

DYNAMIT: Dynamic Zero-Shot Manipulation via Instruction-Grounded Visual Tracking and Interception in Industrial Conveyor Environments

Singh, Harsh; Suzuki, Kei; Wang, Ye; Liu, Jing; Cascante-Bonilla, Paola; Koike-Akino, Toshiaki

TR2026-146 October 03, 2026

Abstract

Manipulation on industrial conveyors is inherently time-sensitive. Conveyor motion can shift the required grasp or placement location before the robot’s end effector reaches it. Existing approaches may require task-specific demonstrations, policy training, instance-specific object models, or configured conveyor motion, which can limit transfer to new objects and motion conditions. We introduce DYNAMIT, an instruction-conditioned, zero-shot framework for conveyor pick-and-place that operates without these requirements. The language instruction specifies the objects and receiving receptacles, while memory-based visual tracking preserves their identities in RGB-D observations. Depth measurements and track histories provide estimates of each instance’s position, dimensions, and velocity. A reachability-aware planner uses these estimates and robot proprioception to plan when and where each grasp or placement should occur, accounting for the end effector’s travel time. The planner updates this timing and location during execution, allowing the same pipeline to grasp moving objects and place objects into moving receptacles. We evaluate two conveyor tasks in simulation at different conveyor speeds and test transfer to industrial objects. DYNAMIT achieves the highest aggregate success rate and Q-score among the evaluated methods in each conveyor study. Evaluation on the Dynamic Object Manipulation (DOM) benchmark further demonstrates transfer to new scenes and task conditions without DOM demonstrations or policy training.

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop 2026

DYNAMIT: Dynamic Zero-Shot Manipulation via Instruction-Grounded Visual Tracking and Interception in Industrial Conveyor Environments

Harsh Singh^{1,2}, Kei Suzuki², Ye Wang², Jing Liu², Paola Cascante-Bonilla¹, and Toshiaki Koike-Akino²

Abstract: Manipulation on industrial conveyors is inherently time-sensitive. Conveyor motion can shift the required grasp or placement location before the robot’s end effector reaches it. Existing approaches may require task-specific demonstrations, policy training, instance-specific object models, or configured conveyor motion, which can limit transfer to new objects and motion conditions. We introduce DYNAMIT, an instruction-conditioned, zero-shot framework for conveyor pick-and-place that operates without these requirements. The language instruction specifies the objects and receiving receptacles, while memory-based visual tracking preserves their identities in RGB-D observations. Depth measurements and track histories provide estimates of each instance’s position, dimensions, and velocity. A reachability-aware planner uses these estimates and robot proprioception to plan when and where each grasp or placement should occur, accounting for the end effector’s travel time. The planner updates this timing and location during execution, allowing the same pipeline to grasp moving objects and place objects into moving receptacles. We evaluate two conveyor tasks in simulation at different conveyor speeds and test transfer to industrial objects. DYNAMIT achieves the highest aggregate success rate and Q-score among the evaluated methods in each conveyor study. Evaluation on the Dynamic Object Manipulation (DOM) benchmark further demonstrates transfer to new scenes and task conditions without DOM demonstrations or policy training.

I. INTRODUCTION

Dynamic manipulation requires robots to coordinate their actions with objects that keep moving. On an industrial conveyor, every perception result has a deadline. An object selected for grasping can move away from its observed position before the end effector reaches it. During placement, the receiving receptacle may leave the workspace before the object is ready for release. While the robot completes one transfer, other objects on the conveyor continue to move. Successful execution therefore requires the robot to identify the objects named in the instruction and keep track of them across successive transfers. It must estimate their geometry and motion, then account for the end effector’s travel time when planning contact. In industrial settings, object appearance, geometry, layout, and conveyor speed can change between production runs. We seek a system that handles these changes from language and visual observations, without collecting new robot demonstrations for each condition.

¹Department of Computer Science, Stony Brook University, Stony Brook, NY 11794-2424, USA. {singh17, paola}@cs.stonybrook.edu.

²Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139-1955, USA. {kesuzuki, jiliu, yewang, koike}@merl.com.

H. Singh conducted this work during an internship at MERL.

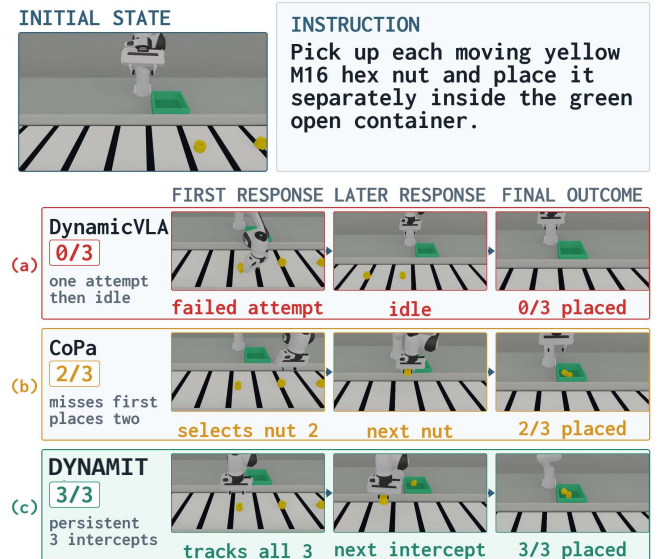


Fig. 1: Given the instruction to place three moving yellow M16 hex nuts separately inside the green open container, the top panel shows the initial observation and task instruction, while each row presents the first response, later response, and final outcome. (a) DynamicVLA makes one failed grasp attempt and then remains idle, placing no objects. (b) CoPa misses the first nut but places the next two. (c) DYNAMIT maintains all three object tracks and updates its interception plan after each placement, successfully completing the task.

Vision-language-action (VLA) models connect language and visual observations to robot actions through pretraining on large robot datasets [1], [2], [3]. More recent dynamic policies incorporate observation history, continuous inference, action streaming, or future-state prediction to react to moving objects [4], [5]. These systems learn motion and timing from their training data. Collecting additional dynamic trajectories can be costly when industrial objects, camera layouts, or conveyor conditions change. This motivates an approach that combines pretrained visual models with motion estimates obtained during execution.

Modular foundation-model systems offer another way to connect language and robot control. They express instructions as executable programs, spatial value maps, or geometric constraints [6], [7], [8], [9]. These representations support new instructions without demonstrations for every task. Conveyor manipulation also requires the robot to maintain the link between the instruction and the same physical

instances as they move. It must update when and where to grasp or place, accounting for both the observed motion and the end effector’s travel time.

Specialized dynamic grasping combines motion prediction, reachability, and online replanning, but assumes known object models and object-specific grasp sets [10], [11]. DGBench provides a physical testbed for comparing camera configurations during dynamic cube grasping [12]. Language-driven closed-loop grasping also supports zero-shot execution, but its demonstrated setup uses CAD-based pose tracking and a predefined placement location [13], [14]. We target transfer across object geometries and conveyor speeds without instance-specific object models, grasp sets, or configured conveyor motion, while preserving task roles and instance identities across successive grasps and placements.

We address this problem with **DYNAMIT**, an instruction-conditioned visual pipeline for zero-shot conveyor pick-and-place. Language specifies the objects, receiving receptacles, and moving task role. Visual grounding locates these instances, and memory-based tracking preserves their identities. Depth measurements and track histories estimate position, dimensions, and velocity. The planner combines these estimates with end-effector travel time to choose reachable grasp or placement points. Tracking continues during each transfer to support subsequent instance selection. Figure 2 shows how these stages work together.

DYNAMIT uses only a natural-language instruction, calibrated RGB-D observations, and robot proprioception during execution. It estimates object geometry and conveyor motion online. We evaluate both tasks at different conveyor speeds and test transfer from cubes to three industrial object types. Every method starts from the same initial conditions for each episode. DYNAMIT achieves the highest aggregate success rate and Q-score among the evaluated methods in each conveyor study. A complementary evaluation on eight axes of the DOM benchmark [4] tests transfer to tabletop manipulation with appearance, spatial, and motion cues, without DOM demonstrations or policy training (Section IV-E). Together, these studies examine zero-shot generalization across object geometries, conveyor speeds, and instruction-defined manipulation conditions.

This work makes three contributions:

- A zero-shot industrial conveyor pipeline using language, calibrated RGB-D, and robot proprioception, without task-specific demonstrations, instance-specific object models, configured conveyor parameters, or privileged simulator state.
- An instruction-grounded representation combining persistent tracking, metric 3-D motion estimation, and reachability-aware predictive interception for moving objects and receptacles.
- A controlled benchmark with the same initial conditions across methods, variations in speed and industrial objects, and metrics for full and partial task completion, together with a transfer study on eight DOM axes.

II. RELATED WORK

A. Language-Grounded Zero-Shot Manipulation

Foundation models allow robots to turn instructions into programs and geometric goals. Code as Policies (CaP) generates programs that combine perception functions with robot primitives [6]. VoxPoser builds 3-D value maps from language instructions and optimizes trajectories over these maps [7]. ReKep expresses tasks as constraints on visual keypoints and optimizes subgoals and paths to satisfy them [8]. CoPa uses object parts to select grasps and define constraints on the end-effector pose after grasping [9]. MALMM assigns planning, code generation, and supervision to separate language-model agents. It uses new observations after each step to support recovery [15]. These methods connect instruction meaning to robot actions. For conveyor execution, DYNAMIT also uses visual motion estimates and robot feedback to determine when each grasp or placement can occur.

B. Learned Visuomotor Policies for Dynamic Manipulation

Dynamic manipulation must account for changes that occur during perception, inference, and execution. Motion prediction and online adaptation are therefore central to dynamic and nonprehensile manipulation [16]. Learned methods use different sources of experience to meet these demands. LAP replans from human demonstration videos using a latent Bayesian model, while ICPI updates its policy from previous interactions [17], [18].

Recent dynamic visuomotor policies focus more directly on motion history and execution latency. DynamicVLA overlaps inference with execution and streams temporally aligned actions [4]. PUMA combines optical-flow history with world queries to predict short-horizon scene state [5]. Their temporal behavior depends on demonstrations or predictive-model training. DYNAMIT reuses pretrained perception while estimating metric motion and interception times online, without task-specific robot demonstrations or policy training.

C. Visual Tracking and Predictive Interception

Open-vocabulary perception can initialize tracks without instance-specific training. Grounding DINO converts language queries into spatial prompts, while SAM 2 segments prompted regions and propagates their masks across video frames. HiM2SAM strengthens this propagation with hierarchical motion estimation and optimized short- and long-term memory [19], [20], [21]. DYNAMIT uses these models to track the instances named in an instruction. It combines their RGB-D histories with robot feedback to plan the timing of grasps and placements.

The dynamic grasping framework in [10] estimates object motion online and supports six-degree-of-freedom grasps, but relies on known object models, precomputed grasp databases, and a grasp-quality network trained for each object. Its prediction horizons are configured experimentally. DGBench compares camera configurations using color-based visual servoing on cubes [12]; its experiments do not evaluate

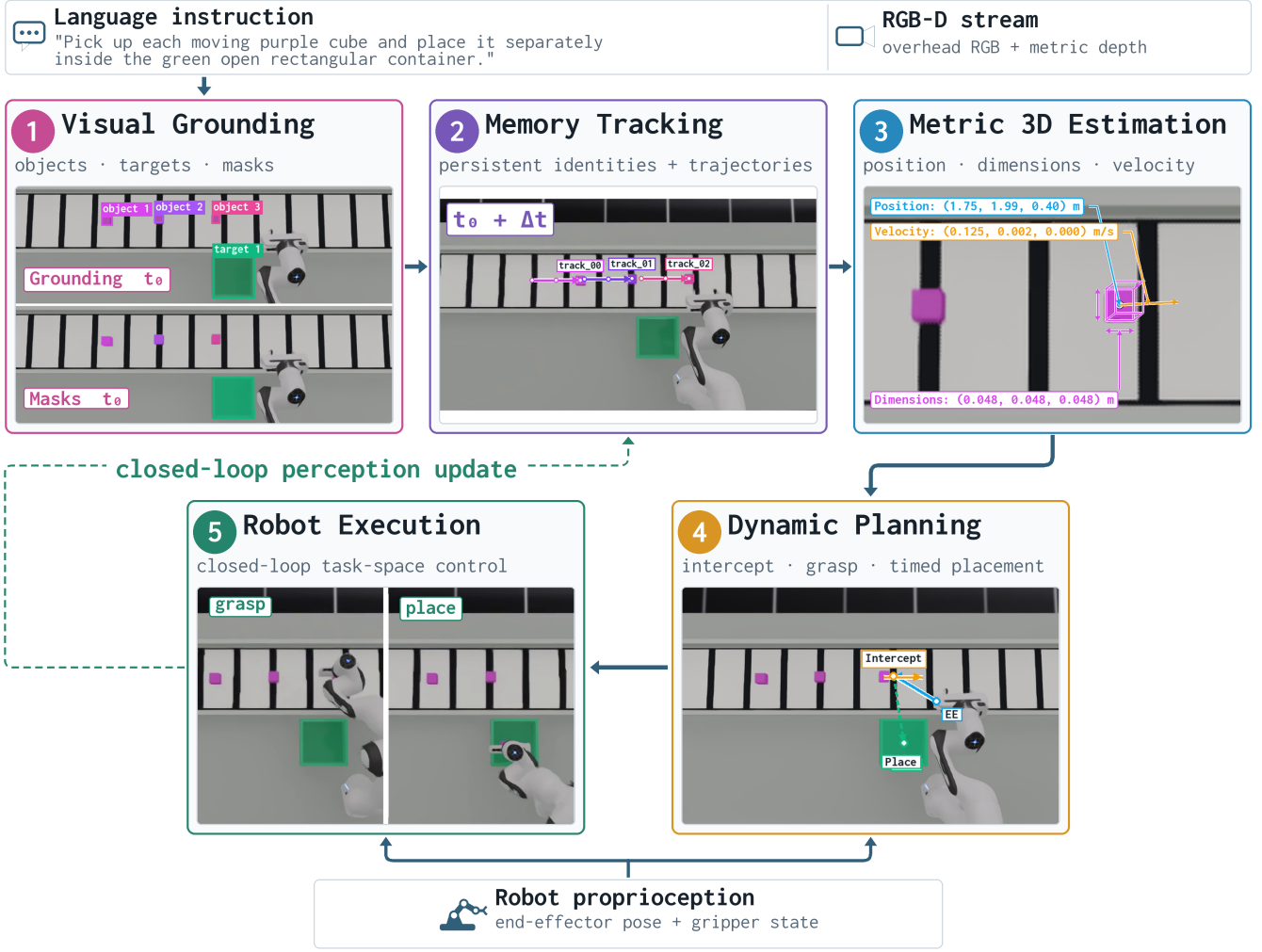


Fig. 2: DYNAMIT pipeline. Given a language instruction and a calibrated RGB-D stream, (1) visual grounding identifies task-relevant object and receptacle instances and initializes their masks; (2) memory tracking maintains their masks and identities over time; (3) depth within each mask provides position and dimension estimates, while track histories provide velocity estimates; (4) dynamic planning combines observed motion with end-effector arrival time to choose reachable grasp or placement points; and (5) robot execution follows these goals using task-space control. Robot proprioception supplies the end-effector pose and gripper state to (4) and (5), while new visual observations update (2) during execution.

language-guided selection or moving-receptacle placement. Constant-time replanning [11] guarantees bounded planning time, but assumes known conveyor velocity, object geometry, and supplied feasible grasps. Language-driven grasping [13] reuses pretrained perception, but its adopted Foundation-Pose [14] interface uses a supplied CAD model and its placement location is predefined. Complementary work addresses semantic reconstruction of dynamic scenes [22]. DYNAMIT estimates geometry and motion from RGB-D, uses end-effector travel time to update interception, and schedules grasps and placements from language. Its zero-shot scope is fixed top-down pick-and-place without instance-specific object models, precomputed grasps, or policy training.

III. METHOD

DYNAMIT turns an instruction and RGB-D observations

into grasp and placement commands. Figure 2 shows the five stages. Visual Grounding (1) finds the named instances and associates them with their task roles. Memory Tracking (2) maintains their masks and identities. Metric 3-D Estimation (3) combines these masks with depth and track histories to estimate position, dimensions, and velocity. Dynamic Planning (4) combines these estimates with end-effector travel time to choose an interaction point. Robot Execution (5) moves the end effector toward the updated grasp or placement position.

A. Problem Formulation and Online Observations

We consider instruction-conditioned pick-and-place beside a conveyor using a robot with a parallel-jaw gripper. The instruction ℓ is parsed once at the start of the task:

$$\Psi(\ell) = (q^{\text{obj}}, q^{\text{rec}}, \mu, \rho), \quad (1)$$

where q^{obj} describes the item to pick up and q^{rec} describes the receiving receptacle (target). These are task roles assigned by the instruction. The variable $\mu \in \{\text{obj}, \text{rec}\}$ states which role moves, while $\rho \in \{\text{many-to-one}, \text{one-to-one}\}$ states whether the objects share one receptacle or require separate receptacles. In Task 1, the objects move and share one receptacle. In Task 2, the receptacles move and each object has a separate target. Instance counts are obtained from the images unless the instruction states them explicitly. This parsed description supplies the roles used by the visual inventory, tracker, and planner.

At time t , DYNAMIT receives observation $\mathbf{o}_t$ and produces command $\mathbf{a}_t$:

$$\begin{aligned}\mathbf{o}_t &= (I_t, D_t, \mathbf{K}, \mathbf{T}_C^W, \mathbf{s}_t^{\text{rob}}), \\ \mathbf{a}_t &= (\mathbf{p}_t^*, g_t^*).\end{aligned}\quad (2)$$

The RGB image I_t and aligned depth map D_t provide the visual observation. Camera calibration consists of the intrinsics $\mathbf{K}$ and the transform $\mathbf{T}_C^W$ from the camera to the robot workspace. Robot feedback $\mathbf{s}_t^{\text{rob}}$ contains joint positions, end-effector position, and gripper opening. The command specifies a desired end-effector position $\mathbf{p}_t^*$ and a close/open gripper command $g_t^* \in \{-1, +1\}$. A fixed top-down wrist orientation is used during task-space control. No privileged simulator state is available online. Here, zero-shot denotes execution without task-specific demonstrations or policy training within the specified pick-and-place tasks.

B. Instruction-Grounded Visual Inventory

Visual Grounding (1) searches for the object and receptacle descriptions separately. An open-vocabulary detector proposes matching regions in I_t , and a promptable segmenter turns each proposal into a mask. Attributes from the instruction, such as color, help reject mismatches and separate nearby instances.

RGB and depth checks remove duplicate regions and masks that are inconsistent with the observed support surface. The remaining masks form the visual inventory. For instance $i \in \{1, \dots, N_t\}$, $M_{i,t}$ is the binary mask and $r_i \in \{\text{obj}, \text{rec}\}$ is its task role. Tracking begins once the inventory is stable and instances assigned to the moving role show consistent displacement.

C. Memory-Based Metric Scene Tracking

Memory Tracking (2) propagates masks through the RGB stream while preserving instance identities. When grounding is repeated, new masks are matched to existing tracks using their 3-D position and dimensions. This corrects drift and adds instances that have newly entered view.

Metric 3-D Estimation (3) converts masks into workspace measurements using depth and camera calibration:

$$(\mathbf{p}_{i,t}, \mathbf{d}_{i,t}) = \text{MetricBox}(M_{i,t}, D_t, \mathbf{K}, \mathbf{T}_C^W), \quad (3)$$

where MetricBox back-projects valid masked depth pixels into the workspace. It trims the lowest and highest 5% of coordinates along each axis to limit isolated depth errors. It then returns the box center $\mathbf{p}_{i,t}$ and its three axis-aligned side

lengths $\mathbf{d}_{i,t}$, shown as *Position* and *Dimensions* in Metric 3-D Estimation (3).

We estimate velocity by fitting a straight line through the last H timestamped positions:

$$\hat{\mathbf{v}}_{i,t} = \text{LinearFit}(\{(t_k, \mathbf{p}_{i,k})\}_{k=1}^H). \quad (4)$$

The velocity estimate is smoothed and constrained to the observed support plane. We infer the motion direction from agreement across moving tracks and the observed lane direction. We project each moving track's velocity onto this direction and take the median as the conveyor speed. During short tracking gaps, positions are predicted from the latest position and velocity estimates. Static tracks retain their last stable position. These estimates form the planner state

$$\mathcal{X}_t = \{(i, r_i, \mathbf{p}_{i,t}, \mathbf{d}_{i,t}, \hat{\mathbf{v}}_{i,t})\}_{i=1}^{N_t}, \quad (5)$$

where N_t is the number of active tracks. Each track contains an identity, task role, position, dimensions, and velocity.

D. Reachability-Aware Predictive Interception

Dynamic Planning (4) predicts where the selected instance will be when the end effector reaches it. Assuming approximately constant velocity over the prediction horizon, the position of moving track i after τ seconds is

$$\hat{\mathbf{p}}_i(\tau) = \mathbf{p}_{i,t} + \hat{\mathbf{v}}_{i,t}\tau. \quad (6)$$

Here $\mathbf{p}_{i,t}$ and $\hat{\mathbf{v}}_{i,t}$ are the estimated position and velocity. To estimate the end-effector arrival time τ_i^{arr} , the planner first computes the travel time from the current end-effector position to a safe point above the tracked instance. It then predicts the instance's position at that time and recomputes the travel time. These updates account for observed motion, nominal end-effector speed, and a short preparation delay.

The predicted interaction must also fall within the robot's reach. Let $\mathbf{p}^b$ be the calibrated robot-base position and R its conservative horizontal reach. We write

$$\begin{aligned}[\tau_i^{\text{in}}, \tau_i^{\text{out}}] &= \text{ReachWindow}(\mathbf{p}_{i,t}, \hat{\mathbf{v}}_{i,t}, \mathbf{p}^b, R), \\ \tau_i^* &= \max(\tau_i^{\text{arr}}, \tau_i^{\text{in}}), \quad \tau_i^* \leq \tau_i^{\text{out}}, \\ \mathbf{p}_i^{\text{int}} &= \hat{\mathbf{p}}_i(\tau_i^*).\end{aligned}\quad (7)$$

The ReachWindow operation returns the interval during which the predicted horizontal distance from the robot base is at most R . The planner selects the earliest time τ_i^* when both the end effector and the instance can reach the interaction point. This time must fall within the reachability window, and the predicted position is $\mathbf{p}_i^{\text{int}}$. The horizontal window approximates reachability for the fixed top-down interaction.

Instances already within reach are prioritized by how soon they will leave. Those not yet within reach are ordered by their earliest feasible interaction time. If the selected instance leaves the reachable region, the planner selects the next candidate. The moving role μ determines whether this schedule is applied to objects for grasping or receptacles for placement. Updated visual tracks and robot feedback update the plan, so its timing follows the observed motion.

E. Task-Conditioned Manipulation

Robot Execution (5) turns the selected track and interception point into approach, grasp, transfer, and release phases. A task-space controller moves the end effector toward the commanded position. Visual tracks and robot feedback guide phase transitions and updates to the grasp or placement position.

a) Moving objects to a static receptacle: The planner chooses the object with the most urgent feasible interception window. During approach and grasping, the end effector follows the grasp position, which is updated using the latest object velocity estimate. After the grasp, the object is transferred to the static receptacle. The observed object and receptacle dimensions are used to select a separate placement location for each object.

b) Static objects to moving receptacles: The controller grasps the nearest unassigned static object. It then selects an unused moving receptacle and predicts a feasible placement position. This position is updated from the receptacle track until release. The receptacle is then removed from the assignment set, enforcing the one-to-one relation.

For both tasks, gripper opening and tracked object motion are used to verify the grasp. A failed lift triggers a bounded re-grasp attempt. Thus, both tasks use the same tracked state and interception planner; only the moving role and placement relation change.

IV. EXPERIMENTS

We evaluate two conveyor tasks across different speeds and test industrial objects in the moving-object task. We then examine transfer to DOM. All experiments are conducted in simulation. The conveyor studies compare DYNAMIT with five baselines on 25 episodes per condition. Every method starts from the same initial scene configuration for each episode. Figure 3 shows the two tasks. The DOM study compares DYNAMIT with the released DOM-trained DynamicVLA checkpoint on eight evaluation axes. It uses 30 episodes per axis, with the same initial conditions for both methods. We report task completion for each integrated system under the stated evaluation protocol.

A. Conveyor Setup and Metrics

a) Environment and tasks: Experiments use Isaac Sim 4.5 and Isaac Lab with a Franka Panda manipulator and a parallel-jaw gripper. We study two complementary tasks. In *Task 1: moving objects to a static receptacle*, three cubes travel on the conveyor and must all be placed in one static open receptacle. In *Task 2: static objects to moving receptacles*, three static cubes must be placed one-to-one into three distinct receptacles moving on the conveyor. Task 1 tests grasp interception, while Task 2 tests placement interception and receptacle assignment.

b) Evaluation conditions: For the speed study, we evaluate both tasks at conveyor speeds of 6.25, 12.5, and 18.75 cm/s. The object study replaces the cubes in Task 1 with one of three industrial object types: a medium gear, an M16 hex nut, or an assembly socket. Each episode contains

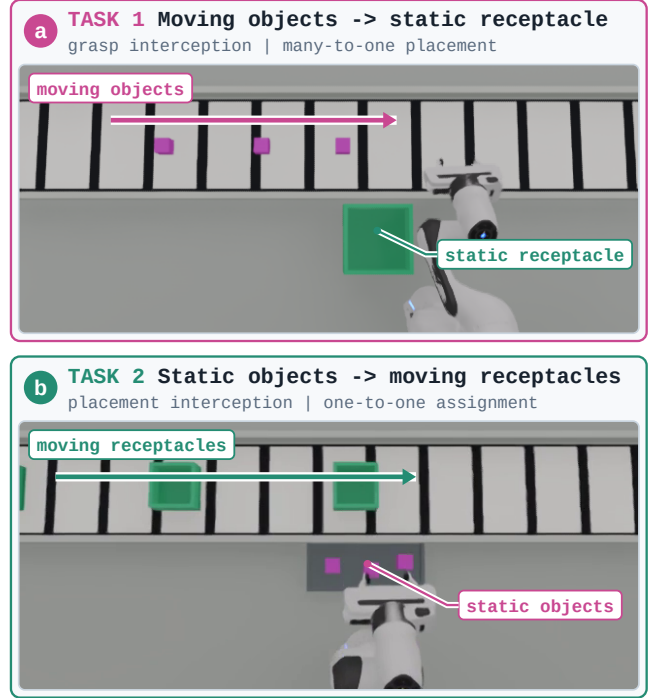


Fig. 3: Experimental tasks. (a) Task 1 requires placing three moving objects into one static receptacle, testing grasp interception and many-to-one placement. (b) Task 2 requires placing three static objects into distinct moving receptacles, testing placement interception and one-to-one assignment. Arrows indicate the moving task role.

three instances of the selected object type, and the conveyor runs at 12.5 cm/s. The initial x and y coordinates of each manipulated object are randomly perturbed within $[-2, 2]$ cm.

c) Online observation protocol: DYNAMIT, VoxPoser, CaP, ReKep, and CoPa receive the same task instruction, a 640×360 overhead RGB-D stream, camera calibration, and robot proprioception. In industrial-object conditions, only the object description in the instruction changes. DYNAMIT uses pretrained Grounding DINO [19] to detect objects and receptacles from language descriptions, SAM 2 [20] to generate their initial segmentation masks, and HiM2SAM [21] to propagate these masks across frames while preserving instance identities. All three models are used without task-specific fine-tuning. For VoxPoser, CaP, ReKep, and CoPa, simulation is paused during external language-model calls. This prevents variable API latency from changing the scene before a plan is returned. DynamicVLA retains the visual and action interface expected by its released checkpoint, including wrist and opposite-view RGB images, end-effector proprioception, and 25-Hz action queries. It receives an equivalent task command in the DOM-style prompt format used during training. No method receives privileged simulator state during execution. Simulator state is used only for offline evaluation, and no method is trained or fine-tuned on our benchmark episodes.

TABLE I: Task 1 (moving-object interception) performance across conveyor speeds. Success rate and Q-score are reported on a $[0, 1]$ scale. All methods use the same 25 initial conditions per speed. Best and second-best nonzero results are highlighted in green and blue, respectively.

Method	6.25 cm/s		12.5 cm/s		18.75 cm/s		Overall	
	Success	Q-score	Success	Q-score	Success	Q-score	Success	Q-score
DYNAMIT	1.000	1.000	0.920	0.973	0.360	0.773	0.760	0.916
VoxPoser	0.000	0.133	0.000	0.160	0.000	0.200	0.000	0.164
CaP	0.000	0.000	0.000	0.160	0.000	0.000	0.000	0.053
ReKep	0.000	0.053	0.000	0.013	0.000	0.000	0.000	0.022
CoPa	0.800	0.920	0.000	0.347	0.000	0.053	0.267	0.440
DynamicVLA	0.000	0.040	0.000	0.147	0.000	0.013	0.000	0.067

TABLE II: Task 2 (moving-receptacle placement) performance across conveyor speeds. All methods use the same 25 initial conditions per speed. Metrics and highlighting follow Table I.

Method	6.25 cm/s		12.5 cm/s		18.75 cm/s		Overall	
	Success	Q-score	Success	Q-score	Success	Q-score	Success	Q-score
DYNAMIT	0.520	0.787	0.920	0.973	0.560	0.747	0.667	0.836
VoxPoser	0.000	0.547	0.000	0.027	0.000	0.000	0.000	0.191
CaP	0.000	0.653	0.280	0.640	0.000	0.000	0.093	0.431
ReKep	0.000	0.067	0.000	0.000	0.000	0.053	0.000	0.040
CoPa	0.000	0.627	0.000	0.053	0.000	0.320	0.000	0.333
DynamicVLA	0.000	0.147	0.000	0.013	0.000	0.000	0.000	0.053

d) Conveyor metrics: We report full-task *success rate* and partial-completion *Q-score*, both on a $[0, 1]$ scale. Following the goal-completion principle used in the BEHAVIOR Challenge [23], let $m_e \in \{0, 1, 2, 3\}$ denote the number of correct placements in episode e . For Task 1, this is the number of objects inside the static receptacle. For Task 2, it is computed using the best one-to-one object–receptacle assignment, so each receptacle contributes at most one placement.

For N episodes, Q-score is

$$Q = \frac{1}{3N} \sum_{e=1}^N m_e, \quad (8)$$

and the full-task success rate is

$$S = \frac{1}{N} \sum_{e=1}^N \mathbf{1}[m_e = 3], \quad (9)$$

where $\mathbf{1}[\cdot]$ is the indicator function. Q-score credits partial completion, while success rate requires all three placements.

B. Baselines

We compare DYNAMIT with four language-model-based manipulation systems and one trained dynamic policy. **VoxPoser** [7] uses its released language-model programs and value-map planner with OWL-ViT, SAM, and XMem for visual grounding and tracking. **Code as Policies (CaP)** [6] uses its released tabletop code-generation prompts, MDETR grounding, and executable pick-and-place primitives. **ReKep** [8] uses its released constraint-generation prompt, DINOv2 keypoint proposals and tracking, and subgoal optimization.

CoPa [9] combines set-of-mark grasp selection, the paper-described segmentation stack, and spatial-constraint optimization. **DynamicVLA** [4] uses its public DOM checkpoint and native action protocol without benchmark fine-tuning. The conveyor study evaluates how DynamicVLA transfers from DOM to our conveyor tasks. Section IV-E compares both systems within DOM.

The public releases lack some perception and robot interfaces required by our conveyor setting. For the four language-model-based baselines, we retain the released prompts and planning components, reconstruct paper-described visual interfaces when needed, and connect their outputs to a common Franka execution interface. ReKep uses its released default `chatgpt-4o-latest`, while VoxPoser uses its released default `gpt-4`. CoPa uses `gpt-4o` because its original `gpt-4-vision-preview` endpoint is unavailable, while CaP uses `gpt-4.1` because the legacy Completion and Edit endpoints used by its released implementation are no longer available. These compatibility changes retain the released prompts and planning logic. Results are reported for these reproduced implementations under the stated interfaces and timing protocol. No baseline receives DYNAMIT’s memory tracker or predictive interception planner. No method receives configured conveyor motion parameters.

C. Performance across Conveyor Speeds

a) Moving-object interception: Table I reports Task 1. DYNAMIT completes 57 of 75 episodes, achieving a success rate of 0.760 and a Q-score of 0.916. The strongest baseline, CoPa, reaches 0.267 success and a Q-score of 0.440, with all successful episodes at the lowest speed. DYNAMIT achieves

TABLE III: Task 1 (moving-object interception) performance on industrial objects at 12.5 cm/s. All methods use the same 25 initial conditions per object type. Metrics and highlighting follow Table I.

Method	Medium Gear		M16 Nut		Assembly Socket		Overall	
	Success	Q-score	Success	Q-score	Success	Q-score	Success	Q-score
DYNAMIT	0.320	0.667	0.440	0.613	0.360	0.787	0.373	0.689
VoxPoser	0.000	0.253	0.000	0.280	0.000	0.000	0.000	0.178
CaP	0.000	0.013	0.000	0.160	0.000	0.000	0.000	0.058
ReKep	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CoPa	0.000	0.000	0.000	0.480	0.000	0.027	0.000	0.169
DynamicVLA	0.000	0.067	0.000	0.040	0.000	0.080	0.000	0.062

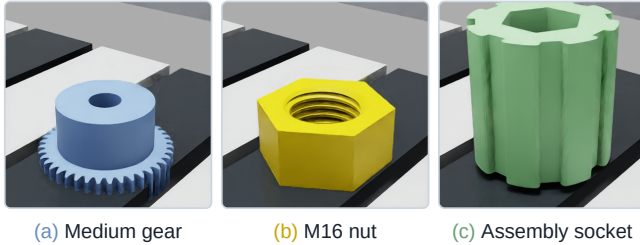


Fig. 4: Industrial objects evaluated in Task 1: (a) medium gear, (b) M16 hex nut, and (c) assembly socket.

1.000 success at 6.25 cm/s and retains 0.920 at 12.5 cm/s. At 18.75 cm/s, success is 0.360 and Q-score is 0.773, showing that the robot still completes most individual placements under faster motion.

b) Moving-receptacle placement: Task 2 applies interception to the moving receptacle and requires a distinct receptacle for each object. Table II shows that DYNAMIT achieves 0.667 success and a Q-score of 0.836 overall, compared with 0.093 success and a Q-score of 0.431 for CaP, the strongest baseline. DYNAMIT completes full tasks at all three speeds, with success rates of 0.520, 0.920, and 0.560. Its highest Q-score, 0.973, occurs at 12.5 cm/s. Together, the two tasks demonstrate zero-shot generalization across the tested conveyor speeds and moving roles: the shared tracking and interception pipeline estimates motion online and maintains object-to-receptacle assignments without policy training or configured conveyor motion.

D. Transfer to Industrial Objects

Figure 4 shows the three industrial objects used to evaluate transfer beyond cubes. As reported in Table III, DYNAMIT completes full tasks for all three object types and is the only evaluated method with nonzero full-task success. It achieves 0.373 overall success and a Q-score of 0.689, compared with the strongest baseline Q-score of 0.178. The M16 hex nut yields the highest success rate, while the assembly socket yields the highest Q-score. These results demonstrate zero-shot transfer from cubes to three industrial object geometries using updated language descriptions and the same top-down grasping strategy, without instance-specific object models or policy training.

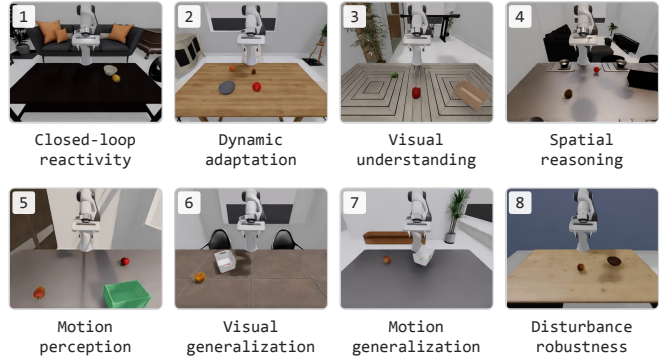


Fig. 5: Example scenes for eight DOM axes: (1) varied object speeds; (2) abrupt motion changes; (3) appearance cues; (4) relative spatial descriptions; (5) speed or direction cues; (6) held-out appearances and scenes; (7) held-out speeds, friction, and trajectories; and (8) external perturbations. Held-out conditions follow the DOM splits.

E. Transfer to the DOM Benchmark

We evaluate zero-shot transfer to eight axes of the Dynamic Object Manipulation (DOM) benchmark [4], using the released task definitions and instructions (Fig. 5). DYNAMIT retains instruction-grounded perception, RGB-D tracking, metric motion estimation, and predictive interception. Camera input, workspace calibration, and the robot execution interface are adapted without DOM demonstrations or policy training. DynamicVLA uses its released DOM-trained checkpoint. Both methods use the same 30 initial conditions per axis, giving 240 episodes per method.

Table IV shows that DYNAMIT achieves 0.700 success on motion perception, compared with 0.600 for DynamicVLA, and 0.467 on spatial reasoning, compared with 0.500. These results demonstrate zero-shot transfer to motion-based instance selection and spatially specified manipulation beyond conveyor scenes. Overall success is 0.279 for DYNAMIT and 0.563 for the DOM-trained DynamicVLA checkpoint. Lower success on visual generalization (0.200), motion generalization (0.233), and disturbance robustness (0.100) shows that transfer remains limited under these shifts.

F. Qualitative Analysis

Figure 1 illustrates successive transfers with persistent tracks. DYNAMIT updates the next interception point after

TABLE IV: Success rates on eight DOM axes. DynamicVLA uses its DOM-trained checkpoint; DYNAMIT uses no DOM demonstrations or policy training. Both methods use the same 30 initial conditions per axis, with 240 episodes per method overall.

Evaluation axis	DynamicVLA	DYNAMIT
Closed-loop reactivity	0.667	0.133
Dynamic adaptation	0.600	0.167
Visual understanding	0.500	0.233
Spatial reasoning	0.500	0.467
Motion perception	0.600	0.700
Visual generalization	0.600	0.200
Motion generalization	0.733	0.233
Disturbance robustness	0.300	0.100
Overall	0.563	0.279

each placement and transfers all three moving M16 hex nuts. CoPa places two after missing the first, while DynamicVLA makes one failed grasp attempt and remains idle. The example distinguishes full-task success from partial completion, as captured by Q-score.

V. CONCLUSION

We presented DYNAMIT, a zero-shot framework that connects language instructions to visual tracking and predictive interception for conveyor pick-and-place. It achieves the highest aggregate success rate and Q-score among the evaluated methods across the conveyor studies. These results demonstrate zero-shot generalization across the tested conveyor speeds and industrial object geometries, with a shared pipeline for moving-object grasping and moving-receptacle placement. Execution requires no task-specific demonstrations, instance-specific object models, configured conveyor motion, or policy training. DOM further demonstrates transfer to motion-based instance selection and spatially specified manipulation without DOM demonstrations or policy training, although the DOM-trained DynamicVLA checkpoint performs better overall. Future work will build on these simulation results through shape-aware grasping and deployment on physical conveyors.

REFERENCES

- [1] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choremanski, T. Ding, D. Driess, A. Dubey, C. Finn, *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” *arXiv preprint arXiv:2307.15818*, 2023.
- [2] 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.
- [3] 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.
- [4] H. Xie, B. Wen, J. Zheng, Z. Chen, F. Hong, H. Diao, and Z. Liu, “Dynamicvla: A vision-language-action model for dynamic object manipulation,” *arXiv preprint arXiv:2601.22153*, 2026.
- [5] H. Fang, S. Li, S. Wang, X. Xi, D. Liang, and X. Bai, “Towards generalizable robotic manipulation in dynamic environments,” *arXiv preprint arXiv:2603.15620*, 2026.

- [6] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *2023 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 9493–9500.
- [7] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “Voxposer: Composable 3d value maps for robotic manipulation with language models,” *arXiv preprint arXiv:2307.05973*, 2023.
- [8] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, “Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation,” *arXiv preprint arXiv:2409.01652*, 2024.
- [9] H. Huang, F. Lin, Y. Hu, S. Wang, and Y. Gao, “Copa: General robotic manipulation through spatial constraints of parts with foundation models,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 9488–9495.
- [10] I. Akinola, J. Xu, S. Song, and P. K. Allen, “Dynamic grasping with reachability and motion awareness,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9422–9429.
- [11] F. Islam, O. Salzman, A. Agarwal, and M. Likhachev, “Provably constant-time planning and replanning for real-time grasping objects off a conveyor belt,” *The International journal of robotics research*, vol. 40, no. 12-14, pp. 1370–1384, 2021.
- [12] B. Burgess-Limerick, C. Lehnert, J. Leitner, and P. Corke, “Dgbench: An open-source, reproducible benchmark for dynamic grasping,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3218–3224.
- [13] H. H. Nguyen, M. N. Vu, F. Beck, G. Ebmer, A. Nguyen, and A. Kugi, “Language-driven closed-loop grasping with model-predictive trajectory replanning,” *arXiv preprint arXiv:2406.09039*, 2024.
- [14] B. Wen, W. Yang, J. Kautz, and S. Birchfield, “Foundationpose: Unified 6d pose estimation and tracking of novel objects,” in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024, pp. 17 868–17 879.
- [15] H. Singh, R. J. Das, M. Han, P. Nakov, and I. Laptev, “Malm: Multi-agent large language models for zero-shot robotic manipulation,” in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2025, pp. 20 386–20 393.
- [16] F. Ruggiero, V. Lippiello, and B. Siciliano, “Nonprehensile dynamic manipulation: A survey,” *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.
- [17] D. Noh, D. Kong, M. Zhao, A. Lizarraga, J. Xie, Y. N. Wu, and D. Hong, “Latent adaptive planner for dynamic manipulation,” *arXiv preprint arXiv:2505.03077*, 2025.
- [18] M. Van der Merwe and D. Jha, “In-context iterative policy improvement for dynamic manipulation,” *arXiv preprint arXiv:2508.15021*, 2025.
- [19] 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.
- [20] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryal, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, *et al.*, “Sam 2: Segment anything in images and videos,” in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 28 085–28 128.
- [21] R. Chen, G. Sun, Y. Li, J. Qin, and L. Benini, “Him2sam: Enhancing sam2 with hierarchical motion estimation and memory optimization towards long-term tracking,” in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2025, pp. 276–291.
- [22] Y. Li and D. Pathak, “Object-aware gaussian splatting for robotic manipulation,” in *ICRA 2024 workshop on 3D visual representations for robot manipulation*, 2024.
- [23] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, B. Martinez, H. Yin, M. Lingelbach, M. Hwang, A. Hiranaka, S. Garlanka, A. Aydin, S. Lee, J. Sun, M. Anvari, M. Sharma, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, Y. Li, S. Savarese, H. Gweon, C. K. Liu, J. Wu, and L. Fei-Fei, “Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation,” *arXiv preprint arXiv:2403.09227*, 2024.