Triplet2Track: A Hierarchical System with Object-Centric Representations for Reliable Long-Horizon Manipulation

Jianxiang Liu[1], Gaojing Zhang[2], Chuan Wen[1], Qipeng Liu[1], Yuxuan Zhao[1], Ning Guo[1], Wenzhao Lian[1,*]

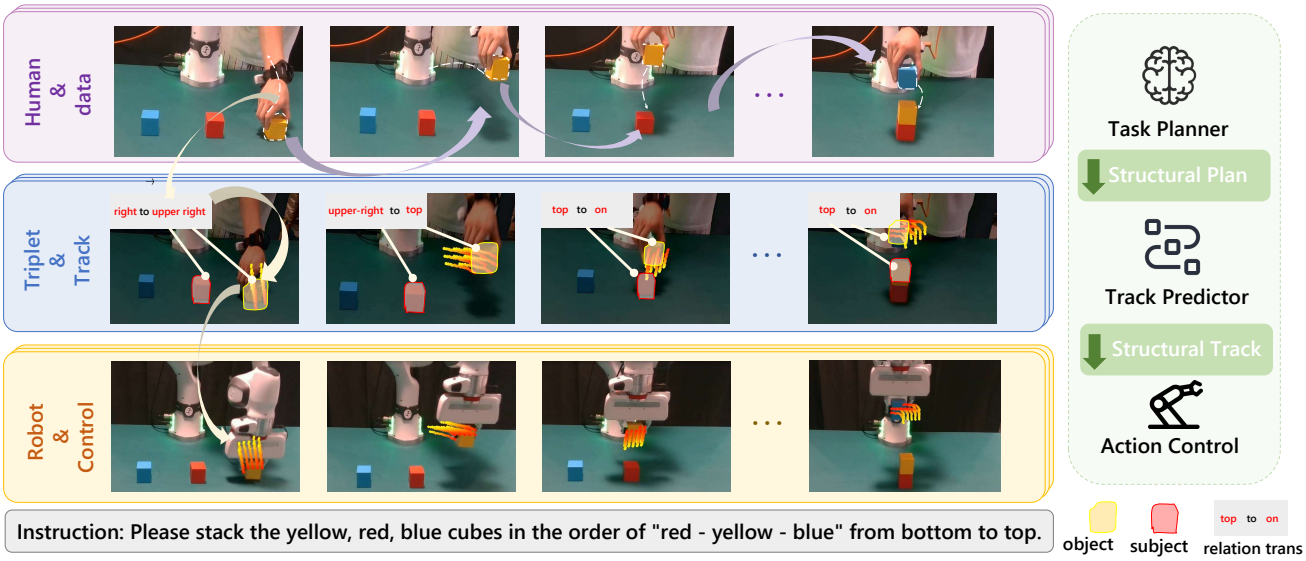

Fig. 1: TTS is a hierarchical closed-loop system for reliable long-horizon manipulation, using instance-grounded triplets as structural task representations for planning and continuous point tracks as physical motion representations for execution. This design reduces planning hallucinations and supports data-efficient manipulation under limited robot data.

***Abstract*— Ensuring reliability in uncertain environments remains difficult for long-horizon robotic manipulation. End-to-end VLA models are data-heavy and opaque, making diagnosis and verification difficult. Hierarchical pipelines are more interpretable, but their plans are often weakly grounded in observations, weakly aligned with low-level actions, and computed without online feedback, leading to open-loop behavior and hallucinations. To address these issues, we introduce the Triplet-to-Track System (TTS), a closed-loop long-horizon imitation learning system that uses human videos to reduce reliance on robot-collected data. TTS represents high-level subgoals as instance-grounded triplets, translates them into continuous track priors for execution, and monitors task progress from observations for online replanning. Across diverse real-world long-horizon tasks, TTS achieves a 74.8% average success rate and supports object-level and compositional generalization.**

I. Introduction

Long-horizon manipulation arises in many everyday tasks [1]–[3], motivating the need for general policies that can reliably execute extended behaviors. Compared with short-horizon settings with explicit goals and near-linear structure [4]–[6], these tasks often involve goals that are only partially specified in the initial observations and become operationally verifiable only after a sequence of intermediate interactions [7]. The central challenge is therefore not merely reactive control, but maintaining plans that remain correct and consistent with evolving observations over extended horizons. A promising direction is to use goals that are directly checkable from visual observations and to monitor execution online so that deviations can be detected before errors accumulate [8], [9]. Existing methods along this direction mainly fall into two paradigms: end-to-end visual–language–action (VLA) systems [10]–[12] and hierarchical pipelines [13]–[15].

End-to-end visual–language–action (VLA) systems typically learn a direct mapping from raw images and language instructions to low-level actions, implicitly coupling perception, language understanding, planning, and control in a shared latent space. While this design avoids hand-crafted abstractions, it also forgoes structured, human-interpretable modeling of task semantics [10]. As a result, it has two main drawbacks. First, it exacerbates the long-standing data-efficiency challenge in imitation learning [16], [17]. In long-horizon settings, extended episodes substantially increase supervision demands and make it far more costly to collect sufficiently diverse demonstrations than in short-horizon

This work was supported by the School of Artificial Intelligence, Shanghai Jiao Tong University.

[1]School of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China.
[2]School of Engineering and Informatics, University of Sussex, Brighton, U.K.
*Corresponding author: lianwenzhao@sjtu.edu.cn.

tasks [18]. Second, because the reasoning process is internalized within the network, such policies are difficult to interpret and diagnose, which can lead to unreliable decisions at execution time [19].

Many hierarchical approaches reduce data requirements by combining human demonstrations with high-level planners built on pretrained LLMs or VLMs, thereby leveraging web-scale knowledge and reducing reliance on robot-collected data [13], [14], [20]. However, these approaches still have important limitations. QA-conditioned policies [13], [20] typically produce free-form language goals without explicit structural or spatial constraints, making hallucinated objects and relations common. Semantic decompositions [14], [21] introduce symbolic task structure, but remain weakly grounded in concrete object instances and scene states. As a result, when hallucinations or semantic drift occur, these systems often lack closed-loop mechanisms to verify subgoals against visual observations or revise inconsistent plans, leaving many failures undetected and unrecovered.

To jointly address the data inefficiency and limited verifiability of end-to-end VLAs, as well as the weak grounding and lack of automatic error detection and recovery in hierarchical schemes, we present the **T**riplet-to-**T**rack **S**ystem (TTS), a framework for reliable long-horizon imitation learning with limited robot data. TTS uses discrete triplets for planning and continuous tracks for execution. Specifically, it represents task goals with object-centric triplets that bind spatial relations to concrete object instances, and maps subgoal transitions to continuous tracks that low-level controllers can consume as physical priors [16]. Unlike prior methods, TTS makes the planning–control interface explicit, instance- and spatially grounded, and visually verifiable.

Our contributions are three-fold:

1) We introduce **T**riplet-to-**T**rack **S**ystem (TTS), a closed-loop long-horizon imitation learning system that uses human videos to reduce reliance on robot-collected data, while improving execution reliability through observation-based monitoring and replanning.
2) We present a structured way to connect planning and control, where high-level subgoals are expressed as instance-grounded triplets and translated into continuous track priors for low-level execution, making task progress explicit and verifiable.
3) We validate TTS on real-world long-horizon manipulation tasks under perturbation and limited-data settings, showing reliable performance together with generalization to unseen object instances and task compositions.

The remainder of this paper reviews related work and problem formulation in Sections II–III, presents TTS in Section IV, reports real-world experiments in Section V, and concludes the paper in Section VI.

II. Related work

A. Methods for Long-horizon tasks

Long-horizon manipulation is commonly approached via two families of methods. End-to-end VLA systems map pixels and language directly to actions [10]–[12], [17], [22]–[24]. These models can scale with data but remain opaque and data-hungry, and their behavior often degrades under distribution shift. Hierarchical pipelines instead use a high-level planner (typically a VLM) coupled with a low-level controller [13]–[15], [25]–[27]. Planners usually emit semantic sketches, which improves interpretability but still risks semantic hallucination [13], [20] and suffers from planner–controller interface mismatches, especially in open-loop settings or under imperfect synchronization.

A complementary line decomposes tasks into atomic units in semantic or relational space [14], [21]. While these approaches increase data efficiency and modularity, their abstractions often remain semantic rather than instance-level and spatially grounded, limiting verifiability against concrete scene geometry and contact. Recent systems [7], [9] add periodic replanning or feedback, yet without predicates that are directly checkable in perception, closed-loop reliability can still be brittle. These limitations motivate a representation that is hierarchical, instance-grounded, visually verifiable, and directly connected to low-level execution.

B. Policy learning from action-free videos

Videos carry rich cues about behavior and scene dynamics [28], [29], but often lack action supervision. Prior work tackling this typically follows three routes: (i) inverse-dynamics pseudo labeling—data-efficient [30], [31] but brittle for continuous actions and long horizons; (ii) self-supervised feature pretraining [32]–[34]—useful for perception yet weak on temporal control structure; and (iii) video prediction or goal-image generation—promising [35], [36] but prone to hallucinated geometry or contact and high inference cost. An alternative is to predict *point tracks* as physically grounded motion priors [16], [37], [38], enabling closed-loop guidance with low compute. However, such track-only priors lack global task-level context, so execution can become myopic and get trapped in local limit cycles. This motivates our use of tracks not as a standalone execution prior, but as a task-conditioned interface driven by explicit and verifiable high-level subgoals.

III. Preliminaries

We model real-world long-horizon tasks as a contextual POMDP $\mathcal{M}_\ell = (\mathcal{S}, \mathcal{A}, P, \Omega, \mathcal{O}, \rho_0)$, where at time t, the state $s_t \in \mathcal{S}$ produces an observation $o_t \in \Omega$ via $\mathcal{O}(o_t \mid s_t)$, after which the agent selects an action $a_t \in \mathcal{A}$ using $\pi(\cdot \mid o_{1:t}, \ell)$, and the next state follows $P(s_{t+1} \mid s_t, a_t)$ with $s_0 \sim \rho_0$.

Guided by this formulation, we decouple high-level goal specification from low-level execution and use two complementary representations. The triplet space $\mathcal{C} = \mathcal{I} \times \mathcal{R} \times \mathcal{I}$ contains predicates $c = (x, r, y)$ over object instances $\mathcal{I}$ and relations $\mathcal{R}$, where R is a predefined vocabulary of spatial and gripper–object relations, such as on, over, and grasped. The function $\mathrm{hold}(o_t, c) \in \{0, 1\}$ indicates whether relation r holds for the ordered instance pair (x, y) in the current scene. The track space $\mathcal{T}$ comprises entity-aligned

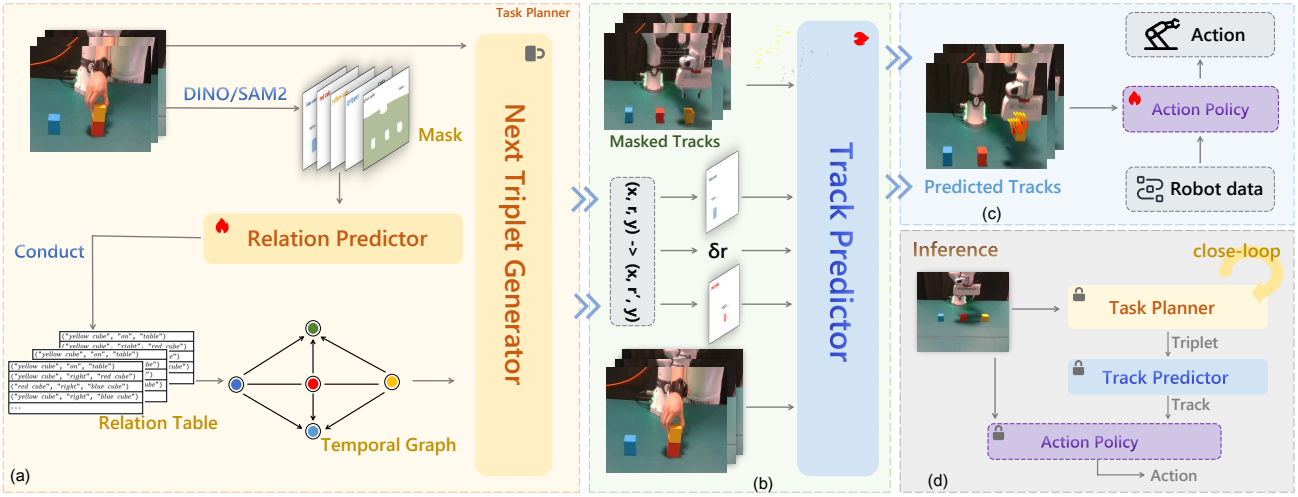

Fig. 2: : TTS pipeline. (a) The Relation Predictor extracts per-frame triplets and constructs a temporal graph, while the Next-Triplet Generator selects the next target triplet. (b) The Track Predictor converts the target transition into continuous point tracks. (c) The Action Policy maps tracks to actions. (d) During inference, TTS detects the current triplets, selects a feasible target, predicts track priors, and executes actions in a closed loop.

motion descriptors $\tau = \{z_t, \ldots, z_{t+L-1}\}$, with $z_i \in \mathbb{R}^d$, that parameterize temporally extended control.

Given (o_t, ℓ, c_k), the Task Planner proposes the next subgoal c_{k+1}, and the Track Predictor maps the discrete transition $c_k \to c_{k+1}$ to a continuous track τ_k, which the low-level policy uses to generate action $a_{t:t+L-1} \sim \pi_L(\cdot \mid o_t, c_k, \tau_k)$. The detailed closed-loop execution rule, including guard evaluation and replanning, is described in Sec. IV.

IV. Method

In this work, we aim to build a robot control system that is reliable, robust, and data-efficient. We propose the Triplet-to-Track System (TTS), which organizes control into three layers: **a Task Planner** that turns human videos into discrete, validatable, instance-spatial triplets; **a Track Predictor** that compiles each triplet into a continuous, entity-aligned track; **an Action Policy** that uses the predicted track as a physical prior to produce actions. Triplets bind object pairs and relations to concrete physical instances and are directly checkable from observations, which helps reduce planning hallucinations and supports spatial reasoning. However, acting directly on triplets leaves a gap between discrete subgoals and continuous control. TTS bridges this gap with a Track Predictor that generates fine-grained, structured, and instance-grounded tracks. The Track Predictor is pretrained on action-free videos, which improves data efficiency under limited robot data [16], [37], [38]. Conditioning tracks on triplet priors further sharpens relational transformation learning and reduces sensitivity to appearance variations, improving object-level and compositional generalization under similar grasp poses. Together with online detection and replanning, these components form a closed-loop system for long-horizon manipulation, as shown in Fig. 2.

A. Task Planner for discrete, abstract triplets

We seek a task planner that (i) infers triplets among objects from an admissible set to extract an object-relation graph from the current frame, and (ii) proposes the next discrete subgoal as a triplet. Operationally, the planner forms a two-stage perception-to-decision pipeline: a **Relation Predictor** and a **Graph-Guarded Next Triplet Generator**.

Relation Predictor. Reliable detection of spatial relations is challenging, as a universal zero-shot detector would require large-scale relation annotations, diverse data, and heavy pretraining [39]–[41]. Since our goal is to reduce planning hallucination and improve reliability, rather than train such a detector, we adopt a low-overhead design with two complementary modules: a **L**ightweight **T**ransformer-based **P**redictor (LTP) and a LoRA-tuned VLM branch. The LTP is a pairwise relation classifier built with four Transformer layers and two linear layers, taking per-instance SAM2 decoder tokens as input and predicting relation logits for each ordered object pair.

Using Grounding-DINO [42] and SAM2 [43] for instance detection, we feed the per-instance tokens from the SAM2 decoder directly into the LTP. The LTP outputs a tensor $P_t \in \mathbb{R}^{n \times n \times |\mathcal{R}|}$, where n is the number of detected object instances in the current frame, (x, y) denotes an ordered object pair among these n instances, and $\mathcal{R}$ is a predefined relation vocabulary. For any object pair (x, y), the predicted relation is obtained by applying a softmax over relation logits: $\hat{r}_{xy}^{\mathrm{LTP}}(t) = \arg\max_{r \in \mathcal{R}} P_t(x, y, r)$.

The dominant error of the LTP is *spurious flips* on relations that should remain stable over short horizons, where the prediction changes for one frame and then quickly reverts. Formally, on adjacent frames $(t, t+1)$, we define FP when $r_{xy}(t) = r_{xy}(t+1)$ but $\hat{r}_{xy}^{\mathrm{LTP}}(t) \neq \hat{r}_{xy}^{\mathrm{LTP}}(t+1)$ (spurious flip), and FN when $r_{xy}(t) \neq r_{xy}(t+1)$ but $\hat{r}_{xy}^{\mathrm{LTP}}(t) = \hat{r}_{xy}^{\mathrm{LTP}}(t+1)$ (missing change). With per-frame accuracy p and per-step change rate ρ, a two-frame analysis yields $\mathbb{E}[\mathrm{FP}] \geq (1-\rho)\, 2p(1-p)$ and $\mathbb{E}[\mathrm{FN}] \leq \rho(1-p^2)$, hence spurious flips dominate whenever $\rho < \frac{2p}{1+3p}$. In practice, we observe $p \in (0.80, 0.90)$ and $\rho \approx 0.02$, so $\rho \ll \frac{2p}{1+3p}$, which

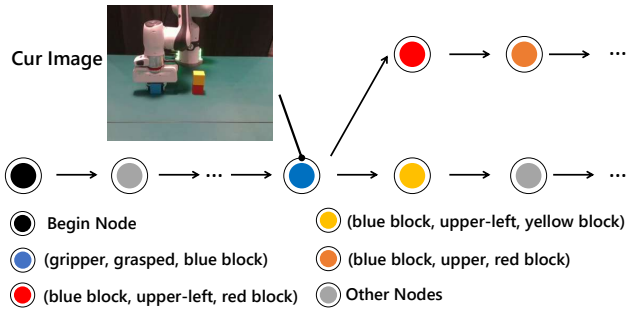

Fig. 3: An example of node transformation in a graph.

matches the transient few-frame flips observed in practice. A naive sliding-window average over per-frame probability vectors reduces but does not remove this jitter.

We therefore fine-tune InternVL2.5 [44] as a high-precision verifier and invoke it only when a change is suspected. This event-triggered VLM improves reliability while avoiding the cost of per-frame VLM inference. When fine-tuning the VLM, we exploit the structured, object-anchored form of triplets to classify within a masked relation set rather than generate free-form text: given (x, y), we condition the prompt on their instance masks and restrict the label space to feasible relations for that pair. For training the LTP and the VLM branch, we use only five sequences per task. Since relation changes are sparse, we annotate only change points to keep supervision cost low.

In summary, inference follows a cascade: a high-rate spatial predictor monitors guarded pairs and raises a flag only when the windowed argmax changes, indicating a suspected transition. Upon a flag, the fine-tuned VLM verifier computes $q(r \mid o_t, x, y, \ell)$ to confirm the change. This high-recall detector together with high-precision verification preserves responsiveness while sharply reducing false positives, especially for context triplets that should remain unchanged.

Graph-Guarded Next Triplet Generator. We use a zero-shot VLM (e.g., GPT-4o [45]) to propose the next subgoal. Unlike prior work, we constrain its proposals with a triplet-based temporal graph to reduce planning hallucinations. We build a directed graph $G = (V, E)$, where each node is a task-relevant triplet $c = (x, r, y)$ and each edge represents a feasible transition. The graph is constructed from action-free videos: at each frame t, the Relation Predictor outputs a relation table R_t; when two consecutive tables R_t and R_{t+1} differ by a transition $(x, r, y) \to (x, r', y)$, we add the edge $c \to c'$ and merge duplicates across demonstrations. Each demonstration forms one execution path, while different execution orders add branches.

The resulting graph serves both as a progress index and as a feasibility filter. At step k, suppose the current node is $c_k = (x, r, y)$. Its outgoing edges enumerate the feasible one-step updates observed in human data, allowing only observed transitions and excluding invalid ones. We encode G by its adjacency matrix A, and use the row $A[i_k, :]$, where $i_k = \mathrm{index}(c_k)$, as a binary mask $\mathcal{M}_k$ over the candidate next nodes. The VLM receives $\{o_t, \ell, c_k, \mathcal{M}_k\}$ through a single prompt template and scores only masked candidates; proposals outside the mask are ignored. In this way, open-ended generation is turned into verifiable instance-level selection, which keeps attention on grounded object relations and suppresses hallucinated transitions. The graph is extracted from human demonstrations rather than manually specified, and can be naturally expanded as additional human videos become available to cover new execution orders and transition patterns. For example, in Fig. 3, if c_k is the blue node, then the next step can choose only the red or yellow nodes, while all other nodes are excluded.

Inference starts from the initial node c_0. During execution, the guard set maintains the latest validated relation for each ordered instance pair (x, y) encountered in completed subgoals: when a subgoal $c = (x, r, y)$ is completed, the entry $(x, y) \mapsto r$ is inserted if absent, and otherwise the previous relation for (x, y) is replaced by r. At each step, the Relation Predictor evaluates the current guard set and returns an indicator $G_k \in \{0, 1\}$, where $G_k = 1$ indicates that all guards hold. If $G_k = 1$, the next subgoal is proposed within the masked neighborhood of c_k; otherwise, the planner relocalizes to c_0 and replans. Formally,

$$c_{k+1} = \begin{cases} \mathrm{VLM}_{\mathrm{planner}}(o_t, \ell, c_k, \mathcal{M}_k), & G_k = 1, \\ c_0, & G_k = 0. \end{cases}$$

B. Track Predictor for continuous tracks

To map discrete triplets to continuous tracks, we train a Track Predictor. Unlike track-only motion priors, our predictor is conditioned on triplet transitions, so the predicted tracks are tied to explicit object pairs and relation changes rather than inferred solely from local motion cues. Track representations also exhibit cross-embodiment capability [16], [37], enabling pretraining on action-free videos and reducing the amount of action-labeled robot data required.

Data preparation. For each sequence, we first derive the triplet transition and the aligned video segment, following the same procedure used to build the temporal graph in the Task Planner. Using Grounding-DINO [42] and SAM2 [43], we obtain pixel-level masks for x and y, which are reused as inputs (M_x, M_y) to explicitly localize the entities and suppress background and appearance distractions. Using the mask-derived boxes, we seed points inside each instance and track them with CoTracker [46] to obtain entity-aligned trajectories $V_{t:t+H}$. The relation change $r \to r'$ is encoded as a discrete token via a text encoder, yielding training samples $\mathcal{D}_{x:y} = \{(O_{t:t+H}, M_x, M_y, V_{t:t+H}, e(r \to r'))\}$. We refer to $(M_x, M_y, e(r \to r'))$ as the triplet-transition prior, which instantiates the discrete transition $(x, r \to r', y)$ for learning.

Multimodal masked track prediction. Inspired by ATM [16], we cast track forecasting as a multimodal masked prediction task. Given the visual observation o_t, current point positions v_t, instance masks M_x, M_y, and a relation-delta embedding $e(r \to r')$, the model predicts the next positions v_{t+1} for each query point. We encode M_x and M_y with a dedicated mask encoder to produce instance-anchored mask tokens, and embed $e(r \to r')$ into the shared token space as a conditioning signal. Compared with formatting

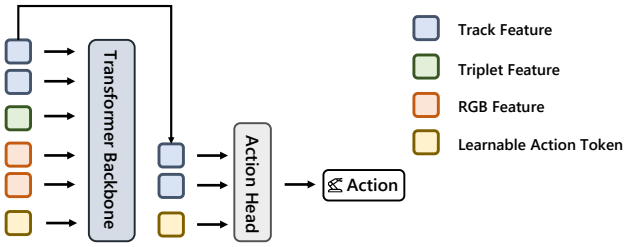

Fig. 4: The Action Policy consists of a Transformer backbone and an MLP-based Action Head.

the same information as a generic text prompt, this triplet-conditioned design provides a stronger inductive bias and clearer spatial supervision for learning relation-dependent motion. All other preprocessing follows ATM [16]. The fused tokens are then processed by a Transformer backbone, and a trajectory decoder produces continuous, entity-aligned tracks that realize the discrete update $(x, r, y) \rightarrow (x, r', y)$ as a motion prior for the low-level policy.

To enhance stability, we introduce two auxiliary objectives. The first is a masked-patch reconstruction objective, in which a decoder reconstructs masked image patches from visual features processed by the Transformer backbone. The second is relation-change recognition: using the object and subject mask decoder features together with intermediate text decoder tokens, the model predicts the relation-change class δr, strengthening its modeling of relation changes. Conditioning tracks on triplet-transition priors also helps reduce sensitivity to appearance variation and improve object-level and compositional generalization under similar grasp poses.

C. Policy Learning for Action

To map the predicted track prior to the robot action space, we train a track-guided behavior cloning policy on a small set of action-labeled robot data. Preprocessing follows the Track Predictor. We denote training samples by $\mathcal{D}_t^{\text{act}} = \{O_{t:t+H}, S_t, A_t, M_x, M_y, e\}$, where S_t denotes the joint and gripper states, and A_t is the action label consisting of a 7-dimensional end-effector pose and a scalar gripper width.

Given $(O_{t:t+H}, M_x, M_y, e)$, the Track Predictor first produces a short-horizon track prior $p_{t:t+H}$. The low-level policy first fuses the observation features, triplet-transition prior, and track prior, and processes them using a Transformer backbone with a learnable action token. The resulting action token is then fused again with the track prior and decoded by an action head to predict the current robot action a_t. In this way, the policy conditions action generation on both the local motion prior and the high-level relational context. The policy architecture is illustrated in Fig. 4. We train the policy with an L_2 action-regression loss against A_t.

Inference follows a receding-horizon scheme. At time t, given (o_t, ℓ, c_k), the Task Planner sets $c_{k+1} = (x, r', y)$ as the active goal. Every 5 steps, the Relation Predictor checks whether c_{k+1} has been achieved. If not, execution continues under the current goal; otherwise, the Next-Triplet Generator issues c_{k+2}, which becomes the new active goal. Conditioned on the transition $c_k \rightarrow c_{k+1}$, we compute a short-horizon track prior $p_{t:t+H}$ from the two camera views, and the low-level policy consumes this prior together with the current observation to predict the current action a_t. The action is executed, the observation is updated, and the horizon recedes. If a guard violation is detected, the planner relocalizes to the initial node of the graph and replans before continuing.

V. Experiments

We evaluate our approach through the following questions:

Q1: Performance & Generalization. Compared with representative baselines, does our model achieve strong performance on real-world long-horizon manipulation while maintaining high success rates under object-level and compositional generalization?

Q2: Reliability & Stability. Does TTS enable closed-loop, disturbance-robust execution through online replanning from observation feedback, while reducing hallucination and keeping goals and execution traceable through its structured, instance-spatial representation?

Q3: Design Justification. Are the components of the Triplet-to-Track architecture necessary and well motivated, and what are their individual contributions?

A. Experimental Setup

Tasks and datasets. We evaluate language-conditioned long-horizon manipulation on a physical 7-DoF Franka robot in the real world using three task families, each probing a different aspect of the system.

- **Dynamic scale balancing (DSB).** This task evaluates closed-loop reasoning and online replanning in a two-pan balancing problem. A pile of toy frogs is placed in the left pan, and the goal is to balance the scale by grasping and placing the brown, blue, and red weights in the right pan. The task is always solvable. For training, we do not enumerate all balancing configurations. Instead, we collect videos containing primitive pick-and-place segments for each weight, so that the policy learns the track-to-action mapping while multi-step decision making is deferred to test-time planning.
- **Sequential stacking (SS).** This task evaluates whether the system can execute a prescribed multi-step order under rigid-body constraints. The goal is to stack the blocks from bottom to top in a specified order, such as red-yellow-blue. During training, the robot sees only one bottom-to-top order, while test-time evaluation requires generalization to unseen orderings.
- **Diverse pick-and-place (DPP).** This task evaluates goal understanding and object-level generalization. We consider two specifications: (i) placing the bitter melon into the blue box and the yellow potato into the green box, and (ii) placing the yellow, red, and pink cubes into the green box.

We label a task instance as *Seen* if both its object set and execution order appear in the robot training data; otherwise, it is labeled *Unseen*. The unseen setting is further divided into *compositional* and *cross-category* generalization. For **SS**, unseen evaluation requires generalizing from the training

order "red–yellow–blue" to a novel order such as "red–blue–yellow," which tests compositional generalization over the same object set. For **DPP**, unseen evaluation substitutes object categories or attributes at test time, e.g., replacing *bitter melon* with *corn* or *red tomato*, which tests cross-category generalization. For **DSB**, after each pick-and-place step, if the scale remains unbalanced, the system replans the next move from the latest observation, which evaluates closed-loop planning under evolving task states.

Objects are randomly positioned within a designated workspace. The evaluated episodes span multi-step horizons, with DPP, SS, and DSB involving up to nine, eleven, and twelve subgoals, respectively. For each task, we collect 30 robot demonstrations and 50 human demonstrations, with 5 trajectory annotations for the robot and human videos, respectively. The human demonstrations contain the object variations and execution orders used for generalization, whereas the robot demonstrations do not.

TABLE I: Success rates on long-horizon tasks (%).

Method	Dynamic Sacl-B		Seq-Stacking		Diverse Pick&Place		
	Seen	Perturbed	Seen	Unseen	Seen	Unseen	Perturbed
FT-Pi0.5 [47]	13.3	0.0	25.0	0.0	40.0	16.7	40.0
SeeDo [13]	26.7	0.0	41.7	40.0	50.0	54.5	0.0
PALO [14]	14.3	0.0	16.7	18.2	42.9	36.4	0.0
TTS	**78.6**	**60.0**	**83.3**	**66.7**	**84.6**	**70.0**	**80.0**

Hardware and implementation. We use two 1280×720 Intel RealSense D435i cameras for third-person and eye-in-hand views of a 7-DoF Franka arm. All models are implemented in PyTorch with CUDA 11.8 and trained on two NVIDIA A100 GPUs. The Action Policy runs at 15 Hz, with relation checks every five steps.

B. Baselines and Results

We compare TTS with three representative baselines. **FT-Pi0.5** [47] denotes a fine-tuned Pi0.5 model, i.e., an end-to-end VLA baseline that maps images and language directly to actions. We include it to evaluate whether a direct action-generation policy can handle long-horizon tasks with both dexterity and combinatorial structure under our low-data setting. **SeeDo** [13] is a hierarchical baseline in which a VLM produces a QA-style text plan executed with code-as-policy, representing QA-driven planning from human videos. **PALO** [14] is a semantic planning baseline in which a VLM recursively decomposes a task embedding into executable atomic actions, representing VLM-driven subgoal setting in semantic space. For fairness, our Relation Predictor is used only to detect task completion and guard violations, and does not provide additional spatial-relation supervision to the zero-shot VLM planners beyond the initial frame.

Results. Across the seven settings in Table I, TTS achieves an average success rate of 74.8%, compared with 30.4% for the best-performing baseline, SeeDo. On **Diverse pick-and-place**, SeeDo and PALO fail mainly because free-form goal generation is weakly constrained by observations, making them prone to spatial hallucinations such as selecting the wrong object after the interaction state has changed.

TABLE II: Planner variants: rollout and overall.

Method	Rollout Time ↓ (s)	Overall Accuracy ↑ (%)
Only LTP	3.4×10^{-3}	84.3
Only VLM	202.6	98.8
LTP + VLM	32.2	98.4

On **Sequential stacking**, the dominant failures are logical misordering and drift in multi-step goal tracking, for example grasping an object that is not required at the current stage or executing the prescribed order incorrectly. These errors become more frequent when subgoals are represented only in free-form or weakly grounded semantic space. In contrast, explicit instance-grounded subgoals together with observation-based checking help mitigate these failures. On **Dynamic scale balancing**, the main challenge is that the correct next action depends on the evolving physical state after each pick-and-place step. Here, baselines are less stable because they do not explicitly maintain closed-loop planning over changing observations. In the case of FT-Pi0.5, visually similar objects often induce indecision or misselection under our task-specific low-data adaptation setting. Overall, these results indicate that structured, instance-grounded subgoals together with track-based execution priors improve reliability on both seen and unseen long-horizon tasks.

C. Reliability and Stability

We evaluate reliability and execution stability through controlled perturbation tests, together with qualitative analyses of interpretability and hallucination suppression.

Closed-Loop Robustness under Perturbations. We test robustness by injecting perturbations during execution. In **Dynamic scale balancing**, we add or remove frog-shaped weights while keeping the task solvable. In **Diverse pick-and-place**, we remove the target immediately after grasping to trigger re-localization and re-grasp planning. Success rates under these perturbations are reported in Table I.

Triplets and Tracks for Interpretability and Hallucination Suppression. Fig. 5 visualizes representative triplet plans and track trajectories. Because both representations are explicit, execution failures can be traced to the corresponding planning or control stage. We observe that replacing explicit, validatable triplets with a semantic VLM plan (e.g., PALO) leads to a clear drop in performance (Table I), with failures often caused by planning hallucinations such as incorrect in–out relations, object mislocalization, and redundant picks. Removing the temporal-graph constraint further increases such errors (Sec. D), suggesting that triplets together with the graph help suppress hallucinated transitions by restricting proposals to observation-consistent and feasible updates.

At execution time, TTS uses object tracks as a physically grounded interface between planning and control. These tracks are explicit and visualizable, keep decision traceable, and support stable closed-loop behavior under limited robot demonstrations. Taken together, these results indicate that triplet-level structure mainly improves planning reliability, while track-based priors help stabilize downstream execution.

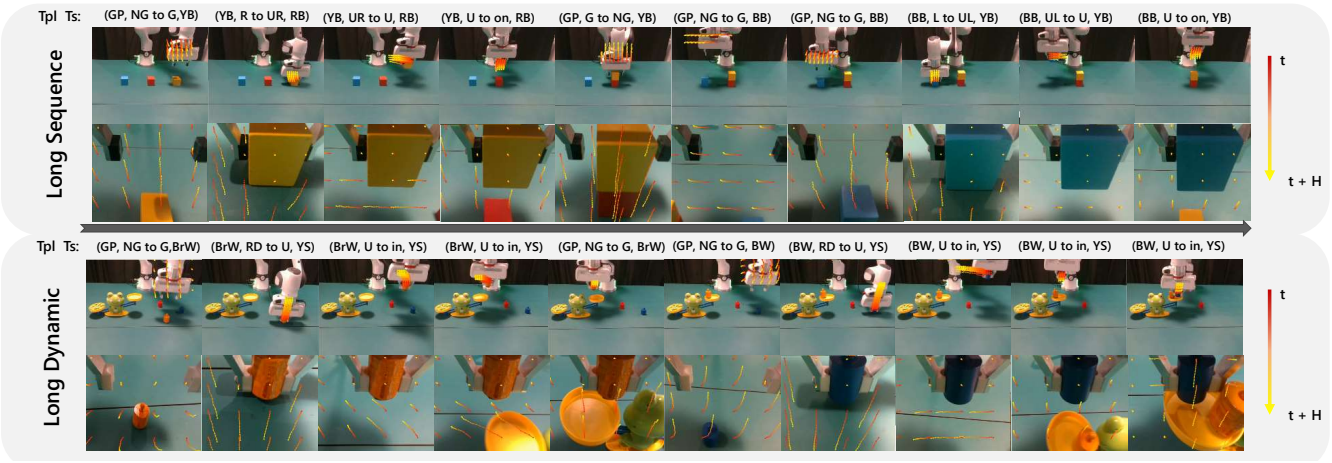

Fig. 5: Partial visualization. Top: triplet transitions from the Task Planner(GP/R/UR/RD/U/UL=gripper/right/upper-right/right-down/upper/upper-left; G/NG=grasp/no-grasp; Y/R/B/Br+B/W/S=yellow/red/blue/Brown+block/weight/scale). Bottom: two-camera point-track views.

TABLE III: Ablation Results: per-family success (%).

Method/Task	DPP	SS	DSB
w/o Track Predictor	45.5 ± 7.4	46.1 ± 3.4	43.8 ± 9.4
w/o Temporal Graph	57.1 ± 8.4	25.0 ± 0.0	0.0 ± 0.0
w/o Relation Predictor	75.0 ± 0.0	37.5 ± 5.5	14.3 ± 0.0
w/o Task Planner (ATM)	7.1 ± 2.0	0.0 ± 0.0	0.0 ± 0.0
Only Action Policy	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0

D. Design Justification and Ablation Analysis

We justify the design through real-world ablations that quantify the contribution of each component, including the two relation-generation branches, the temporal graph, task decomposition (Task Planner), trajectory generation (Track Predictor), a policy-only baseline, and the effect of action-labeled data scale on performance and generalization.

Effect of Task Planner Components. We ablate the relation-generation branches and temporal graph. As shown in Table II, combining both branches improves accuracy with acceptable latency. Removing the graph increases next-triplet errors by making the VLM more prone to hallucinations and dependency-tracking failures, while graph constraints reduce these errors by limiting transitions to feasible updates observed in human data. Table III shows that zero-shot VLMs can infer salient relations in some cases, such as DPP without the Track Predictor, but struggle with fine-grained distinctions required for task switching.

Effect of Hierarchical Architecture. Layer-wise ablations show that the hierarchy is essential. Without the Task Planner, long-horizon tasks collapse into short-horizon behavior and the system cannot maintain multi-step task progress. Removing the Track Predictor leaves a gap between discrete subgoals and continuous control, which substantially degrades performance. The policy-only baseline (BC) fails consistently because it lacks explicit task-level structure and therefore cannot reliably maintain global progress during execution, as shown in Table III.

Effect of Action-Labeled Data Size. Finally, we vary the amount of robot action labels used to train the policy and plot the corresponding performance curves. As shown in Fig. 6,

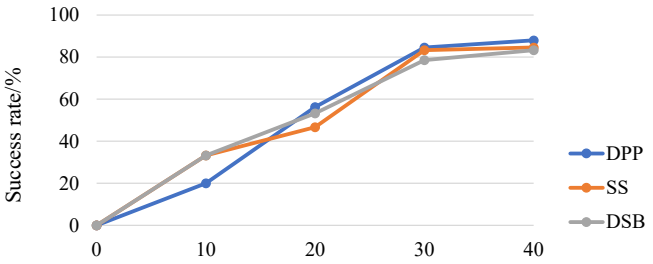

Fig. 6: TTS Success rate with 0–40 robot Demos.

TTS achieves reasonable performance with only a few dozen action-labeled demonstrations, supporting its data efficiency.

VI. CONCLUSIONS

We introduced TTS, a closed-loop system using discrete triplets for planning and continuous tracks for execution. Its structured, instance-spatially grounded design reduces hallucinations, improves long-horizon reliability under limited data, and supports object-level and compositional generalization. Limitations include out-of-graph failures, perception errors, and non-triplet tasks such as deformable-object manipulation. Future work will explore state relocalization and richer object representations.

REFERENCES

[1] Y. Liu, W. Chen, Y. Bai, X. Liang, G. Li, W. Gao, and L. Lin, "Aligning cyber space with physical world: A comprehensive survey on embodied ai," *IEEE/ASME Transactions on Mechatronics*, 2025.

[2] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 230–244, 2022.

[3] Z. Wang, B. Yu, J. Zhao, W. Sun, S. Hou, S. Liang, X. Hu, Y. Han, and Y. Gan, "Karma: Augmenting embodied ai agents with long-and-short term memory systems," *arXiv preprint arXiv:2409.14908*, 2024.

[4] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on robot learning*. PMLR, 2020, pp. 1094–1100.

[5] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[6] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[7] Y. Yang, J. Sun, S. Kou, Y. Wang, and Z. Deng, "Lohovla: A unified vision-language-action model for long-horizon embodied tasks," 2025.

[8] Y. Feng, J. Han, Z. Yang, X. Yue, S. Levine, and J. Luo, "Reflective planning: Vision-language models for multi-stage long-horizon robotic manipulation," *arXiv preprint arXiv:2502.16707*, 2025.

[9] T. Zhou, Z. Wang, H. Ao, G. Chen, B. Xing, J. Cheng, Y. Yang, and Y. Yue, "Step planner: Constructing cross-hierarchical subgoal tree as an embodied long-horizon task planner," *arXiv preprint arXiv:2506.21030*, 2025.

[10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[11] J. Zhang, Y. Guo, X. Chen, Y.-J. Wang, Y. Hu, C. Shi, and J. Chen, "Hirt: Enhancing robotic control with hierarchical robot transformers," *arXiv preprint arXiv:2410.05273*, 2024.

[12] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[13] B. Wang, J. Zhang, S. Dong, I. Fang, and C. Feng, "Vlm see, robot do: Human demo video to robot action plan via vision language model," *arXiv preprint arXiv:2410.08792*, 2024.

[14] V. Myers, B. C. Zheng, O. Mees, S. Levine, and K. Fang, "Policy adaptation via language optimization: Decomposing tasks for few-shot imitation," *arXiv preprint arXiv:2408.16228*, 2024.

[15] P. Li, H. Wu, Y. Huang, C. Cheang, L. Wang, and T. Kong, "Gr-mg: Leveraging partially-annotated data via multi-modal goal-conditioned policy," *IEEE Robotics and Automation Letters*, 2025.

[16] 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.

[17] 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.

[18] C. Gao, Z. Liu, Z. Chi, J. Huang, X. Fei, Y. Hou, Y. Zhang, Y. Lin, Z. Fang, Z. Jiang *et al.*, "Vla-os: Structuring and dissecting planning representations and paradigms in vision-language-action models," *arXiv preprint arXiv:2506.17561*, 2025.

[19] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma, "Vlatest: Testing and evaluating vision-language-action models for robotic manipulation," *Proceedings of the ACM on Software Engineering*, vol. 2, no. FSE, pp. 1615–1638, 2025.

[20] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[21] Y. Wu, J. Zhang, N. Hu, L. Tang, G. Qi, J. Shao, J. Ren, and W. Song, "Mldt: Multi-level decomposition for complex long-horizon robotic task planning with open-source large language model," in *International Conference on Database Systems for Advanced Applications*. Springer, 2024, pp. 251–267.

[22] C.-L. Cheang, G. Chen, Y. Jing, T. Kong, H. Li, Y. Li, Y. Liu, H. Wu, J. Xu, Y. Yang *et al.*, "Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation," *arXiv preprint arXiv:2410.06158*, 2024.

[23] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y.-G. Jiang *et al.*, "Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks," *arXiv preprint arXiv:2412.18194*, 2024.

[24] Y. Fan, P. Ding, S. Bai, X. Tong, Y. Zhu, H. Lu, F. Dai, W. Zhao, Y. Liu, S. Huang *et al.*, "Long-vla: Unleashing long-horizon capability of vision language action model for robot manipulation," *arXiv preprint arXiv:2508.19958*, 2025.

[25] J. Sun, A. Curtis, Y. You, Y. Xu, M. Koehle, L. Guibas, S. Chitta, M. Schwager, and H. Li, "Hierarchical hybrid learning for long-horizon contact-rich robotic assembly," *arXiv preprint arXiv:2409.16451*, 2024.

[26] P. Sundaresan, H. Hu, Q. Vuong, J. Bohg, and D. Sadigh, "What's the move? hybrid imitation learning via salient points," *arXiv preprint arXiv:2412.05426*, 2024.

[27] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2081–2088.

[28] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2robot: Learning manipulation concepts from instructions and human demonstrations," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1419–1434, 2021.

[29] K. Shaw, S. Bahl, and D. Pathak, "Videodex: Learning dexterity from internet videos," in *Conference on Robot Learning*. PMLR, 2023, pp. 654–665.

[30] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 639–24 654, 2022.

[31] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[32] 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.

[33] 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," *arXiv preprint arXiv:2210.00030*, 2022.

[34] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 1134–1141.

[35] 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.

[36] H. Bharadhwaj, A. Gupta, S. Tulsiani, and V. Kumar, "Zero-shot robot manipulation from passive human videos," *arXiv preprint arXiv:2302.02011*, 2023.

[37] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani, "Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation," *CoRR*, 2024.

[38] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song, "Flow as the cross-domain manipulation interface," *arXiv preprint arXiv:2407.15208*, 2024.

[39] C. Wen, D. Jayaraman, and Y. Gao, "Can transformers capture spatial relations between objects?" *arXiv preprint arXiv:2403.00729*, 2024.

[40] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia, "Spatialvlm: Endowing vision-language models with spatial reasoning capabilities," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14 455–14 465.

[41] A.-C. Cheng, H. Yin, Y. Fu, Q. Guo, R. Yang, J. Kautz, X. Wang, and S. Liu, "Spatialrgpt: Grounded spatial reasoning in vision-language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 135 062–135 093, 2024.

[42] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.

[43] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[44] Z. Chen, W. Wang, Y. Cao, Y. Liu, Z. Gao, E. Cui, J. Zhu, S. Ye, H. Tian, Z. Liu *et al.*, "Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling," *arXiv preprint arXiv:2412.05271*, 2024.

[45] R. Islam and O. M. Moushi, "Gpt-4o: The cutting-edge advancement in multimodal llm," in *Intelligent Computing-Proceedings of the Computing Conference*. Springer, 2025, pp. 47–60.

[46] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht, "Cotracker3: Simpler and better point tracking by pseudo-labelling real videos," *arXiv preprint arXiv:2410.11831*, 2024.

[47] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai *et al.*, "$\pi_{0.5}$: a vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.