MULTIPACK: HIERARCHICAL PLANNING WITH PHYSICS FEEDBACK FOR 3D MULTI-BIN PACKING

Anonymous authors
Paper under double-blind review

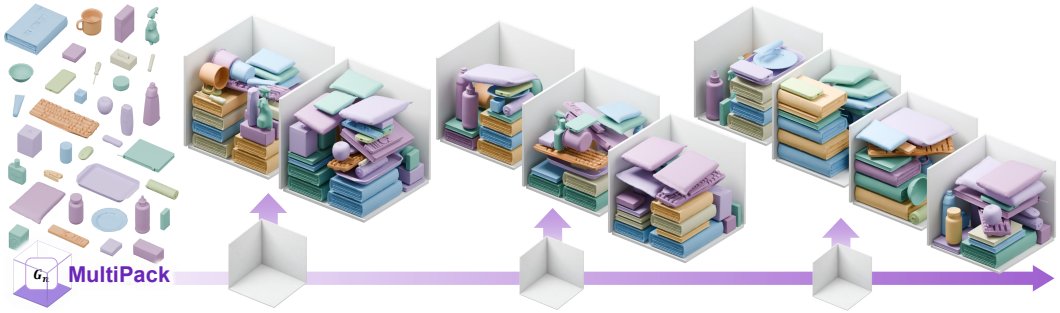

Figure 1: **MultiPack for irregular 3D multi-bin packing.** MultiPack combines hierarchical planning based on LLM-derived scheduling attributes with physics feedback to determine bin assignments, placement order, and physically validated object poses. The resulting packing layouts are shown for three different bin sizes.

ABSTRACT

Packing household objects into storage bins requires deciding which objects to pack together, in what order, and in which poses. These decisions are tightly coupled: locally compact placements may increase overall bin usage, while physical settling can change the space available for subsequent objects. In this paper, we introduce MultiPack, a hierarchical multi-bin packing framework that integrates cross-bin composition planning with physically validated in-bin pose search. Our Object–Group–Bin planning strategy groups objects by LLM-derived support, protection, and access priorities, connecting cross-bin allocation with in-bin scheduling. A capacity-based mixed-integer linear program assigns groups to bins by minimizing volume overflow. Within each bin, physics-feedback geometric search combines heightmap-guided candidate generation with height-envelope and weighted surface-clearance scoring. Physical validation, rollback, and settled-state synchronization close the loop between placements. Experiments on two datasets show reduced bin usage and competitive or improved layout quality under a shared evaluation protocol, while a real-world demonstration with 54 objects shows how the generated plans can guide human packing.

1 INTRODUCTION

As embodied AI advances toward everyday assistance in homes, agents must decide not only how to manipulate individual objects, but also how to organize them to accomplish broader tasks (Chang et al., 2025). Packing household objects into storage bins is a familiar example: even an agent capable of picking up and placing every object must still decide which objects to pack together, in what order, and in which poses. These choices determine how many bins are needed and must accommodate diverse object shapes, physical interactions, and preferences for protection and subsequent access. We study this planning problem as irregular 3D multi-bin packing. Given object names and 3D models, together with bins of identical dimensions, our goal is to generate bin assignments, placement sequences, and physically validated poses that pack the objects into as few bins as possible while incorporating scheduling preferences. Figure 1 illustrates the resulting arrangements across different bin sizes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This task is challenging because bin assignment, packing order, and spatial placement are tightly coupled. A locally compact arrangement need not minimize the total number of bins: assigning an object to the current bin changes its remaining space and affects which objects remain to be packed together in subsequent bins. Even an assignment that satisfies a volume budget may be unrealizable because of incompatible shapes or placement-order constraints. Physical interactions make these decisions harder to predict. A collision-free candidate need not settle stably, and settling can shift previously placed objects, making the scene representation used for subsequent search outdated. An effective planner must therefore (i) coordinate cross-bin allocation with in-bin placement, (ii) reason over the full object set to improve overall bin utilization, and (iii) validate each placement physically while keeping its internal scene up to date.

Existing methods address local placement, cross-bin planning, and physical feedback largely in isolation. For local placement, heightmap minimization (HM) (Wang & Hauser, 2019) and SDF-Pack (Pan et al., 2023) favor small height-envelope growth and close geometric fit, respectively. Hierarchical reinforcement learning (Huang et al., 2022a) further couples object ordering with placement through task-specific training. These approaches improve in-bin decisions without explicitly planning the composition of subsequent bins. Multi-bin formulations (Liu et al., 2023b; Cheng et al., 2024) extend planning across containers, but do not explicitly incorporate the protection and access preferences considered here. Meanwhile, stability checks (Theodoropoulos et al., 2025) and closed-loop correction support physically feasible dense packing (Wang & Hauser, 2021; Qin et al., 2026), but do not by themselves coordinate cross-bin allocation. Jointly addressing cross-bin allocation, scheduling-aware composition, and physically validated placement therefore remains a challenge.

To address these challenges, we present **MultiPack**, an LLM-guided framework that combines hierarchical planning with physics feedback for irregular 3D multi-bin packing. Our key idea is to plan bin composition and individual placements at different levels while linking them through an Object–Group–Bin hierarchy: LLM-derived scheduling attributes organize objects into groups that guide both bin allocation and placement order. Geometric search and rigid-body simulation then turn coarse assignments into physically validated placements, with settling outcomes updating the scene for subsequent decisions and placement failures informing later allocation.

Specifically, at the cross-bin level, MultiPack uses an LLM to infer support, protection, and access priorities from the task description and object names. Objects with similar priorities are grouped, and a mixed-integer linear program (MILP) assigns these groups to bins by minimizing volume overflow. Groups are assigned as units and processed in an order guided by their scheduling priorities, with their members handled consecutively within each bin. The resulting assignments provide a coarse plan whose geometric and physical feasibility is determined during placement.

At the in-bin level, we couple heightmap-guided 3D voxel search with rigid-body simulation in a closed loop. The solver generates candidate placements from the current heightmap, filters collisions using 3D voxel occupancy, and ranks the remaining candidates by volume-normalized height-envelope growth, surface clearance, and placement height. Simulation then evaluates selected candidates during descent and settling. An accepted trial updates the search representation using the settled poses of all objects in the bin; a rejected trial restores the previous state. Experiments on two irregular-object datasets across multiple bin sizes show that MultiPack achieves lower mean bin counts than the evaluated baselines under a shared physical revalidation and completion protocol, with competitive or improved layout quality. A real-world demonstration with 54 household objects further illustrates how the generated plans can guide human packing.

Our contributions are threefold. First, **MultiPack** integrates cross-bin composition planning with physically validated in-bin pose search. Second, our **Object–Group–Bin strategy** links allocation and scheduling through shared groups, assigned to bins by a capacity-based MILP. Third, **physics-feedback geometric search** combines heightmap-guided candidate generation with height-envelope and weighted surface-clearance scoring, using physical validation, rollback, and settled-state synchronization to close the loop between placements.

2 Related Work

Irregular 3D Packing Packing irregular 3D objects requires deciding the order, orientation, and position of each object under geometric and stability constraints (Gupta & Raman, 2026). Geometric

heuristics rank candidate placements by criteria: DBL-style heuristics (Karabulut & İnceoğlu, 2004) favor the deepest-bottom-left feasible position, HM (Wang & Hauser, 2019) minimizes height-envelope growth, and SDF-Pack (Pan et al., 2023) evaluates geometric fit using a signed distance field. These heuristics evaluate placements statically: they do not capture how settling changes the poses of placed objects and the space available for subsequent ones. Another line of work is learning-based (Perovic et al., 2025): PPN-Pack (Pan et al., 2024) learns placement proposals to prune the candidate space, HRL (Huang et al., 2022a) uses hierarchical policies to jointly decide object order and placement, and IR-BPP (Zhao et al., 2023) extends this line to online packing. These methods require task-specific training, and their generalization across object distributions and bin configurations must be established empirically. In contrast, MultiPack requires no packing-specific training: a pretrained LLM provides scheduling attributes for cross-bin assignment and in-bin ordering, while geometric search, coupled with rigid-body validation, produces the placements.

LLM-Guided Planning Early work prompts LLMs to directly generate action sequences for embodied agents (Huang et al., 2022b; Song et al., 2023), but the resulting plans often violate state and action constraints, as PlanBench (Valmeekam et al., 2023) shows. To improve executability, one line of work grounds LLM outputs in explicit representations or external solvers (Hao et al., 2025; Shu et al., 2025; Choi et al., 2025): SayCan (Ahn et al., 2022) combines LLM scores with skill value functions; ProgPrompt (Singh et al., 2022) enumerates available objects and actions in program-structured prompts; and LLM+P (Liu et al., 2023a) translates task descriptions into PDDL for classical planning. A second line incorporates feedback and spatial grounding (Duan et al., 2024; Liu et al., 2026): Inner Monologue (Huang et al., 2022c) revises plans using success detection and scene descriptions; SayPlan (Rana et al., 2023) plans over hierarchical 3D scene graphs with simulation-based feedback; and VoxPoser (Huang et al., 2023) converts language-derived affordances and constraints into 3D value maps that guide manipulation. In contrast, MultiPack uses the LLM only to produce reusable scheduling attributes; bin assignment and placement are delegated to combinatorial optimization, geometric search, and physical validation.

3 METHOD

3.1 OVERVIEW

Given objects $\mathcal{O} = \{o_i\}_{i=1}^{N}$ with names and 3D models, and identical bins of dimensions (L_x, L_y, L_z), the goal is to pack all objects into as few bins as possible while respecting scheduling preferences. The output specifies bin assignments, placement order, and physically validated poses.

MultiPack separates packing decisions into two levels (Figure 2). At the cross-bin level, objects are grouped by LLM-derived scheduling attributes that encode these preferences, and the groups are assigned to bins by minimizing volume overflow. At the in-bin level, geometric candidates are generated from the current voxel scene and evaluated through physics simulation. An accepted placement is committed and the scene is updated from the settled poses. If a trial is rejected, the previous state is restored before the next candidate is attempted. Before packing, each object is voxelized under preset orientations. Appendix A details geometric preprocessing.

3.2 CROSS-BIN ALLOCATION

Multi-bin packing requires assigning objects to bins and organizing their placement order within each bin. We cluster objects with similar scheduling requirements into a group set $\mathcal{G}$, providing a shared unit for cross-bin allocation and in-bin scheduling. Each group is assigned as a whole, and its members are processed consecutively in the assigned bin. For n objects and K candidate bins, this reduces the number of binary assignment variables from nK to $|\mathcal{G}|K$.

Scheduling-attribute grouping. Given the task description and object names, the LLM assigns each object three five-level scheduling attributes, $\mathbf{p}_i = (p_i^{\text{sup}}, p_i^{\text{pro}}, p_i^{\text{acc}}) \in \{0, \ldots, 4\}^3$. Support priority favors early placement in the base layout, protection priority limiting subsequent coverage, and access priority facilitating later retrieval. Protection and access capture different needs but both favor later scheduling. These attributes guide coarse planning through scheduling preferences; they do not measure surface flatness, load-bearing capacity, or retrieval feasibility.

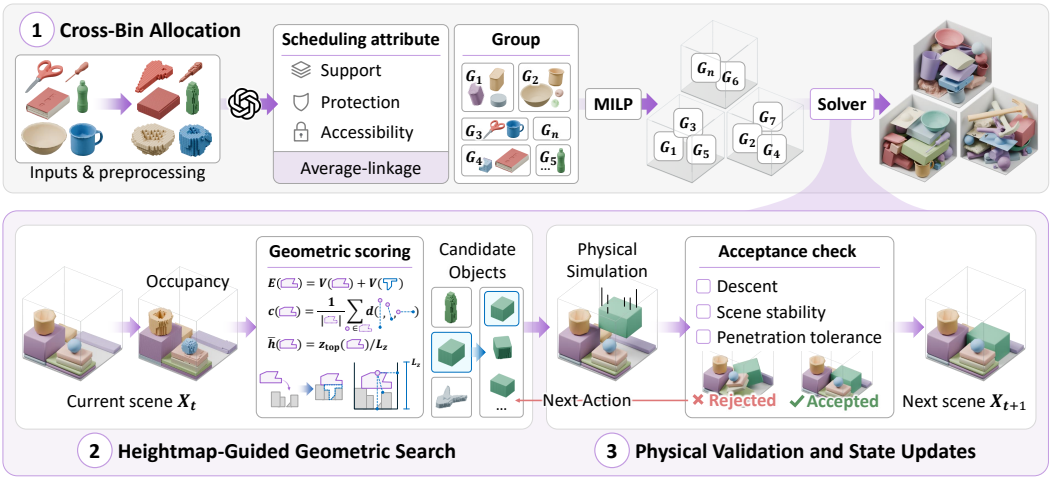

Figure 2: **Overview of MultiPack. (1)** LLM-derived scheduling attributes guide grouping, followed by capacity-based MILP allocation. **(2)** Heightmap-guided voxel search generates candidates and scores envelope growth, surface clearance, and placement height. **(3)** Physics simulation validates placements: accepted trials update the voxel scene from settled poses, while rejected trials restore the previous state.

Starting from singleton clusters, we perform average-linkage agglomerative clustering with similarity $\mathrm{sim}(i,j) = 1 - \|\mathbf{p}_i - \mathbf{p}_j\|_1/12 \in [0,1]$, where the denominator is the maximum distance between two attribute vectors. At each step, we merge the most similar pair of clusters whose average cross-cluster similarity is at least τ and whose combined size does not exceed m, until no admissible merge remains.

Capacity-based MILP allocation. We assign groups to bins using a mixed-integer linear program (MILP) that minimizes volume overflow. Let $C = L_x L_y L_z$ denote the bin volume, which serves as the volume budget for coarse allocation. Since total object volume alone does not determine packing feasibility, the resulting assignment is subsequently evaluated by geometric search and physical validation. For each group g, let $V_g = \sum_{i \in g} v_i$ denote its total voxel volume, where v_i is the object-volume proxy used for allocation.

For K candidate bins, let $x_{gb} \in \{0,1\}$ indicate whether group g is assigned to bin b. Each group is assigned exactly once, and every candidate bin receives at least one group:

$$\sum_{b=1}^{K} x_{gb} = 1 \quad \forall g \in \mathcal{G}, \qquad \sum_{g \in \mathcal{G}} x_{gb} \geq 1 \quad \forall b = 1, \ldots, K. \tag{1}$$

Writing $\mathbf{x} = (x_{gb})$, define the assigned volume of bin b as $V_b = \sum_g V_g x_{gb}$. Subject to Eq. (1), we solve

$$\min_{\mathbf{x}} \ \Omega(\mathbf{x}), \qquad \Omega(\mathbf{x}) = \frac{1}{C} \sum_{b=1}^{K} [V_b - C]_+, \tag{2}$$

where $[z]_+ = \max(z, 0)$. This objective penalizes overflow separately for each bin, preventing unused capacity elsewhere from offsetting a local excess. It does not require equal assigned volumes across bins. Scheduling attributes influence allocation through group construction and guide subsequent in-bin ordering; they do not enter the MILP objective directly.

The planner estimates an initial bin count from group volumes and searches over K within a time budget, keeping group membership fixed during each round. When solved to optimality, the assignment minimizes the overflow surrogate under the fixed grouping and bin count. Under a time limit, a validated integer feasible solution may instead be retained. Geometric and physical realizability are subsequently assessed by in-bin search and physical validation. MILP linearization and bin-count search are detailed in Appendix B.

3.3 HEIGHTMAP-GUIDED GEOMETRIC SEARCH

Given coarse assignments and scheduling attributes, the in-bin solver determines object orientations and positions using a voxel scene that maintains 3D occupancy, object ownership, and a heightmap.

Candidate generation. Groups and their members are ordered using scheduling attributes, favoring objects with higher support priority. The first $w = 3$ unprocessed objects in the current group form the candidate window. For each object, we enumerate preset orientations and discrete horizontal positions. The scene heightmap and object bottom profile determine the lowest placement height that avoids the scene surface and bin floor. Physical-dimension, grid-boundary, and 3D occupancy checks then discard out-of-bounds or colliding candidates.

Geometric scoring. For a candidate action a, let H_t and $H_{t,a}$ denote the current heightmap and the provisional heightmap after adding the candidate, both in physical length units. With horizontal grid spacing Δ, the increase in height-envelope volume is

$$E(a) = \Delta^2 \sum_{x,y} \left[H_{t,a}(x, y) - H_t(x, y) \right], \tag{3}$$

where $H_{t,a}$ is computed before physical settling.

Let $V(a) > 0$ denote the candidate's voxel volume, and let $c(a) \in [0, 1]$ denote its mean normalized surface clearance. This term combines individually truncated distances to the bin floor, existing occupancy, and side walls; smaller values favor closer geometric fit. We also define $\bar{h}(a) = z_{\text{top}}(a)/L_z$, where $z_{\text{top}}(a)$ is the candidate's highest occupied height above the floor before settling. We score placements by

$$q_{\text{geom}}(a) = \frac{1}{\max\{1, E(a)/V(a)\} + c(a) + \lambda_h \bar{h}(a)}. \tag{4}$$

The three terms penalize normalized envelope growth, surface clearance, and placement height, respectively. We use $\lambda_h = 6$ throughout. The height term continues to favor lower placements after floor clearance reaches its truncation distance.

For each candidate object, we rank actions by q_{geom} and retain at most $J = 4$ actions with diverse horizontal positions. We then use the volume-weighted score $s(a) = V(a) q_{\text{geom}}(a)$ to select the object with the highest-scoring retained action and order its physical trials. Trials continue until a placement is accepted or the candidate pool or trial budget is exhausted. After acceptance, subsequent candidates are reevaluated on the updated scene. Scheduling rules, clearance computation, and the connection to HM are detailed in Appendix C.

3.4 PHYSICAL VALIDATION AND STATE UPDATES

Geometric scoring determines trial order, while physical validation determines whether a placement is committed. Let X_t denote the accepted physical scene. Before each trial, the system snapshots X_t, initializes the candidate above its target pose, and performs local descent. If descent triggers a penetration or disturbance rejection, the pre-descent state is restored. The candidate may then be released from its original elevated pose if penetration at that pose remains within tolerance. Both paths proceed to whole-bin settling, producing a provisional state X_t^a. Acceptance is

$$\text{Accept}(a \mid X_t) = \mathbf{1}[\mathsf{Init}(X_t, a) \wedge \mathsf{Stable}(X_t^a) \wedge \mathsf{Geom}(X_t^a, a)], \tag{5}$$

with three predicates:

- Init: the candidate enters settling through either checked descent or a validated release pose after restoring the pre-descent state.
- Stable: the scene settles within the simulation budget, judged by velocity thresholds, contact conditions, and bounded pose changes over a consecutive sampling window.
- Geom: candidate penetration stays within tolerance, and all objects satisfy the bin boundary conditions after settling.

On acceptance, the system commits X_t^a and synchronizes the settled poses of all placed objects with the voxel representation, updating object occupancy and ownership, the heightmap, and support information. Subsequent distance queries are computed from the updated scene. On rejection, the snapshot is restored, the search representation is synchronized with the restored state, and the next candidate in score order is attempted. Settling-induced changes to existing placements therefore inform the next geometric search, while a rejected trial leaves the accepted scene unchanged. Physical validation establishes acceptance within a finite simulation window. Appendix D specifies the validation criteria; the experimental simulation settings and thresholds are reported in Appendix E.1.

After processing the assigned bins, we retry unpacked objects in existing bins. Objects that still remain unpacked are regrouped and reallocated using the procedure in Section 3.2.

4 EXPERIMENTS

We evaluate MultiPack on two irregular-object datasets. Multi-bin packing is our main setting, with a focus on compactness of the packing and the number of boxes. Single-bin experiments provide a supplementary assessment of the in-bin solver.

4.1 EXPERIMENTAL SETUP

Datasets. We use **PPN-Pack AutoStore** as our main evaluation dataset. It contains 96 object shapes, primarily from YCB and Rutgers APC RGB-D. We use the first 100 of its 2,000 released test sequences, each containing 80 objects. **Mixed Household Objects (MHO)** serves as a supplementary dataset and contains 198 asset templates from YCB and RoboTwin. We evaluate 100 fixed-seed sequences of 80 objects each.

Baselines. We compare the geometric heuristics HM (Wang & Hauser, 2019) and DBL (Karabulut & İnceoğlu, 2004), the learning-based method PPN-Pack (Pan et al., 2024), and GPT6-Astra, which directly predicts packing order and poses. HM and DBL both use descending bounding-box volume order. PPN-Pack uses its publicly available pretrained weights and native reordering buffer, without retraining on either dataset. We extend the geometric and learned placement baselines from single-bin placement to multi-bin packing through the adaptation described in Appendix E.1. Single-bin experiments directly compare their in-bin packing performance.

Task settings. Multi-bin experiments use the same 80 input objects from each sequence with identical cubic bins of side length 0.25, 0.30, or $0.35\,\mathrm{m}$. Single-bin experiments use the first 20, 40, or 60 objects from the same sequences. The bin base is fixed at $0.40 \times 0.40\,\mathrm{m}$, with respective heights of 0.25, 0.35, and $0.45\,\mathrm{m}$. We use 100 input sequences for each dataset and setting. Implementation details, evaluation protocols, and metric definitions are provided in Appendix E.

4.2 COMPARISON WITH BASELINES

Multi-bin packing. Table 1 shows that MultiPack achieves the highest compactness, the best or tied-best pyramidality, and the lowest mean bin count at all three bin sizes on both AutoStore and MHO. On AutoStore, MultiPack reduces the mean bin count by approximately 0.6%–7.2% relative to PPN-Pack. The gains over HM and DBL are larger: for $0.35\,\mathrm{m}$ bins, the mean bin count decreases from 3.07 to 2.46, a reduction of about 19.9%. On MHO, MultiPack uses approximately 4.3%–12.1% fewer bins than PPN-Pack. Its compactness advantage over PPN-Pack is also more pronounced on MHO: at $0.35\,\mathrm{m}$, the scores are 0.557 versus 0.493, compared with 0.654 versus 0.652 on AutoStore. These results indicate that MultiPack improves packing efficiency over the evaluated geometric heuristics and pretrained placement policy across both datasets, without requiring packing-specific training. Figure 3 compares the methods on AutoStore using the same 80 input objects. MultiPack uses fewer bins in this example, consistent with the aggregate results in Table 1. Additional visualizations for MHO and the single-bin experiments are provided in Appendix F.

Single-bin packing. We also evaluate single-bin packing as a special case of the multi-bin task. Table 2 shows that the same MultiPack framework maintains competitive layout quality: it achieves the highest pyramidality in all six settings and the highest compactness in four. On AutoStore, it



Figure 3: Qualitative comparison of multi-bin packing on PPN-Pack AutoStore using cubic bins with side lengths of 0.25, 0.30, and $0.35\,\mathrm{m}$. All methods and bin sizes use the same 80 objects.

also achieves the highest pyramidality and the lowest mean top-layer object height at all three input sizes. HM and DBL remain competitive with 20 objects on AutoStore but fall behind MultiPack with 40 and 60 objects. This highlights the limits of local heuristics when more objects share a container. In compactness, PPN-Pack performs slightly better only in the AutoStore setting with 60 objects. These results support the use of a unified framework for both single-bin and multi-bin packing. Single-bin visualizations are provided in Appendix F.

4.3 Ablation Study

We conduct ablation experiments on AutoStore in the multi-bin setting. Table 3 compares the full framework with five variants: **Shuffled Attributes:** Randomly permutes complete support–protection–access attribute vectors within each sequence while preserving the attribute distribution. **Object-Level planning:** Uses individual objects instead of groups as the bin-allocation units. The candidate window contains the first $w = 3$ unprocessed objects in the full object sequence.

Table 1: Multi-bin packing results (80 objects per sequence). The best and second-best results are highlighted by color. Column groups indicate cubic bin side lengths. C, P, and N_{bin} denote compactness, pyramidality, and mean nonempty bin count.

Dataset	Method	0.25 m			0.30 m			0.35 m		
		$C\uparrow$	$P\uparrow$	$N_{\text{bin}}\downarrow$	$C\uparrow$	$P\uparrow$	$N_{\text{bin}}\downarrow$	$C\uparrow$	$P\uparrow$	$N_{\text{bin}}\downarrow$
PPN-Pack AutoStore	HM + BBox Order	0.597	0.815	6.98	0.579	0.782	4.40	0.576	0.775	3.07
	DBL + BBox Order	0.597	0.816	6.99	0.579	0.784	4.41	0.580	0.778	3.07
	GPT6-Astra	0.485	0.784	8.72	0.476	0.790	4.26	0.471	0.797	3.20
	PPN-Pack	0.641	0.836	5.44	0.619	0.818	4.11	0.652	0.820	2.65
	MultiPack	0.653	0.847	5.41	0.634	0.818	3.92	0.654	0.832	2.46
Mixed Household Objects (MHO)	HM + BBox Order	0.493	0.751	14.2	0.437	0.695	10.6	0.428	0.667	6.67
	DBL + BBox Order	0.492	0.752	14.2	0.434	0.693	10.7	0.426	0.666	6.73
	GPT6-Astra	0.504	0.763	14.3	0.485	0.757	9.56	0.401	0.753	5.78
	PPN-Pack	0.499	0.765	13.9	0.475	0.736	9.43	0.493	0.725	5.72
	MultiPack	0.523	0.789	13.3	0.528	0.759	8.67	0.557	0.761	5.03

Table 2: Single-bin packing results. The best and second-best results are highlighted by color. Column groups indicate input object counts. C, P, and H denote compactness, pyramidality, and mean top-layer object height (m), respectively.

Dataset	Method	$N=20$			$N=40$			$N=60$		
		$C\uparrow$	$P\uparrow$	$H\downarrow$	$C\uparrow$	$P\uparrow$	$H\downarrow$	$C\uparrow$	$P\uparrow$	$H\downarrow$
PPN-Pack AutoStore	HM + BBox Order	0.512	0.886	0.107	0.578	0.780	0.221	0.573	0.733	0.343
	DBL + BBox Order	0.511	0.886	0.108	0.576	0.786	0.219	0.591	0.746	0.337
	GPT6-Astra	0.430	0.858	0.121	0.415	0.723	0.267	0.523	0.636	0.450
	PPN-Pack	0.489	0.899	0.0898	0.593	0.831	0.158	0.639	0.806	0.253
	MultiPack	0.511	0.936	0.0875	0.613	0.880	0.153	0.637	0.836	0.237
Mixed Household Objects (MHO)	HM + BBox Order	0.447	0.739	0.163	0.470	0.647	0.246	0.484	0.598	0.343
	DBL + BBox Order	0.450	0.736	0.163	0.476	0.644	0.250	0.478	0.592	0.344
	GPT6-Astra	0.427	0.695	0.146	0.494	0.622	0.278	0.514	0.603	0.387
	PPN-Pack	0.464	0.784	0.141	0.515	0.695	0.234	0.519	0.651	0.328
	MultiPack	0.519	0.842	0.140	0.571	0.757	0.238	0.555	0.683	0.336

No Surface Clearance: Sets only the surface-clearance term $c(a) = 0$ in the action score. **No Height Penalty:** Sets only the height weight $\lambda_h = 0$. **No Physics Feedback:** Maintains only the predicted voxel state during native planning. Once planning is complete, the fixed plan is passed to physical simulation for final validation.

Scheduling attributes and grouping. Shuffled Attributes lowers compactness and pyramidality and increases the bin count at all three sizes. Since the attribute distribution is unchanged and only the attribute–object correspondence is permuted, this gain must come from matching LLM-generated attributes to the right objects: with mismatched support–protection relationships, same-group objects lose their reason to be placed together. Object-Level Planning likewise lowers compactness at every bin size. It increases the mean bin count at $0.25\,\text{m}$ and $0.30\,\text{m}$, while matching the full framework at $0.35\,\text{m}$, where both use 2.46 bins on average at the reported precision. Together, these results support the group as the shared unit for both cross-bin allocation and intra-bin scheduling.

Geometric scoring. Removing the surface-clearance term lowers compactness and increases the bin count at all sizes, with pyramidality rising only at $0.25\,\text{m}$. This suggests that the term trades a small amount of immediate stacking quality for free surface that later placements can still use; without it, surfaces are broken up too early. Removing the height penalty causes the largest compactness drop among all variants, and the effect grows with bin size: larger bins are less constrained, so without the height penalty objects tend to be stacked upward too early, wasting horizontal space.

Physics feedback. Replacing online physics feedback with post-hoc validation increases the bin count at all three sizes, and compactness drops at $0.30\,\text{m}$ and $0.35\,\text{m}$. At $0.25\,\text{m}$, pyramidality increases from 0.847 to 0.857, while compactness remains unchanged at 0.653 at the reported pre-

Table 3: Multi-bin ablation results on AutoStore. The best and second-best distinct values are highlighted in red and orange, respectively, with ties determined at the reported precision. Metrics follow the multi-bin comparison experiments.

Method	0.25 m			0.30 m			0.35 m		
	$C \uparrow$	$P \uparrow$	$N_{\text{bin}} \downarrow$	$C \uparrow$	$P \uparrow$	$N_{\text{bin}} \downarrow$	$C \uparrow$	$P \uparrow$	$N_{\text{bin}} \downarrow$
Full	0.653	0.847	5.41	0.634	0.818	3.92	0.654	0.832	2.46
Shuffled Attributes	0.646	0.830	5.62	0.626	0.804	3.95	0.639	0.816	2.55
Object-Level Planning	0.642	0.847	5.65	0.623	0.817	3.98	0.640	0.828	2.46
No Surface Clearance	0.645	0.851	5.67	0.620	0.814	3.95	0.646	0.828	2.51
No Height Penalty	0.636	0.845	5.67	0.607	0.812	4.08	0.605	0.818	2.69
No Physics Feedback	0.653	0.857	5.62	0.612	0.814	3.97	0.631	0.830	2.55

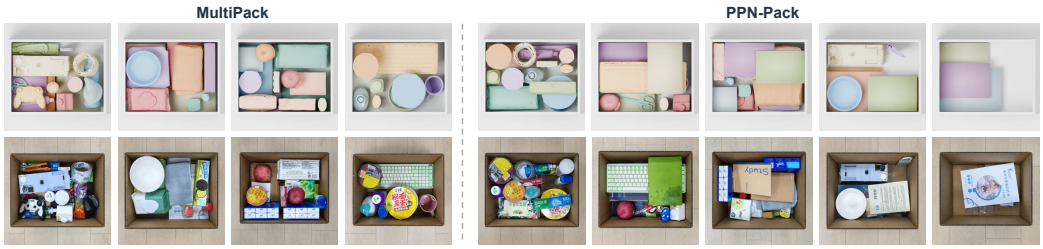

Figure 4: Real-World Packing Demonstration. The top row shows the planned arrangements, and the bottom row shows the corresponding physical arrangements executed by participants.

cision. However, the mean bin count rises from 5.41 to 5.62, indicating that improved pyramidality does not translate into better bin efficiency. Online feedback instead lets the planner correct placements against real physics before committing, and its benefit appears most consistently in bin count.

4.4 Real-World Packing Demonstration

To evaluate whether the generated plans can guide human packing, we conduct a real-world demonstration using 54 everyday objects. We obtain their 3D models through calibrated video capture and Meshy-based reconstruction, and calibrate the model scales using measured object dimensions. MultiPack and PPN-Pack receive the same object models and bins with internal dimensions of $0.40 \times 0.30 \times 0.20\,\text{m}$. Participants then pack the physical objects following the visual guidance generated by each method.

As shown in Figure 4, MultiPack uses fewer bins for the same set of objects, and this advantage is retained during physical execution. This example demonstrates that MultiPack's plans can serve as actionable guidance for real-world packing. The physical arrangements also provide qualitative support for the executability and placement stability of plans generated with geometric and physical constraints. The complete execution process is provided in the supplementary video.

5 Conclusion

We presented MultiPack, a hierarchical framework for irregular 3D multi-bin packing that integrates cross-bin composition planning with physically validated in-bin pose search. The Object–Group–Bin strategy connects allocation with scheduling through LLM-derived priorities, while physics-feedback geometric search uses settled object configurations to guide subsequent placements. Experiments on two datasets show reduced bin usage and competitive or improved layout quality under a shared evaluation protocol. A real-world demonstration with 54 household objects further illustrates how the generated plans can guide human packing. The current framework assumes known geometry and physical parameters and does not account for robotic manipulation constraints. Future work will address these limitations through uncertainty-aware planning and integration with grasp and motion planning.

AI USE STATEMENT

AI tools were used to improve the clarity and readability of the manuscript and assist with literature search. The authors take full responsibility for the final work.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes MultiPack's grouping and cross-bin allocation, geometric search, and physical validation procedures. Appendices A–D provide further details on geometric preprocessing, MILP linearization and bin-count search, scheduling rules, geometric scoring, and physical acceptance criteria. Appendix E documents data preparation and sequence selection, key parameters and simulation thresholds, baseline adaptations, and the shared multi-bin post-processing protocol. It also specifies how object coverage and evaluation metrics are computed and aggregated, providing concrete details for reproducing the reported experiments. Our code will be released publicly soon.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, et al. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. In *International Conference on Learning Representations*, volume 2025, pp. 65205–65268, 2025.

Hanni Cheng, Ya Cong, Weihao Jiang, and Shiliang Pu. Learning to solve class-constrained bin packing problems via encoder-decoder model. In *International Conference on Learning Representations*, volume 2024, pp. 24837–24860, 2024.

Wonje Choi, Jinwoo Park, Sanghyun Ahn, Daehee Lee, and Honguk Woo. Nesyc: A neuro-symbolic continual learner for complex embodied tasks in open domains. In *International Conference on Learning Representations*, volume 2025, pp. 85933–85969, 2025.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024.

Palak Gupta and Shanmuganathan Raman. Differentiable packing of irregular 3d objects with adaptive container estimation. *arXiv preprint arXiv:2606.16333*, 2026.

Yilun Hao, Yang Zhang, and Chuchu Fan. Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming. In *International Conference on Learning Representations*, volume 2025, pp. 64335–64391, 2025.

Sichao Huang, Ziwei Wang, Jie Zhou, and Jiwen Lu. Planning irregular object packing via hierarchical reinforcement learning. *IEEE Robotics and Automation Letters*, 8(1):81–88, 2022a.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022b.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022c.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Korhan Karabulut and Mustafa Murat İnceoğlu. A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In *International Conference on Advances in Information Systems*, pp. 441–450. Springer, 2004.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023a.

Huwei Liu, Li Zhou, Jianglong Yang, and Junhui Zhao. The 3d bin packing problem for multiple boxes and irregular items based on deep q-network: H. liu et al. *Applied Intelligence*, 53(20): 23398–23425, 2023b.

Yuecheng Liu, Dafeng Chi, Shiguang Wu, Zhanguang Zhang, Yuzheng Zhuang, Bowen Yang, He Zhu, Lingfeng Zhang, Pengwei Xie, David Gamaliel Arcos Bravo, et al. Omnieva: Embodied versatile planner via task-adaptive 3d-grounded and embodiment-aware reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 76044–76095, 2026.

Jia-Hui Pan, Ka-Hei Hui, Xiaojie Gao, Shize Zhu, Yun-Hui Liu, Pheng-Ann Heng, and Chi-Wing Fu. Sdf-pack: Towards compact bin packing with signed-distance-field minimization. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10612–10619. IEEE, 2023.

Jia-Hui Pan, Xiaojie Gao, Ka-Hei Hui, Shize Zhu, Yun-Hui Liu, Pheng-Ann Heng, and Chi-Wing Fu. Ppn-pack: Placement proposal network for efficient robotic bin packing. *IEEE Robotics and Automation Letters*, 9(6):5086–5093, 2024.

Gojko Perovic, Nuno Ferreira Duarte, Atabak Dehban, Gonçalo Teixeira, Egidio Falotico, and José Santos-Victor. Herb: Human-augmented efficient reinforcement learning for bin-packing. *arXiv preprint arXiv:2504.16595*, 2025.

Tianhao Qin, Connor McCann, Berk Calli, and Jing Xiao. Gripper-aware automatic dense packing of irregular objects. *arXiv preprint arXiv:2609.22062*, 2026.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning. *arXiv preprint arXiv:2307.06135*, 2023.

Xiang Shu, Hong Qian, Xingyu Lu, JUN ZHOU, Aimin Zhou, Yang Yu, et al. Llmopt: Learning to define and solve general optimization problems from scratch. In *International Conference on Learning Representations*, volume 2025, pp. 101580–101606, 2025.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2998–3009, 2023.

Nikolaos Theodoropoulos, Dionisis Andronas, Emmanouil Kampourakis, and Sotiris Makris. Automated pallet loading of irregularly shaped objects: A deep reinforcement learning and multi-criteria optimization method. *Journal of Manufacturing Systems*, 80:916–932, 2025.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987, 2023.

Fan Wang and Kris Hauser. Stable bin packing of non-convex 3d objects with a robot manipulator. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8698–8704. IEEE, 2019.

Fan Wang and Kris Hauser. Dense robotic packing of irregular and novel 3d objects. *IEEE Transactions on Robotics*, 38(2):1160–1173, 2021.

Hang Zhao, Zherong Pan, Yang Yu, and Kai Xu. Learning physically realizable skills for online packing of general 3d shapes. *ACM Transactions on Graphics*, 42(5):1–21, 2023.

A GEOMETRIC PREPROCESSING

Given a voxel size Δ, we discretize the bin into

$$(n_x, n_y, n_z) = (\lfloor L_x/\Delta \rfloor, \lfloor L_y/\Delta \rfloor, \lfloor L_z/\Delta \rfloor).$$

Each object is voxelized under its preset orientations. For each orientation, we record its physical dimensions, bottom and top profiles, and surface voxels. Surface voxels are obtained by subtracting the six-neighbor erosion of the occupancy from the original occupancy.

Let Θ_i^{fit} contain the orientations of object i that fit an empty bin in both physical and discrete dimensions. The volume proxy used for coarse allocation is

$$v_i = \min_{\theta \in \Theta_i^{\mathrm{fit}}} \Delta^3 |\mathcal{V}_i^\theta|, \tag{A1}$$

where $\mathcal{V}_i^\theta$ is the occupied voxel set under orientation θ. Taking the minimum reduces the influence of orientation-dependent voxelization differences on allocation. Local candidate scoring instead uses the occupied volume of the candidate's own orientation. If Θ_i^{fit} is empty, the object is recorded as having no fitting orientation among those tested. This does not imply that no feasible pose exists in the continuous pose space.

B MILP DETAILS

B.1 LINEARIZATION AND OBJECTIVE PROPERTIES

For a fixed candidate bin count K, we introduce a nonnegative auxiliary variable o_b for each bin and solve

$$\begin{aligned} \min_{\mathbf{x},\mathbf{o}} \quad & \sum_{b=1}^{K} o_b \\ \text{subject to} \quad & o_b \geq \frac{V_b - C}{C}, \quad o_b \geq 0, \qquad \forall b = 1, \ldots, K, \\ & \mathbf{x} \text{ satisfies Eq. (1)}, \quad x_{gb} \in \{0, 1\}. \end{aligned} \tag{B1}$$

At an optimum, $o_b = [V_b - C]_+/C$, so this linear formulation is equivalent to minimizing $\Omega(\mathbf{x})$. We use the SciPy MILP interface. Solutions returned at the time limit are retained only after passing integrality and constraint checks, with an integrality tolerance of 10^{-6}.

The objective is zero whenever all assigned bin volumes are at most C. Such assignments are equally optimal under this surrogate: the objective does not further balance bin loads or rank their geometric realizability. Volume overflow is penalized rather than forbidden, allowing a coarse assignment even when it cannot be eliminated for the current grouping and bin count.

B.2 BIN-COUNT SEARCH

For a nonempty planning set with at least one new bin available, let K_{avail} be the maximum number of new bins allowed in the current round, and define $\bar{K} = \min(K_{\mathrm{avail}}, |\mathcal{G}|)$. The initial candidate bin count and a lower bound on overflow under the fixed grouping are

$$\begin{aligned} K_0 &= \min\left(\bar{K}, \max\left\{1, \left\lceil \sum_{g \in \mathcal{G}} \min(V_g/C, 1) - \varepsilon \right\rceil \right\}\right), \\ \Omega_{\mathrm{irr}} &= \sum_{g \in \mathcal{G}} [V_g/C - 1]_+, \end{aligned} \tag{B2}$$

where $\varepsilon = 10^{-8}$ is a numerical tolerance. The lower bound accounts for groups whose individual volumes exceed the bin volume. Since each group must remain intact, their overflow cannot be eliminated by adding bins.

If the current optimal overflow is proven to exceed $\Omega_{\text{irr}} + \varepsilon$ and $K < \bar{K}$, we increase the candidate bin count by one and solve again. All candidate bin counts share the planning budget for the current round, which defaults to 30 seconds. Group membership remains fixed within the round. If a later solve returns no valid assignment, a previously validated solution may be retained.

Bins in the resulting coarse plan are processed in descending order of assigned volume. This procedure minimizes a volume-overflow surrogate for each tested bin count; it does not guarantee the minimum number of bins in the final packing.

C Geometric Search Details

Scheduling rules. Groups are processed in descending order of mean support priority, with ties broken by the ascending sum of mean protection and access priorities. Within each group, objects are ordered by descending support priority and then volume. The candidate window contains the first $w = 3$ unprocessed objects, or all remaining members if fewer than three are available.

Weighted surface clearance. For each scene-grid voxel $\mathbf{u}$, we compute discrete clearances to existing occupancy, the bin floor, and the side walls, denoted by $d_t^{\text{occ}}(\mathbf{u})$, $d^{\text{floor}}(\mathbf{u})$, and $d^{\text{wall}}(\mathbf{u})$, respectively. All distances are expressed in physical length units. Clearance to existing occupancy is obtained from a Euclidean distance transform, reduced by one voxel length and clamped at zero to approximately account for voxel-center distances. Floor and wall clearances are computed directly from grid coordinates. In an empty bin, occupancy clearance is set to the truncation distance. The open top is excluded from clearance scoring.

Each source distance is truncated before taking the weighted average:

$$d_t(\mathbf{u}) = \sum_{k \in \{\text{floor},\text{occ},\text{wall}\}} \lambda_k \min\{d_t^k(\mathbf{u}), r_c\}, \qquad \lambda_k \geq 0, \quad \sum_k \lambda_k = 1. \tag{C1}$$

Here, the floor and wall distances do not depend on t. We use $r_c = 0.08\,\text{m}$ and weights 0.50, 0.35, and 0.15 for the floor, existing occupancy, and side walls, respectively. Distances are computed from the scene before adding the candidate, so its own occupancy is excluded.

Let $\mathcal{S}(a)$ denote the candidate's surface voxels in scene coordinates. The mean normalized surface clearance is

$$c(a) = \frac{1}{|\mathcal{S}(a)|} \sum_{\mathbf{u} \in \mathcal{S}(a)} \frac{d_t(\mathbf{u})}{r_c} \in [0, 1]. \tag{C2}$$

The normalized weights and per-source truncation ensure $d_t(\mathbf{u}) \in [0, r_c]$. Smaller $c(a)$ indicates closer average proximity to surrounding surfaces. This is a geometric proxy, not a measure of contact area or load-bearing capacity. Distances beyond r_c incur no additional clearance penalty, whereas the separate height term continues to penalize higher placements.

Connection to heightmap minimization. For candidates of equal volume, omitting the clearance and height terms and assuming $E(a) \geq V(a)$ reduces the geometric score to

$$q_{\text{geom}}(a) = \frac{V(a)}{E(a)}. \tag{C3}$$

Maximizing this score therefore preserves the ranking induced by the height-envelope term in HM. This correspondence does not include the additional horizontal-position preference in the original HM heuristic.

D Physical Validation

This appendix specifies the descent, settling, and geometric checks used in MultiPack's native placement trials (Section 3.4). The simulation settings and thresholds used in the experiments are listed in Table 4 in Appendix E.1.

Descent checks. A descent is rejected if the candidate penetrates an existing object beyond the allowed tolerance or causes a contacted object to exceed the displacement or velocity thresholds.

Settling checks. The scene is checked every eight simulation steps. All objects must satisfy the velocity thresholds and have contact with a positive normal force. The system retains poses from five consecutive checks. Settling is considered converged only if the position and orientation changes at each of the last four checks, relative to the first, remain below the corresponding thresholds. Contact normals need not point upward, allowing friction at side-wall contacts to contribute to stability.

Geometric acceptance. The candidate's maximum penetration depth at any contact must remain within tolerance, and all objects must satisfy the bin boundary conditions after settling. The side walls and floor use a geometric tolerance of $\max(0.008\,\mathrm{m}, 0.4\Delta)$, while no object may extend above the bin height. Previously placed objects may undergo natural settling if the resulting scene passes validation. Their final displacement from the previous poses is recorded for diagnostics and is not, by itself, a rejection criterion. These conditions define acceptance within a finite simulation window; they do not guarantee long-term stability, protection from damage, or safety during transport.

E Experimental Details

E.1 Additional Settings

Data preparation. AutoStore is our main evaluation dataset. It uses watertight meshes with convex-decomposition collision assets. We evaluate the first 100 released test sequences, each containing 80 objects. MHO provides a supplementary evaluation on a different object collection. It contains 78 local YCB asset templates and 120 RoboTwin templates. The 100 source sequences are generated with seed 20260919 and contain 100 instances each. The 80-object evaluation sequences in Section 4.1 use their first 80 instances. Repeated templates represent distinct instances. For templates that do not fit a $0.40\,\mathrm{m}$ reference cube in any tested yaw orientation, we scale the longest side in the canonical pose to $0.20\,\mathrm{m}$. This affects 29 templates. All methods use the same processed assets. Single-bin experiments use the first 20, 40, or 60 objects from the same sequences, with bin dimensions as specified in Section 4.1.

MultiPack settings. MultiPack uses the same configuration across both datasets and all single-bin and multi-bin settings. It uses $0.0125\,\mathrm{m}$ search voxels, cached scheduling attributes from GPT-5.6-sol, and PyBullet for physical validation. No packing-specific training is required. MultiPack tests yaw angles of $0°$, $90°$, $180°$, and $270°$. The grouping threshold is $\tau = 0.5$, the group size limit is $m = 9$, and the MILP budget is $30\,\mathrm{s}$ per round. We use $r_c = 0.08\,\mathrm{m}$, a candidate window of $w = 3$ objects, and at most $J = 4$ action trials per selected object. The height weight is $\lambda_h = 6$, and object selection uses $s(a) = V(a)q_{\mathrm{geom}}(a)$. Table 4 reports the PyBullet settings and thresholds used for native physical validation in all experiments. The corresponding acceptance checks are specified in Appendix D.

Baseline adaptation. For the geometric and learned placement baselines, the multi-bin adapter starts from descending bounding-box volume order. PPN-Pack retains its native five-object reordering buffer and uses the same publicly available pretrained weights on both datasets, without retraining. This setting evaluates the transfer of the pretrained policy to a different object collection. A new bin is opened when a full pass over the remaining objects yields no placement; PPN-Pack's scene state is reset for each new bin. Single-bin experiments compare the in-bin solvers without cross-bin allocation.

E.2 Shared Multi-Bin Evaluation Protocol

All methods undergo the same post-processing procedure on both datasets, including layout revalidation and completion. Within each dataset and bin-size setting, all methods use the same processed assets and input sequences. The multi-bin results in Table 1 are computed from the final validated layouts produced by this procedure.

Table 4: Experimental settings for MultiPack's native physical validation. These values are shared across datasets and task settings. The shared multi-bin evaluation procedure is specified separately in Appendix E.2.

Parameter	Value
Simulation frequency	$240\,\mathrm{Hz}$
Solver iterations	110
Initial offset above the target	$0.03\,\mathrm{m}$
Maximum local descent steps	100
Maximum whole-bin settling steps	1,200
Linear speed threshold	$0.04\,\mathrm{m/s}$
Angular speed threshold	$0.3\,\mathrm{rad/s}$
Position change threshold within the settling window	$0.001\,\mathrm{m}$
Orientation change threshold within the settling window	1°
Penetration tolerance into existing objects during descent	$0.0015\,\mathrm{m}$
Displacement threshold for existing objects during descent	$0.0015\,\mathrm{m}$
Candidate penetration tolerance after settling	$0.003\,\mathrm{m}$

Layout revalidation. Each bin undergoes gravitational settling and boundary revalidation with a $3\,\mathrm{mm}$ containment tolerance. Objects outside the bin or failing stability checks are removed from the validated layout and returned to the completion queue. If the remaining contents still fail to settle as a whole, all remaining objects in that bin are also returned to the queue.

Oversized exemption. An oversized exemption excludes an instance from the eligible-object count but not from the total requested count. Starting from the processed asset's canonical pose, let $d_{i,k}(\theta)$ denote the extent of its continuous collision-geometry axis-aligned bounding box along axis $k \in \{x, y, z\}$ after a yaw rotation $\theta \in \Theta = \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$. Instance i is exempt if and only if

$$\forall \theta \in \Theta, \quad \exists k \in \{x, y, z\} : \; d_{i,k}(\theta) > L_k + \epsilon, \qquad \epsilon = 10^{-7}\,\mathrm{m}.$$

Thus, at every tested orientation, at least one bounding-box extent exceeds the corresponding bin dimension by more than ϵ. This test uses continuous collision-geometry bounds rather than rounded voxel dimensions. It determines size eligibility under the prescribed orientations, not feasibility under arbitrary 3D rotations. The exemption rule is independent of packing success and yields the same exempt set across methods for identical processed assets, canonical poses, bin dimensions, and input instances. Search failure, insufficient remaining space, and instability do not qualify for exemption. Passing the size test establishes eligibility but does not guarantee a stable placement.

Completion. All missing, non-exempt instances enter a common gravity-based completion procedure. The procedure first attempts placement in existing bins, using a $0.01\,\mathrm{m}$ heightmap to select a region with the lowest mean height. The object is released $0.03\,\mathrm{m}$ above the proposed placement and allowed to settle. Completion tests the same four yaw angles used for the oversized check. New bins are opened when existing bins cannot accept an object, subject to a limit of 50 nonempty bins. Objects rejected during revalidation may be returned to the queue. Completion does not guarantee that all eligible objects are packed.

Relation to native packing. The shared procedure uses its own settling and boundary checks, rather than the native solvers' scoring or acceptance and rollback rules. All multi-bin metrics are computed from the final validated layouts, and N_{bin} includes bins added during completion. The reported results therefore characterize each native method combined with the shared post-processing procedure.

E.3 Evaluation Metrics and Aggregation

Layout metrics. We measure final layouts on a shared $\Delta = 0.025\,\mathrm{m}$ voxel grid. Let V denote the union volume of occupied voxels, $h(x, y)$ the height of the highest occupied surface above the bin floor in grid column (x, y), and $h_{\max} = \max_{x,y} h(x, y)$. Empty columns have height zero. We

compute compactness and pyramidality as

$$C = \frac{V}{L_x L_y h_{\max}}, \qquad P = \frac{V}{\Delta^2 \sum_{x,y} h(x,y)}. \tag{E1}$$

Higher values of C and P indicate greater compactness and pyramidality, respectively. We also report the number of nonempty bins, N_{bin}, for multi-bin packing and the mean top-layer object height, H, for single-bin packing.

Object coverage. For each evaluated multi-bin trajectory, let M denote the number of requested instances, S the number exempted by the size rule in Appendix E.2, and A the number of finally accepted instances among the remaining $M - S$ eligible inputs. Repeated copies of an asset count as separate instances. Exempt instances are excluded from A. Eligible instances that remain unplaced or fail final validation remain in the eligible-object denominator. Overall and eligible-object coverage are defined as

$$R_{\text{all}} = \frac{A}{M}, \qquad R_{\text{eligible}} = \begin{cases} A/(M-S), & M > S, \\ 1, & M = S. \end{cases}$$

The value of R_{eligible} is set to one by convention when no eligible inputs remain. An eligible-object coverage of 100% indicates that all non-exempt instances were accepted; it does not necessarily mean that all requested instances were packed.

Multi-bin aggregation. For a trajectory containing nonempty bins with occupied volumes V_b, we compute volume-weighted compactness and pyramidality:

$$C_{\text{traj}} = \frac{\sum_b V_b C_b}{\sum_b V_b}, \qquad P_{\text{traj}} = \frac{\sum_b V_b P_b}{\sum_b V_b}.$$

An empty valid layout receives zero compactness and pyramidality.

Single-bin height. For single-bin evaluation, a top-layer object is one that forms the highest occupied surface in at least one horizontal grid column. The height metric H is the mean of these objects' individual maximum occupied heights.

Aggregation across trajectories. Within each setting, valid trajectories are averaged equally, including both complete and validated partial packings. Errors and missing runs are excluded from these averages and recorded separately. Bin counts and layout metrics summarize the accepted layouts; they do not, by themselves, establish that all eligible inputs were packed. Representative AutoStore multi-bin layouts are shown in Figure 3. Additional visualizations for MHO and single-bin packing are provided in Appendix F.

F More Experiment Results

F.1 Comparison with baselines

Figure 6 shows a qualitative comparison of multi-bin packing on MHO dataset. Figure 5 shows a qualitative comparison of single-bin packing on AutoStore and MHO.

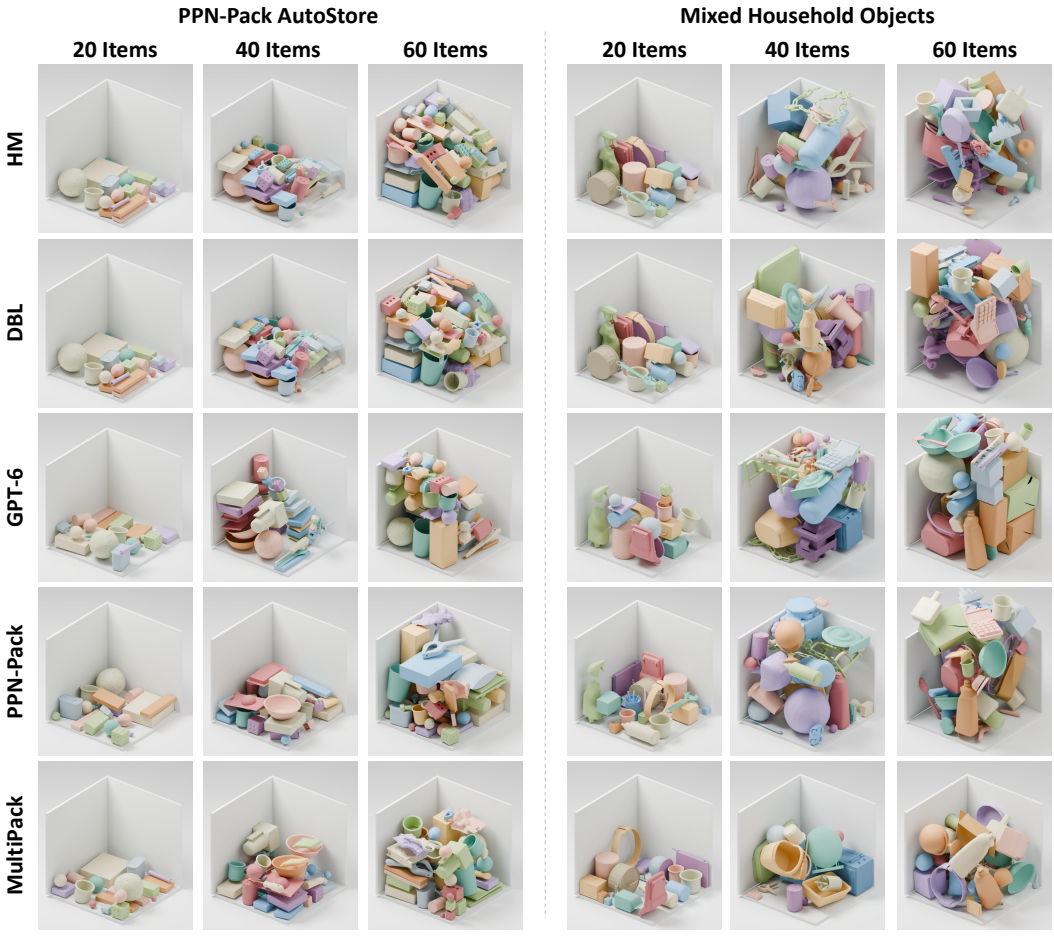

Figure 5: Qualitative comparison of single-bin packing on PPN-Pack AutoStore and Mixed Household Objects (MHO) with 20, 40, and 60 objects.



Figure 6: Qualitative comparison of multi-bin packing on the Mixed Household Objects(MHO) dataset using cubic bins with side lengths of 0.25, 0.30, and $0.35\,\mathrm{m}$.