

Receding-Horizon Pushing with Composable Object-Centric Policies

Zhiyi Yuan, Tianrun Hu, Anxing Xiao, Yuhong Deng, David Hsu, Hanbo Zhang

Abstract—Non-prehensile manipulation is practical for relocating large, heavy, or geometrically ungraspable objects. Yet, long-horizon pushing of arbitrarily-shaped 3D objects couples three problems: 1) where to push the object so as to approach the target pose, 2) whether each push is stable and reachable, 3) whether subsequent actions remain feasible. We present an object-centric pushing policy within a feedback-guided hierarchical framework. At the low level, a learning-based policy predicts contact actions from a pose- and scale-normalized point cloud, conditioned on a near single-step subgoal. A stability score is applied to evaluate the predicted contacts by a quasi-static sliding-versus-tipping analysis. At the high level, BIT* first searches for an object path, and the next several subgoals are checked by contact prediction and robot motion planning for future feasibility. Failed motion plans, as feedback, change the local path costs and trigger re-planning. During execution, only the first feasible action is executed. In simulation, we evaluate 22 objects in six different scenes, upon which we also conduct comprehensive ablation studies. Results demonstrate that our method outperforms baselines with a clear margin and can reliably achieve long-horizon object pushing tasks under different situations. We also report quantitative real-robot experiments with a Franka arm and qualitative demonstrations with a mobile manipulator for large and heavy objects, with directly zero-shot sim-to-real transfer.

I. INTRODUCTION

Pushing is practical when an object is too large or too heavy for grasping. Planar pushing of simple objects is well studied [1]–[3], whereas practical objects may have irregular 3D geometry and can tip or roll. For such objects, a contact that produces a desired planar push also depends on the object’s support geometry and force direction. Moreover, the error compounds over a long horizon. For example, moving along a collision-free object path may require robot contacts that are unreachable, and an inaccurate push may leave the object in a corner or near clutter where future actions are infeasible. These challenges expose a gap between local pushing policies and long-horizon pushing tasks. Analytical and model-predictive methods provide explicit physical reasoning but usually rely on precise and expensive contact models [4]–[6]. Learned object-centric policies get rid of contact modeling and reduce computation, but a local policy does not account for robot kinematics, scene collisions, or the feasibility of future actions [7]–[13]. Therefore, a question arises: *how can we achieve generalizable learned contacts and compose them for long-horizon pushing reliably?*

Zhiyi Yuan, Tianrun Hu, Anxing Xiao, Yuhong Deng, and David Hsu are with the School of Computing and Smart Systems Institute, National University of Singapore, Singapore. Hanbo Zhang is with the Shanghai Innovation Institute, Shanghai, China and the School of Computing and Smart Systems Institute, National University of Singapore, Singapore. Correspondence to: yuanzy9920@gmail.com, zhanghanbo@sii.edu.cn.



Fig. 1. A single-step contact policy is composed with stability scoring and feedback-guided hierarchical planning for long-horizon deployment.

To do so, our key idea is that a push is a contact on the object, so the action should be directly associated with the geometry of the object itself, which can be learned inside the object-canonical frame. Simultaneously, object stability and long-horizon feasibility, subject to environmental constraints, should be addressed by explicit, model-based verification rather than left to the learned policy. We therefore formulate the low-level pushing actions in the object frame normalized for pose and scale. In this frame, the map from a desired pose change to a contact is learned for each object shape, so a network trained on a small dataset, collected from a single push direction, generalizes to unseen poses, scales, and objects. Stability is checked explicitly by a quasi-static analysis of the object’s support polygon, which softly rejects contacts that would tip the object. This stability test is scale-invariant and consistent with the normalized policy. Finally, a bi-level planner, considering both object trajectories and robot motions, shapes the object’s path to keep future pushes reachable and feasible, with closed-loop replanning addressing model inaccuracy.

We evaluate 22 ShapeNet [14] objects in six simulated representative scenes and report quantitative real-world tabletop experiments on a Franka arm. The result concludes that a policy trained on only ten objects, each pushed from a single direction, moves unseen, geometrically complex objects to

their goals. Long-horizon pushing results demonstrate that our policy is composable for bi-level planners, and can effectively execute long-horizon tasks reliably. We additionally show a qualitative deployment on a mobile manipulator pushing large furniture, showing zero-shot generalization to different robots and task settings.

Our contributions include: (1) An object-centric learning approach for generalizable and composable pushing of geometrically complex 3D objects, which enables zero-shot transfer across embodiments, unseen objects and environments; (2) A feedback-guided bi-level planner integrated with the object-centric policy, which enables reliable long-horizon object pushing.

II. RELATED WORK

A. Object-Centric Manipulation Policies

Object-centric representations have been used to improve the transfer of robotic manipulation policies across scenes and tasks [10], [15]. Existing methods enhance the policy’s perception of task-relevant objects [16]–[20] or isolate the object to learn its representation independently [12], [13], [21]. However, these policies may still be coupled with scene- or robot-specific configurations, or suffer from large spatial variations caused by different object pose. To address these limitations, our method learns goal-conditioned contacts in an object-centric canonical space that normalizes both object pose and scale variations, decoupling the policy from robot- and scene-specific configurations and yielding composability and generalization with high data efficiency.

B. Object Pushing

Early methods established the foundation for model-based pushing through quasi-static analysis, contact modeling, and related techniques [1]–[3]. Recent model-based methods use online optimization and feedback replanning [6], [22]–[24]. However, these methods mostly focus on 2D objects and do not explicitly account for full 3D geometry. Learning-based methods have been explored for 3D object pushing. Some recent works [7] model objects as 3D point clouds for non-prehensile manipulation, but focus on contact-rich 6D pose adjustment rather than coupling contact prediction with future robot-motion feasibility. Other methods consider the toppling stability of 3D object pushing, but use geometric approximations most directly applicable to boxes, cylinders, or extruded shapes [25], [26]. We focus on the practical long-horizon pushing tasks, considering both geometrically complex 3D object stability and long-horizon feasibility by integrating a receding-horizon planner with a composable low-level pushing policy.

C. Hierarchical Manipulation Planning

Hierarchical manipulation is an important strategy to address long-horizon tasks [27]–[30]. Typical hierarchical methods decompose long-horizon tasks into high-level sub-goals and execute them with low-level planners or controllers [31]–[33]. However, their high-level modules operate without considering low-level execution previously. Bi-level planning

considers the trajectory feasibility of the complete task before action execution [34], [35]. However, standard bi-level planners typically assume known dynamics and use feasibility mainly to reject or rank high-level candidates, rather than exploiting failures to adapt the high-level planner, which is difficult to model pushing in practice. In contrast, we integrate a high-level planner that generates subgoals and preserve space for future pushes with a learned goal-conditioned low-level policy that executes them. Low-level feasibility failures are fed back to update the high-level cost distribution, enabling robust receding-horizon push planning without requiring accurate dynamics.

III. PROBLEM FORMULATION

We consider the problem of a rigid 3D object that is moved by planar pushes on a horizontal support surface with static obstacles. The inputs are its initial point cloud P_0 , $X_0 \in SE(2)$, target pose $X_g \in SE(2)$, and scene geometry $\mathcal{W}$, which is reconstructed from perceptions. The output is an object-pose sequence $X_{0:K}$ and a push sequence $u_{0:K-1}$, where K is determined by the planner and $u_k = (p_k, v_k, d_k)$ contains a surface contact point, a push direction, and a push distance. For every k , the contact must satisfy $u_k = \pi(X_k, X_{k+1}, P_k)$, $u_k \in \mathcal{S}(X_k, P_k) \cap \mathcal{R}(X_k, \mathcal{W})$. Here, P_k is the estimated object point cloud at k , π is the pushing policy that associates each desired single-step pose transform with a contact on P_k achieving it, $\mathcal{S}(X_k, P_k)$ collects the contacts that do not tip the object at pose X_k , and $\mathcal{R}(X_k, \mathcal{W})$ collects the contacts the robot’s end-effector can reach at X_k without colliding with $\mathcal{W}$. We assume known robot kinematics, a static scene during pushing, and single-point end-effector contact. The stability model further uses a quasi-static approximation and estimates the support polygon and center of mass from the observed geometry.

IV. METHOD

Our method combines three components: an object-centric contact-based policy, a quasi-static stability score τ , and a feedback-guided hierarchical planner. The high-level state is the planar object pose $X_k \in SE(2)$. The policy consumes a canonical point cloud $P_k^{(\phi)}$ and a canonical desired transform T_k (Sec. IV-B). At execution step k , the high-level planner generates an object path from X_k to the goal. For at most the next h path segments, the low level predicts contacts, evaluate them with τ , selects the best for each pushing step, and plans the associated robot motions. If all checked steps are feasible, the first pushing action is executed. The object is then perceived again, and the loop repeats until the target pose is reached. Figure 2 demonstrates the pipeline.

A. High-Level Object Path Planning

1) *Object Path Planner*: We leverage BIT* [36] as the high-level object planner. At step k , BIT* searches for a collision-free pose sequence $X_{k:K}$ for the target object. Let $D(X_i, X_{i+1}) \geq 0$ denote the $SE(2)$ edge metric and

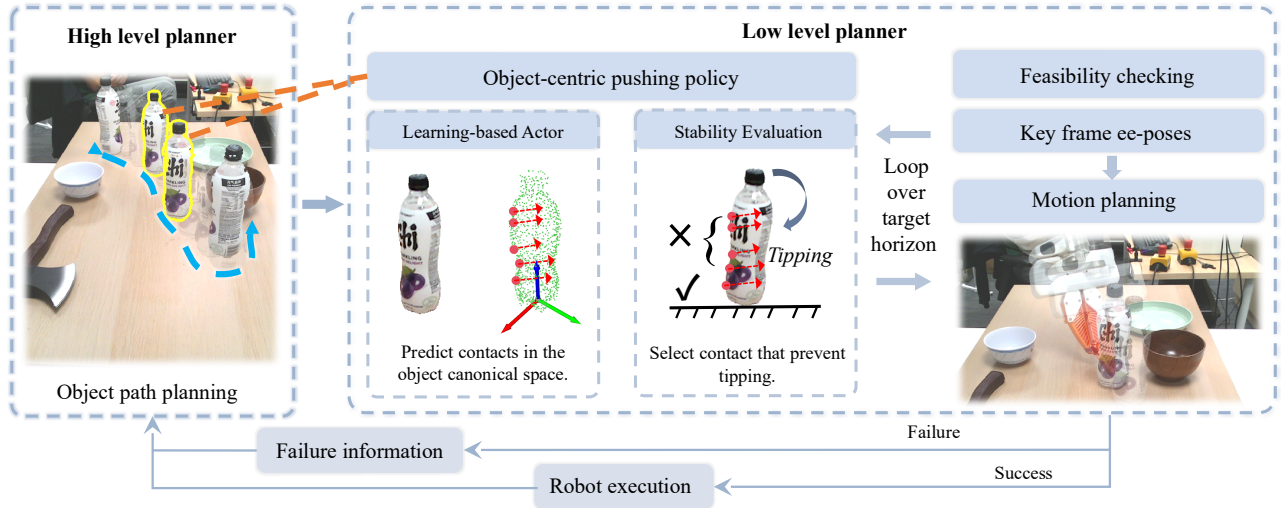


Fig. 2. Method overview. The high level plans an object path, the low level checks contact and robot-motion feasibility, and a failed check updates the high level cost distribution before replanning.

$C(X_i, X_{i+1}) \geq 0$ an adaptive edge penalty initialized to zero at the start of a task. BIT* minimizes

$$J(X_{k:K}) = \sum_{i=k}^{K-1} [1 + C(X_i, X_{i+1})] D(X_i, X_{i+1}). \quad (1)$$

Thus, before any low-level planning or motion failures, the objective is the usual path cost. Feedback changes the cost of path segments softly by updating the cost term $C(X_i, X_{i+1})$ after each failure. Besides, we inject clearance preference into the search prior. The planner first tries to search the object path with a minimum 5 cm distance to any obstacles. If it fails or does not return a path within time limits, the distance decreases until a feasible path is found.

2) *Future Feasibility Checking and Feedback*: At time step k , the contact policy proposes candidates for the desired moves in the next h steps. The stability score contributes to selecting the best one for each step, and the robot motion planner checks motion feasibility from its corresponding pre-push and post-push keyframes. If a motion cannot be found within a fixed planning time, the candidate object path is rejected. Denote $\bar{X}$ as the xy -position on the path at which the planner feasibility check fails. The corresponding 2D transition is denoted by $\bar{e}$. Their nearby edges are penalized. The penalty has two components: one decays with planar distance from the failed waypoint, while the other decays with angular difference and is further modulated by distance. Both components use truncated Gaussian decay, and accumulate across replans:

$$C(e) \leftarrow C(e) + 5K_p(e; \bar{X}) + 5K_t(e; \bar{e}), \quad (2)$$

The kernels are $K_p = e^{-d_p^2/(2\sigma_p^2)}$ and $K_t = e^{-(d_t^2/\sigma_p^2 + d_\theta^2/\sigma_\theta^2)/2}$ for $d_p \lesssim 3\sigma_p$, and zero otherwise. Here, d_p, d_t are the planar distances from X to $\bar{X}$ and the segment $\bar{e}$. The angle d_θ is between the planar directions of e and $\bar{e}$; $\sigma_p, \sigma_\theta > 0$ set the spatial and angular scales. The high level then replans with Eq. (1).

If all h checks succeed, the robot executes only the first push, observes the new object pose, and replans. The

procedure terminates when the goal is reached or the fixed replanning limit is exhausted. Notably, an important planner-specific parameter is h , which affects how far the robot is seeing before each physical action. We ablate the selection of h in Sec. V-B.

B. Low-Level Object-Centric Pushing Policy

1) *Object Frame Canonicalization*: At the beginning of each task, an object-canonical frame is constructed from the world-frame point cloud P_0 . We assume a known gravity direction. The z -axis of the canonical space passes through the centroid of P_0 and is oriented opposite to gravity. To determine the horizontal axes, P_0 is projected onto the support plane, and PCA is performed on the projected points. The dominant principal direction is defined as the x -axis, while the y -axis is determined according to the right-hand rule. Once this frame is established, the normalization length ℓ is the maximum Euclidean distance from its origin o_0 to the observed object surface.

Because a PCA axis has two possible signs, the sign chosen at task initialization is held fixed by pose tracking. At each step, let P_k be the observed world-frame point cloud. Pose tracking gives the world-frame pose $Y_k = (Q_k, o_k) \in SE(3)$ of the canonical frame, where $Q_k \in SO(3)$. The normalization length ℓ is obtained from the origin o_k and P_k . Combining the desired next planar pose X_{k+1} with the tracked roll, pitch, and support height defines $Y_{k+1} \in SE(3)$. Let $R_k \in SO(3)$ and $r_k \in \mathbb{R}^3$ be the rotation and translation of $Y_k^{-1}Y_{k+1}$, and let the superscript (ϕ) denote the canonical mapping. The two policy inputs are

$$P_k^{(\phi)} = Q_k^T(P_k - o_k)/\ell, \quad T_k = \begin{bmatrix} R_k & r_k/\ell \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (3)$$

For a predicted canonical contact $u = (p, v, d)$, the corresponding world-frame contact is $(Q_k p \ell + o_k, Q_k v, d\ell)$. This construction analytically removes the tracked rigid pose and scalar length from the policy input, making it generalizable regardless of object poses and scales.

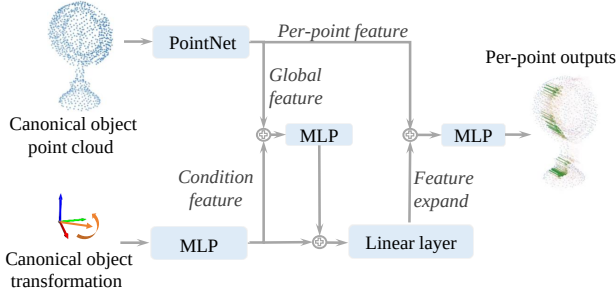


Fig. 3. Pushing policy network.

2) *Pushing-Action Representation*: A single-step push is represented in the canonical frame as contact $u = (p, v, d)$, where $p \in P_k^{(\phi)}$ is a contact point on the object surface, $v \in \mathbb{S}^2$ is a unit push direction, and $d \in \mathbb{R}_+$ is the push distance. Our policy predicts it densely over the observed geometry, generating at every point $p_i \in P_k^{(\phi)}$ a contact probability $\hat{a}_i \in [0, 1]$, a direction vector $\hat{v}_i \in \mathbb{R}^3$, and a distance $\hat{d}_i \in \mathbb{R}_+$. The direction vector is normalized during training. The predicted contacts remain explicit, so the stability score and robot-motion planner can evaluate them before execution. The canonical representation reduces pose and scale variations while decoupling camera configurations and robot embodiments from the policy.

3) *Data Generation*: Each training example is a tuple $(P^{(\phi)}, T, \{(a_i, v_i, d_i)\}_{i=1}^N)$ in the canonical frame. The inputs are the canonical object cloud $P^{(\phi)} = \{p_i\}_{i=1}^N \subset \mathbb{R}^3$ with $N = 1024$ and the canonical single-step transform T , and the per-point targets are a binary contact label $y_i \in \{0, 1\}$, a push direction v_i , and a push distance d_i , with v_i and d_i defined on the positive points $\{i : a_i = 1\}$.

Examples are collected in ManiSkill3 [37] with random pushing actions. Per rollout, an object is placed at a fixed position with random yaw, three RGB-D cameras reconstruct its cloud, and the end-effector pushes it along a fixed direction with a random distance, then retracts while it settles. From the world-frame object poses before and after the push and the recorded contact trajectory, we extract the single-step transform T and distance d . The five points nearest each recorded contact are labeled positive ($a_i = 1$, inheriting v and d) and the rest negative. The cloud is reduced to $N = 1024$ points by farthest-point sampling with the true contacts retained, then mapped to the canonical frame by ϕ . Rollouts with no object motion or incomplete reconstruction are discarded.

The main training set consists of 100 objects with 500 examples each, totaling 50,000 examples. A 10-object set with 5,000 examples per object, also totaling 50,000 examples, is used for ablation studies. Both are fivefold-augmented by canonical-frame jitter (random x, y translation within the object's convex hull and random yaw) to tolerate frame-estimation error at test time.

4) *Pushing Policy*: The policy operates entirely in the canonical frame. It is a point-wise network that maps the input pair $(P_k^{(\phi)}, T_k)$ to the per-point outputs $(\hat{a}_i, \hat{v}_i, \hat{d}_i)$. A PointNet [38] backbone of three 1×1 convolution layers with group normalization and ReLU extracts per-point and

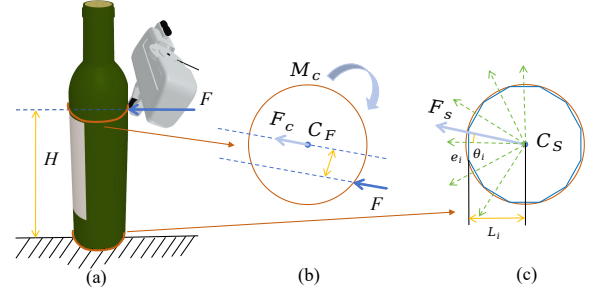


Fig. 4. Quasi-static stability model. (a) A horizontal push is applied at height H . (b) On the horizontal contact plane, the force F is decomposed about the projected center of mass C_F into F_c and M_c . (c) Support-plane, including the support polygon, the projection of F_c , the projected center of mass C_S and candidate tipping edges. (In a more general setting, the blue convex polygon $\mathcal{H}$ is the support convex hull.); dashed arrows show outward normals.

global features from $P_k^{(\phi)}$. A separate MLP embeds T_k . The global and transform features are fused, expanded to all points, and concatenated with the per-point features, as shown in Fig. 3. A shared decoder MLP maps each fused feature through three heads, producing a contact logit passed through a sigmoid to give $\hat{a}_i$, a direction vector $\hat{v}_i$, and a scalar distance $\hat{d}_i$.

The probability head is supervised by a weighted binary cross-entropy, the direction and distance heads by mean squared error for only the positive points. The loss is

$$\begin{aligned} \mathcal{L} = & -\frac{1}{N} \sum_{i=1}^N [\alpha a_i \log \hat{a}_i + (1 - a_i) \log(1 - \hat{a}_i)] \\ & + \frac{\lambda_v}{\max(N_+, 1)} \sum_{i=1}^N a_i \left(\frac{\hat{v}_i}{\|\hat{v}_i\|_2} - v_i \right)^2 \\ & + \frac{\lambda_d}{\max(N_+, 1)} \sum_{i=1}^N a_i (\hat{d}_i - d_i)^2. \end{aligned} \quad (4)$$

Here $N_+ = \sum_{i=1}^N a_i$ is the number of positive points, and $\alpha, \lambda_v, \lambda_d$ are weighting coefficients.

C. Stability Evaluation

We use a quasi-static model to evaluate the policy's contact candidates. For each candidate, H is its contact height, and the push direction is horizontal. Let m be the object mass, g the gravitational acceleration, and μ the object-support friction coefficient. The model compares the force required to initiate sliding with the force required to tip the object about an edge of its support polygon (Fig. 4).

At runtime, the support polygon is approximated by the convex hull of bottom-surface points, and the center of mass is approximated by the point-cloud centroid. Let $\mathcal{E} = \{e_i\}_{i=1}^M$ denote the set of edges of $\mathcal{H}$. For edge e_i , gravity acts at C_S , and the effective component of F forms an angle θ_i with its outward normal. Balancing the tipping and restoring moments yields the critical tipping force $F_{\text{tip},i} = \frac{mgL_i}{H \cos \theta_i}$. For fixed m, g , and H , the edge most likely to induce toppling is therefore the one $i^* = \arg \min_i \frac{L_i}{\cos \theta_i}$. For e_{i^*} , L is the perpendicular distance from the projected center of mass to that edge, and θ is the angle between the push

direction and the edge’s outward normal. The force required for tipping is $mgL/(H \cos \theta)$, whereas that required for Coulomb sliding is μmg . Their common factor mg cancels, giving the sliding-before-tipping ratio

$$\tau = \frac{L}{H \cos \theta}. \quad (5)$$

Under this model, sliding is predicted to start before tipping when $\mu < \tau$. The ratio is dimensionless and unchanged by ideal uniform scaling of both L and H . For evaluation, each candidate’s learned contact probability is multiplied by $\tanh(\beta\tau)$, and the candidate with the largest product is selected. Here $\beta > 0$ controls saturation. The ratio is a soft ranking cue, not a stability constraint.

V. EXPERIMENTS

The experiments are designed to comprehensively evaluate the performance of our method on long-horizon, cluttered pushing tasks, including both simulated ones and real-world scenes. Besides, we also conduct ablation studies on the horizon h as well as other proposed components. Results demonstrate that:

- *Our method can do reliable long-horizon pushing with limited training data, and directly generalize to the real world, including large-scale mobile pushing tasks;*
- *Seeing forward with larger h is not always beneficial due to inaccurate dynamics of pushing, but it helps significantly when there is high-risk alternative paths;*
- *The learned object-centric policy plays a crucial role for reliable performance, and failure-guided cost updates are also important for long-horizon pushing success.*

A. Experimental Setup

We first evaluate the method with 22 ShapeNet objects [14] in the six simulation scenes shown in Fig. 5, including 10 seen and 12 unseen objects. In each trial, the robot pushes an object from an initial pose to a target pose. The initial and target positions are sampled within a circular region of diameter 5, cm, while their orientations are sampled uniformly at random. Each object is evaluated in 50 trials per setting.

For simulation, we reconstruct the object point cloud from three cameras and obtain object masks using SAM2 [39]. Object poses are estimated with the PCA-ICP pipeline. To support diverse 3D geometries, task success is defined as the square root of the Chamfer distance between the current and target point clouds being below 1.5 cm. Each boxplot observation is an object-scene success rate from 50 trials, with a total of $22 \times 6 = 132$ observations, and 6600 long-horizon pushing trials. For group-mean success rates, we compute 95% percentile bootstrap intervals using 20,000 whole-object resamples within the 10 seen and 12 unseen objects, retaining each object’s scene and horizon records. Our experiments are run on an RTX 2080 GPU and an Intel i9-14900KF CPU.

For real-world deployment, we use a dual-arm robotic system with 4 objects in three real-world obstacle settings.

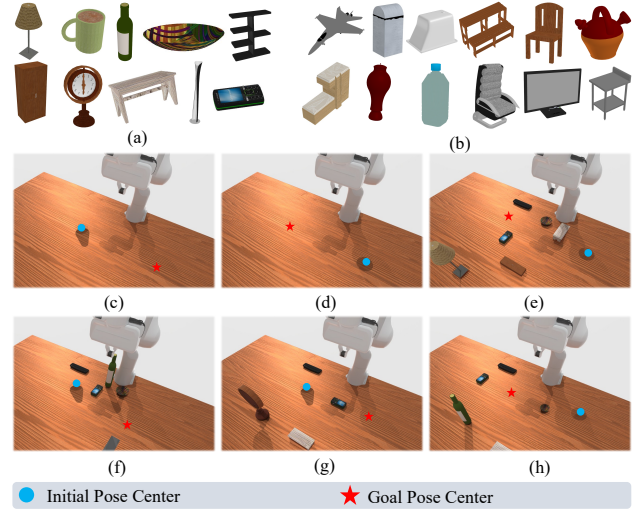


Fig. 5. Simulation benchmark. Subfigures (a) and (b) show the tested objects, with the former containing seen objects and the latter unseen objects; Scenes (c) and (d) are denoted as *Empty*, Scenes (e) and (f) as *Cluttered*, and Scenes (g) and (h) as *Adversarial*.

Each scene is set so that: 1) there exists a path (e.g., a line, a U-shape, or a S-shape) from the initial pose to the target pose; 2) obstacles and the target are visible. Object poses are estimated from monocular observations using SAM3D [40] and FoundationPose [41]. For each object under each setting, we run 5 different trials.

B. Results with Different Horizons

We evaluate $h = 1$, $h = 3$, and full-horizon planning in three kinds of scenes (Fig. 6). For Empty and Cluttered scenes, $h = 1$ achieves the highest mean success of 87.23% and 84.27%. By contrast, full-horizon planning yields 82.14% and 73.23%, respectively. In these tasks, a longer planning horizon may accumulate prediction errors during forward search, and hence deviate from the underlying true optimal path. As a result, failures here are mainly due to planning, in which it filters out some physically feasible paths and finds no solutions. For the Adversarial scenes, which include an apparently feasible object path whose later robot motions are infeasible, success rises from 49.82% at $h = 1$ to 73.05% with full-horizon planning, which highlights the significance of long-horizon planning.

C. Results of Different Methods

We compare to 3 baselines, including: *GRP* [31] (Greedy Reactive Policy with stability scoring), which selects goal-conditioned contacts heuristically and evaluates them with our stability model; *P-Net* (Push-Net) [31], adapted with multi-view reconstruction, rotation augmentation, and object-centric inputs; and *ManiFM* (ManiFoundation) [11], adapted with motion-feature extraction, sparse-contact-aware training, and robot-policy decoupling. Besides, we include an additional object-centric baseline, Ours-10, which is only trained with 10 objects. By comparison, Ours-100 is the default model, which uses 100 training objects with the same total number of training examples. Since the high-level planner setting h severely affects the end-to-end testing

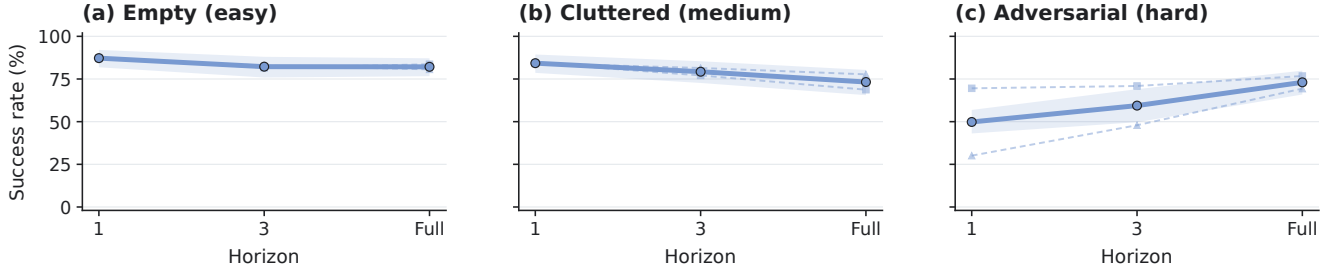


Fig. 6. Horizon sensitivity in (a) empty (S1-2), (b) cluttered (S3-4), and (c) adversarial (S5-6) scenes. Solid lines show group means, dashed lines show individual scene means, and shading shows 95% object-bootstrap confidence intervals. Each group mean uses 22 objects in two scenes.

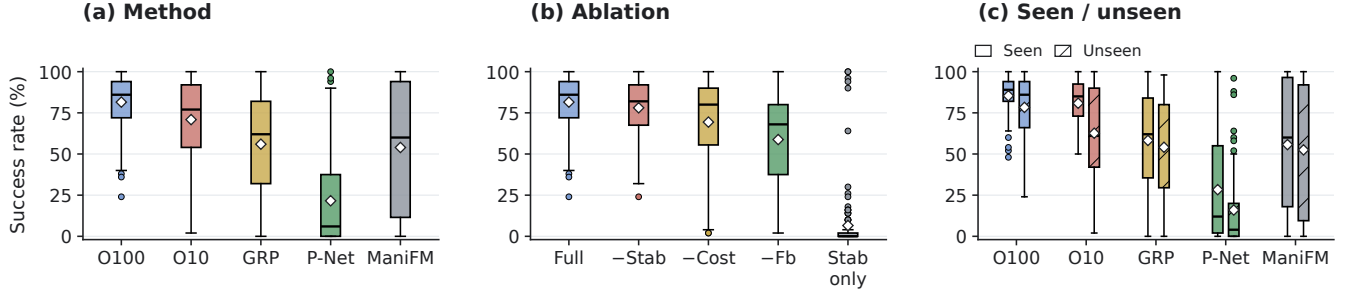


Fig. 7. Success-rate distributions for (a) methods, (b) ablations, and (c) seen/unseen objects. Each box contains 132 object-scene rates in (a,b), or 60/72 in (c), with 50 trials per rate. Boxes span the interquartile range; black lines mark medians, diamonds mark means, whiskers use 1.5 times the interquartile range, and circles mark outliers. O100/O10 denote Ours-100/Ours-10; Full denotes Ours-100.

TABLE I

MEAN SUCCESS RATES (%). EACH METHOD USES 6,600 TRIALS

Method	All objects	Seen objects	Unseen objects
Ours-100	81.52	85.37	78.31
Ours-10	70.89	80.83	62.61
GRP	55.97	58.23	54.08
P-Net	21.59	28.33	15.97
ManiFM	53.95	55.70	52.50

TABLE II

COMPARISON WITH CEM-MPC.

Object type	Method	Position (cm)	Orientation (rad)	Time (s)	Falls
Seen	Ours	1.31	0.466	0.026	2
Seen	CEM-MPC	3.41	0.608	81.8	14
Unseen	Ours	1.82	0.463	0.027	4
Unseen	CEM-MPC	3.44	0.557	71.8	6

performance, for fair comparisons, we use $h = 1$ for the Empty and Cluttered scenes, and a full-horizon planner for the Adversarial scenes for all baselines and our methods.

Table I reports mean success rates, while Fig. 7(a,c) shows their distribution across objects and scenes. Ours-100 reaches 81.52% overall and 78.31% on unseen objects. Increasing the training objects improves seen success by 4.54% and unseen success by 15.70%. The boxplots also expose variation hidden by scene averages: P-Net has a 6% median and 34 completely failed object-scene combinations; ManiFM’s interquartile range is 11.5-94%, with 13 completely failed combinations. Ours-100 has no zero-success combination and an interquartile range of 72-94%. These comparisons support transfer within the evaluated object set.

We also compare our method with a model-based approach, *CEM-MPC*, under a short-horizon pushing setting. It uses MUJOCO simulation for rollouts and the cross-entropy method for optimization [42]. In this setting, the robot is allowed at most two pushing actions to relocate the object to a near target pose. The results are shown in Table II. We report the final pose error between the object and the goal, whether the object tips over, and the inference time for a single push action. Within this two-push setting, our method has lower reported position and orientation errors, fewer falls, and lower per-action inference time than CEM-MPC.

D. Ablation Study

We demonstrate comparisons among four variants in Figure 7(b): *-Stab*, without stability scoring; *-Cost*, without local path-cost updates after failed robot motion planning; *-Fb*, without future feasibility check and feedback; and *Stab only*, selecting contacts using only the stability score, as a heuristic ablation. The mean (diamonds) are 81.52%, 78.06%, 69.38%, 58.77%, and 6.55%, respectively. Notably, *Stab only* has a 0% median with 81/132 zero-success pairs, indicating that only stability scores alone are insufficient for reliable goal-directed pushing in these tasks. Particularly, the cost-update difference is concentrated in the adversarial scenes. This is because the planner does not inherently favor routes near feasible regions, requiring feedback for replanning. Stability scoring brings marginal performance gain, but leads to better performance for geometrically complex unseen objects. For example, on the eighth unseen object, a vase (Fig. 5(b)), the overall success rate drops from 93% to 50% without the stability evaluation (300 trials in 6 scenes).



Fig. 8. Real-robot large-scale object relocation with a mobile manipulator. Top row: a table. Bottom row: a chair.

TABLE III

TABLETOP SUCCESSES AND MEAN SUCCESS RATE (%).

Object	Scene 1	Scene 2	Scene 3	Average
Packing tape	3/5	4/5	4/5	73.3
Hammer	2/5	2/5	3/5	46.7
Bottle	4/5	3/5	3/5	66.7
Chair	4/5	4/5	4/5	80.0

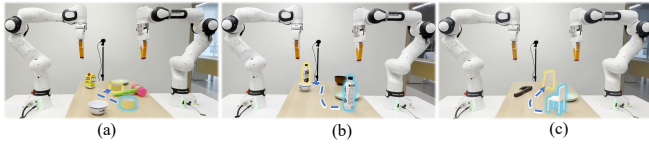


Fig. 9. Dual-arm robot scenes. The object marked in blue indicates the initial position, while the object inside the yellow indicates the target pose. (a), (b), and (c) show the scenarios with 4, 3, and 2 obstacles, respectively.

E. Real-world Deployment

We conduct real-world experiments on two Franka FR3 arms mounted on opposite sides of a shared tabletop and an AutoLife mobile manipulator. The tabletop setup uses an Intel RealSense L515 RGB-D camera for perception and a VAMP [43] planner for collision-aware motion planning, with one arm executing each push. AutoLife leverages the object-level planner and learned pushing policy, integrating base navigation with whole-body IK and collision-aware planning over the leg, waist, and arm joints. Both re-estimate the object pose and replan after each push. Across the tabletop experiments (Fig. 9), the system succeeds in 40/60 trials (66.7%; Table III). We also include a full failure analysis for the real-robot deployment in Fig. 10) for a deeper understanding of the system-level bottleneck. Among all failures, perception accounts for more than half, 12/20 failures. The remaining comprises mainly four types of failures including infeasible IK (IK error), object pushed out of workspace (Workspace), and collision failure (Collision). Perception is therefore the main recorded limitation, revealing that geometry or pose errors can affect planning, push action prediction, and feasibility checking, motivating better reconstruction and tracking alongside pushing components.

Finally, we deploy our method on the AutoLife mobile robot to relocate a chair and a coffee table in household environments (Fig. 1). The complete relocation process for both objects is shown in Fig. 8.

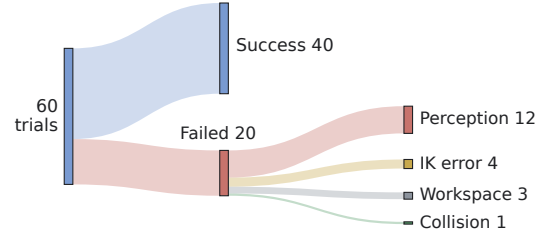


Fig. 10. Outcomes of 60 tabletop trials. Ribbon widths are proportional to counts; the 20 failures are partitioned by recorded cause.

VI. CONCLUSION

We presented a long-horizon pushing system that integrates object-level path search, object-centric push policy learning, quasi-static stability ranking, and robot-motion planning in a hierarchical planning framework. Performance in both simulation and real-world tabletop experiments supports generalization to unseen object geometries and shows benefits from the feedback and stability modules. The mobile-manipulator demonstrations show that our model can directly generalize to large-scale objects, without retraining or data collection. The main system-level bottleneck lies in the object-centric perception pipeline, which relies on stable object pose estimation and tracking. Future work can consider enhancing perception modules [44] or absorbing perception noise into data-driven policies. Besides, for non-prehensile manipulation like pushing, physical parameters like the center of mass, friction, and their uncertainties are critical for robustness. Future work includes estimating these parameters with uncertainty-aware models and integrating our pushing policy into longer-horizon tasks that require diverse skill primitives [45]–[47].

ACKNOWLEDGMENT

We used vibe coding tools including Codex and Claude Code to help implement parts of the algorithm and analyze bugs. Besides, we used AI tools to design visuals for Figure 6, 7, and 10, and to polish the text and grammar.

REFERENCES

- [1] M. T. Mason, "On the scope of quasi-static pushing," in *International Symposium on Robotics Research*, 1986, 1986, pp. 229–233.
- [2] K. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 1992, pp. 416–421.

- [3] S. Akella and M. T. Mason, "Posing polygonal objects in the plane by pushing," *The International Journal of Robotics Research*, pp. 70–88, 1998.
- [4] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *Frontiers in Robotics and AI*, vol. Volume 7 - 2020, 2020.
- [5] T. Meriçli, M. Veloso, and H. L. Akın, "Push-manipulation of complex passive mobile objects using experimentally acquired motion models," *Autonomous Robots*, vol. 38, no. 3, pp. 317–329, 2015.
- [6] H. Bui, *et al.*, "Push anything: Single- and multi-object pushing from first sight with contact-implicit mpc," 2026. [Online]. Available: <https://arxiv.org/abs/2510.19974>
- [7] W. Zhou, B. Jiang, F. Yang, C. Paxton, and D. Held, "Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation," 2024.
- [8] C. Chi, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.
- [9] B. Serhan, H. Pandya, A. Kucukyilmaz, and G. Neumann, "Push-to-see: Learning non-prehensile manipulation to enhance instance segmentation via deep q-learning," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1513–1519.
- [10] Y. Zhu, Z. Jiang, P. Stone, and Y. Zhu, "Learning generalizable manipulation policies with object-centric 3d representations," in *7th Annual Conference on Robot Learning*, 2023.
- [11] Z. Xu, *et al.*, "Manifoundation model for general-purpose robotic manipulation of contact synthesis with arbitrary objects and robots," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.
- [12] R. Wu and Y. Zhao, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," in *International Conference on Learning Representations (ICLR)*, 2022, 2022.
- [13] Y. Chen, C. Wang, Y. Yang, and K. Liu, "Object-centric dexterous manipulation from human motion data," in *8th Annual Conference on Robot Learning*, 2024.
- [14] A. X. Chang, *et al.*, "ShapeNet: An Information-Rich 3D Model Repository, Tech. Rep. arXiv:1512.03012, 2015.
- [15] K. Yu, *et al.*, "Genflowrl: Shaping rewards with generative object-centric flow in visual reinforcement learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2025, pp. 13 183–13 192.
- [16] J. Wen, *et al.*, "Object-centric instruction augmentation for robotic manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 4318–4325.
- [17] W. Yuan, C. Paxton, K. Desingh, and D. Fox, "Sornet: Spatial object-centric representations for sequential manipulation," in *6th Annual Conference on Robot Learning*, 2022.
- [18] X. Li, *et al.*, "Manipllm: Embodied multimodal large language model for object-centric robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 18 061–18 070.
- [19] J. Qian, A. Panagopoulos, and D. Jayaraman, "Recasting generic pretrained vision transformers as object-centric scene encoders for manipulation policies," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 17 544–17 552.
- [20] S. Ferraro, P. Mazzaglia, T. Verbelen, and B. Dhoedt, "Focus: object-centric world models for robotic manipulation," *Frontiers in Neuro-robotics*, vol. Volume 19 - 2025, 2025.
- [21] M. Pan, *et al.*, "Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025, pp. 17 359–17 369.
- [22] A. Heins and A. P. Schoellig, "Force push: Robust single-point pushing with force feedback," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6856–6863, 2024.
- [23] K. Ren, G. Wang, A. S. Morgan, L. E. Kavraki, and K. Hang, "Object-centric kinodynamic planning for nonprehensile robot rearrangement manipulation," *IEEE Transactions on Robotics*, 2025.
- [24] J. Moura, T. Stouraitis, and S. Vijayakumar, "Non-prehensile planar manipulation via trajectory optimization with complementarity constraints," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 970–976.
- [25] I. Dadiotis, M. Mittal, N. Tsarakakis, and M. Hutter, "Dynamic object goal pushing with mobile manipulators through model-free constrained reinforcement learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 13 363–13 369.
- [26] Z. Bi, *et al.*, "Interactive navigation for legged manipulators with learned arm-pushing controller," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 9–16.
- [27] E. Triantafyllidis, F. Acero, Z. Liu, and Z. Li, "Hybrid hierarchical learning for solving complex sequential tasks using the robotic manipulation network roman," *Nature Machine Intelligence*, vol. 5, no. 9, 2023.
- [28] Z. Zhang, *et al.*, "Universal visual decomposer: Long-horizon manipulation made easy," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [29] Z. Xie, J. Tseng, S. Starke, M. van de Panne, and C. K. Liu, "Hierarchical planning and control for box loco-manipulation," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, pp. 1–18, 2023.
- [30] A. Xiao, H. Zhang, T. Hu, and D. Hsu, "Hypothesis-driven model expansion under uncertainty for open-world robot planning," in *Robotics: Science and Systems (RSS)*, 2026.
- [31] J. K. Li, D. Hsu, and W. S. Lee, "Push-net : Deep planar pushing for objects with unknown physical properties," in *Robotics: Science and System*, 2018.
- [32] O. Kroemer, C. Daniel, G. Neumann, H. van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015.
- [33] S. Nasiriany, H. Liu, and Y. Zhu, "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7477–7484.
- [34] N. Kumar, W. McClinton, K. Le, and T. Silver, "Bilevel planning for robots: An illustrated introduction," 2023, <https://lis.csail.mit.edu/bilevel-planning-for-robots-an-illustrated-introduction>.
- [35] B. Li, T. Silver, S. Scherer, and A. Gray, "Bilevel Learning for Bilevel Planning," in *Proceedings of the Robotics: Science and Systems (RSS)*, 2025.
- [36] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.
- [37] S. Tao, *et al.*, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," *Robotics: Science and Systems*, 2025.
- [38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [39] N. Ravi, *et al.*, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024. [Online]. Available: <https://arxiv.org/abs/2408.00714>
- [40] S. D. Team, *et al.*, "Sam 3d: 3dfy anything in images," 2025. [Online]. Available: <https://arxiv.org/abs/2511.16624>
- [41] 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)*, 2024, pp. 17 868–17 879.
- [42] A. H. Li, P. Culbertson, V. Kurtz, and A. D. Ames, "Drop: Dexterous reorientation via online planning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 14 299–14 306.
- [43] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in *ICRA*, 2024, pp. 8749–8756. [Online]. Available: <http://arxiv.org/abs/2309.14545>
- [44] A. Anand, A. Agarwal, and L. P. Kaelbling, "Which reconstruction model should a robot use? routing image-to-3d models for cost-aware robotic manipulation," *arXiv preprint arXiv:2603.27797*, 2026.
- [45] B. Zandonati, T. Lozano-Pérez, and L. P. Kaelbling, "Rational inverse reasoning: Few-shot imitation by inferring intent through planning," *arXiv preprint arXiv:2508.08983*, 2025.
- [46] Y. Huang, *et al.*, "Kinder: A physical reasoning benchmark for robot learning and planning," in *Robotics: Science and Systems (RSS)*, 2026.
- [47] W. Shen, *et al.*, "Tiptop: A modular open-vocabulary planning system for robotic manipulation," *arXiv preprint arXiv:2603.09971*, 2026.