PLAN2POSE: PRESCIENT DOWNWARD REFINEMENT OF SYMBOLIC PLANS OVER LONG HORIZONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Task and motion planning uses symbolic abstractions to determine a high-level, symbolic action plan, and then refines the actions into continuous robot configurations. This downward refinement typically makes use of hand-designed samplers, constraint solvers, or predicate-specific learned models, which require hand-crafted predicate-specific supervision or losses. Learned models are myopic and only consider the current configuration's requirements, producing refinements that may cause subsequent actions to be unrealizable. This is particularly problematic in long-horizon tasks that require many sequential refinements. We propose a refinement learning method that is prescient where needed, i.e., it learns the relevant aspects of the remaining plan instead of myopically focusing on the current action. In contrast to previous approaches, we use a predicate-agnostic approach that avoids hand-crafted predicate-specific losses.

Given a PDDL plan and an object-segmented point cloud of the initial scene, a transformer embeds the whole action sequence. For a given state, we learn placement fields that are composed to produce an SE(3) pose that satisfies all object relations described in the state. These placement fields are learned without predicate-specific losses, evaluators, or hand-coded geometric semantics, but rather purely from learning to replicate the distributions seen in demonstrations. Our approach, PLAN2POSE, is trained on synthetic rearrangement plans of at most nine states and outperforms existing baselines on target-atom satisfaction, especially on OOD plan lengths. Under autoregressive rollout on 200 plans of length 17–20, PLAN2POSE satisfies 95.5% of state atoms in the final state, compared to 58.4–76.8% achieved by learned baselines, demonstrating substantially more robust long-horizon refinement. PLAN2POSE transfers zero-shot to real point clouds, and fine-tuning adds new predicates without forgetting.

1 INTRODUCTION

Autonomous robots operate in high-dimensional continuous spaces, but also have to carry out complex tasks that require long-horizon planning. *Task and motion planning* (TAMP) (Srivastava et al., 2014; Garrett et al., 2021; Kaelbling & Lozano-Pérez, 2011) decomposes the problem into two layers: a symbolic planner determines an abstract plan that achieves the long-term goal, and a motion planner or a learned policy realizes each action in the continuous space. The interface between the two is *refinement*, which translates symbolic actions into continuous parameters that satisfy the high-level constraints of the plan. We consider refinement at the level of object poses: a refinement of a symbolic plan assigns to each action a geometric state, i.e., a pose for every object, that satisfies the symbolic state after that action.

Two properties determine whether a plan can be refined action by action. A symbolic plan is *refinable* if it admits a refinement; the *downward refinement property* requires this of every plan (Bacchus & Yang, 1994). A refinable plan is *serializable* if every refinement of its first action can be extended to a refinement of the remaining plan, which must in turn be serializable (Kaelbling & Lozano-Pérez, 2011), adapting the serializable subgoals of Korf (1987). Refinability does not imply serializability: placing a small block at the center of a larger one is a valid refinement of the first action, but leaves no room for a second block the plan places next (cf. Lagriffoul et al., 2014). A plan that is not refinable must be replaced by searching over plan skeletons (Srivastava et al., 2014; Garrett

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.

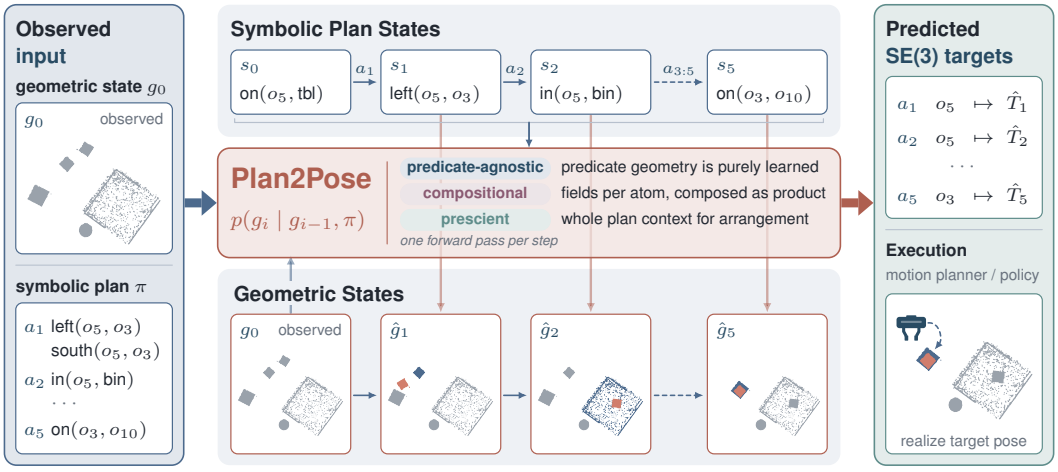

Figure 1: Overview of PLAN2POSE. Given an observation of the initial scene (3 cubes, 1 bin, and 1 ball) in the form of a segmented point cloud and a symbolic plan, PLAN2POSE refines the symbolic states (here shortened into just the visualized atoms) into geometric states by providing the $SE(3)$ target poses of the objects that were moved. These new poses can then be realized for each step with a motion planner or a learned policy. Moved objects are apricot, relative objects are blue.

et al., 2020). A plan that is refinable but not serializable may still fail if an earlier action is refined such that the remaining plan admits no refinement. Classical systems recover by backtracking over earlier continuous parameters, at a worst-case cost exponential in the plan length (Lagriffoul et al., 2014), by optimizing all parameters jointly (Toussaint, 2015), or by keeping goal-relevant regions clear (Kaelbling & Lozano-Pérez, 2011), all with manually specified constraint models or samplers.

Learned feasibility models decide *which* plan to refine (Wells et al., 2019; Yang et al., 2023a) and are complementary to our work. Learned samplers and placement models decide *how* (Kim et al., 2019; Wang et al., 2021; Simeonov et al., 2023; Liu et al., 2023; Kim et al., 2024), but condition each placement only on the current action or state. They are therefore myopic: they choose among all refinements of the current action, not among those from which the remaining plan stays refinable. Non-myopic learned refiners (Yang et al., 2023b; Mishra et al., 2023) sample the parameters of the whole plan jointly, but compose diffusion models trained per constraint type or per skill and sample iteratively at inference time (Section 2).

We propose PLAN2POSE, a learned intermediate layer between the symbolic planner and the motion planner. Given an object-segmented point cloud of the initial scene and a symbolic plan, PLAN2POSE predicts for each action the target SE(3) pose of every moved object. Since each pose is conditioned on the whole plan, the poses carry the long-horizon information, and the motion planner or policy that reaches them can operate myopically. A transformer encodes the plan as a sequence of atom additions and deletions together with per-object shape features. For the object to be placed, every atom that mentions it yields a placement field in the frame of the atom's reference object, and the fields are composed as a product of experts, so that each additional atom narrows the admissible region.

Predicates are opaque symbols: their geometry is learned from labelled demonstrations with a single loss on the composed prediction. PLAN2POSE thus amortizes downward refinement: it refines the plan autoregressively with one forward pass per step and without backtracking.

In summary, we make the following contributions:

- We formulate downward refinement as a conditional distribution over object poses given the whole symbolic plan, whose support should be restricted to refinements from which the remaining plan stays refinable (Section 3).
- We present PLAN2POSE, a single model that encodes the whole plan with a transformer and composes per-atom placement fields as a product of experts. It requires no backtracking,

manually defined constraints, samplers, object models, or predicate-specific losses, and acquires new predicates by fine-tuning (Section 4).

- We show that PLAN2POSE, trained on plans of at most nine states, satisfies 95.5 % of final-state atoms on plans of 17–20 states, compared to 58.4 %–76.8 % for the baselines, and that, after fine-tuning, its placements anticipate constraints of later steps in distribution (Section 5).
- We show that PLAN2POSE transfers zero-shot to real point clouds, acquires new predicates without forgetting, and refines each step in 30 ms, 18–30 times faster than diffusion-based refiners (Section 5).

2 RELATED WORK

Refinement in TAMP. Most TAMP systems replace unrefinable plan skeletons by feeding geometric failures back to the task planner (Srivastava et al., 2014; Dantam et al., 2018; Garrett et al., 2020). For plans that are refinable but not serializable, they backtrack over earlier continuous choices, pruned by constraint propagation and culprit detection (Lagriffoul et al., 2014; Lagriffoul & Andres, 2016; Bidot et al., 2017), optimize all continuous parameters jointly (Toussaint, 2015), or choose values that keep goal-relevant regions clear (Kaelbling & Lozano-Pérez, 2011). All of these rely on hand-specified constraint models, samplers, or geometric predicates.

Learned feasibility and samplers. Learned feasibility models discard unrefinable plans early by classifying motion feasibility (Wells et al., 2019), predicting action sequences from a scene image (Driess et al., 2020), localizing infeasible subsets of a nonlinear program (Ortiz-Haro et al., 2023), or exploiting that no extension of an infeasible prefix is feasible (Noseworthy et al., 2021). PIGINet (Yang et al., 2023a) is architecturally closest to PLAN2POSE: it encodes plan skeleton, goal, and scene with a transformer, but outputs a feasibility score rather than a refinement. Within a fixed skeleton, learned heuristics, samplers, and operators (Chitnis et al., 2016; Kim et al., 2019; Wang et al., 2021; Silver et al., 2021; 2022; 2023) and diffusion samplers (Fang et al., 2024) reduce manual specification, but are conditioned on the current operator and state, so a search-then-sample loop is retained at test time.

Non-myopic and anticipatory refinement. Diffusion-CCSP (Yang et al., 2023b) trains one diffusion model per constraint type and composes them over the constraint graph of the whole plan. Generative skill chaining (Mishra et al., 2023) trains one diffusion model per skill and samples all skill parameters jointly; relational constraints such as leaving room for an object placed later must be added as hand-specified likelihoods. Generative factor chaining (Mishra et al., 2024) adds factors between objects, again composed from separately trained diffusion models. All three require models and data per constraint type or skill and sample iteratively, whereas PLAN2POSE trains one model with one loss and places each object in one forward pass. Deep affordance foresight (Xu et al., 2021) and anticipatory TAMP (Dhakal et al., 2024) use learned foresight to choose among actions or candidate plans, and NeRP (Qureshi et al., 2021) chooses its own sequence of rearrangement moves towards a goal scene; none of them refines a given plan.

Relational placement. StructFormer and StructDiffusion (Liu et al., 2022; 2023) generate multi-object arrangements from language, RPDiff (Simeonov et al., 2023) diffuses relative poses, AnyPlace (Zhao et al., 2025) combines a vision-language model with a local pose predictor, and GOPLA (Zhong et al., 2025) distills symbolic states into placement poses via 3D fields. LINGO-Space (Kim et al., 2024) composes per-relation distributions as a mixture, which widens the admissible region with each relation, whereas Gkanatsios et al. (2023) compose one energy-based model per predicate. These methods ground a single target state and do not condition on the rest of a plan.

3 PROBLEM FORMULATION

We first define symbolic states and actions, then geometric states, relate the two through a labelling function, and finally state the refinement problem for a given plan.

Symbolic States and Actions. We consider a finite set of K objects $\mathcal{O}$ and a finite set of predicate symbols Pred, each with an arity. A *ground atom* $p(o_1, \ldots, o_k)$ consists of a predicate $p \in \mathrm{Pred}$ of arity k and objects $o_1, \ldots, o_k \in \mathcal{O}$, e.g., $on(o_1, o_3)$. We write Φ for the finite set of all ground atoms over Pred and $\mathcal{O}$. A *symbolic state* $s \subseteq \Phi$ is the set of atoms that are true in the state; all other atoms are false. All predicates in our domains are binary, but the method does not depend on this. For an atom $p(o_1, o_2)$, we call o_1 its *subject* and o_2 its *reference object*. A (ground) action a has a precondition $\mathrm{pre}(a) \subseteq \Phi$, positive effects $\mathrm{add}(a) \subseteq \Phi$, and negative effects $\mathrm{del}(a) \subseteq \Phi$. Action a is *applicable* in state s if $\mathrm{pre}(a) \subseteq s$, and applying it yields the successor state $s' = (s \setminus \mathrm{del}(a)) \cup \mathrm{add}(a)$. A *plan* $\pi = (a_1, \ldots, a_n)$ is applicable in an initial state s_0 if it induces a state sequence $s_0, s_1, \ldots, s_n$ such that each a_i is applicable in s_{i-1} and s_i is the result of applying a_i in s_{i-1}. We assume that π is given by a symbolic planner and achieves the task goal in s_n.

Geometric States. Each object $o \in \mathcal{O}$ is rigid, with its shape given as a segmented point cloud $C_o \subset \mathbb{R}^3$ in the object frame. A *geometric state* $g \colon \mathcal{O} \to \mathrm{SE}(3)$ assigns a pose to each object.

Connecting Symbolic and Geometric States. Each predicate $p \in \mathrm{Pred}$ of arity k is interpreted by a geometric test h_p that decides, based on the shapes $C_{o_1}, \ldots, C_{o_k}$, whether p holds for objects $o_1, \ldots, o_k$ placed at poses $T_1, \ldots, T_k$. We assume that objects are represented by point clouds, which capture object geometry directly from depth sensing, without requiring object-specific geometric models and are commonly used in robotics. For example, h_{on} holds for o_1, o_2 at poses T_1, T_2 if o_1 rests on the top surface of o_2. A ground atom $\phi = p(o_1, \ldots, o_k)$ holds in g, written $g \models \phi$, if h_p holds for $o_1, \ldots, o_k$ at poses $g(o_1), \ldots, g(o_k)$. This induces the *labelling function* $l \colon \mathrm{SE}(3)^{\mathcal{O}} \to 2^{\Phi}$ with $l(g) = \{\phi \in \Phi \mid g \models \phi\}$, which maps a geometric state to its symbolic abstraction. The initial scene is a geometric state g_0 with $s_0 = l(g_0)$. The *satisfying region* of a symbolic state s is $\mathcal{G}(s) = \{g \in \mathrm{SE}(3)^{\mathcal{O}} \mid l(g) = s\}$.

Refinement as a Conditional Distribution Each action a moves a set of objects $mv(a) \subseteq \mathcal{O}$. For a plan $\pi = (a_1, \ldots, a_n)$, a geometric state g_i is *valid for step* i given g_{i-1} if $g_i \in \mathcal{G}(s_i)$ and $g_i(o) = g_{i-1}(o)$ for all $o \notin mv(a_i)$. A *refinement* of π from the initial scene g_0 is a sequence of geometric states $g_1, \ldots, g_n$ such that each g_i is valid for step i given g_{i-1}. The *refinement problem* is to find a refinement of π from g_0. We approach this problem by modelling refinement as a conditional distribution over geometric states given the initial scene and the whole plan,

$$p(g_{1:n} \mid g_0, \pi) = \prod_{i=1}^{n} p_i(g_i \mid g_{i-1}, \pi),$$

where $p_i(g_i \mid g_{i-1}, \pi) > 0$ only if g_i is valid for step i given g_{i-1} and there are $g_{i+1}, \ldots, g_n$ such that g_j is valid for step j given g_{j-1} for all $j \in \{i+1, \ldots, n\}$. Conditioning on the whole plan rather than only on a_i makes refinement non-myopic, and modelling a distribution rather than a single configuration represents the multiple valid placements that typically exist.

4 PLAN2POSE

We parameterize $p(g_i \mid g_{i-1}, \pi)$ as a single transformer-based neural network (Figure 2) which represents planar position and height as distributions and decodes a single pose from them. For a given step i, PLAN2POSE takes the segmented point clouds of the initial scene, the whole plan π, and the previous poses g_{i-1} as input, and predicts a pose for every object. The model is rolled out autoregressively over the whole plan to achieve the desired downward refinement. PLAN2POSE first encodes the scene and plan as a token sequence and processes it with a transformer encoder. Second, every atom relevant to an object at step i is decoded by one shared decoder into a placement field around its reference object, with additional readouts for rotation and z-placement. Third, the placement fields are composed as a product of experts, so each atom narrows the admissible region, the rotations are averaged with the same weights, and the pose is read off the composed field. We train PLAN2POSE end-to-end with a single objective on the composed prediction, instead of per-predicate losses or hand-coded geometric constraints.

Plan and scene encoding. To process the scene and the plan, we encode them as a sequence of tokens, which is processed by the transformer(Vaswani et al., 2017) encoder. Each object o_k

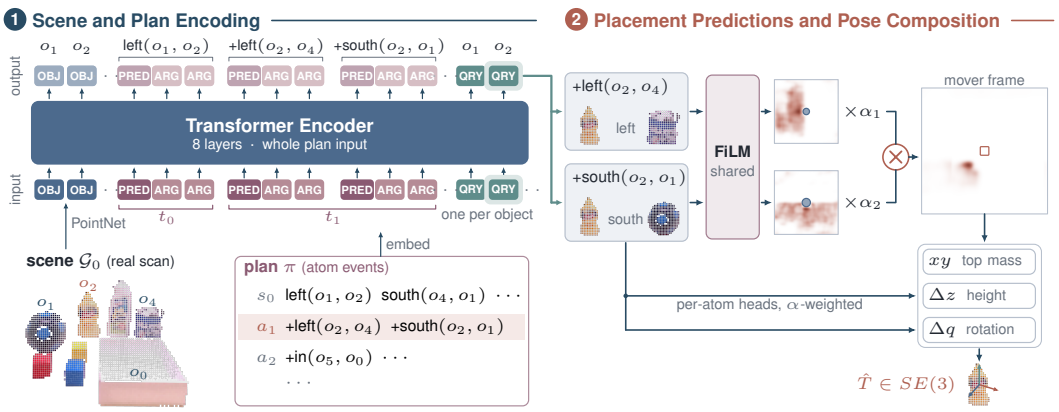

Figure 2: Architectural overview of PLAN2POSE. (1) The scene and the whole plan are encoded as a sequence of tokens, which is processed by a transformer. (2) The query embeddings of the moved objects are passed to the prediction head, which, using a shared FiLM decoder, translates every atom relevant to the current step into a placement field centered on its reference object. The fields are composed as a product of experts. The z-placement and rotation are predicted separately. The planar position is decoded from the composed field as the probability-weighted mean of the smallest set of cells that holds half of its mass.

is represented by its segmented point cloud C_k from the given initial observation. We center the point cloud on its mean and encode it with a shared PointNet (Qi et al., 2017) network. A learned projection of the initial position to the embedding size is added to the PointNet feature, so the object token carries the shape and the initial position of the object.

We construct atom-level tokens to encode the symbolic plan. Instead of encoding every state $s_0, \ldots, s_n$, we encode the initial state plus the changes at each state in the plan, i.e., the deletions $\mathrm{del}(a_i)$ followed by the additions $\mathrm{add}(a_i)$ of each action a_i. Each n-ary atom is encoded as $n + 1$ tokens: a predicate token, whose learned embedding encodes the predicate and whether the atom is added or deleted, and one token per argument, which encodes the object and its argument position. Each predicate is represented only by a learned identifier. Every plan token also carries its step relative to the step i being refined. As absolute step embeddings do not extrapolate to plan lengths unseen in training (cf. Table 17), we encode only the offset $\Delta = i_t - i$, which is negative for past steps, positive for future steps, and zero for object and query tokens. Δ enters as an additive sinusoidal code, a bucketed attention bias (Raffel et al., 2020), and a rotary encoding (Su et al., 2024) on half of each head's channels.

We append one query token per object, and the prediction head reads its input from these tokens. Every object is assigned a learned identity embedding, which is added to its object token, every argument token that names it, and potential query tokens, binding them across the sequence. Identity embeddings are drawn from a set of learned vectors and assigned to the objects of each scene with randomization during training.

Relational placement fields. Given the query embeddings, a shared prediction head translates the symbolic atoms into a distribution over object poses. For every predicate, the domain designates which argument is placed relative to the other; in our domain, it is the first. The relevant atoms for an object o_k at step i are those in which o_k is the moved object's argument and that are either added or deleted by a_i, or *preserved* by a_i, i.e., hold in both s_{i-1} and s_i and name a moved object. Atoms in which o_k is only the reference object act on it through attention. In addition, every object has one base factor that depends on no atom, so that the composition is defined for objects without relevant atoms. The prediction head translates each of these atoms into a placement field centered on its reference object, which is then composed with the other fields to yield a distribution over poses for o_k.

A convolutional decoder with FiLM conditioning (Perez et al., 2018) decodes the context c_j of each atom, computed from the query embeddings of its two objects and the embeddings of its predicate



Figure 3: Examples of decoded placement fields for left, south, in, on, and near relations learned by PLAN2POSE purely from demonstrations. The object embeddings of the reference objects influence the shape and structure, as can be seen for `in` and `on`. PLAN2POSE composes these fields as a product and takes the probability-weighted mean of the smallest set of cells that holds half of the composed mass as the final placement in the 2D plane of the workspace.

and operation, into a 100×100 map of logits over cells of 10 mm. A softmax over the map yields a distribution p_j over positions relative to the reference object. Since the decoder is shared, the geometry of a predicate is not stored in dedicated weights but is provided in the embeddings. The fields in Figure 3 emerge purely from demonstrations, with the reference object's shape influencing them through its query embedding.

Composition and pose decoding. The per-atom fields of an object are combined on a common canvas, from which its pose is decoded. Each object has a canvas of the same resolution, centered on its position in g_{i-1}. The previous poses g_{i-1} enter the model only through these canvases and through the relative height and orientation predictions below. Since each field is centered on its reference object, it is shifted onto the canvas of its subject by the integer cell offset between their previous positions, similar to the object-centric frames of Migimatsu & Bohg (2020). The canvas of object o_k composes its factors as a weighted product of experts (Hinton, 2002),

$$\log P_k(u) \;=\; \sum_j \alpha_j \log p_j(u - t_j) \;+\; \text{const},$$

with cells u and a confidence α_j predicted from c_j. A cell keeps high probability only if every factor supports it, so each additional atom narrows the admissible region instead of widening it like an additive mixture. We decode the planar position without sampling, as the probability-weighted mean of the smallest set of cells that holds half of the composed mass.

Per object, the head also predicts a distribution over 160 height bins of 5 mm relative to the previous pose, composed with the weights α_j. Each factor also predicts a rotation of the previous orientation. The resulting quaternions are aligned in sign, averaged with the weights α_j, and renormalized. Refinement is thus one forward pass per step.

Training. We train on demonstrations with teacher forcing (Williams & Zipser, 1989), i.e., the previous poses are taken from the demonstration at a randomly drawn step. Every term of the objective $\mathcal{L} = \mathcal{L}_{xy} + \mathcal{L}_z + \mathcal{L}_{\text{rot}}$ acts on the composed prediction. The geometry of a predicate is learned only through its contribution to the demonstrated pose.

$\mathcal{L}_{xy}$ is the cross-entropy between the composed canvas and a Gaussian ($\sigma = 1.5$ cells) centered on the continuous demonstrated position. A demonstration is one sample from a region of valid placements, so neighbouring cells should not be penalized. $\mathcal{L}_z$ is the analogous cross-entropy over height bins, and $\mathcal{L}_{\text{rot}}$ the squared error between predicted and demonstrated quaternion. $\mathcal{L}_{xy}$ is applied to every object, while $\mathcal{L}_z$ and $\mathcal{L}_{\text{rot}}$ weight objects that move by more than 10 mm by a factor of three. An auxiliary regression of sub-cell offsets, not used for decoding, is added during training.

5 EXPERIMENTS

We evaluate PLAN2POSE against learned placement models to answer three questions: how well it refines long plans autoregressively, whether its placements anticipate constraints that only later steps of the plan state, and how it generalizes to unknown scenes.

Table 1: Main results on autoregressive refinement on plans of 9–12 / 13–16 / 17–20 states. Each set uses 200 plans, totalling 28,441 / 38,654 / 51,161 atoms. Mean over 3 seeds.

	TP			FS			MS			CV ↓		
Method	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20
B_0 identity	0.630	0.610	0.575	0.005	0.000	0.005	0.524	0.519	0.501	0.299	0.326	0.374
B_2 random	0.594	0.577	0.559	0.003	0.003	0.002	0.494	0.496	0.490	0.339	0.366	0.393
Rej. sampling	0.780	0.770	0.753	0.058	0.038	0.065	0.723	0.724	0.717	0.182	0.198	0.223
LINGO-Space^{z,s}	0.733	0.701	0.672	0.027	0.013	0.010	0.655	0.629	0.613	0.213	0.248	0.287
Diffusion-CCSP	0.839	0.838	0.827	0.100	0.115	0.105	0.795	0.803	0.797	0.130	0.137	0.152
StructDiffusion	0.817	0.806	0.804	0.075	0.097	0.075	0.767	0.760	0.767	0.149	0.162	0.170
PLAN2POSE	**0.972**	**0.971**	**0.966**	**0.557**	**0.585**	**0.542**	**0.962**	**0.963**	**0.958**	**0.022**	**0.022**	**0.029**

5.1 SETUP, METRICS, BASELINES

We compare against three learned baselines for placing objects under spatial relations. LINGO-Space (Kim et al., 2024) grounds each relation as a mixture of distributions around a reference object and updates the distribution by iteratively applying the relations to determine a placement in the plane. Since z-level movements are not considered by LINGO space, we provide the height computed from the reference object's geometry. The original LINGO-Space uses a language model to parse an instruction into tuples of a relation and a reference object. We replace the language component with the plan: every atom that the step adds with the moved object as first argument becomes one tuple, whose relation is the frozen CLIP embedding of the predicate name and whose reference object is given by its identity and position instead of a description. Diffusion-CCSP (Yang et al., 2023b) learns one diffusion model per constraint type (i.e., relation in our case) and samples from their composition using a diffusion-based approach. StructDiffusion (Liu et al., 2023) diffuses the poses of all objects of an arrangement jointly with one transformer. As additional references, the identity (B_0) leaves every object in place, and random valid placement (B_2) puts the moved object at a random collision-free pose. For rejection sampling we draw placements until one satisfies the step's atoms, with the height computed from the reference object's geometry, as for LING-Space.

We train PLAN2POSE on data from two pools of demonstrations, a bin-packing pool (primary), and a general arrangement pool (replay). The primary pool contains 1,965 episodes verified by a motion planner, where two or four blocks are packed into a bin sized for exactly that many, in scenes of 6–8 objects with plans of 3–5 states. The replay pool contains 10,400 general tabletop episodes with 2–9 states and 3–7 objects, built with primitive geometric and YCB objects (Calli et al., 2015), placed in simulated scenes similar to those in the primary pool.

Each baseline is applied per step and conditioned on the atoms that the step newly establishes for the moved object. StructDiffusion reads the same point clouds of the initial scene as our model, while Diffusion-CCSP and LINGO-Space receive object poses and box extents instead.

We evaluate all methods with four metrics by refining every plan $\pi \in \Pi$ autoregressively from the initial state. At each step i, we compare the atoms $\hat{s}_i = l(\hat{g}_i)$ that hold in the refined state with the demanded atoms s_i, computed from the demonstration with the same labelling function l. With sums over all steps of all plans, we define:

- *Task progress (TP):* how much of the plan is realized. TP is the recall of the demanded atoms: $\sum |s_i \cap \hat{s}_i| / \sum |s_i|$.
- *Final state correct (FS):* whether refining the whole plan ends in the correct state, i.e., the fraction of plans whose final refined state satisfies every demanded atom, $s_n \subseteq \hat{s}_n$.
- *Mover satisfaction (MS):* whether the moved objects are placed as the plan demands, i.e., TP restricted to the atoms s_i^M that name an object moved at step i.
- *Collateral violation (CV):* fraction of violated atoms c_i that name no moved object, and are thus unchanged since step $i-1$.

Our primary evaluation is conducted on plans of 9–12, 13–16, and 17–20 states with 4–6 primitive objects (200 plans each). Each plan makes use of 4 primary predicates `on`, `south`, `left`, and `in`.

Table 2: Prescience of learned models in two future-conditioned domains (platform and bridge). Each has the same initial state condition, but their final states require different refinements. **ext.:** measures whether the first placement satisfies its atoms and the rest of the plan stays satisfiable. **shift:** distance between the first placements under the two futures (i.e., the effect of prescience). **compl.:** autoregressive completion. Learned rows fine-tuned with equal budget. Mean over 3 seeds.

	Platform (6 mm, full support)						Bridge (5 mm inside)	
Method	ext. k=2	ext. k=4	ext. k=6 OOD	shift k=4 [mm]	compl. ID	compl. OOD	ext.	compl.
NULL A$^{\flat}$	0.355	0.095	0.090	0.0	0.013	0.000	0.070	0.080
Demonstrations$^{\flat}$	1.000	1.000	1.000	39.7	1.000	1.000	1.000	1.000
LINGO-Space^{z,s}	0.000	0.000	0.000	0.0	0.117	0.000	n/a	n/a
Diffusion-CCSP	0.095	0.000	**0.265**	0.0	0.221	0.177	0.245	0.002
StructDiffusion	0.405	0.172	0.108	0.0	0.087	0.000	0.260	0.232
PLAN2POSE no future (NF)	0.332	0.327	0.098	0.0	0.562	0.365	0.458	0.745
PLAN2POSE future masked	0.175	0.172	0.007	0.0	0.546	0.188	0.397	0.610
PLAN2POSE	**0.780**	**0.893**	0.115	36.5	**0.954**	**0.567**	**0.978**	**0.893**

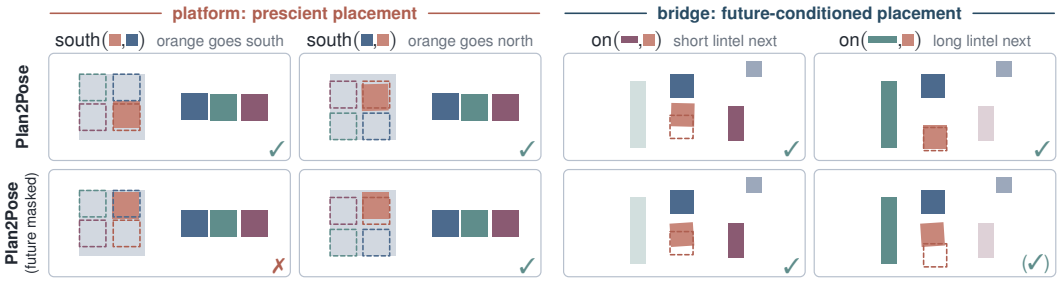

Figure 4: Two evaluation tasks testing whether PLAN2POSE's placements are prescient, with and without future masking. Platform: the first placement decides whether later relations (e.g., orange south or north of blue) remain satisfiable. With the future masked, PLAN2POSE always puts the orange block top right, failing when the blue block must later be north of it. Bridge: placements are shaped by the geometry of objects used later. For the short lintel both versions place the pillars close together; for the long one only PLAN2POSE with future context places them farther apart.

Additional evaluation is performed on scenes with 9–10 objects, on new scenes of YCB objects, and on 51 real captures recorded with an IFM O3P time-of-flight camera.

5.2 REFINEMENT QUALITY

Table 1 reports autoregressive refinement on plans of increasing length across our approach and the baselines. Accross all plan lengths, PLAN2POSE satisfies 0.97 of the atoms, compared to at most 0.84 for the best baseline (Diffusion-CCSP), and ends 0.54–0.59 of the plans in a correct final state, compared to at most 0.12. Considering purely myopic placements of a single steps, PLAN2POSE satisfies 0.964 of the atoms, compared to Diffusion-CCSP and StructDiffusion at 0.955 and 0.956. However, their placements deviate further from the demonstrations (Table 11), leading to worse refinement in autoregressive chaining. On steps where the moved object must satisfy two or more atoms at once, PLAN2POSE satisfies 0.911 of them, compared to 0.856 and 0.859 for Diffusion-CCSP and StructDiffusion (Table 10). Composing the fields as an additive mixture instead of a product lowers task progress by 0.11–0.13 and final-state correctness by 0.44–0.48 (Table 17).

5.3 PRESCIENT REFINEMENT

We test whether our refinements are prescient on two domains (Figure 4), platform and bridge. In platform, the first placement decides whether the rest of the plan stays satisfiable. In bridge, we test whether the geometry of future objects is considered during placement. We fine-tune PLAN2POSE

Table 3: Generalization on OOD datasets. We test plans with 9–10 objects (when trained with up to 8), new synthetic scenes of YCB objects, and 51 real captures. Mean over 3 seeds.

	TP			FS			CV ↓		
Method	9–10	YCB	real	9–10	YCB	real	9–10	YCB	real
LINGO-Space^{z,s}	0.881	0.834	0.821	0.063	0.074	0.000	0.081	0.097	0.138
Diffusion-CCSP	0.911	0.894	0.836	0.142	0.186	0.029	0.062	0.064	0.127
StructDiffusion	0.933	0.894	0.847	0.145	0.191	0.039	0.044	0.063	0.120
PLAN2POSE	**0.990**	**0.965**	**0.929**	**0.601**	**0.557**	**0.294**	**0.006**	**0.026**	**0.054**

and the baselines on these tasks, and ablate against fine-tuning without the future (NF) and future masking at test time. Table 2 shows that in distribution, only PLAN2POSE moves beyond myopic placements. When only the later steps of the plan differ, our first placement on the platform moves by 36.5 mm on average, close to the 39.7 mm of the demonstrations. The models without future context place it identically. Prescience holds in distribution but does not transfer to the out-of-distribution platform with six blocks, where the first placement is extendable in only 0.115 of the cases, close to NF (0.098). Appendix F.2) analyzes this failure in more detail.

5.4 GENERALIZATION

We evaluate PLAN2POSE's ability to generalize to new scenes and objects in Table 3. Trained on the same data, PLAN2POSE performs better than any of the baselines, especially on final-state correctness. We further look into a pre-trained PLAN2POSE's ability to learn additional predicates. To this end, we fine-tune on two new predicates `near` and `right`. After fine-tuning on 4000 episodes mixed with the original dataset, both new predicates are placed with similar quality to the established ones, while the established relations retain their quality.

6 LIMITATIONS

PLAN2POSE was evaluated on synthetic and real data, as well as with a MuJoCo simulation in the loop, but not on a real robot. PLAN2POSE decodes a single $\mathrm{SE}(3)$ pose from a multimodal field, which optimizes for compositional satisfaction, but could limit executability, where grasping constraints would sometimes require re-sampling. This also limits some OOD generalization aspects such as adapting to unseen platform sizes. PLAN2POSE anticipates constraints of later states, not unstated ones such as kinematic reachability. Finally, PLAN2POSE is limited to a bounded tabletop workspace (in our case $\pm 0.5\,\mathrm{m}$ around the origin) and takes symbolic plans rather than language instructions, being primarily designed for classical TAMP settings.

7 CONCLUSION

We formulated downward refinement as learning a distribution over object poses conditioned on the whole symbolic plan, and presented PLAN2POSE, which encodes scene and plan with one transformer and composes one placement field per atom as a product of experts. Predicates enter only as learned identifiers, and their geometry is learned from demonstrations with a single loss.

On single steps, PLAN2POSE is only slightly ahead of diffusion-based refiners, but its placements stay closer to the demonstrations, so errors compound less along the plan. On plans twice as long as seen during training, it satisfies 0.97 of the atoms and ends 0.54–0.59 of the plans in a correct final state, against at most 0.84 and 0.12 for the baselines, at 30 ms per step. PLAN2POSE transfers zero-shot to real point clouds, and in distribution its placements anticipate constraints that only later steps of the plan state.

This foresight cannot yet be used in out-of-distribution configurations. Constraints the plan leaves unstated, such as reachability, are not anticipated. Natural next steps are decoding several candidate poses and checking them with a motion planner, real-robot experiments, and richer predicate vocabularies.

AI USE STATEMENT

In this work, we used generative AI tools to refine hypotheses, provide feedback on experiments, to help develop conceptual frameworks, to implement methods, baselines, to clean and reformat datasets for release, and to interpret some preliminary results. Parts of the appendix were generated with the help of generative AI. Additionally, we used generative AI tools to identify, summarize and analyze existing literature, to brainstorm, to create and edit code, to draft and edit parts of this paper. We have not used generative AI tools to assist with the remaining required-disclosure tasks.

We have reviewed all AI-assisted work. Every quantitative result reported in this paper was produced by code executed by the authors on their own compute infrastructure. No reported number was produced, estimated, or extrapolated by a language model. AI-proposed explanations were treated as hypotheses requiring empirical test, never as findings. Multiple of these were falsified by our own measurements and withdrawn. Success criteria, decision thresholds, and stopping rules for the experiments reported here were defined by the authors. Every work cited in this paper was verified by retrieving the source where applicable. Descriptions of prior methods in our related work section were checked by the authors against the papers themselves. We conducted a deliberate manual search for prior work. Anonymized code implementing the method and the experiments was created with generative AI assistance and reviewed by the authors. Implementation errors found during this process will be documented in our final release code.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The model is described in Section 4; architecture and training settings, the fine-tuning recipes and the model card are given in Appendix A. Appendix B describes the data sets, their splits and the real captures, Appendix C defines every evaluation metric, and Appendix D describes how the baselines were ported and trained. The supplementary material and the anonymous repository `https://github.com/plan2pose/plan2pose-code` contain a reviewed and anonymized re-implementation of PLAN2POSE, which reproduces the reported predictions bitwise on the same GPU type, the baseline ports, the evaluation package, the exact training, validation and evaluation splits, and the script that renders every table from stored result files. The data sets and the checkpoints behind every reported number, for our model and all baselines, are release assets of the tagged repository; for licensing reasons, the released replay pool omits the episodes built from GraspNet-1Billion scans. All results use fixed evaluation sets and three independently trained seeds, and evaluation is deterministic for a given checkpoint.

REFERENCES

Fahiem Bacchus and Qiang Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, 71(1):43–100, 1994. doi: 10.1016/0004-3702(94)90062-0.

Julien Bidot, Lars Karlsson, Fabien Lagriffoul, and Alessandro Saffiotti. Geometric backtracking for combined task and motion planning in robotic systems. *Artificial Intelligence*, 247:229–265, June 2017. doi: 10.1016/j.artint.2015.03.005.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pp. 510–517, 2015.

Rohan Chitnis, Dylan Hadfield-Menell, Abhishek Gupta, Siddharth Srivastava, Edward Groshev, Christopher Lin, and Pieter Abbeel. Guided search for task and motion plans using learned heuristics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 447–454, 2016.

Neil T Dantam, Zachary K Kingston, Swarat Chaudhuri, and Lydia E Kavraki. An incremental constraint-based framework for task and motion planning. *International Journal of Robotics Research*, 37(10):1134–1151, September 2018. doi: 10.1177/0278364918761570.

Roshan Dhakal, Duc M. Nguyen, Tom Silver, Xuesu Xiao, and Gregory J. Stein. Anticipatory task and motion planning, July 2024.

Danny Driess, Jung-Su Ha, and Marc Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Proceedings of Robotics: Science and Systems*, 2020.

Hao-Shu Fang, Chenxi Wang, Minghao Gou, and Cewu Lu. GraspNet-1Billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11444–11453, 2020.

Xiaolin Fang, Caelan Reed Garrett, Clemens Eppner, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. DiMSam: Diffusion models as samplers for task and motion planning under partial observability. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1412–1419, October 2024. doi: 10.1109/IROS58592.2024.10802746.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pp. 440–448, 2020. doi: 10.1609/icaps.v30i1.6739.

Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):265–293, 2021. doi: 10.1146/annurev-control-091420-084139.

Nikolaos Gkanatsios, Ayush Jain, Zhou Xian, Yunchu Zhang, Christopher G. Atkeson, and Katerina Fragkiadaki. Energy-based models are zero-shot planners for compositional scene rearrangement. In *Robotics: Science and Systems XIX*, volume 19, July 2023. `https://roboticsproceedings.org/rss19/p030.html`.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. doi: 10.1162/089976602760128018.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1470–1477. IEEE, 2011. doi: 10.1109/ICRA.2011.5980391.

Beomjoon Kim, Zi Wang, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning to guide task and motion planning using score-space representation. *The International Journal of Robotics Research*, 38(7):793–812, 2019. doi: 10.1177/0278364919848837.

Dohyun Kim, Nayoung Oh, Deokmin Hwang, and Daehyung Park. LINGO-Space: Language-conditioned incremental grounding for space. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10314–10322, 2024. doi: 10.1609/aaai.v38i9.28898.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023.

Richard E. Korf. Planning as search: A quantitative approach. *Artificial Intelligence*, 33(1):65–88, September 1987. doi: 10.1016/0004-3702(87)90051-8.

Fabien Lagriffoul and Benjamin Andres. Combining task and motion planning: A culprit detection problem. *The International Journal of Robotics Research*, 35(8):890–927, July 2016. doi: 10.1177/0278364915619022.

Fabien Lagriffoul, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti, and Lars Karlsson. Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14):1726–1747, December 2014. doi: 10.1177/0278364914545811.

Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox. Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6322–6329, 2022. doi: 10.1109/ICRA46639.2022.9811931.

Weiyu Liu, Yilun Du, Tucker Hermans, Sonia Chernova, and Chris Paxton. StructDiffusion: Language-Guided Creation of Physically-Valid Structures using Unseen Objects. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023. ISBN 978-0-9923747-9-2. doi: 10.15607/RSS.2023.XIX.031.

Toki Migimatsu and Jeannette Bohg. Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 5(2):844–851, April 2020. doi: 10.1109/LRA.2020.2965875.

Utkarsh Aashu Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative skill chaining: Long-horizon skill planning with diffusion models. In *7th annual conference on robot learning*, 2023. URL https://openreview.net/forum?id=HtJE9ly5dT.

Utkarsh Aashu Mishra, Yongxin Chen, and Danfei Xu. Generative factor chaining: Coordinated manipulation with diffusion-based factor graph. In *8th Annual Conference on Robot Learning*, September 2024. https://openreview.net/forum?id=p6Wq6TjjHH.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp. 220–229, 2019.

Michael Noseworthy, Isaiah Brand, Caris Moses, Sebastian Castro, Leslie Kaelbling, Tomas Lozano-Perez, and Nicholas Roy. *Active Learning of Abstract Plan Feasibility*. July 2021. doi: 10.15607/RSS.2021.XVII.043.

Joaquim Ortiz-Haro, Jung-Su Ha, Danny Driess, Erez Karpas, and Marc Toussaint. Learning feasibility of factored nonlinear programs in robotic manipulation planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3729–3735, May 2023. doi: 10.1109/ICRA48891.2023.10160887.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3942–3951, 2018. doi: 10.1609/aaai.v32i1.11671.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, Honolulu, HI, July 2017. IEEE. ISBN 978-1-5386-0457-1. doi: 10.1109/CVPR.2017.16. URL http://ieeexplore.ieee.org/document/8099499/.

Ahmed H. Qureshi, Arsalan Mousavian, Chris Paxton, Michael Yip, and Dieter Fox. NeRP: Neural rearrangement planning for unknown objects. In *Robotics: Science and Systems XVII*, volume 17, July 2021. https://roboticsproceedings.org/rss17/p072.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3182–3189, September 2021. doi: 10.1109/IROS51168.2021.9635941.

Tom Silver, Ashay Athalye, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 701–714, 2022.

Tom Silver, Rohan Chitnis, Nishanth Kumar, Willie McClinton, Tomás Lozano-Pérez, Leslie Kaelbling, and Joshua B. Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12120–12129, June 2023. doi: 10.1609/aaai.v37i10.26429.

Anthony Simeonov, Ankit Goyal, Lucas Manuelli, Lin Yen-Chen, Alina Sarmiento, Alberto Rodriguez, Pulkit Agrawal, and Dieter Fox. Shelving, stacking, hanging: Relational pose diffusion for multi-modal rearrangement. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2030–2069, 2023.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 639–646, 2014. doi: 10.1109/ICRA.2014.6906922.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, pp. 1930–1936, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6–7):866–894, 2021. doi: 10.1177/02783649211004615.

Andrew M. Wells, Neil T. Dantam, Anshumali Shrivastava, and Lydia E. Kavraki. Learning feasibility for task and motion planning in tabletop environments. *IEEE Robotics and Automation Letters*, 4(2):1255–1262, 2019. doi: 10.1109/LRA.2019.2894861.

Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, June 1989. doi: 10.1162/neco.1989.1.2.270.

Danfei Xu, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Deep affordance foresight: Planning through what can be done in the future. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6206–6213, May 2021. doi: 10.1109/ICRA48506.2021.9560841.

Zhutian Yang, Caelan Reed Garrett, Tomás Lozano-Pérez, Leslie Pack Kaelbling, and Dieter Fox. Sequence-based plan feasibility prediction for efficient task and motion planning. In *Proceedings of Robotics: Science and Systems*, 2023a. doi: 10.15607/RSS.2023.XIX.061.

Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3242–3265, 2023b.

Yuchi Zhao, Miroslav Bogdanovic, Chengyuan Luo, Steven Tohme, Kourosh Darvish, Alan Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Anyplace: Learning generalizable object placement for robot manipulation. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4038–4057, 2025.

Yao Zhong, Hanzhi Chen, Simon Schaefer, Anran Zhang, and Stefan Leutenegger. Gopla: Generalizable object placement learning via synthetic augmentation of human arrangement, 2025.

A Implementation Details

Table 4: Architecture and training settings of the final model.

Component	Setting
Object encoder	512 points per object; PointNet 3–64–128–1024, max-pool, 1024→128→512; initial position added through a linear map 3→512
Tokens	width 512; 20 identity labels shared by object, argument and query tokens
Plan position	relative step Δ: sinusoidal code, bucketed attention bias (32 buckets), rotary encoding on half of each head's channels
Transformer	8 layers, 8 heads; only the query rows are read out
Factor context	MLP 2048→256→256
Field decoder	shared; 256→64×8×8, four FiLM blocks up to 128^2, resized to 100×100
Canvas	100×100 cells of 10 mm (±0.5 m), centred on the previous position
Height	160 bins of 5 mm (±0.4 m), relative to the previous height
Rotation	per-factor quaternion change, α-weighted mean
Decoding	centroid of the top half of the probability mass; no sampling
Loss	soft cross-entropy on xy ($\sigma = 1.5$ cells) and height ($\sigma = 1$ bin); squared quaternion error; height and rotation terms weighted ×3 for objects moved by more than 10 mm; auxiliary Huber loss on sub-cell offsets (not used for decoding)
Optimiser	AdamW, learning rate 10^{-4}, constant; no gradient clipping; 50-epoch warm-up of the α temperature
Batches	16 episodes (8 primary, 8 replay), 222 per epoch
Length	200 epochs (44,400 steps)
Regularisation	dropout 0.05; atom-tag permutation; identity relabelling on the primary pool from epoch 161
Seeds	3
Parameters	24.0 M, 87 % in the transformer

A.1 Fine-Tuning

Our fine-tunes start from the final checkpoints of the three seeds and keep the batch composition of eight fine-tuning and eight replay episodes. For the two foresight families, every learned method is fine-tuned on the family's 1,000 training episodes with the same budget of 160,000 training examples, counted in each method's own unit. For our model, this is 10,000 steps of batch size 16, i.e. 80 epochs over the family's episodes, with identity relabelling from the first epoch. The no-future control (NF) is fine-tuned with the future plan tokens removed. For the new predicates, the predicate vocabulary is extended from five to seven entries, and training continues for 60 epochs with a fresh optimiser on 4,000 episodes (3,600 for training, 400 for validation).

A.2 Model Card

Following Mitchell et al. (2019). Table 4 lists the settings of the final model.

Model. Plan2Pose, the final configuration of Section 4 (checkpoint after 200 epochs), three training seeds, and the fine-tuned variants of Section 5 (new predicates, platform and bridge families). Transformer encoder over scene and plan tokens with a compositional field head; 24.0 M parameters.

Inputs. Segmented point clouds of the initial scene (512 points per object), a symbolic plan given as its sequence of states over binary predicates, the index of the step to refine, and the objects' previous poses.

Outputs. For every object, a distribution over planar position (100×100 cells of 10 mm around its previous position), a distribution over height change (160 bins of 5 mm), and a rotation; a single pose per object is decoded without sampling.

Training data. Demonstrations generated in simulation: the primary pool (bin packing with two or four blocks) and the replay pool (general tabletop episodes), mixed at batch level; predicates *on*, *in*, *left*, *south*, *smaller*; at most eight objects per scene. 2,850 replay episodes use real object scans from GraspNet-1Billion (Fang et al., 2020) in simulated scenes; no capture of our own setup is used for training (Appendix B).

Training procedure. One objective on the composed prediction, teacher forcing, AdamW with a constant learning rate, identity relabelling and atom-tag permutation (Appendix A).

Intended use. Research on refining symbolic plans into object poses for tabletop rearrangement with rigid objects, as an intermediate layer between a symbolic planner and a motion planner or policy that reaches the predicted poses.

Out of scope. Direct execution without a downstream feasibility check; safety-critical use; deformable or articulated objects; more than 20 objects (the number of identity labels); placements more than 0.5 m from an object's previous position or height changes beyond 0.4 m, which the output cannot represent; trajectories and grasps, which the model does not predict.

Evaluation. Plans of 9–20 states, scenes with 9–10 objects, new scenes of YCB objects (the objects also occur in training), real captures of our setup (none in training), new predicates, and the foresight families; the metrics of Appendix C, extendable rate, shift, completion and latency (Section 5, Appendix C, Appendix F).

Limitations. The model anticipates constraints stated in later steps of the plan in distribution, but not in the out-of-distribution platform setting, and not constraints that the plan does not state, such as kinematic reachability. Decoding the centroid of a multimodal field can place an object between two valid regions. Atoms that name the moved object as relatum act on it only through attention. The rotation loss does not identify q with $-q$. Results vary slightly across GPU types.

Ethical considerations. No human data is used. Predicted poses are targets for a robot and must be checked for feasibility and safety before execution.

Licence. Code MIT; data and checkpoints CC BY 4.0. For licensing reasons, the released replay pool omits the episodes built from GraspNet-1Billion scans; the omitted episodes are listed. Baseline code keeps its original licences.

B Data

B.1 Training and Evaluation Sets

Figure 5 shows one scene of each kind, and Table 5 lists every data set. The primary pool contains bin-packing episodes in which two blocks (1,050 episodes) or four blocks (915) are packed into a bin; every demonstrated transition was executed successfully by the executor of subsection B.3. It contains no *on* atom. The replay pool contains general tabletop episodes built from geometric primitives (2,850 episodes), YCB objects (Calli et al., 2015) (3,300), real object scans from GraspNet-1Billion (2,850) and cubes (1,400, of which 450 contain a bin); every *on* atom our model sees in training comes from this pool. The evaluation sets share no initial scene and no file with the training data. In 25 of the 204 YCB scenes, an object lies off the table; since every method is scored on the same stored poses, this affects all rows alike.

B.2 Real Point-Cloud Captures

We recorded 51 manipulation episodes with a single pre-production ifm O3P RGB-D time-of-flight camera, whose specifications may differ from the production model. The scenes contain cubes, YCB

Table 5: Data sets. Training and validation splits in parentheses. States count the initial state; a plan with n states has $n - 1$ steps.

Set	episodes	objects	used in
Training			
Primary pool (3–5 states)	1,965 (1,769 / 196)	6–8	all models
Replay pool (up to 9 states)	10,400	3–7	all models
Platform family	1,000	7–8	fine-tuning, Table 2
Bridge family	1,000	5–6	fine-tuning, Table 2
New predicates	4,000 (3,600 / 400)	4–6	fine-tuning, Table 12
Evaluation			
Plans of 9–12 / 13–16 / 17–20 states	200 each	4–6	Tab. 1, 11, 13
Nine to ten objects	101	9–10	Table 3
YCB scenes	204	3–7	Table 3
Real captures	51	6–7	Table 3
Containment-heavy	150	5–6	Table 10
In-distribution	300	4–6	Table 12
Five-block family (held out)	170	8	Table 17
New predicates	400	4–6	Table 12
Platform, $k = 2$–4 / $k = 6$	700 / 300	7–8 / 9	Table 2
Bridge	200	5–6	Table 2
Grasp-blocking problems	40	–	Tab. 16, 14

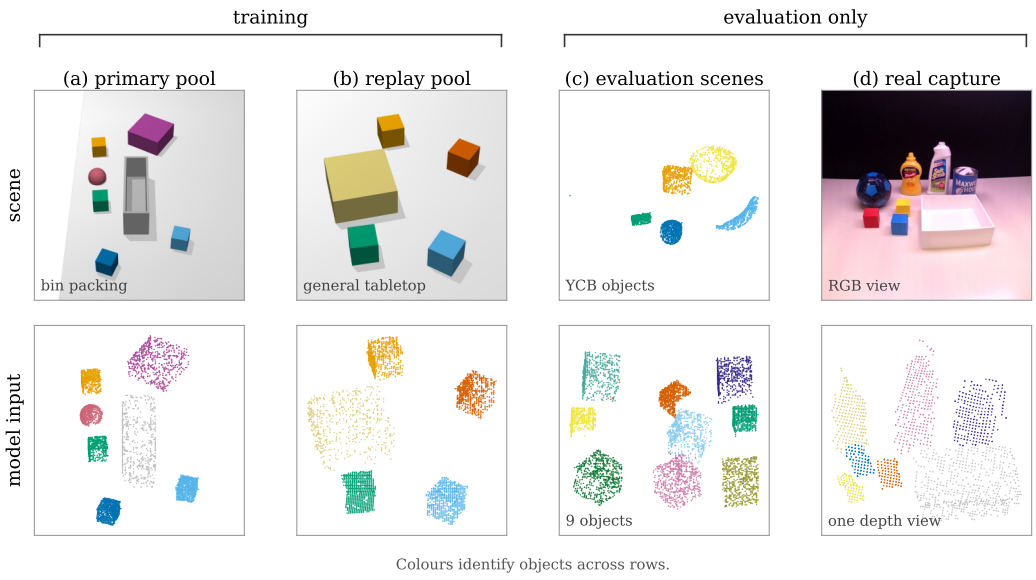

Figure 5: One scene of each kind: (a) the primary pool (bin packing), (b) the replay pool (general tabletop), (c) evaluation scenes with YCB objects and with nine objects, and (d) a real capture. Top: the scene; bottom: the per-object point clouds the model receives (512 points each).

objects and a bin. Each episode is a sequence of states; 399 of the 413 transitions move one object, the others two to four. Every state contains a 1008×1344 colour image, a 180×240 time-of-flight depth map with its point cloud, and the calibration between the depth and colour cameras. Object masks and per-object point clouds are obtained with the pipeline in Figure 6. The released data contain the per-object point clouds and the symbolic states, not the images.

Masks. For each state, the automatic mask generator of SAM (Kirillov et al., 2023) with a frozen ViT-B backbone proposes class-agnostic masks from a 32×32 grid of point prompts, with a predicted-IoU threshold of 0.88 and a stability threshold of 0.92; masks outside 500–80,000 pix-

els are discarded. Where small or adjacent cubes are not separated, we use a 64×64 grid and thresholds of 0.82 and 0.88.

Identities. Object identities are assigned by hand in one verified reference state. Each mask is described by a hue–saturation histogram, an a^*b^* histogram in Lab space, and its log aspect ratio and log area. In every other state, the candidate masks are matched to the reference objects with the Hungarian algorithm on a weighted distance between these descriptors. Since masks are generated and matched independently in each state, no box is propagated between states. Matches with a high descriptor cost or an implausible displacement from the last accepted position are rejected rather than forced, and the remaining errors are corrected by hand during visual verification.

Per-object point clouds. Cube masks are eroded, and their points are restricted to the largest DBSCAN cluster to remove points from neighbouring surfaces. YCB objects keep all points in their mask, since their visible parts can be disconnected. The visible top and front of the bin are matched separately and merged. Valid depth points are transformed into the colour-camera frame and projected into the image; a point belongs to an object if its pixel lies in the object's mask, and overlapping masks are made mutually exclusive beforehand, so that each point belongs to at most one object. No depth is interpolated and no points are created from mask pixels. An object with fewer than eight points is treated as missing in that state. A real cube keeps about 70 points (median; YCB objects about 240), and the model draws 512 points per object with replacement, as for the simulated sets.

Geometry and symbolic states. Object centres are the coordinate-wise median of the points. Since the camera does not see the lower part of a cube, the bottom of a cube's box is placed 50 mm below its observed top; its horizontal extent is the observed one, and the point cloud keeps only measured points. The pipeline also computes relations with its own thresholds, which we use only to verify the masks and poses visually. The symbolic states of the released episodes, which form both the plan given to every method and the demanded atoms, are computed from the extracted poses and sizes with the labelling function l, as for every simulated set.

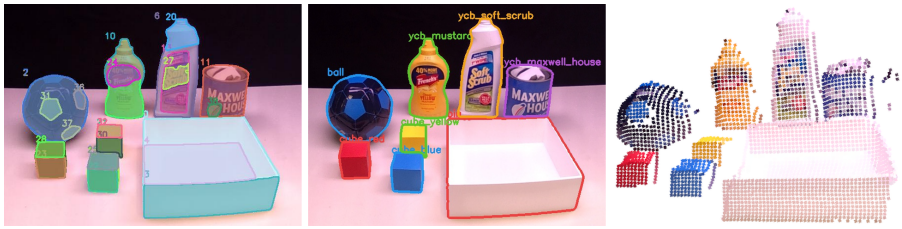

Figure 6: Annotation of a real capture: SAM mask candidates (a), masks after matching and naming (b), and the resulting per-object point clouds (c).

B.3 Execution in Simulation

We check which transitions a robot can execute with a MuJoCo simulation of a Franka Panda arm and our grasp, pick-and-place and motion-planning components (Figure 7). This executor gates the primary pool, filters the cube episodes of the replay pool, labels the grasp-blocking problems, and executes the refinements in the experiments of subsection E.3. No column of Table 2 uses it.

Each episode is checked on its own. The simulator is initialised with every object at its demonstrated initial pose, and the transitions are executed in order: for each transition, the moved object and its target pose are taken from the demonstration, and the planner plans and executes the pick and the place. The evaluator can keep the executable prefix of a failed episode, but every gate used in this paper keeps only episodes whose transitions all succeed; none of the 1,965 primary episodes is a prefix. The simulator is then reset to the initial state of the next episode.

All episodes use one configuration (Table 6), except that the experiments of Table 14 let objects settle for 640 simulation steps instead of 160. We extended the default setup with diagonal grasp orientations and a dedicated grasp for spheres. During placement, alternative gripper orientations

are tried if the first attempt fails, and if inverse kinematics cannot reach the approach pose, the placement is retried with lower approach heights, each retry starting from the saved pre-placement state. The grasped object is attached to the end effector during transport, so grasp stability is not simulated, and every object except spheres is simulated as a box of its measured size. The check establishes which transitions are executable; it does not evaluate the planner.

Table 6: Planning and execution parameters of the executor. The experiments of Table 14 use 640 settling steps.

Component	Parameter	Value	Description
Inverse kinematics	Position threshold	$0.005\,\mathrm{m}$	Maximum accepted positional error
	Orientation threshold	$0.01\,\mathrm{rad}$	Maximum accepted orientation error
	Maximum iterations	1000	Maximum number of IK iterations per attempt
	Maximum retries	3	Maximum number of IK retries
RRT*	Maximum iterations	5000	Maximum number of search iterations
	Step size	0.1	Extension step size of the planner
	Search radius	0.5	Neighbourhood radius used during the search
	Goal threshold	0.05	Threshold for considering the goal reached
	Goal sampling rate	0.1	Probability of sampling the goal directly
Grasp planning	Approach distance	$0.12\,\mathrm{m}$	Distance used for approaching a grasp pose
	Lift height	$0.15\,\mathrm{m}$	Vertical lifting distance after grasping
	Table clearance	$0.025\,\mathrm{m}$	Clearance above the table surface
	Grasp orientations	Front + diagonal	Front grasps extended with diagonal candidates
	Sphere handling	Enabled	Sphere-specific grasp handling
Placement	Initial approach height	$0.12\,\mathrm{m}$	Initial approach height for placement
	Height attempts	3	Approach heights tested after an IK failure
	Height range	$0.12 \rightarrow 0.04\,\mathrm{m}$	Approach heights reduced linearly
	Placement clearance	$0.035\,\mathrm{m}$	Clearance applied during placement
Execution	Approach step size	$0.01\,\mathrm{m}$	Step size during the approach
	Grasp/lift/place step size	$0.003\,\mathrm{m}$	Step size during grasping, lifting and placing
	Retreat step size	$0.01\,\mathrm{m}$	Step size during the retreat
	Settling steps	160	Simulation steps allowed for settling

C Evaluation Protocol

For an evaluation episode e with plan $\pi_e = (a_1, \ldots, a_{n_e})$, demonstrated states $g_0^D, \ldots, g_{n_e}^D$ and moved objects M_i, the demanded atoms are computed from the demonstration with the labelling function that also scores the predictions, $s_i = l(g_i^D)$, restricted to the evaluated predicates. Each method is rolled out autoregressively from the observed state, $\hat{g}_0 = g_0$: $\hat{g}_i$ equals $\hat{g}_{i-1}$ except for the objects in M_i, which take the poses the method predicts from $(\hat{g}_{i-1}, \pi_e)$, and $\hat{s}_i = l(\hat{g}_i)$ are the atoms that hold in the refined state. M_i is taken from the demonstration for every method: from the generator's record of the moved object on the simulated sets, and on the real captures as the object with the largest displacement, provided it clearly exceeds the displacement of the objects that do not move. A step is scored if it establishes at least one atom, $S_e = \{\, i : s_i \setminus s_{i-1} \neq \emptyset \,\}$. We write s_i^M for the atoms of s_i that name an object in M_i, and $c_i = \{\varphi \in s_i \cap s_{i-1} : \varphi \text{ names no object in } M_i\}$ for the unchanged atoms that name no moved object. Sums over e range over the evaluation set E and sums over i over S_e; every metric is computed per training seed and averaged over seeds. The four metrics of the main text are:

- *Task progress (TP):* the fraction of demanded atoms that hold, pooled over all scored steps, i.e., the recall of the demanded atoms,
$$\mathrm{TP} = \frac{\sum_{e,i} |s_i \cap \hat{s}_i|}{\sum_{e,i} |s_i|}.$$
- *Final state correct (FS):* the fraction of episodes whose final state satisfies every demanded atom,
$$\mathrm{FS} = \frac{1}{|E|} \sum_{e} \mathbf{1}\big[s_{n_e} \subseteq \hat{s}_{n_e}\big].$$

Table 7: Criteria FP per-atom satisfaction in the final state and AS every state perfect, same rows, rungs and seeds as Table 1. $^{\flat}$ reference rows; † one seed; ‡ two seeds.

	FP			AS		
Method	9–12	13–16	17–20	9–12	13–16	17–20
B_0 identity$^{\flat}$	0.557	0.528	0.511	0.000	0.000	0.000
B_2 random valid$^{\flat}$	0.513	0.491	0.485	0.000	0.000	0.000
Rejection sampling (oracle)$^{z\flat}$	0.722	0.705	0.689	0.000	0.000	0.000
LINGO-Space^{z,s}	0.654	0.606	0.584	0.000	0.000	0.000
Diffusion-CCSP	0.795	0.800	0.768	0.000	0.003	0.000
StructDiffusion	0.771	0.747	0.747	0.002	0.000	0.000
PLAN2POSE (ours)	**0.961**	**0.963**	**0.955**	**0.158**	**0.087**	**0.043**

- *Mover satisfaction (MS):* TP restricted to the atoms that name a moved object, whether newly established or carried over,

$$\mathrm{MS} = \frac{\sum_{e,i} |s_i^M \cap \hat{s}_i|}{\sum_{e,i} |s_i^M|}.$$

- *Collateral violation (CV, ↓):* the fraction of unchanged atoms that name no moved object but do not hold,

$$\mathrm{CV} = \frac{\sum_{e,i} |c_i \setminus \hat{s}_i|}{\sum_{e,i} |c_i|}.$$

  Since only the objects in M_i move at step i, such atoms can only be broken by earlier placements; CV is zero under teacher forcing and measures how errors propagate along the rollout.

We additionally report three metrics:

- *Final progress (FP):* TP in the final state only,

$$\mathrm{FP} = \frac{\sum_{e} |s_{n_e} \cap \hat{s}_{n_e}|}{\sum_{e} |s_{n_e}|}.$$

- *All states correct (AS):* the fraction of episodes in which every scored state satisfies every demanded atom,

$$\mathrm{AS} = \frac{1}{|E|} \sum_{e} \prod_{i \in S_e} \mathbf{1}\big[s_i \subseteq \hat{s}_i\big].$$

- *Precision (PR):* the fraction of atoms that hold in the refined state and are also demanded,

$$\mathrm{PR} = \frac{\sum_{e,i} |s_i \cap \hat{s}_i|}{\sum_{e,i} |\hat{s}_i|}.$$

Teacher-forced variants replace $\hat{g}_{i-1}$ by g^D_{i-1}. Differences between methods are tested with a paired cluster bootstrap over episodes (10,000 resamples). Table 7 reports FP and AS for the rollouts of Table 1. Every state is fully correct in 0.158, 0.087 and 0.043 of our rollouts on the three plan lengths, and in at most 0.003 for every other method.

TP measures recall: it counts the demanded atoms that hold, but not the atoms a placement creates without being asked to. Table 8 reports PR on the same rollouts; re-running them reproduces TP to within 0.0003 in every seed. Our precision is 0.963–0.975, against 0.674–0.820 for the learned baselines. The largest difference is on *on*: 0.917–0.950 for our model, 0.428–0.671 for the learned baselines, which create many stacking relations that the plan does not contain.

Table 8: Precision of the rollouts of Table 1: of the atoms that hold after a placement, the fraction that also holds in the demonstrated state, $\sum_{e,i} |s_i \cap l(\hat{g}_i)| \,/\, \sum_{e,i} |l(\hat{g}_i)|$, overall and per relation (no set contains an *in* atom). –: the method never produces the relation. 3 seeds, mean. $^\flat$ reference rows; † one seed.

	9–12				13–16				17–20			
Method	PR	on	left	south	all	on	left	south	PR	on	left	south
B_0 identity$^\flat$	0.633†	–	0.617†	0.646†	0.603†	–	0.577†	0.625†	0.581†	–	0.547†	0.608†
B_2 random valid$^\flat$	0.551	–	0.508	0.590	0.529	–	0.496	0.558	0.513	–	0.474	0.548
Rejection sampling (oracle)$^{z\flat}$	0.737	0.773	0.705	0.764	0.713	0.702	0.686	0.737	0.700	0.634	0.674	0.724
LINGO-Space^{z,s}	0.738	0.541	0.745	0.741	0.696	0.441	0.693	0.712	0.674	0.428	0.658	0.700
Diffusion-CCSP	0.820	0.600	0.807	0.840	0.806	0.570	0.805	0.818	0.802	0.564	0.797	0.818
StructDiffusion	0.800	0.671	0.776	0.822	0.783	0.634	0.766	0.801	0.782	0.664	0.767	0.798
PLAN2POSE (ours)	**0.975**	**0.950**	**0.980**	**0.971**	**0.968**	**0.942**	**0.972**	**0.966**	**0.963**	**0.917**	**0.964**	**0.964**

D BASELINES

Common adapter. Every baseline is applied one step at a time and conditioned on the atoms that the step newly establishes and that name the moved object; no baseline sees carried atoms, the full target state or later steps. Diffusion-CCSP, StructDiffusion and rejection sampling use atoms in which the moved object is subject or relatum; LINGO-Space uses only those in which it is the subject (^{s}), and a step whose only atoms name the moved object as relatum leaves it in place. The geometry each method receives follows its original design: StructDiffusion reads point clouds, while Diffusion-CCSP and LINGO-Space receive object poses and box sizes.

LINGO-Space. The original method uses a language model to parse an instruction into tuples of a relation and a reference object, encodes each relation and the description of each reference object with the CLIP text encoder (Radford et al., 2021), and encodes each scene-graph node by its coordinate, its box and a CLIP image feature of its crop. We keep the estimator and its released training objective and replace these inputs. The relation is the bare predicate name (*on*, *in*, *left*, *south*), encoded once with the frozen CLIP ViT-L/14 text encoder and followed by a new trained linear layer. The reference object is encoded by its identity and position instead of a description. A node's coordinate is its xy position, scaled to $[-1, 1]$ with the original Fourier encoding; its size enters the identity feature, and the image-feature slot is fed by a learned layer over the size and a flag for fixed objects. At each step, the estimator is called once per atom, in a fixed order (predicate name, then related object), passing its state forward as in the original. The per-relation distributions are combined on a 640×320 grid by summing log-densities, which selects the same cell as the original product and avoids underflow. LINGO-Space places objects in the plane; the height is computed from the supporting object's geometry (^{z}).

Diffusion-CCSP. The original method represents a problem as a graph of constraints between objects and learns one denoising network per constraint type. The networks share an encoder of object geometry, are trained jointly on whole constraint graphs, and their outputs for an object are averaged and decoded by a shared pose decoder. Objects are represented by their bounding boxes, and a pose by its position and its rotation about the vertical axis. At test time, the original draws many samples and keeps one that satisfies all constraints. We keep the model and change four things. First, the constraint types are our four relations, so there is no collision constraint, which the original uses to keep objects apart. Second, the only decision variable is the pose of the moved object; the poses of the other objects are fixed, as in the original when some poses are given. Third, the constraints of a step are the atoms it newly establishes for the moved object, as subject or relatum. Fourth, we draw a single sample per placement, like every other learned method; Table 13 gives it more samples.

StructDiffusion. The original method encodes the partial point cloud of every object and the words of a language goal as tokens of one transformer, which predicts the poses of all objects of a structure jointly by diffusion; learned discriminators then rank the samples by collisions and by the validity of the structure. We change three things. First, the language goal is replaced by the

Table 9: Table 1 with the learned baselines in their pool-only exposure (trained on the primary pool alone). 3 seeds, mean. † one seed; ‡ two seeds.

	TP			FS			MS			CV ↓		
Method	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20
LINGO-Space^{z,s}	0.690	0.653	0.622	0.010	0.010	0.007	0.599	0.570	0.553	0.249	0.287	0.330
Diffusion-CCSP	**0.772**	**0.760**	**0.746**	**0.038**	**0.042**	**0.045**	**0.713**	**0.711**	**0.707**	**0.189**	**0.206**	**0.227**
StructDiffusion	0.670	0.659	0.633	0.008	0.007	0.003	0.586	0.589	0.571	0.273	0.291	0.324

atoms of the step: each atom enters as tokens for its relation and its argument positions, so that the model sees whether the moved object is subject or relatum. Second, we apply it per step and diffuse only the pose of the moved object, given its point cloud and those of up to seven related objects (64 points each), their sizes, and the related objects' current poses; the point clouds are those of the initial scene. A pose is its position and its rotation about the vertical axis, as for Diffusion-CCSP, instead of a full rotation. Third, we draw a single sample per placement and do not rank samples with the discriminators.

References. The identity (B_0) leaves every object in place. Random valid placement (B_2) puts the moved object at a random collision-free pose. Rejection sampling draws up to 50 placements and accepts the first that satisfies the step's atoms under l, with the height computed from the supporting object's geometry; it therefore has exact access to the predicate definitions.

Training exposure. The learned baselines are trained on the same data as our model: the primary and replay pools merged, with every primary episode repeated six times, so that the primary pool keeps about half of the training steps as in our batches (22,189 episodes). LINGO-Space, whose training is slower, is trained for five epochs on this pool and selected by its validation loss. Trained on the primary pool alone, every baseline is weaker (Table 9), most clearly StructDiffusion, so the rows of Table 1 are the stronger comparison.

Fidelity. We did not run the original authors' code. Each port is tested against frozen copies of its own earlier trainer, and its fidelity to the original was checked in written audits: StructDiffusion line by line against the released code, LINGO-Space in its optimisation settings and architecture, and Diffusion-CCSP against its paper, since its code carries no licence. An earlier LINGO-Space port trained on only part of the released objective; it was retrained, and its earlier runs were discarded. The divergences of every port from its original are documented in the release.

E Further Results

E.1 Single Steps and Compounding

With the earlier steps at their demonstrated poses, the diffusion baselines are close to our model on single placements (Table 10). On *left*, Diffusion-CCSP (0.973) and StructDiffusion (0.964) are slightly above ours (0.951); our advantage is mainly on *on* (0.996 against 0.837 and 0.713). When a step demands two or more atoms of the moved object, our model satisfies all of them in 0.911 of the steps, against 0.859 and 0.856 (differences $+0.052$ $[+0.028, +0.075]$ and $+0.055$ $[+0.028, +0.083]$). For all three methods, this rate lies about 0.01 above what independent per-atom outcomes would give ($+0.008$, $+0.010$, $+0.010$), so the gap follows from per-atom satisfaction rather than from composition.

The difference appears along the rollout (Table 11). At steps 17–20, the three methods satisfy 0.964, 0.955 and 0.956 of the atoms under teacher forcing, but 0.946, 0.787 and 0.760 on their own rollouts. The baselines' placements deviate further from the demonstrations (median 98.1 and 112.9 mm against 41.5 mm; more than 100 mm in 0.485 and 0.569 of the steps against 0.086), and later steps build on these placements.

Table 10: Teacher-forced single-step satisfaction per predicate, per argument role of the moved object, and for steps demanding two or more atoms of the mover (conj.) on the containment-heavy set (150 plans). ^{z} height analytic; ^{s} conditioned on subject-role atoms only. 3 seeds.

	per predicate				role of the mover		
Method	on	in	left	south	subject	relatum	conj.
B_2 random valid$^{\flat}$	0.000	0.000	0.502	0.488	0.495	0.493	0.236
LINGO-Space^{z,s}	0.961	1.000	0.612	0.600	0.923	0.285	0.421
Diffusion-CCSP	0.837	0.973	0.973	0.945	0.956	0.959	0.856
StructDiffusion	0.713	0.992	0.964	0.963	0.961	0.965	0.859
PLAN2POSE (ours)	0.996	0.988	0.951	0.973	0.973	0.953	0.911

Table 11: Compounding on 17–20-step plans. Per-atom satisfaction teacher-forced (TF, earlier steps at ground truth) and autoregressive (AR, earlier steps at the method's own placements) for steps 1–4 and 17–20; the xy deviation from the demonstrated position written into the next step's reference (median over steps, and fraction above 100 mm) at steps 1–4 and 17–20; collateral violation (CV). 3 seeds.

	steps 1–4		steps 17–20		xy deviation p50 [mm]		xy deviation >100 mm		
Method	TF	AR	TF	AR	1–4	17–20	1–4	17–20	CV 17–20
LINGO-Space^{z,s}	0.596	0.550	0.616	0.460	122.5	150.0	0.600	0.738	0.287
Diffusion-CCSP	0.956	0.918	0.955	0.787	76.2	98.1	0.352	0.485	0.152
StructDiffusion	0.950	0.873	0.956	0.760	83.2	112.9	0.411	0.569	0.170
PLAN2POSE (ours)	0.970	0.969	0.964	0.946	27.4	41.5	0.031	0.086	0.029

E.2 NEW PREDICATES

After fine-tuning on the two new relations *near* and *right_of*, both are satisfied as often as the pre-trained ones (Table 12). On the in-distribution set, which contains neither new relation, per-atom satisfaction under autoregressive rollout is 0.9617 against 0.9616 for the model before fine-tuning.

E.3 EXECUTION IN THE LOOP

We place the methods in a TAMP loop with the executor of subsection B.3 (Figure 7). At each step, a method proposes B candidate poses for the moved object: its own prediction first, then $B-1$ further candidates, for our model positions drawn from its predicted field and for the diffusion baselines further samples. The executor tries the candidates and accepts an executable one, and the rollout continues from the accepted placement. A task counts as a strict success if every accepted placement passes the executor, and as a chain success if every step's atoms hold along the executed chain. Table 13 reports both on 40 plans of 13–16 states for $B = 1, 5, 20$. Without search ($B = 1$), our model succeeds in 0.300 (strict) and 0.250 (chain) of the plans, above every baseline even at $B = 20$ (at best 0.225 and 0.150). More candidates raise our strict success to 0.375 but lower chain success to 0.200: the additional candidates execute, but satisfy the plan's atoms less often. These runs use one seed, and most differences between budgets lie within the interval of the $B = 1$ rate.

Table 14 reports execution on the 40 grasp-blocking problems, where the first placement must leave a second object graspable. Our first placements execute less often at the first attempt than those of the diffusion baselines (0.794 against 0.975 and 1.000), but more of the demanded steps are executed (0.617 against 0.500 and 0.487). No method completes a plan, while the demonstrations complete all of them.

Table 12: New predicates. Arm 23 fine-tuned from PLAN2POSE with two relations absent from pretraining (near, right_of); per-atom satisfaction teacher-forced and autoregressive on the new-predicate set (mean [range] for the new relations) and, for the carried-over relations, on the in-distribution set as a forgetting control. 3 seeds.

		new-predicate set		in-distribution set	
Predicate	in pretraining	TF	AR	TF	AR
near	no	0.974 [0.964–0.982]	0.970 [0.962–0.976]	–	–
right_of	no	0.966 [0.963–0.969]	0.955 [0.953–0.957]	–	–
left	yes	0.968	0.961	0.967	0.953
south	yes	0.979	0.974	0.974	0.969
on	yes	0.988	0.992	0.969	0.971

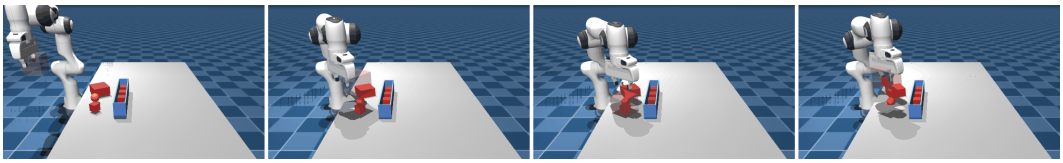

Figure 7: The executor in simulation: a Franka Panda arm picks a block and places it at the target pose (transparent) of one step.

F PRESCIENT REFINEMENT

F.1 TASK FAMILIES AND PROTOCOL

Platform. A plan places k blocks on the top face of a platform, and the directional relations of the later steps decide on which side of the first block the others go. Each scene is posed with two futures that mirror each other: they share the initial state and differ only in the side the later blocks take, and their extendable regions for the first block are disjoint. In distribution, $k = 2$, 3 or 4 (300 pairs of futures); out of distribution, $k = 6$ on a larger 2×3 platform (100 pairs). A first placement is extendable if it satisfies its own atoms and the remaining blocks can still be placed entirely on the top face, which we certify geometrically on a 2 mm grid. With the clearance between slots first registered, 10 mm, almost every first placement stayed extendable for $k = 2$ and 3 (only 0.012 of the satisfying placements did not), so we reduced it to 6 mm (0.111 at $k = 2$, 0.556 at $k = 4$) before any model was evaluated.

Bridge. Two pillars are placed, and the last step names a short (90 mm) or a long (170 mm) lintel that must rest on both pillars with each end at least 5 mm inside the supports. The two futures of a scene differ only in the lintel, and the pillar spacings that allow the short (60–105 mm) and the long lintel (120–145 mm) do not overlap. There are 100 pairs of futures.

Protocol. The criterion and the reading rule were written down before any model was evaluated on these families, and every later change was dated: the 6 mm clearance, mirror futures instead of nested extendable regions, the regeneration of both families before any result entering the rule was computed, and dropping an executed column for the bridge, whose demonstrations execute in only 156 of 200 cases. The rule compares our model with NF and with the masked model at $k = 4$, at $k = 6$ and on the bridge; by this rule, foresight is not claimed at $k = 6$.

F.2 ROBUSTNESS

Every seed of our model exceeds 0.5 at $k = 4$ (0.915, 0.895, 0.870) and on the bridge (0.990, 1.000, 0.945), and no seed of NF or of the masked model does; at $k = 6$, one seed reaches 0.315 and the others stay below 0.05. Permuting the identity labels at test time changes these rates to 0.870, 0.990 and 0.735 at $k = 4$ and to 0.960, 0.995 and 0.980 on the bridge, with NF still below 0.5, so the model does not read the future through the labels.

Table 13: Search against the candidate budget B on test_len_13_16; final model and composition only, one seed, $n = 40$ episodes per row, every budget of a row from one run. Candidate 1 is each method's own prediction; the further $B-1$ are, for PLAN2POSE, xy drawn from its predicted placement distribution (seeded per query; height and rotation from its prediction); for Diffusion-CCSP and StructDiffusion, further samples from their own sampler; for rejection sampling (oracle), further draws that satisfy the step's atoms (at most 50). Strict task success (every accepted placement passes the executor) and chain success (every step's atoms hold along the continued chain). The B=5 and B=20 rates lie inside the 95 % Wilson interval of the row's B=1 rate, except the strict success of Diffusion-CCSP (above) and the B=20 chain success of rejection sampling (below). No demonstration ceiling is available for this set: search is not defined for a single demonstration.

	strict			chain		
Method	B=1	B=5	B=20	B=1	B=5	B=20
Rejection sampling[♭]	0.025[†]	0.075[†]	0.075[†]	0.050[†]	0.025[†]	0.000[†]
Diffusion-CCSP	0.075[†]	0.225[†]	0.225[†]	0.125[†]	0.100[†]	0.150[†]
StructDiffusion	0.075[†]	0.150[†]	0.150[†]	0.025[†]	0.025[†]	0.025[†]
PLAN2POSE	0.300[†]	0.375[†]	0.375[†]	0.250[†]	0.225[†]	0.200[†]

Table 14: Execution in the TAMP loop on the 40 certified grasp-blocking problems (kinematic-grasp executor: physics on, object welded to the end-effector during transport, so grasp stability is not simulated). First-attempt executable at R=0; steps executed over steps demanded; plans completed (every step executed and the plan's own atoms hold). Demonstrations are the ceiling on the same problems.

Method	first-attempt executable (R=0)	steps executed / demanded	plans completed
Demonstrations[♭]	1.000	1.000	1.000
Rejection sampling (oracle)[z♭]	0.950	0.500	0.000
Diffusion-CCSP	0.975	0.500	0.000
StructDiffusion	1.000	0.487	0.000
PLAN2POSE (ours)	0.794	0.617	0.000

The bound of 0.5 for a placement that ignores the future rests on the full-support criterion: under it, no first placement is extendable under both futures of any scene (Table 15). If a later block only needs its *on* atom or a stable centre of mass, every scene has such a placement and the bound no longer holds; LINGO-Space, whose first placements all lie within 8.5 mm of the platform's centre line, then reaches 1.000. Our model stays above NF under every criterion (at $k = 4$: 1.000 and 0.995 against 0.737 and 0.613), but the difference shrinks, so the foresight result is specific to full support.

At $k = 6$, reading our model's composed field directly shows that it still puts more mass into the extendable region of the future it is shown than into the other one (by 0.142, against 0.149 at $k = 4$); the decoded placement nevertheless is extendable in only 0.115 of the cases. The failure out of distribution therefore arises when a single position is decoded from the field. This reading is exploratory; it was added after an intermediate preview.

F.3 KINEMATIC FORESIGHT

On the 40 grasp-blocking problems, the first placement must leave a second object graspable, a constraint that no atom of the plan states (Table 16). Masking the future lowers the satisfaction of the current step from 0.700 to 0.583 in distribution and from 0.300 to 0.067 out of distribution, while shuffling the future events leaves it unchanged (0.717 and 0.317). Among satisfying placements, the second object remains graspable in 0.571 and 0.944 of the cases, against 0.614 and 0.952 for NF. Removing the future changes the placement, but the model does not anticipate a constraint that the plan does not state.

Table 15: Extendable rate of the first placement (final checkpoints, the placements scored for Table 2) under three readings of "a later block fits". *full*: the registered criterion (platform: every later cube fully on the top face; bridge: both lintel ends 5 mm inside the supports). *label*: a later block only needs its *on* atom at the labelling threshold (overlap ≥ 0.25 of its own footprint; bridge: the lintel's *on* and left/south atoms). *stab.*: each later block's centre of mass lies over the supporting face (bridge: the lintel's atoms hold, its centre lies between the supports and it overlaps nothing; no 5 mm margin). The first placement's own *on* atom is required in every reading. Last line: scenes (of 100) in which some first placement, on a 4 mm lattice (bridge: within 10 cm of the demonstrated pillar), is extendable under both futures. Under the full criterion there is none, which caps a placement that ignores the future at 0.5; under the relaxed readings every scene has one, so that cap no longer holds. The full column reproduces Table 2 exactly. Learned rows: mean over 3 seeds; references: one draw. n/a: the LINGO-Space adapter cannot move a bridge pillar (as in Table 2).

	Platform k=2			Platform k=4			Platform k=6 OOD			Bridge		
Row	full	label	stab.	full	label	stab.	full	label	stab.	full	label	stab.
NULL A[♭]	0.355	0.935	0.780	0.095	0.835	0.625	0.090	0.680	0.475	0.070	0.175	0.175
Demonstrations[♭]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
LINGO-Space	0.000	1.000	1.000	0.000	1.000	1.000	0.000	1.000	1.000	n/a	n/a	n/a
Diffusion-CCSP	0.095	0.627	0.518	0.000	0.532	0.410	0.265	0.835	0.500	0.245	0.607	0.607
StructDiffusion	0.405	0.790	0.500	0.172	0.210	0.210	0.108	0.650	0.353	0.260	0.532	0.532
PLAN2POSE no future (NF)	0.332	0.737	0.613	0.327	0.737	0.613	0.098	0.832	0.713	0.458	0.815	0.815
PLAN2POSE future masked	0.175	0.952	0.777	0.172	0.952	0.777	0.007	0.985	0.983	0.397	0.835	0.835
PLAN2POSE (ours)	0.780	1.000	0.990	0.893	1.000	0.995	0.115	1.000	1.000	0.978	0.983	0.983
Both-futures placement exists (scenes of 100)	0	100	100	0	100	100	n/c	100	100	0	100	100

Table 16: Kinematic foresight on the 40 grasp-blocking problems: satisfaction of the first step with the future plan shown, masked or shuffled, and whether a second object remains graspable, among satisfying placements (count in parentheses) and over all placements. In distribution: two- and four-block families; out of distribution: a wide four-block and the five-block family. The baselines read no future; for NF, masking and shuffling change nothing.

	In distribution					Out of distribution				
	First step satisfied			Graspable		First step satisfied			Graspable	
Method	shown	masked	shuffled	if sat. (n)	all	shown	masked	shuffled	if sat. (n)	all
NULL A[♭]	1.000	–	–	0.517 (20)	0.517	1.000	–	–	0.590 (20)	0.590
NULL B[♭]	0.283	–	–	1.000 (17)	0.933	0.233	–	–	0.857 (14)	0.683
Diffusion-CCSP	0.417	–	–	0.640 (25)	0.717	0.383	–	–	0.826 (23)	0.833
StructDiffusion	0.600	–	–	0.833 (36)	0.900	0.283	–	–	0.882 (17)	0.950
PLAN2POSE no future (NF)	0.733	0.733	0.733	0.614 (44)	0.717	0.350	0.350	0.350	0.952 (21)	0.917
PLAN2POSE (ours)	0.700	0.583	0.717	0.571 (42)	0.683	0.300	0.067	0.317	0.944 (18)	0.917
Demonstrations[♭]	1.000	–	–	1.000 (20)	1.000	1.000	–	–	1.000 (20)	1.000

G ABLATIONS

Table 17 changes one variable of the final recipe at a time. Three changes cause large losses: composing the fields as a mixture instead of a product (-0.11 to -0.13 on TP and -0.44 to -0.48 on FS), training without the replay pool (-0.27 to -0.32 and -0.54 to -0.58), and an absolute instead of a relative step index (-0.25 to -0.50 on FS). Removing carried atoms, a smaller transformer and a primary share of 0.75 cost less but consistently. Without future context, TP barely changes, but FS drops by 0.068 and 0.100 on the two longer plan lengths. A pre-specified check of where this gain arises did not find it concentrated in steps whose placement depends on the future, so we do not attribute it to foresight. Future dropout and a lower learning rate have no significant effect; dropping the label permutation changes only the held-out five-block family ($+0.037$). Using the general pool as primary helps slightly on TP but costs 0.241 on FS for the five-block family. Finally, fine-tuning on the 79 recorded real captures (a quarter of each batch, 100 further epochs) lowers autoregressive per-atom satisfaction on the 51 held-out captures from 0.869 to 0.817; all real-data results in the paper are therefore zero-shot.

Table 17: Ablations, each arm against PLAN2POSE at the arm's own matched epoch ($^{m\,e}$); a positive difference favours the arm. Bold: the paired cluster-bootstrap 95% interval (over episodes, 10,000 draws) excludes zero. Rows with fewer than three seeds on either side are not rendered.

	ΔTP			ΔFS			
Arm	9–12	13–16	17–20	9–12	13–16	17–20	n5line
Composition and head							
mixture instead of product	**-0.109**	**-0.109**	**-0.129**	**-0.443**	**-0.483**	**-0.458**	**-0.173**
no carried atoms	**-0.011**	**-0.017**	**-0.016**	**-0.105**	**-0.160**	**-0.165**	**-0.092**
no base factor	-0.002	**-0.004**	-0.003	**-0.055**	**-0.058**	**-0.065**	**+0.110**
per-predicate decoders	**-0.002**	**-0.004**	**-0.003**	-0.030	-0.018	-0.025	**-0.055**
scalar height$^{m\,125}$	-0.002	**-0.003**	**-0.003**	-0.022	-0.038	-0.037	**-0.110**
no offset loss$^{m\,125}$	**-0.002**	**-0.005**	**-0.004**	**-0.047**	**-0.047**	**-0.050**	+0.025
no argument-slot channel$^{m\,125}$	**-0.003**	-0.001	**-0.002**	**-0.053**	+0.003	-0.015	+0.037
$\alpha \equiv 1$	**-0.005**	**-0.005**	**-0.003**	-0.028	**-0.042**	-0.040	**-0.071**
no point-cloud geometry	-0.002	**-0.006**	**-0.003**	+0.018	-0.037	-0.033	**-0.088**
Encoder and plan conditioning							
absolute step index	**-0.017**	**-0.076**	**-0.110**	**-0.253**	**-0.497**	**-0.500**	**-0.063**
no future context	-0.001	**-0.005**	**-0.004**	-0.015	**-0.068**	**-0.100**	**-0.049**
future dropout 0.5	+0.001	-0.001	0.000	-0.005	0.000	+0.012	-0.012
small trunk$^{m\,150}$	**-0.009**	**-0.010**	**-0.014**	**-0.085**	**-0.092**	**-0.130**	+0.004
Data							
no replay (primary only)	**-0.269**	**-0.294**	**-0.316**	**-0.543**	**-0.580**	**-0.537**	**-0.116**
primary share 0.25	-0.001	**-0.005**	**-0.004**	-0.027	-0.033	-0.038	-0.012
primary share 0.75	**-0.012**	**-0.016**	**-0.013**	**-0.100**	**-0.152**	**-0.123**	**+0.039**
general pool as primary$^{m\,100}$	**+0.004**	**+0.004**	**+0.007**	+0.047	+0.040	+0.038	**-0.241**
no label permutation	+0.001	-0.002	0.000	+0.007	0.000	0.000	**+0.037**
Optimisation							
learning rate $5 \cdot 10^{-5}$	-0.001	-0.001	+0.001	-0.008	+0.008	-0.012	-0.027

H DEPLOYMENT AND TRAINING COST

PLAN2POSE places an object with one forward pass followed by a deterministic decode, with no iterative sampling, denoising or rejection loop, so its cost per placement does not depend on how hard the constraints are to satisfy. All methods were measured back to back in one session on a single NVIDIA L4 GPU, as deployed in our rollouts: batch size 1, the same query, 100 timed calls after 10 warm-up calls, three repeats (Table 18). No other process was observed on the GPU during the session. PLAN2POSE takes 29.9 ms per placement, 18.6 times less than StructDiffusion and 29.2 times less than Diffusion-CCSP, which run an iterative denoising procedure for every placement. LINGO-Space takes about half as long, since it predicts the parameters of a small mixture of polar distributions rather than dense fields. Rejection sampling is cheap because it tests candidates against the predicates' exact geometry, which no learned method has. For a 16-step plan, refinement takes 0.48 s with PLAN2POSE, against 8.9 and 14.0 s for the two diffusion baselines.

Table 19 lists model sizes and training times. PLAN2POSE has 24.0 M parameters and trains in 11.7 h on an NVIDIA L40S or 19.2 h on an A10, with a peak memory of 12.4 GiB.

Table 18: Latency per placement on one NVIDIA L4, every row in one session (session 2) on the same query (first admitted step of a 14-step plan with five objects), final checkpoints; batch 1, 100 timed calls after 10 warm-up, three repeats, median [min–max] over repeats. Latency is a property of the architecture, not of the training seed. A reporting table, not a criterion.

Method	latency [ms], median [min–max]
LINGO-Space	16.4 [16.3–16.4]
Rejection sampling	17.0 [17.0–17.1]
PLAN2POSE (ours)	29.9 [29.8–30.2]
StructDiffusion	557.2 [544.8–562.3]
Diffusion-CCSP	873.7 [872.3–892.9]

Table 19: Model size and training cost per method. Parameters counted on the model each adapter builds; LINGO-Space's total includes the frozen CLIP ViT-L/14 text encoder its shim loads, which is not trained. Training wall-clock per checkpoint from the scheduler, mean [min–max] over three seeds, with the GPU class the job ran on. Rejection sampling has no weights and draws up to the stated number of candidates per placement, accepting by oracle.

Method	trainable params	total params	training [h]	GPU	draws / placement
StructDiffusion	4,334,469	4,334,469	5.2 [3.6–6.4]	A10	1
StructDiffusion (pool-only)	4,334,469	4,334,469	0.4 [0.4–0.5]	A10	1
Diffusion-CCSP	2,697,221	2,697,221	14.0 [13.5–14.8]	A10	1
Diffusion-CCSP (pool-only)	2,697,221	2,697,221	1.9 [1.8–2.2]	A10	1
LINGO-Space (compute-matched)	27,189,847	150,840,151	8.7 [7.9–9.1]	A10	1
LINGO-Space (pool-only)	27,189,847	150,840,151	9.3 [8.1–9.9]	A10	1
Rejection sampling	0	0	–	–	50
PLAN2POSE (ours)	24,041,961	24,041,961	16.7 [11.7–19.2]	A10, L40S	1

Table 20: Reproducibility of PLAN2POSE. The pinned seeds (mean, and their range over three seeds) against two retrains of the same recipe, per criterion and rung; * marks a retrain mean outside the pinned seeds' range at the printed precision. Hardware: pinned s0 on an L40S, s1 and s2 on A10s; the retrains all-A10 and all-L40S, so the hardware-matched pairs are pinned s0 with the L40S retrain and pinned s1/s2 with the A10 retrain.

	TP			FS			MS			CV ↓		
Row	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20	9–12	13–16	17–20
pinned seeds	0.972	0.971	0.966	0.557	0.585	0.542	0.962	0.963	0.958	0.022	0.022	0.029
range over seeds	.968–.974	.970–.972	.964–.969	.500–.610	.560–.605	.535–.550	.958–.965	.961–.964	.956–.962	.019–.025	.022–.023	.025–.031
A10 retrain	0.971	0.969*	0.963*	0.552	0.560	0.503*	0.961	0.959*	0.955*	0.022	0.025*	0.031
L40S retrain	0.970	0.967*	0.964	0.525	0.587	0.532*	0.959	0.958*	0.956	0.023	0.027*	0.031