.2176	0.01024
	Point prompt	0.1183	0.0674	0.2045	**0.00844**
	BBox prompt	0.1220	0.0689	0.2213	0.01022
	Visual prompt	0.1301	0.0713	**0.2024**	0.00964
	BBox + visual	**0.1116**	**0.0632**	0.2093	0.00975
Scene 3	No prompt	0.1875	0.1037	0.2512	0.01970
	Point prompt	0.1292	0.0812	0.2402	0.01981
	BBox prompt	0.1345	0.0831	0.2478	0.02184
	Visual prompt	0.1307	0.0832	**0.2290**	**0.01966**
	BBox + visual	**0.1224**	**0.0779**	0.2446	0.02048

Table 2: Ablation on visual encoder type. We compare frozen base-size visual encoders under the same SPOT configuration.

Split	Encoder	FDE	Pos. L2	Rot. L2	Grip. L1
All	PE-Base	0.1544	0.0891	0.2136	0.01085
	SigLIP-Base	0.1204	0.0724	**0.1890**	**0.00921**
	SAM-Base	0.1759	0.0929	0.2173	0.00928
	EVA2-Base	0.1422	0.0826	0.2077	0.01050
	DINOv2-Base	**0.1147**	**0.0699**	0.1965	0.01133
Scene 1	PE-Base	0.1137	0.0687	0.1574	0.00749
	SigLIP-Base	**0.1083**	**0.0664**	**0.1517**	**0.00674**
	SAM-Base	0.1418	0.0752	0.1601	0.00676
	EVA2-Base	0.1096	0.0665	0.1582	0.00728
	DINOv2-Base	0.1123	0.0700	0.1588	0.00693
Scene 2	PE-Base	0.1888	0.1019	0.2358	0.00967
	SigLIP-Base	0.1248	0.0702	**0.2009**	**0.00858**
	SAM-Base	0.1998	0.1001	0.2386	0.01017
	EVA2-Base	0.1564	0.0861	0.2206	0.00952
	DINOv2-Base	**0.1116**	**0.0632**	0.2093	0.00975
Scene 3	PE-Base	0.1822	0.1083	0.2813	0.01788
	SigLIP-Base	0.1361	0.0853	**0.2372**	0.01399
	SAM-Base	0.2090	0.1154	0.2898	**0.01254**
	EVA2-Base	0.1796	0.1055	0.2753	0.01700
	DINOv2-Base	**0.1224**	**0.0779**	0.2446	0.02048

translation accuracy, reducing FDE from 0.1739 to 0.1147 and Pos. L2 from 0.0912 to 0.0699. Visual prompt alone gives the lowest overall Rot. L2, while point prompts give the lowest overall Grip. L1. Per-scene results show that visual bounding-box prompts are especially useful for Scene 1, and combining them with coordinate box prompts is strongest for endpoint and Pos. L2 on Scenes 2 and 3. This suggests that coordinate prompts provide explicit object-target geometry, while visual prompts make the same spatial intent directly visible to the first-frame visual encoder.

5.3 Ablation Studies

We organize ablations around the major design decisions in SPOT.

Vision encoder type. We compare DINOv2-Base [36] with other base-size vision foundation model encoders when available, including SAM [20], Perception Encoder (PE) [4], EVA2 [9], and SigLIP [55]. This ablation tests whether SPOT depends on a specific visual foundation model or benefits generally from strong image tokens.

Table 2 shows that the choice of visual encoder affects different trajectory factors differently. On the overall split, DINOv2-Base achieves the best endpoint and translation accuracy, with FDE 0.1147 and Pos. L2 0.0699, outperforming SigLIP-Base (0.1204 / 0.0724), EVA2-Base (0.1422 / 0.0826), PE-Base (0.1544 / 0.0891), and SAM-Base (0.1759 / 0.0929). This advantage is most pronounced on Scenes 2 and 3, where DINOv2-Base gives the lowest FDE and Pos. L2, indicating stronger spatial localization for prompt-conditioned trajectory prediction. EVA2-Base is competitive on Scene 1, with FDE 0.1096 and Pos. L2 0.0665, but its spatial errors increase on Scenes 2 and 3. In contrast, SigLIP-Base achieves the best overall Rot. L2 and Grip. L1 (0.1890 and 0.00921), and performs best across all metrics on Scene 1, suggesting that language-aligned visual features can help orientation and gripper prediction in some layouts. SAM-Base and PE-Base are less competitive on endpoint and translation errors. Since spatial accuracy is the primary objective for SP-VTP, we keep frozen DINOv2-Base as the default encoder.

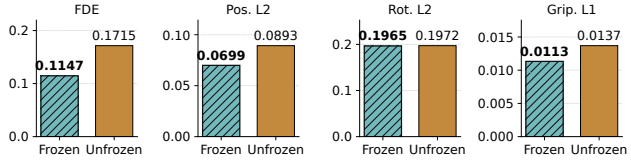

Figure 3: **Ablation on tuning DINOv2-Base.** We compare the default frozen DINOv2-Base encoder with an unfrozen variant on the full validation split.

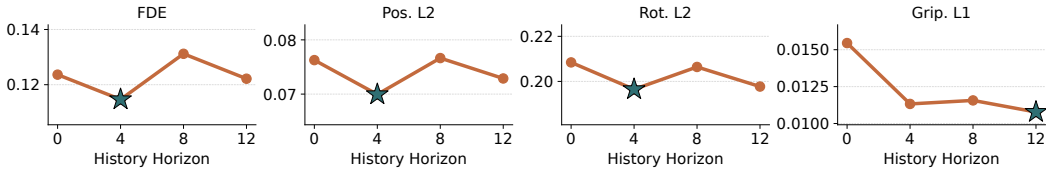

Figure 5: **Ablation on history horizon.** We compare history horizons of 0, 4, 8, and 12 on the full validation split. Stars mark the best value for each metric. Lower is better for all metrics.

Trajectory head. We compare the default flow-matching head with a diffusion head under the same task, observation tokens, and the architecture. Figure 4 shows that the flow-matching head consistently outperforms the diffusion head on the full validation split, reducing FDE from 0.2279 to 0.1147 and Pos. L2 from 0.1222 to 0.0699. It also improves Rot. L2 and Grip. L1, indicating that the flow-matching objective is better aligned with prompt-conditioned trajectory generation across all output factors. We therefore keep flow matching as the default trajectory head.

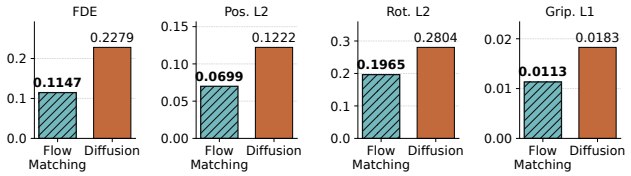

Figure 4: **Ablation on trajectory head.** We compare the default flow-matching head with a diffusion head on the full validation split. Lower is better for all metrics.

Tuning vision backbone. We compare the default frozen DINOv2-Base encoder with an unfrozen variant that updates the visual backbone during policy training. This ablation tests whether adapting the visual features to EgoSPT improves trajectory prediction or instead overfits the limited task distribution. Figure 3 shows that keeping DINOv2-Base frozen outperforms tuning the backbone on all metrics, reducing FDE from 0.1715 to 0.1147 and Pos. L2 from 0.0893 to 0.0699. The degradation from end-to-end tuning suggests that updating the visual backbone weakens the general visual representation needed for prompt-conditioned spatial prediction. We therefore keep the DINOv2-Base encoder frozen in the default SPOT configuration.

History horizon. We compare history horizons of 0, 4, 8, and 12. This measures how much recent motion context helps trajectory generation beyond the current egocentric frame and first-frame task prompt. Figure 5 shows that a moderate history horizon is most effective for trajectory prediction. Using $K = 4$ gives the best FDE, Pos. L2, and Rot. L2, indicating that recent motion context helps localize the execution phase without overloading the policy with stale observations. Increasing the horizon to $K = 12$ slightly improves Grip. L1, but it does not improve endpoint, translation, or rotation accuracy. We therefore use $K = 4$ as the default history horizon.

5.4 Qualitative Evaluation

To analyze the performance of SPOT on SP-VTP beyond scalar metrics, we visualize both full stitched trajectories and first-chunk predictions. More results are provided in the appendix.

First-chunk visualization. Fig. 7 shows the first-frame prompts, current frame, predicted and ground-truth 3D trajectory chunks, and Pos. L2/Rot. L2/Grip. L1 curves. The prompt specifies the object and target regions, while the current frame captures the end effector inside the manipulation workspace. The predicted trajectory matches the direction and curvature of the ground truth, indicating that SPOT infers the correct local motion phase from a static prompt and current observation. Pos. L2 grows over the horizon, suggesting compounding translation errors, whereas Rot. L2 and Grip. L1 remain relatively stable, indicating more consistent orientation and gripper prediction over short horizons.

Full-trajectory visualization. Fig. 6 shows stitched chunk predictions over a full episode, together with the ground-truth 3D trajectory, first-frame prompt, and per-frame Pos. L2. SPOT preserves the coarse structure of the manipulation trajectory: the stitched prediction follows the main geometric pattern of the ground truth, including the approach and return motions. The remaining error mainly appears as local drift along the path rather than task-level failure. Pos. L2 is low at the beginning

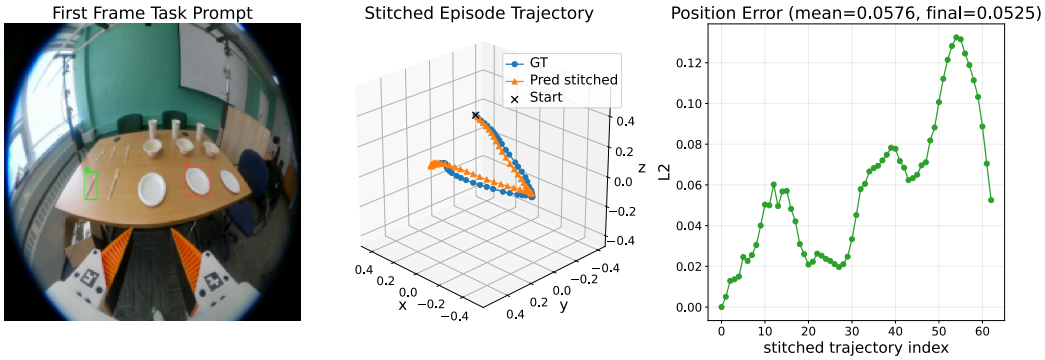

Figure 6: **Full-trajectory visualization.** We show stitched predictions over full episodes together with ground-truth 3D trajectories, first-frame spatial prompts, and per-frame position errors.

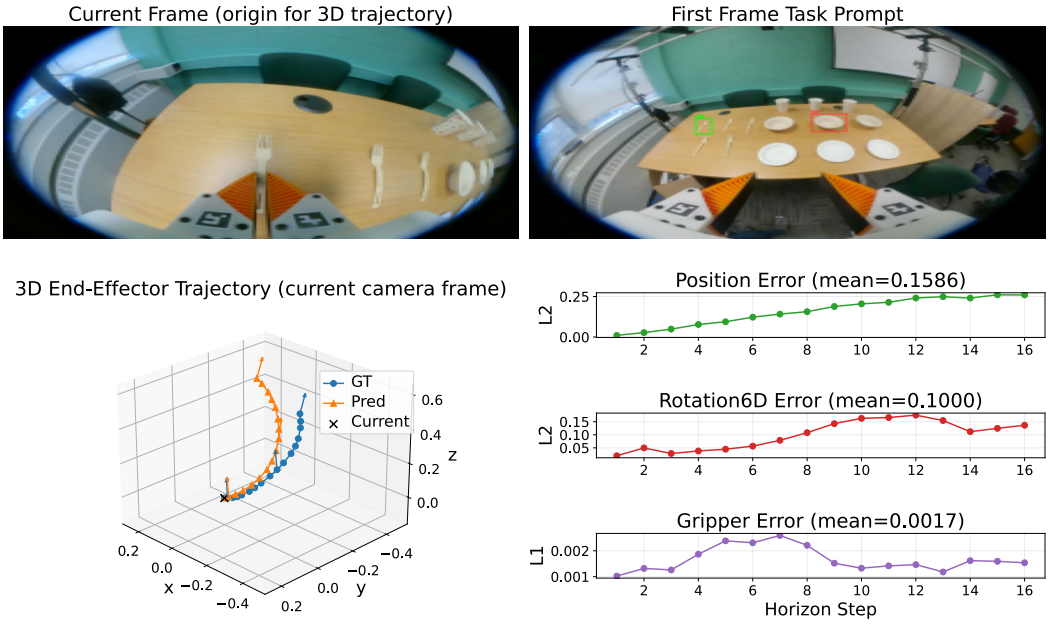

Figure 7: **First-chunk visualization.** We show the first-frame prompt, current observation, predicted and ground-truth trajectory chunks, and chunk-level Pos. L2, Rot. L2, and Grip. L1 curves.

and end of the episode, with a final error of 0.0525, but increases in the middle and late stages as chunk-level errors accumulate under larger camera and EE motion. These results suggest that SPOT captures the intended spatial task and overall trajectory geometry, while long-horizon stitching remains sensitive to compounding local prediction errors.

6 Conclusion

We introduce *Spatially Prompted Visual Trajectory Prediction* (SP-VTP), where first-frame object and target prompts condition future egocentric end-effector trajectory prediction. We instantiate this setting with EgoSPT, a dataset with spatial grounding and recovered 3D motion, and SPOT, a policy that fuses visual and coordinate prompts with current observation and motion history for flow-based trajectory generation. Scene-aware experiments show that spatial prompts substantially improve trajectory accuracy, with the combined visual and coordinate prompt giving the strongest endpoint and translation performance. Ablations further identify frozen DINOv2 features, flow matching, and short motion history as key factors for robust prediction. These results establish spatial prompting as a compact and effective interface for visually grounded manipulation.

References

[1] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, et al. Hot3d: Hand and object tracking in 3d from egocentric multi-view videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7061–7071. IEEE, 2025.

[2] Wentao Bao, Lele Chen, Libing Zeng, Zhong Li, Yi Xu, Junsong Yuan, and Yu Kong. Uncertainty-aware state space transformer for egocentric 3d hand trajectory forecasting. In *Int. Conf. Comput. Vis.*, pages 13702–13711, 2023.

[3] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conf. Robot Learn.*, 2025.

[4] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv:2504.13181*, 2025.

[5] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Robotics: Science and Systems*, 2024.

[6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *Int. J. Robot. Res.*, 44(10-11):1684–1704, 2025.

[7] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *Int. J. Comput. Vis.*, 130(1): 33–55, 2022.

[8] Irving Fang, Yuzhong Chen, Yifan Wang, Jianghan Zhang, Qiushi Zhang, Jiali Xu, Xibo He, Weibo Gao, Hao Su, Yiming Li, et al. Egopat3dv2: Predicting 3d action target from 2d egocentric vision for human-robot interaction. In *IEEE Int. Conf. Robot. Autom.*, pages 3036–3043. IEEE, 2024.

[9] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. Eva-02: A visual representation for neon genesis. *arXiv:2303.11331*, 2023.

[10] Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt2: Learning precise manipulation from few demonstrations. *Robotics: Science and Systems*, 2024.

[11] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 18995–19012, 2022.

[12] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, et al. Ego-exo4d: Understanding skilled human activity from first-and third-person perspectives. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 19383–19400, 2024.

[13] Harsh Gupta, Xiaofeng Guo, Huy Ha, Chuer Pan, Muqing Cao, Dongjae Lee, Sebastian Scherer, Shuran Song, and Guanya Shi. Umi-on-air: Embodiment-aware guidance for embodiment-agnostic visuomotor policies, 2025. URL `https://arxiv.org/abs/2510.02614`.

[14] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. UMI-on-legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. In *Conf. Robot Learn.*, 2024. URL https://openreview.net/forum?id=3i7j8ZPnbm.

[15] Masashi Hatano, Ryo Hachiuma, and Hideo Saito. Emag: Ego-motion aware and generalizable 2d hand forecasting from egocentric videos. In *Eur. Conf. Comput. Vis.*, pages 119–136, 2024.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Adv. Neural Inform. Process. Syst.*, volume 33, pages 6840–6851, 2020.

[17] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. In *Conf. Robot Learn.*, 2023.

[18] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video. In *IEEE Int. Conf. Robot. Autom.*, pages 13226–13233. IEEE, 2025.

[19] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conf. Robot Learn.*, 2024.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Int. Conf. Comput. Vis.*, pages 4015–4026, 2023.

[21] Taein Kwon, Bugra Tekin, Jan Stühmer, Federica Bogo, and Marc Pollefeys. H2o: Two hands manipulating objects for first person interaction recognition. In *Int. Conf. Comput. Vis.*, pages 10138–10148, 2021.

[22] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. *arXiv preprint arXiv:2502.05485*, 2025.

[23] Yiming Li, Ziang Cao, Andrew Liang, Benjamin Liang, Luoyao Chen, Hang Zhao, and Chen Feng. Egocentric prediction of action target in 3d. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 20971–20980. IEEE, 2022.

[24] Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. In *Int. Conf. Learn. Represent.*, 2025. URL https://openreview.net/forum?id=pISLZG7ktL.

[25] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Int. Conf. Learn. Represent.*, 2020.

[26] Kehui Liu, Zhongjie Jia, Yang Li, Pengan Chen, Song Liu, Xin Liu, Pingrui Zhang, Haoming Song, Xinyi Ye, Nieqing Cao, et al. Fastumi-100k: Advancing data-driven robotic manipulation with a large-scale umi-style dataset. *arXiv preprint arXiv:2510.08022*, 2025.

[27] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3282–3292, 2022.

[28] Zeyi Liu, Cheng Chi, Eric Cousineau, Naveen Kuppuswamy, Benjamin Burchfiel, and Shuran Song. Maniwav: Learning robot manipulation from in-the-wild audio-visual data. In *Conf. Robot Learn.*, pages 947–962. PMLR, 2025.

[29] Yunhao Luo and Yilun Du. Grounding video models to actions through goal conditioned exploration. In *Int. Conf. Learn. Represent.*, 2025. URL https://openreview.net/forum?id=G6dMvRuhFr.

[30] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

[31] Junyi Ma, Xieyuanli Chen, Jingyi Xu, and Hesheng Wang. Diff-ip2d: Diffusion-based hand-object interaction prediction on egocentric videos. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 4291–4298. IEEE, 2025.

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[33] Ruiqian Nai, Boyuan Zheng, Junming Zhao, Haodong Zhu, Sicong Dai, Zunhao Chen, Yihang Hu, Yingdong Hu, Tong Zhang, Chuan Wen, et al. Humanoid manipulation interface: Humanoid whole-body manipulation from robot-free demonstrations. *arXiv preprint arXiv:2602.06643*, 2026.

[34] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Adv. Neural Inform. Process. Syst.*, 31, 2018.

[35] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems*, Delft, Netherlands, 2024.

[36] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024.

[37] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems*, 2023. URL https://www.roboticsproceedings.org/rss19/p028.pdf.

[38] Mingyo Seo, H Andy Park, Shenli Yuan, Yuke Zhu, and Luis Sentis. Legato: Cross-embodiment imitation using a grasping tool. *IEEE Robot. Autom. Lett.*, 10(3):2854–2861, 2025.

[39] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conf. Robot Learn.*, pages 894–906. PMLR, 2022.

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

[41] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *Int. Conf. Machi. Learn.*, pages 4732–4741. PMLR, 2018.

[42] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Adv. Neural Inform. Process. Syst.*, volume 33, pages 7537–7547, 2020.

[43] Hao Tang, Kevin J Liang, Kristen Grauman, Matt Feiszli, and Weiyao Wang. Egotracks: A long-term egocentric visual object tracking dataset. *Adv. Neural Inform. Process. Syst.*, 36: 75716–75739, 2023.

[44] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies. *Robotics: Science and Systems*, 2025.

[45] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, volume 30, 2017.

[47] Zixuan Wang, Yuxin Chen, Yuqi Liu, Jinhui Ye, Pengguang Chen, Changsheng Lu, Shu Liu, and Jiaya Jia. Vp-vla: Visual prompting as an interface for vision-language-action models. *arXiv preprint arXiv:2603.22003*, 2026.

[48] Yifei Wei, Linqing Zhong, Yi Liu, Yuxiang Lu, Xindong He, Maoqing Yao, and Guanghui Ren. Libra-vla: Achieving learning equilibrium via asynchronous coarse-to-fine dual-system. *arXiv preprint arXiv:2604.24921*, 2026.

[49] Zhenyu Wu, Yuheng Zhou, Xiuwei Xu, Ziwei Wang, and Haibin Yan. Momanipvla: Transferring vision-language-action models for general mobile manipulation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1714–1723, 2025.

[50] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. In *Conf. Robot Learn.*, pages 437–459. PMLR, 2025.

[51] Jianwei Yang, Reuben Tan, Qianhui Wu, Ruijie Zheng, Baolin Peng, Yongyuan Liang, Yu Gu, Mu Cai, Seonghyeon Ye, Joel Jang, Yuquan Deng, and Jianfeng Gao. Magma: A foundation model for multimodal ai agents. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14203–14214, June 2025.

[52] Ruihan Yang, Huazhe Xu, YI WU, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Adv. Neural Inform. Process. Syst.*, volume 33, pages 4767–4777. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/32cfdce9631d8c7906e8e9d6e68b514b-Paper.pdf`.

[53] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conf. Robot Learn.*, pages 1094–1100. PMLR, 2020.

[54] Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conf. Robot Learn.*, pages 3157–3181. PMLR, 2025.

[55] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Int. Conf. Comput. Vis.*, pages 11975–11986, 2023.

[56] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.

[57] Zhaxizhuoma, Kehui Liu, Chuyue Guan, Zhongjie Jia, Ziniu Wu, Xin Liu, Tianyu Wang, Shuai Liang, Pengan Chen, Pingrui Zhang, Haoming Song, Delin Qu, Dong Wang, Zhigang Wang, Nieqing Cao, Yan Ding, Bin Zhao, and Xuelong Li. Fastumi: A scalable and hardware-independent universal manipulation interface with dataset. *arXiv*, 2025. URL `https://arxiv.org/abs/2409.19499`.

[58] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R Sanketi, Grecia Salazar, Michael S Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J Joshi, Alex Irpan, brian ichter, Jasmine Hsu, Alexander

Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conf. Robot Learn.*, 2023.

Figure 8: More visualization demonstrations of modified UMI device.

A Dataset Documentation

Collection setup. EgoSPT contains 2,841 egocentric pick-and-place episodes collected with a modified UMI (see Fig. 8) by nine trained experts. Each episode records an RGB video and tracking streams. The recording interface stores raw episode videos as `camera_<i>.mp4` and per-camera arrays such as pose, receive time, capture time, and intrinsics. In the processed dataset used by SPOT, `camera_1.mp4` is the RGB video consumed by the vision model, while the tracking stream is used to recover time-aligned 6-DoF EE motion. Videos are recorded at 1920×1080 and 30 fps, and are temporally subsampled with stride 3 for trajectory prediction.

Trajectory processing pipeline. The raw demonstrations are converted into the format consumed by SPOT data loader. The practical pipeline is: record raw videos and tracking arrays, synchronize RGB and tracking timestamps, interpolate EE pose onto valid RGB frame times, detect ArUco tags on the gripper, calibrate and interpolate gripper width, copy the RGB video into a processed episode directory, and attach first-frame object/target bounding-box annotations. The processed layout expected by the dataset loader is

`<data.root>/<scene>/<task>/recording_output_processed/<episode>/`

containing `camera_1.mp4`, `valid_indices`, `pose_interp`, and `gripper_widths`. The raw videos and processed trajectories are organized under `umi_day/output_data`, and first-frame prompt annotations are stored separately in `annotations_merged.json`.

Time alignment and trajectory recovery. During synchronization, camera and tracking receive times are corrected using configured camera and tracking latencies. The overlapping time range is used to define `valid_indices` over RGB frames. EE translations are linearly interpolated, rotations are interpolated with spherical linear interpolation, and the camera pose is converted to an EE pose using the calibrated camera-to-EE transform. Gripper width is obtained from ArUco detections of finger tags, calibrated with a gripper-range episode, and interpolated to the valid frame times. The resulting `pose_interp` and `gripper_widths` are aligned with the valid RGB frames.

Annotations and sample construction. Nine trained experts are assigned the collected videos and annotate the object and target bounding boxes using our annotation tool, as shown in Fig. 9. The tool is a lightweight video bounding-box annotator designed for frame-0 task specification: annotators open an episode video, jump to the first frame, and draw exactly two boxes, where the first box denotes the manipulated object and the second denotes the target region. It also supports video browsing, playback, zooming, box editing, and in-place correction of merged annotations. Annotations are saved in JSON format with both the ordered box list and explicit `object/target` fields in original video pixel coordinates. The annotation results are merged and stored in `annotations_merged.json`.

After annotation, we manually check each bounding box with a separate annotation modification tool shown in Fig. 10. This tool loads videos from `annotations_merged.json`, supports direct navigation through annotated videos, allows frame browsing and playback, and restricts box editing to frame 0. Annotators can move, resize, create, or delete boxes, and the corrected object/target boxes are written back to the original merged annotation file in place.

The annotation JSON maps each episode video key to the object and target boxes in the original first-frame pixel coordinates. The dataset parser converts keys into `scene`, `task`, and `episode`, locates the corresponding processed episode directory, and skips episodes with missing videos,

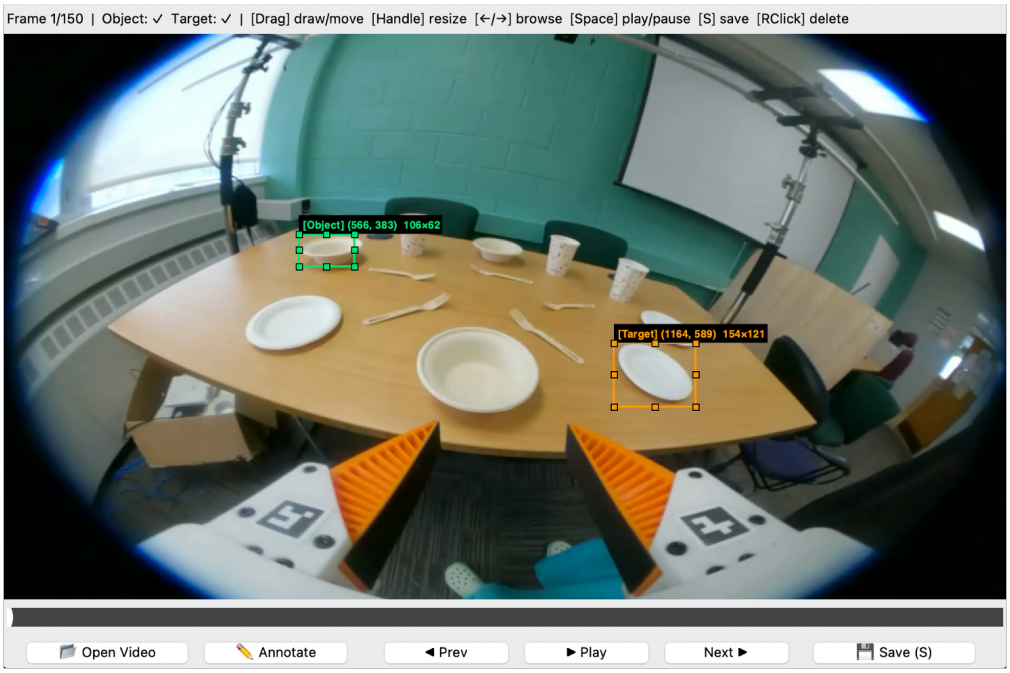

Figure 9: **Annotation interface.** Annotators label the manipulated object and target region on the first egocentric frame. The resulting bounding boxes are stored in the annotation JSON and used as spatial prompts for SP-VTP.

missing prompts, missing zarr arrays, or too few valid frames. For a sample at timestep t, the loader reads the first frame, the current RGB frame, normalized object/target prompts, a history window, and a future action window. Future actions are computed as relative EE motions in the current frame:

$$T_{\text{rel}}(t, h) = T_t^{-1} T_{t+h},$$

then represented as relative translation, 6D rotation, and gripper width. Action history is built analogously from previous poses relative to the current pose. Each returned sample contains `first_frame`, `current_frame`, `prompt_object`, `prompt_target`, `action_history`, `future_actions`, and `is_final_chunk`.

Preprocessing and scale. The default preprocessing uses image size 224×224, bounding-box prompts, action horizon $H = 16$, history horizon $K = 4$, and stride 3. This produces 112,856 sliding-window trajectory samples in total, including 102,832 training samples from 2,584 episodes and 10,024 validation samples from 257 episodes. The mean episode length after preprocessing is 59.7 trajectory steps, and each episode yields 39.7 trajectory samples on average. Across all samples, the mean relative-translation magnitude is 0.2159, the mean final displacement is 0.3529, and the mean gripper width is 0.0303.

Scene and task coverage. The data cover five visually similar forks and nine target receptacles across three scenes. Scene 1 uses structured layouts, Scene 2 uses a cluttered layout, and Scene 3 uses diverse cluttered subscenes with fewer demonstrations per configuration. The full dataset contains 931 episodes / 43,972 samples from Scene 1, 919 episodes / 35,902 samples from Scene 2, and 991 episodes / 32,982 samples from Scene 3. Scene 1 and Scene 2 are organized around object–target tasks such as `put_fork1_to_plate1`; Scene 3 is organized into 22 cluttered subscenes, each covering the object–target combinations with sparse demonstrations. The train/validation split is performed at the episode/task level while preserving scene coverage; the scene-distribution total variation between splits is 0.0015. All reported experiments therefore evaluate both overall validation performance and per-scene behavior.

Quality checks and statistics files. Before training, we verify that every annotation key maps to an existing processed episode, each processed episode has `camera_1.mp4`, `valid_indices`, `pose_interp`, and `gripper_widths`, each episode has enough valid frames for $K + H + 1$, the

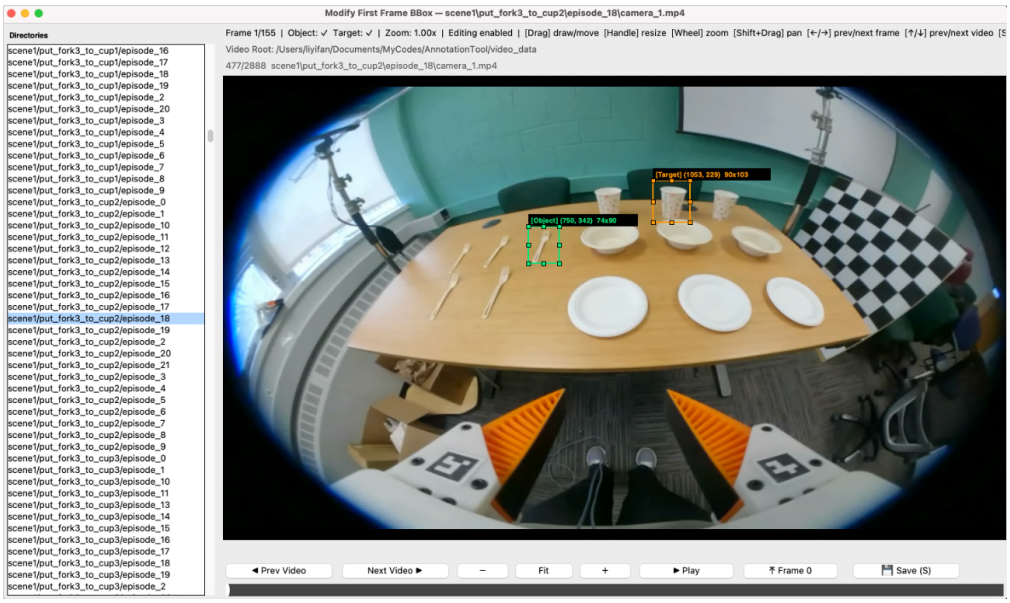

Figure 10: **Annotation modification interface.** After initial annotation, annotators inspect each video and correct frame-0 object/target bounding boxes directly in the merged annotation file.

pose and gripper arrays share compatible temporal indexing, and first-frame boxes are valid in the original image coordinate system. We also check that scene/task/episode names in the annotations match directory names after path normalization and that the gripper calibration and tag detections are available for width interpolation.

We generate auxiliary statistics files to support reproducibility and data inspection. `summary.json` stores the machine-readable dataset summary, `episode_stats.csv` stores per-episode lengths, sample counts, and video metadata, `count_stats.csv` stores scene/task/scene-task counts, `prompt_stats.csv` stores bounding-box size and location statistics, `image_stats.csv` stores sampled RGB statistics, and `outliers_*.csv` stores samples with the largest final-displacement values for train and validation splits. These files are used to verify dataset scale, split quality, prompt validity, and trajectory target ranges. These files are provided in the supplementary material.

B Model Architecture Details

Policy interface. The implementation is centered on `VisionTrajPolicy`, which maps a first-frame task specification and a current execution state to a future action chunk:

$$(\text{first frame}, \text{prompt}, \text{current frame}, \text{history}) \rightarrow A_t \in \mathbb{R}^{H\times 10}.$$

We use $H = 16$ by default. Each action vector is $[\Delta x, \Delta y, \Delta z, \text{rot6d}, g]$, where the pose component is expressed relative to the current camera/EE frame and g is the gripper width. Architecturally, the policy has three parts: the task encoder builds prompt-conditioned first-frame tokens, the observation encoder builds current-state tokens, and the action head decodes future trajectory tokens from their concatenation.

Prompt forms. The dataset loader exposes five task-input variants. `bbox` uses raw first-frame pixels with object/target bounding-box coordinates; `point` uses the centers of the boxes; `none` removes coordinate prompts and falls back to learned null prompt tokens; `vision_bbox` renders object/target boxes into the first frame without coordinate prompts; and `vision_bbox_and_bbox` combines rendered boxes with coordinate bounding-box prompts. The final SPOT configuration uses `vision_bbox_and_bbox`, so the task is visible both in image space and in explicit coordinate-token form.

Shared visual backbone. Both the task encoder and observation encoder use the same visual backbone interface. The backbone is implemented with `timm` and returns image tokens in (B, N, C)

format. The default backbone is `vit_base_patch14_dinov2`, with $C = 768$ and policy dimension $D = 768$. We also support SAM-Base, Perception Encoder-Base, SigLIP-Base, and EVA2-Base through their corresponding `timm` model names. When the backbone is frozen, its parameters are excluded from the optimizer and the module is kept in evaluation mode; when unfrozen, it is trained together with the policy.

Task encoder. The task encoder first applies the shared visual backbone to the first frame, optionally after visual prompt rendering. Image tokens are projected to dimension D and receive image type embeddings. Coordinate prompts are encoded separately: a bounding box contributes two corner tokens for the object and two corner tokens for the target, a point prompt contributes one object token and one target token, and the no-prompt setting uses two learned null tokens. Prompt tokens receive role/type embeddings before fusion.

The default fusion path is prompt-to-image cross-attention. Prompt tokens query first-frame image tokens through a Transformer decoder with 4 layers and 12 attention heads. With image-summary return enabled, the task encoder returns the first image summary token followed by the fused prompt tokens. Thus, the default bounding-box coordinate prompt produces five task tokens: one image-summary token and four fused coordinate-prompt tokens.

Observation encoder and conditioning sequence. The observation encoder applies the same image encoder and projection layer to the current frame, giving current-frame image tokens in the same feature space as the first-frame task tokens. When the history horizon $K > 0$, each previous 10D action is embedded by a lightweight MLP, `Linear(10,D)-GELU-Linear(D,D)`, then augmented with learnable temporal position embeddings and a history type embedding. The observation sequence is the concatenation of current-frame image tokens and history-action tokens.

The final conditioning sequence is

$$Z_{\text{cond}} = [Z_{\text{task}}; Z_{\text{obs}}],$$

which contains first-frame task evidence, current egocentric visual context, and recent motion context. The trajectory head uses this sequence as cross-attention memory while decoding the future action chunk.

C Training and Evaluation Details

C.1 Training Details

Default configuration. All baseline and ablation experiments use the same training protocol and change one factor at a time from the default SPOT configuration. The default model uses `vision_bbox_and_bbox` prompts, a frozen DINOv2 ViT-B/14 visual encoder, policy dimension $D = 768$, cross-attention task fusion, 4 fusion layers, 12 fusion heads, history horizon $K = 4$, a 6-layer 12-head trajectory decoder, and a flow-matching trajectory head. Final experiments use learning rate 2×10^{-4}, batch size 16, 30 epochs, validation every 2 epochs, checkpointing every 10 epochs, and 4 dataloader workers. The launch configuration uses 8 A6000Ada GPUs with bf16 mixed precision; the DINOv2 tuning ablation reduces the batch size to 8 because the visual backbone is unfrozen.

Split construction. The training split is deterministic. Episodes are grouped by scene and task, task names are shuffled with a seed-dependent scene-specific random generator, and validation tasks are selected according to the validation ratio. All episodes of a selected validation task are assigned to validation, preventing sample-level leakage across the same task split decision. Rank 0 writes a `split_manifest.json` containing train/validation episode keys, scene/task names, and paths, and training checks that no episode key appears in both splits.

Optimization. Training uses `accelerate` for distributed execution. The optimizer is AdamW over parameters with `requires_grad=True`; thus frozen visual backbones are excluded from optimization. Gradients are clipped to norm 1.0 when synchronized. The learning-rate scheduler is constructed with `diffusers.optimization.get_scheduler`, and the code supports constant, warmup, linear, cosine, cosine-with-restarts, and polynomial schedules.

Action normalization. Before optimizer construction, the policy computes per-action-dimension mean and standard deviation from training samples. Standard deviations are clamped to at least

Table 3: Ablation on DINOv2 encoder size. Lower is better for all metrics.

Split	Encoder	FDE	Pos. L2	Rot. L2	Grip. L1
All	DINOv2-Small	**0.1129**	0.0701	0.2124	0.01386
	DINOv2-Base	0.1147	0.0699	0.1965	0.01133
	DINOv2-Large	0.1135	**0.0679**	**0.1862**	**0.01081**
Scene 1	DINOv2-Small	0.1085	0.0685	0.1695	0.00719
	DINOv2-Base	0.1123	0.0700	0.1588	0.00693
	DINOv2-Large	**0.0988**	**0.0609**	**0.1500**	**0.00676**
Scene 2	DINOv2-Small	**0.1057**	**0.0617**	0.2187	**0.00885**
	DINOv2-Base	0.1116	0.0632	0.2093	0.00975
	DINOv2-Large	0.1244	0.0679	**0.2008**	0.00978
Scene 3	DINOv2-Small	0.1282	0.0829	0.2781	0.03091
	DINOv2-Base	**0.1224**	**0.0779**	0.2446	0.02048
	DINOv2-Large	0.1251	0.0796	**0.2300**	**0.01871**

10^{-4}. During training, future action chunks are normalized as $(a - \mu)/\sigma$ before being passed to the trajectory head; predictions are denormalized before evaluation or visualization.

Flow-matching head. The default flow-matching head predicts velocity over normalized action chunks. During training, time t is sampled from a Beta distribution with $\alpha = 1.5$ and $\beta = 1.0$, then clamped away from zero by $t = 0.999t + 0.001$. Given a clean action chunk x_0 and Gaussian noise ϵ, the interpolated sample is $x_t = (1 - t)x_0 + t\epsilon$, and the target velocity is $\epsilon - x_0$. Time is encoded with a continuous sinusoidal embedding over periods from 0.004 to 4.0, followed by a two-layer MLP.

Diffusion head. The alternative diffusion head uses a DDPM scheduler with the `squaredcos_cap_v2` beta schedule during training. It predicts either noise or the clean sample depending on the configured prediction type. The diffusion and flow-matching heads share the same high-level decoder structure: trajectory tokens with a time embedding attend to the conditioning tokens through a Transformer decoder and output an action chunk.

C.2 Evaluation Details

Validation protocol. Training-time validation runs every two epochs on a capped number of batches for fast feedback. It reports both overall metrics and per-scene metrics using validation episodes selected at training start. Final evaluation reuses the same validation episodes and evaluates the full validation set without batch truncation.

Inference. For the default flow-matching head, inference starts from Gaussian noise at $t = 1$ and integrates to $t = 0$ using forward Euler with 10 steps. The final evaluation uses deterministic initial action noise with sample seed 0. For the diffusion head, inference uses a DDIM scheduler with the same `squaredcos_cap_v2` beta schedule used during training.

Metrics and outputs. We report FDE for endpoint translation accuracy, Pos. L2 for mean relative translation error, Rot. L2 for mean 6D rotation error, and Grip. L1 for gripper-width error. Final evaluation writes overall and per-scene summaries to `eval_metrics/all/summary.json`, `eval_metrics/all/metrics.csv`, and the corresponding scene-specific files. For full-episode visualizations, predicted chunks are stitched in temporal order to inspect accumulated drift in addition to single-chunk accuracy.

D Additional Ablation: Vision Encoder Size

We further compare DINOv2 backbones of different sizes under the same SPOT setting. All variants use the combined visual and coordinate bounding-box prompt, frozen visual features, flow matching, and the same scene-aware evaluation protocol. Table 3 reports the results.

The results show that increasing encoder size mainly improves orientation and gripper prediction, while endpoint accuracy does not improve monotonically. DINOv2-Large [36] achieves the best

overall Pos. L2, Rot. L2, and Grip. L1, and is strongest on Scene 1. DINOv2-Small gives the lowest overall FDE and performs best on Scene 2 translation metrics, suggesting that larger visual capacity is not always necessary for short-horizon endpoint prediction. DINOv2-Base remains competitive and is strongest on Scene 3 FDE and Pos. L2, which is why we keep it as the default in the main experiments for a balanced accuracy-cost tradeoff.

E More Visualization Results

We provide more visualization results of full-episode trajectories from the three scenes in Figs. 11, 12, and 13. These examples complement the main qualitative results and show stitched predictions under structured layouts, cluttered layouts, and diverse cluttered subscenes. We also provide some first chunk visualization trajectories in Fig. 14.

F Limitations

SP-VTP is evaluated as open-loop trajectory prediction rather than closed-loop robot control. Although stitched trajectories indicate that SPOT preserves the coarse episode structure, local chunk errors can accumulate over long horizons, especially under large camera and EE motion. The current dataset focuses on fork pick-and-place tasks with table-top receptacles, so the scope of the claims is limited to spatially prompted egocentric manipulation in this task family. Broader objects, deformable items, multi-step tasks, and real-time closed-loop execution remain important directions for future work.

The method also assumes that the first-frame object and target prompts are available and reasonably accurate. Strong prompt noise, severe occlusion, or target regions that leave the camera view may reduce performance. Finally, while frozen foundation visual encoders improve generalization in our experiments, their behavior may depend on the visual domain, camera viewpoint, and object categories.

G Ethics and Broader Impact

EgoSPT is collected and annotated by trained experts rather than crowdsourced workers. The videos are captured from an egocentric device during table-top manipulation and are intended to avoid collecting identifiable personal information. The dataset is designed for research on visually grounded manipulation and does not involve high-risk generated media, scraped web data, or personal decision making.

Spatial prompting can make robot task specification more direct and accessible, especially in cluttered scenes where language instructions are ambiguous. At the same time, improved manipulation policies may be unsafe if deployed without closed-loop monitoring, collision checking, or task-level constraints. Any deployment should therefore include safety checks, workspace limits, and human supervision appropriate to the robot platform and environment.

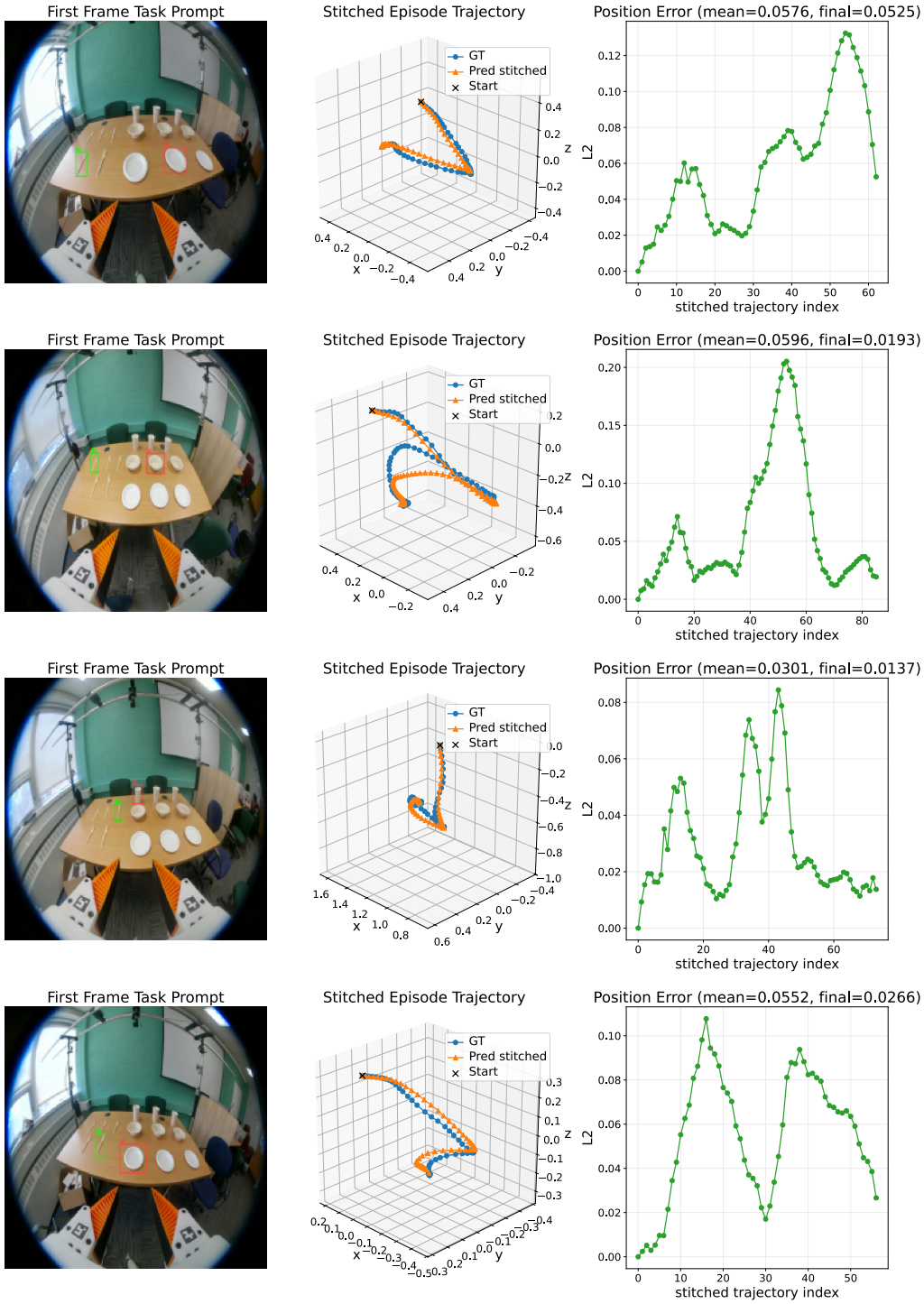

Figure 11: **Additional full-trajectory visualization for Scene 1.** We show a representative stitched full-episode prediction under a structured layout.

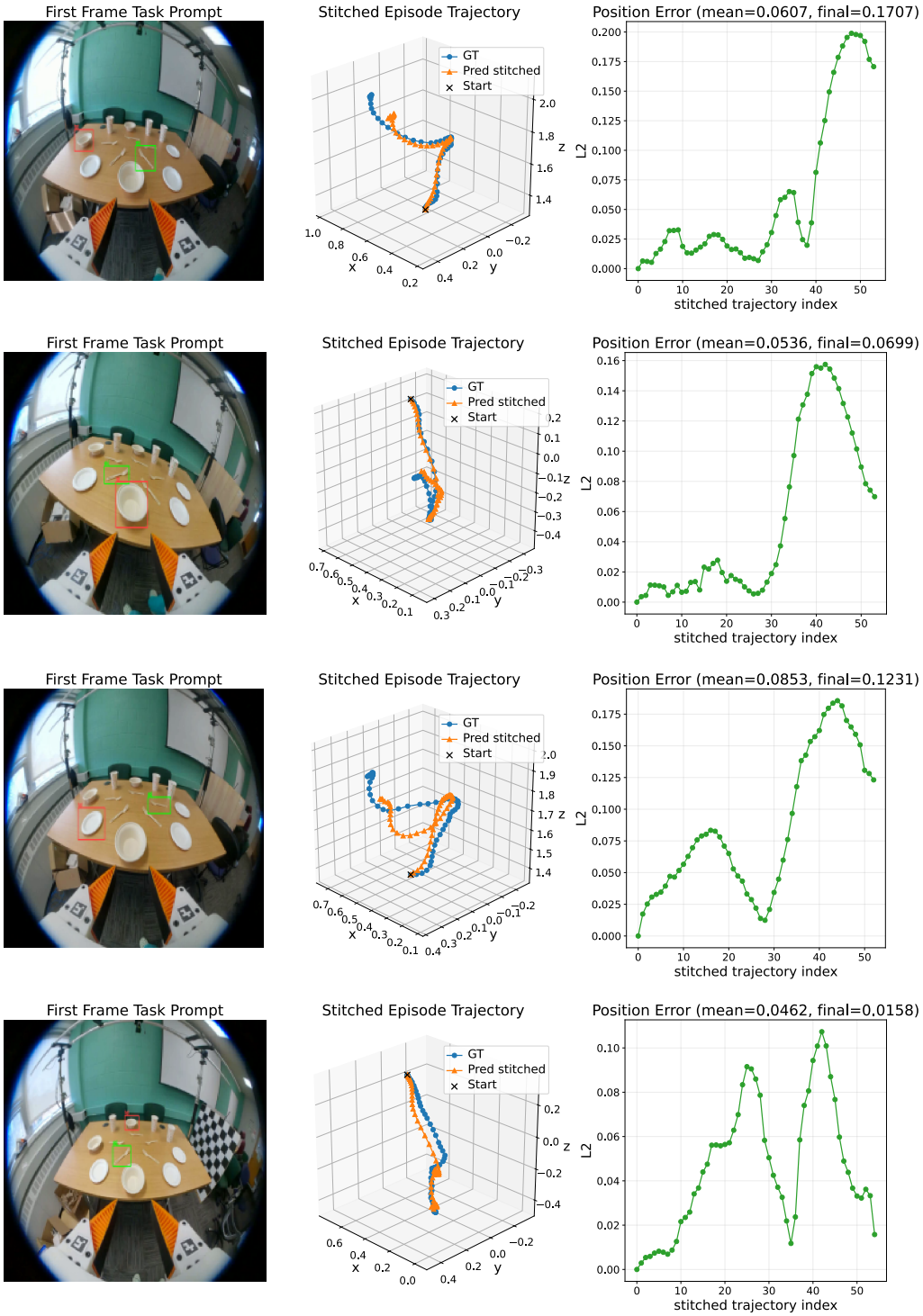

Figure 12: **Additional full-trajectory visualization for Scene 2.** We show a representative stitched full-episode prediction under a cluttered layout.

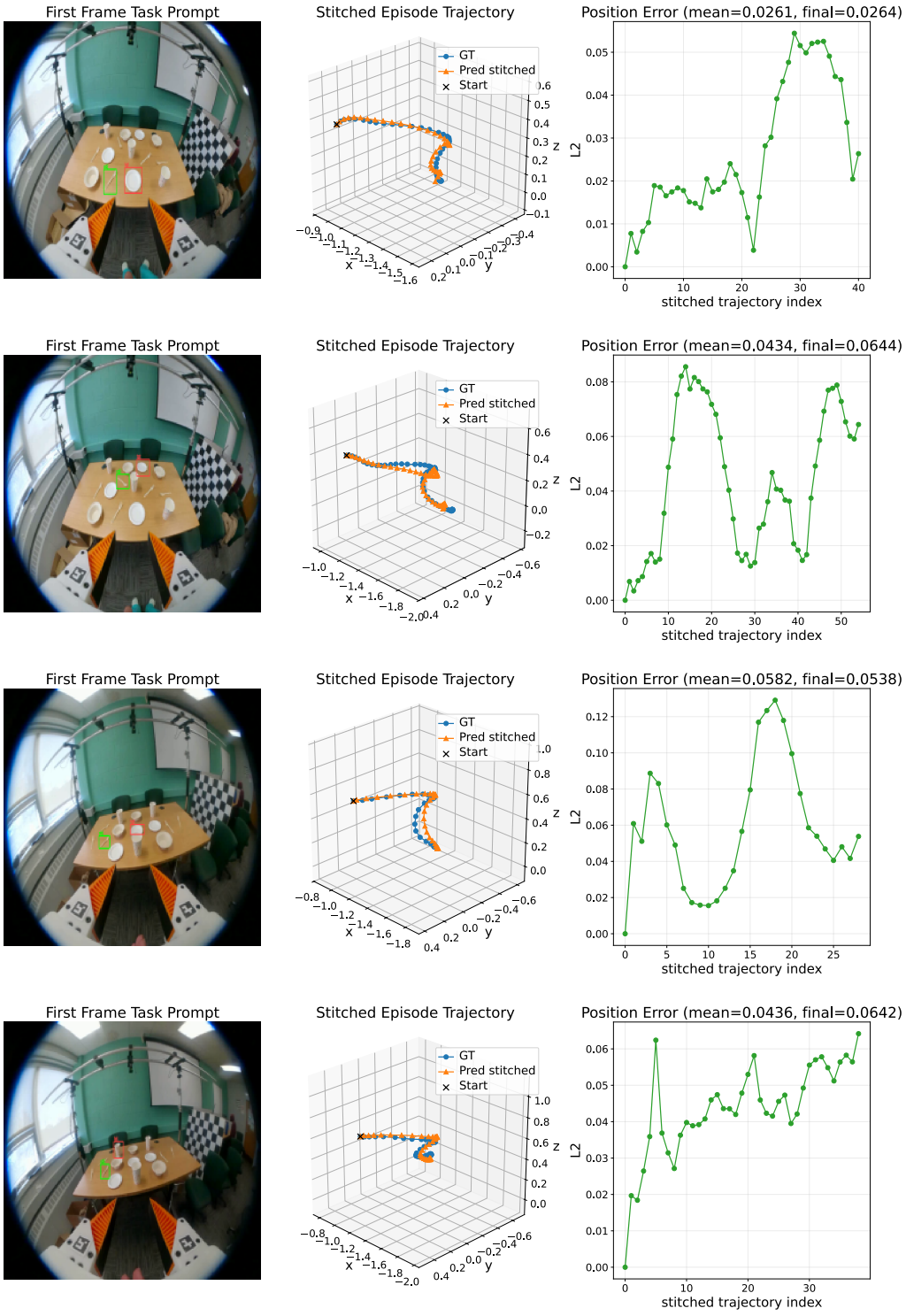

Figure 13: **Additional full-trajectory visualization for Scene 3.** We show a representative stitched full-episode prediction under diverse cluttered subscenes.

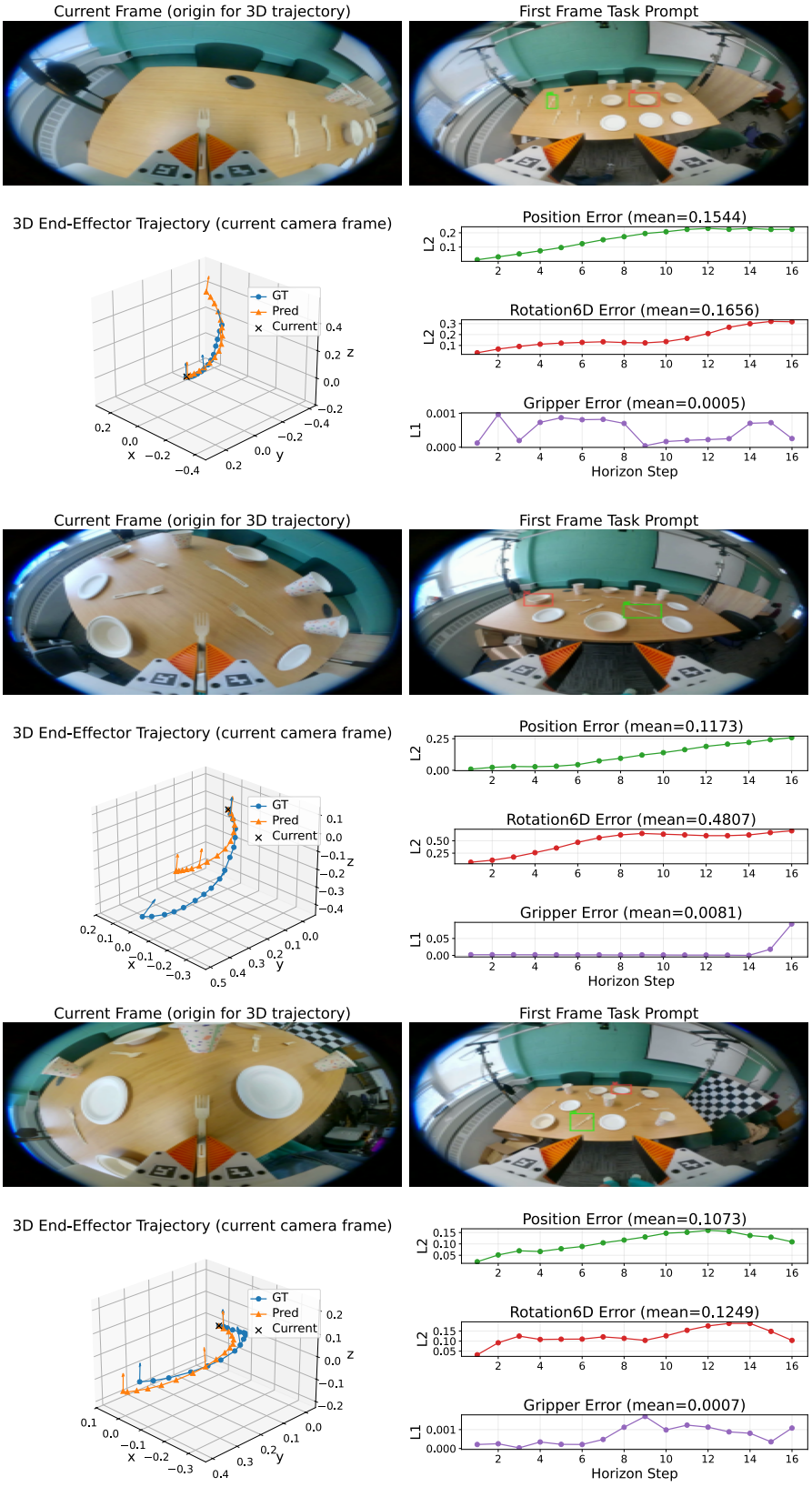

Figure 14: **Additional first-chunk visualization for three scenes.** We present three representative first-chunk visualization results of three scenes from top to down.