# RelArt: Relation-Aware Articulated Object Synthesis from Executable Structural Priors

**Anonymous authors**
Paper under double-blind review

## Abstract

Existing methods for articulated object generation often predict joint structure and part geometry together from limited observations. The observations may not determine hidden joint axes or the space needed for parts to move, and errors in the predicted structure can cause disconnected parts or collisions. We therefore use an executable structural prior to specify the articulation structure instead of inferring it from the observation alone. We study geometry generation conditioned on this prior, which specifies parts, joints, interfaces, and motion constraints. Our framework, RelArt, includes an agent-based module for constructing these priors and an autoregressive model for generating part geometry. The construction module derives relations between parts from template code and a small set of valid parameter instances. It executes each proposed instance, checks the relations, and repairs parameters that affect failed checks. The verified relations also form the graph supplied to the geometry model. When generating a part, the decoder retrieves its relations to previously generated neighbors and pairs each neighbor's relations with that neighbor's generated geometry. This gives the decoder both the required connection and the geometry to which the new part must fit. With the same image and structural prior, RelArt improves F-score from 0.733 to 0.788 and motion success from 0.781 to 0.869 over the strongest existing generator in our comparison. In a separate construction experiment with equal proposal and executor-call budgets, repairing parameters that affect failed checks produces more valid articulated instances and covers more templates than resampling all parameters.

## 1 Introduction

Articulated object generators often predict part geometry and joint structure together from limited observations Lei et al. (2023); Su et al. (2025); Liu et al. (2025). A single view may be consistent with several joint configurations: appearance alone may not determine hidden joint axes, internal clearances, or motion limits. Errors in these estimates can produce disconnected parts or collisions even when individual part shapes look plausible.

We study geometry generation from an executable structural prior that specifies part roles, joints, interfaces, and motion constraints. Given this prior and a visual observation, the generator must produce part surfaces that meet at the specified interfaces and move without prohibited collisions. The prior supplies the structure; the generator must produce geometry that satisfies it, as illustrated in Figure 1.

Constructing a structural prior manually for every object requires repeated specification of parts, joints, and interfaces. Inspired by code-based representations Jones et al. (2020); Sun et al. (2024; 2025), we start from reusable parameterized templates. Our agent-based Relation Induction and Verification module derives constraints between parts from template code and a small set of valid parameter instances. It proposes parameter values, executes the template, and checks the resulting instance. When a check fails, the module traces the quantities used by that check to the template parameters and repairs the parameters that can affect the result. The resulting relation graph supports both prior verification and geometry generation.

The structural prior specifies how parts should connect and move, while previously generated geometry determines the surfaces and space available to each new part. A drawer, for example, must fit the generated cabinet interior and follow the supplied sliding axis. RelArt therefore conditions each



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.

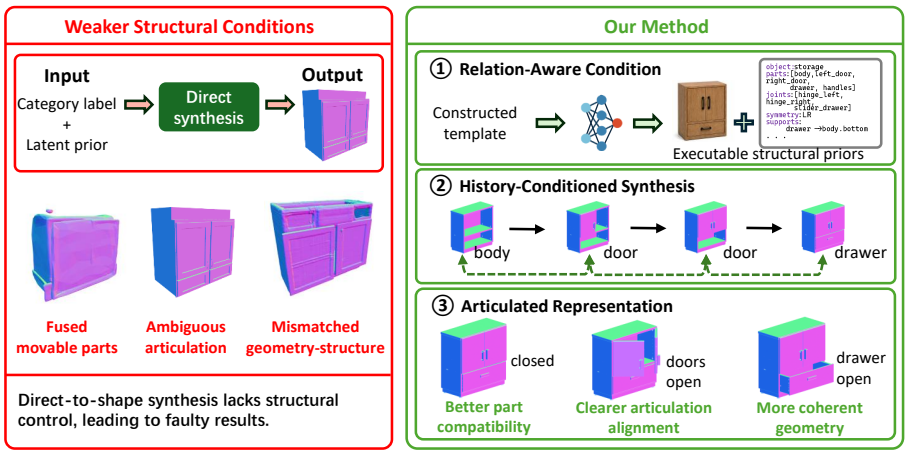



Figure 1: Motivation and method overview. The left panel illustrates disconnected or fused parts and ambiguous articulation when the input does not specify joints and interfaces. The right panel shows generation from an executable structural prior. RelArt uses the geometry of previously generated parts when generating each new part.

generation step on both the specified relations and the geometry of previously generated parts. The decoder retrieves relations between the current part and its generated neighbors, pairs each relation descriptor with the corresponding neighbor geometry, and uses these pairs to predict the new surface.

In controlled comparisons, each generator receives the same image and structural prior. We measure agreement with reference geometry and satisfaction of the supplied constraints. Under these inputs, RelArt improves geometric accuracy and motion success and reduces interface distance relative to the compared generators.

Our contributions are as follows. First, we formulate articulated object generation from executable structural priors. We separate the underdetermined problem of structural inference from geometry generation through a generative task based on executable structural priors. Second, we propose an agent-based method for relation induction and prior expansion. The method automatically generates and verifies parameter instances in batches to construct executable structural priors that directly guide geometry generation. Third, we propose an autoregressive generation method jointly conditioned on relations and generation history. This lets the structural prior guide specific connections and motion interfaces at every step, improving the structural consistency of the generated result.

We evaluate prior construction separately from decoding with fixed priors so that changes to the input structure are not attributed to the decoder.

## 2 Related Work

### 2.1 Articulated Object Generation

NAP models part geometry and articulation graphs Lei et al. (2023); CAGE conditions on connectivity graphs Liu et al. (2024), SINGAPO uses an input image Liu et al. (2025), and ArtFormer represents geometry and kinematics in a tree Su et al. (2025). A comparison using each method's original inputs measures both the effect of those inputs and the behavior of the generator. We therefore also compare generators after supplying the same image and executable structural prior to each method.

### 2.2 Executable Structure Construction

ShapeAssembly represents part structures as executable assembly programs Jones et al. (2020), and ConceptFactory uses reusable parameterized templates Sun et al. (2024). Arti-PG describes artic-

ulated objects through generalized structure programs Sun et al. (2025). Real2Code predicts articulation code from reconstructed parts Zhao et al. (2025); URDF-Anything and URDF-Anything+ construct articulated models Li et al. (2025); Wu et al. (2026), while Articraft writes and validates asset programs Zhou et al. (2026). We use template code and valid instances to derive relations, verify them through execution, and trace failed checks to parameters that can be repaired. The resulting graph is also an input to the geometry generator.

### 2.3 Structure-Conditioned Part Geometry

StructureNet and PQ-NET model part hierarchies or sequences Mo et al. (2019a); Wu et al. (2020); progressive assembly studies part compatibility Narayan et al. (2022); Chen et al. (2022), and MeshArt conditions part-mesh generation on local connectivity to previously generated parts Gao et al. (2025). Our decoder pairs each typed relation from an executable structural prior with the geometry of the generated neighbor to which it applies. The construction module verifies those same relations. We test the decoder with a fixed prior so that its gains can be distinguished from gains due to additional constructed relations. Appendix A.4 discusses further work.

## 3 Problem Formulation

The construction stage takes a reusable geometry template $T$, a set of valid parameter instances $D_T$, and, for articulated templates, a joint adapter $J_T$ checked against reference articulation states. The adapter computes part transforms from joint frames, types, and ranges. The construction stage proposes parameter values $x$, executes the template and joint adapter to obtain a base prior $C_0(T, x; J_T)$, and adds verified part relations to form $C$. The synthesis stage takes a visual observation $O$ and a supplied or constructed prior $C$, and generates the articulated object $\hat{\mathcal{A}}$. The construction stage does not infer $C$ from $O$. We report articulation metrics only for the subset $\mathcal{T}_{\text{kin}}$ of templates whose joint adapters provide the frames, joint types, and ranges needed to compute part motion. Templates without these adapters are evaluated only for spatial relations. In fixed-prior experiments, $O$ and $C$ describe the same test object. When constructing a new asset, we render $O$ from the template geometry produced by $x$. We define the prior as

$$C = (C_0, \mathcal{R}), \qquad C_0 = (V, E, F, K, B, \Omega),$$

where $V$ lists the parts to be generated, $E$ specifies their parent--child kinematic hierarchy, $F$ contains part coordinate frames, $K$ contains joint parameters, $B$ describes support regions, and $\Omega$ contains interface and motion constraints. A part slot in $V$ specifies a part before its final geometry is generated. For $T \in \mathcal{T}_{\text{kin}}$, the geometry template and $J_T$ together produce these fields. The graph $\mathcal{R} = \{r_{ij}\}$ represents typed relations for attachment, containment, support, alignment, symmetry, stacking, size compatibility, front-surface placement, and motion limits. Each edge is

$$r_{ij} = (\rho_{ij}, m_{ij}, s_{ij}, \gamma_{ij}),$$

where $\rho_{ij}$ is the relation type, $m_{ij}$ contains numerical attributes such as normalized distances and margins, $s_{ij}$ identifies the template parameters that can affect the relation check when those dependencies are known, and $\gamma_{ij}$ records confidence or validity. We call the parameter set $s_{ij}$ the relation's repair scope. Several relation checks may connect the same part pair, so $\mathcal{R}$ is a typed multigraph; we omit the edge index when it is unnecessary. Repair scopes are used only for parameter repair and are not passed to the geometry encoder. Static analysis identifies scopes for relations supplied by the base prior and for newly induced relations. A relation without a verified scope can condition geometry generation and be checked, but a violation of that relation cannot initiate parameter repair. Parameter proposal, execution checks, and repair with a finite budget define a distribution $q_\psi(C \mid T, D_T)$ over priors that pass the construction checks. Given a constructed or manually specified prior $C$, geometry synthesis is defined by

$$p_\theta(\hat{G}_{1:N} \mid O, C) = \prod_{t=1}^{N} p_\theta\Big(\hat{G}_{\pi_t} \mid Z_g, z^C_{\pi_t}, z^R_{\pi_t}, H_{<t}\Big),$$

where $\pi$ is the generation order determined by structural dependencies, $Z_g$ contains encoded image and prior features, and $z^C_{\pi_t}$ represents the current part descriptor. The history $H_{<t}$ contains feature

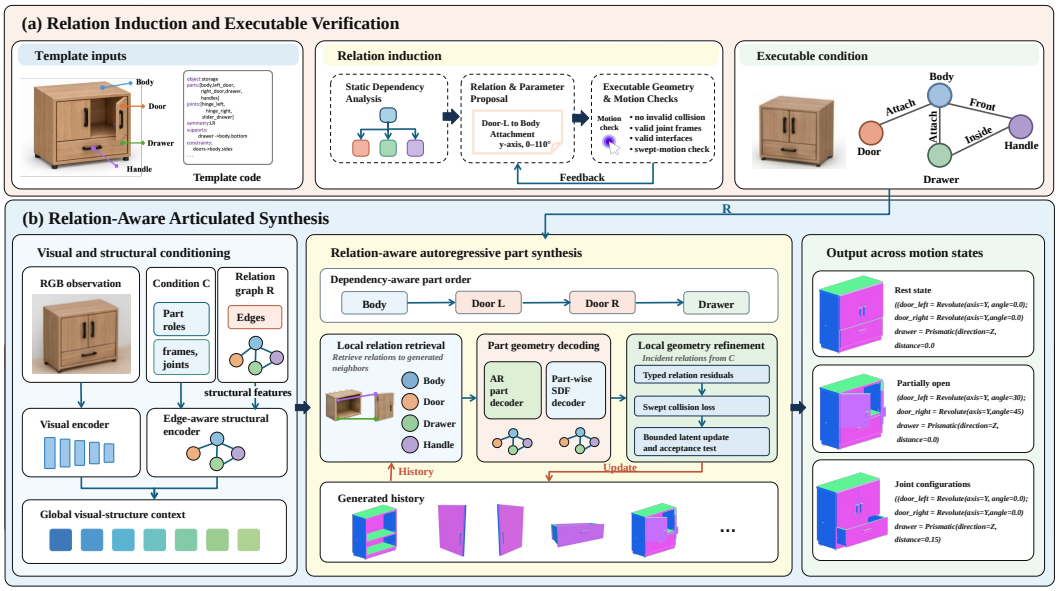



Figure 2: Overview of RelArt. The construction module derives part relations from template code and valid parameter instances, then executes templates to check the relations. The generator uses the resulting relation graph, image features, and representations of previously generated parts to predict each new part. The optional local geometry refinement step adjusts the current part's SDF latent vector after decoding. This step is disabled in fixed-prior comparisons.

vectors computed from previously generated parts. The vector $z^R_{\pi_t}$ combines relations to those parts with their geometry features, as defined in Section 4.4. The output $\hat{\mathcal{A}} = \{\hat{G}_i, K_{ij}\}$ consists of generated part geometry and the supplied joint parameters. The objective is to match the observation and satisfy the prior's constraints throughout its motion range. We hold $C$ fixed across generators in controlled synthesis comparisons and evaluate changes to $C$ in separate construction experiments.

# 4 Method

## 4.1 Relation Induction and Verification

The Relation Induction and Verification module uses the template's abstract syntax tree and valid parameter instances to identify relations between parts. Static analysis records which parameters affect each part, frame, joint, and interface. The valid instances provide candidate equalities, inequalities, axis correspondences, and ratios between dimensions. Using these records, the LLM proposes relation types and repair scopes. Each proposed relation is expressed as a deterministic relation check from the supported residual library. The construction executor runs the template, computes joint transforms through $J_T$ when available, evaluates the relation check, and reports any violation together with a verified repair scope when one is available. The LLM proposes relations; the executor determines whether an instance satisfies them. Appendix A.2 specifies the construction executor and the separate evaluation checker. Each relation is represented by a relation check $f_k(C) \leq 0$ or $f_k(C) = 0$. Static analysis identifies the parameters and frames used by part constructors and, for templates in $\mathcal{T}_{\text{kin}}$, by the joint adapter. The module proposes relations only over these variables. A candidate relation's empirical support is the fraction of valid seed instances that satisfy its relation check within the tolerance for that relation type. We select this tolerance and the minimum support on training templates and keep them fixed for held-out templates. We reject a candidate relation if it fails an executor check or contradicts a higher-confidence relation check designated as required. If the seed instances provide insufficient evidence, the module proposes no relation. Each retained relation stores its type, numerical attributes, confidence, and verified repair scope, if available.

### 4.2 Parameter Repair within Verified Scopes

We sample parameter values in dependency order, choosing parent dimensions before the dimensions of dependent openings, panels, and handles. The template builds each candidate instance; when $J_T$ is available, the executor also checks the instance at sampled articulation states. If a relation check fails, the module traces its inputs back to the parameters that can affect the check. It changes continuous parameters only within their verified repair scopes and template bounds, and enumerates the allowed discrete alternatives. After changing the parameters, it executes the template again and repeats every affected relation check. It rejects a candidate if required constraints remain violated after at most three repair rounds. When the prior was constructed by this module, violations found after geometry generation may trigger parameter repair followed by another generation pass. Parameter repair after generation is disabled in fixed-prior comparisons because it changes the generator's input. Appendix A.2 gives the dependency and acceptance rules. For templates in $\mathcal{T}_{\text{kin}}$, parameter values that pass the construction checks define $C_{\text{agent}} = (C_0(T, x; J_T), \mathcal{R}_{\text{agent}}(x))$. Evaluation checks more articulation states than construction; neither finite set establishes collision-free motion at every state in a continuous range.

### 4.3 Structural Interface and Relation Encoder

We convert priors from different template formats into a common set of descriptors. The mapping $\Phi$ extracts relations already specified by the base prior, giving $\mathcal{R}_{\text{fix}} = \Phi(C_0)$. The construction module adds relations derived from template code and valid instances, giving $\mathcal{R}_{\text{agent}}(x) = \Phi(C_0(T, x; J_T)) \cup \mathcal{R}_{\text{new}}(T, D_T; x)$. Here $C_0 = (V, E, F, K, B, \Omega)$ is obtained by executing the template with $x$ and, when available, its joint adapter $J_T$. The mapping $\Phi$ uses only the part, frame, joint, support, interface, and motion fields supplied to every model in the controlled comparison. It does not use additional examples, template source code, or test geometry. The construction module removes duplicate or conflicting relation checks before adding new relations. The two graphs use the same edge format and encoder but provide different information. Decoder comparisons keep $O$, $C_0$, and $\mathcal{R}_{\text{fix}}$ fixed. Separate experiments compare the two graph sources and the components of prior construction. We parse the executable structural prior $C$ into shared descriptors:

$$S_C = (V, E, F, K, B, \Omega, \mathcal{R}),$$

where $\mathcal{R}$ contains kinematic and spatial relations. Appendix A.3 gives the full schema. For each part $v_i$, we construct a part descriptor

$$c_i = [r_i, F_i, B_i, \eta_i],$$

where $r_i$ is the part role, $F_i$ is the canonical frame, $B_i$ is the support descriptor, and $\eta_i$ contains local attributes. For each typed relation between $i$ and $j$, we construct an extended edge descriptor

$$\tilde{e}_{ij}^{(k)} = [\tau_{ij}, p_{ij}, a_{ij}, \ell_{ij}, R_{ij}, \rho_{ij}^{(k)}, m_{ij}^{(k)}, \gamma_{ij}^{(k)}],$$

where $\tau_{ij}$ is the joint type, $p_{ij}$ its pivot or origin, $a_{ij}$ its axis, $\ell_{ij}$ its motion range, and $R_{ij}$ the interface region. These five fields apply when the pair has a joint interface. The remaining fields specify the relation type, normalized numerical attributes, and confidence. A mask marks fields that do not apply to a pair. The interface computes the generation order $\pi$ from support and kinematic dependencies. For example, it schedules a cabinet body before its doors and drawers, and schedules handles after their parent panels. The order determines which part is generated at each step; it is not supplied as a separate feature to the global encoder. The structural encoder embeds part descriptors and typed relation edges, aggregates multiple edges on each part pair, and exchanges information through the graph. It produces part features $z_i^C$ and pair features $g_{ij}^C$. The visual backbone and cross-attention combine image and structure features into $Z_g$. The encoder is trained with the geometry generator and does not reconstruct the supplied prior. Appendix A.5 defines the feature representations.

### 4.4 Relation-Aware History-Conditioned Synthesis

RelArt generates parts sequentially according to the order $\pi$. The conditional distribution is factorized as

$$p_\theta(\hat{G}_{1:N} \mid O, C) = \prod_{t=1}^{N} p_\theta\Big(\hat{G}_{\pi_t} \mid Z_g, z_{\pi_t}^C, z_{\pi_t}^R, H_{<t}\Big),$$

where the local conditioning vector $z^R_{\pi_t}$ uses only relation edges between the current part and previously generated parts:

$$z^R_{\pi_t} = \mathrm{RelAttn}\big(z^C_{\pi_t}, \{[g^C_{\pi_t j}, h_j] \mid j \in \pi_{<t} \cap \mathcal{N}_{\mathcal{R}}(\pi_t)\}\big)\,.$$

Here $\pi_{<t}$ contains the indices of previously generated parts, and $\mathcal{N}_{\mathcal{R}}(\pi_t)$ contains the parts connected to the current part by relation edges. Attention uses only neighbors in both sets. If there are none, $z^R_{\pi_t}$ is a learned vector representing the absence of a generated relation neighbor. The geometry-history encoder computes $h_j$ from samples of part $j$'s SDF in its canonical frame. This vector summarizes the geometry retained for that part, including any refinement that passes the criteria in Appendix A.6. Appendix A.8.3 specifies the encoder. For each generated neighbor $j$, the retrieval module concatenates the pair feature $g^C_{\pi_t j}$ with the geometry feature $h_j$. The former describes the required relations; the latter describes the geometry already generated for that neighbor. After decoding the current part, relation-specific residuals measure whether its geometry satisfies those requirements. These residuals provide training supervision and can also be used in the optional refinement and repair procedures. The autoregressive decoder and part-wise SDF decoder use the current part descriptor, global context, generated-part history, and $z^R_{\pi_t}$ to predict geometry in the specified canonical frame. For a door, the retrieved features can describe the hinge, the required alignment, and the generated body. For a drawer, they can describe containment, sliding clearance, and the generated cabinet interior. The history encoder then computes $h_{\pi_t}$ from the current part's SDF and stores it for subsequent generation steps. If refinement is enabled, the encoder uses the refined SDF only when the refinement passes the criteria in Appendix A.6; otherwise it uses the original SDF. Appendices A.8.3 and A.7 specify the decoders and geometry losses. After all parts have been generated, the executor checks their interfaces, containment, and, for templates in $\mathcal{T}_{\mathrm{kin}}$, motion constraints. When parameter repair after generation is enabled, a failed relation with a verified repair scope triggers a parameter change, template re-execution, and another generation pass. A relation without a verified scope is checked but cannot initiate repair. This procedure changes the prior and is disabled in fixed-prior comparisons. To obtain the object at articulation state $q$, we transform each generated part by $T_i(q; C)$. These transforms change part poses while preserving their canonical geometry. An optional local refinement can adjust a decoded SDF latent vector while holding the prior and previously generated parts fixed. It is disabled in fixed-prior comparisons; Appendix A.6 gives its objective and acceptance rule.

### 4.5 Training Objective

RelArt is trained with losses for part reconstruction, articulation consistency, motion consistency, and relation satisfaction:

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{art}}\mathcal{L}_{\mathrm{art}} + \lambda_{\mathrm{mot}}\mathcal{L}_{\mathrm{mot}} + \lambda_{\mathrm{rel}}\mathcal{L}_{\mathrm{rel}}.$$

Here $\mathcal{L}_{\mathrm{rec}}$ supervises part geometry, $\mathcal{L}_{\mathrm{art}}$ penalizes errors in support, joint-interface alignment, and part separation, and $\mathcal{L}_{\mathrm{mot}}$ penalizes inter-part penetration at sampled articulation states. The relation loss $\mathcal{L}_{\mathrm{rel}}$ is a weighted average of relation-specific residuals, defined in Eq. equation 9; it is zero for an empty graph. Type weights and normalization are fixed across graph sources. Averaging prevents edge count alone from multiplying the loss, although adding edges can change its value. In fixed-prior comparisons, the loss uses $\mathcal{R}_{\mathrm{fix}}$ and can supervise a decoder even when the decoder does not receive relation features. Every variant marked ``+ loss'' uses the same relation loss and coefficient. Experiments with $\mathcal{R}_{\mathrm{agent}}(x)$ use the same residual functions and weights. Appendix A.7 gives the definitions.

## 5 Experiments

### 5.1 Experimental Setup

**Dataset and splits.** The fixed-prior synthesis benchmark uses articulated PartNet-Mobility categories Xiang et al. (2020) and the base autoregressive generator's 5:1 train--test split. Every object used for motion evaluation has a joint specification checked against reference articulation states. The separate construction study uses partitions that hold out parameter regions, template programs grouped by normalized signatures, combinations of relation types, or categories composed of relation types seen in training. Except in the parameter-region partition, near-duplicate programs and

all their instances remain in the same partition. Motion scores for constructed priors use only $\mathcal{T}_{\text{kin}}$; templates without joint specifications contribute only to spatial-relation analysis. Test images are rendered from the corresponding objects with fixed cameras and lighting. Appendix A.8.3 describes the partitions and numbers of seed instances.

**Baselines and controlled comparisons.** The main controlled comparison gives NAP Lei et al. (2023), CAGE Liu et al. (2024), ArtFormer Su et al. (2025), MeshArt Gao et al. (2025), and our model the same image $O$ and executable structural prior. The prior includes part roles, hierarchy, joints, interfaces, and $\mathcal{R}_{\text{fix}}$. We convert these fields to each method's conditioning format and train a projection where needed, while retaining its generation backbone. The methods receive the same information through different representations. The base autoregressive generator and our model share the visual backbone and part-wise SDF decoder. These comparisons use no induced relations, local geometry refinement, or prior repair. Appendix Table 5 compares ways of encoding and retrieving relations within the base generator architecture, using the same data and optimization budget. A global-fusion control adds attention capacity comparable to that used by step-wise retrieval. Appendix Tables 3 and 4 separately report NAP, ArtFormer, SINGAPO Liu et al. (2025), and URDF-Anything+ Wu et al. (2026) with each method's original inputs. Those tables compare complete systems, including differences in input information. Another experiment evaluates both graph sources on the same target objects with each generator. For NAP, part slots become graph nodes, joint descriptors become directed articulation edges, and spatial relations become edge features. For CAGE, part and joint fields enter its controllable articulation representation, and a conditioning projection adds relation attributes. ArtFormer receives part slots and joint descriptors in dependency order, with relation features associated with the relevant part pairs. MeshArt receives the shared hierarchy and joint edges, with relation attributes mapped by a projection. The methods receive no additional labels, target geometry, or template code.

**Metrics.** Minimum Matching Distance and Coverage computed with Instantiation Distance (ID-MMD and ID-COV) compare generated and reference object sets. Part Overlapping Ratio (POR) measures inter-part overlap. F-score and Chamfer Distance (CD) compare each generated object with its corresponding reference; State-CD repeats this comparison at synchronized articulation states. Interface distance (Intf.), collision rate across evaluated states (Coll.), and motion success measure whether the geometry satisfies the supplied interfaces and motion constraints. Collision rate records whether a prohibited collision occurs at any evaluated state; it does not establish collision-free motion between those states. Construction metrics are articulation validity, template coverage, valid yield, and runtime. Rates lie in $[0, 1]$ unless stated otherwise, and distances use normalized object coordinates. Static CD is displayed in units of $10^{-3}$; State-CD is reported without that display scaling. Results are averaged within categories and then equally across categories. Appendix A.8.3 defines the metrics and their denominators.

**Implementation details.** The within-family controlled models use a frozen DINOv2 ViT-B/14 image encoder, 256-dimensional structure and relation tokens, the same part-wise SDF decoder, and 4096 SDF queries per part. The structural encoder has four transformer layers, and the autoregressive decoder has six. They are trained for 500 epochs with a global batch size of 32 on four NVIDIA RTX 4090 (24GB) GPUs. Appendix A.8.3 describes the architecture and preprocessing.

### 5.2 Comparison with Existing Methods

Table 1 holds the image and prior fixed across its listed generators. Figure 3 shows methods using their original inputs, while Figure 4 and Appendix Figures 10 and 11 show comparisons with a shared structural prior. Appendix Table 5 compares relation encodings within one generator architecture under the same training data and optimization budget. The left panel of Figure 5 displays representative outputs from that controlled comparison. With the same structural input, the relation-aware model improves F-score from 0.732 to 0.788, reduces State-CD from 0.125 to 0.078, and raises motion success from 0.742 to 0.869 relative to the base autoregressive generator. It also exceeds MeshArt, the strongest existing generator in this comparison, by 0.055 in F-score and 8.8 percentage points in motion success. Appendix Table 5 shows that step-wise retrieval exceeds global fusion with comparable capacity.

Table 1: Generation from the same image and executable structural prior. Part roles, joints, interfaces, and typed relations are mapped to each method's conditioning input; generation backbones are retained.

| Method | F-score ↑ | CD ↓ | State-CD ↓ | Intf. ↓ | Coll. ↓ | Motion ↑ |
|---|---|---|---|---|---|---|
| NAP Lei et al. (2023) | 0.692 | 5.35 | 0.142 | 0.031 | 0.159 | 0.721 |
| CAGE Liu et al. (2024) | 0.703 | 5.18 | 0.134 | 0.029 | 0.148 | 0.736 |
| ArtFormer Su et al. (2025) | 0.724 | 4.82 | 0.112 | 0.027 | 0.131 | 0.769 |
| MeshArt Gao et al. (2025) | 0.733 | 4.65 | 0.105 | 0.026 | 0.120 | 0.781 |
| Base autoregressive generator | 0.732 | 4.96 | 0.125 | 0.037 | 0.187 | 0.742 |
| RelArt (ours) | 0.788 | 3.94 | 0.078 | 0.019 | 0.096 | 0.869 |

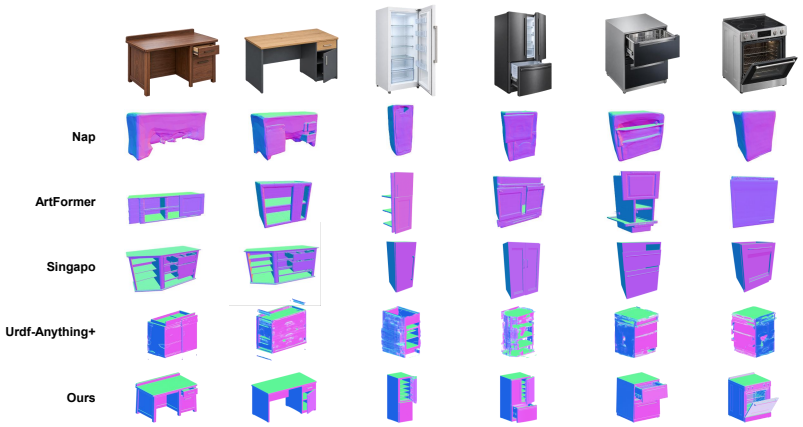



Figure 3: Examples generated with each method's original inputs. The rows compare complete systems, including differences in available input information.

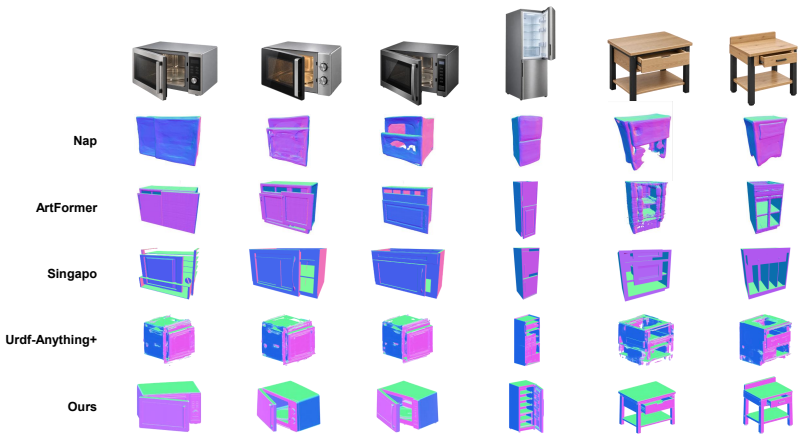



Figure 4: Examples generated with the same structural prior in open configurations, showing interior geometry and movable-part placement.

Appendix Table 6 adds structural fields in stages, giving all methods the same information at each stage. The ``+ Relation graph'' column represents relations already specified by the prior as explicit typed edges; it adds no independent annotations. With the full prior, step-wise retrieval achieves relation satisfaction of 0.90, compared with 0.81 for flat relation encoding.

Figure 3 compares complete systems with their original inputs; Figure 4 uses a shared structural prior. Several displayed baseline outputs retain a front panel but omit or distort the surrounding body, while the displayed RelArt outputs retain more of the body and the placement of movable

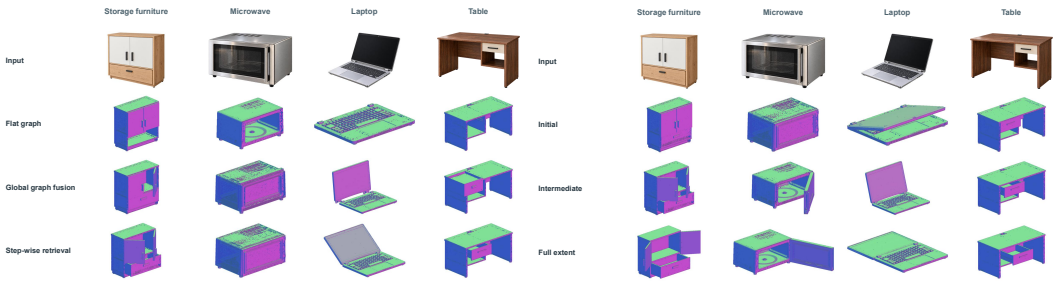



Figure 5: Qualitative results for four object categories. Left: fixed-prior relation-encoding variants. Right: reference images and RelArt outputs at three joint states. Test-set metrics are reported in Appendix Table 5 and Table 1.

parts. Open configurations also expose differences in interior geometry. Appendix Tables 3 and 4 report original-input metrics, while Table 1 reports controlled metrics for its listed methods. These static views do not establish collision-free motion.

In the left panel of Figure 5, flat graph encoding omits movable parts in some examples and global fusion leaves an interface incomplete; step-wise retrieval connects those parts to the body. The right panel shows geometry hidden in closed views.

**Prior construction and repair.** With equal budgets, repair within verified scopes improves articulation validity from 0.74 to 0.82 and template coverage from 0.70 to 0.84 over full-parameter resampling (Appendix Table 12). Per-object runtime excludes authoring the reusable templates and joint adapters.

**Scope of the evaluation.** The fixed-prior comparison isolates geometry generation: every model receives the same image and structural input, and no method uses prior repair in this comparison. The separate construction study tests expansion of an existing executable template library. Its held-out-program split uses new programs built from supported relation types, while the category-held-out split tests new combinations of known part roles and relations. Neither split evaluates unseen joint types. The construction runtime excludes the work of writing templates and joint adapters, so it does not measure the total labor required to extend the library. Motion success is checked at finitely many articulation states; passing that test does not establish collision-free motion throughout a joint range. Further details of the splits and checks appear in Appendix A.8.3.

**Limits of repair.** Parameter repair changes only values within verified scopes and cannot change the topology of a template. It may also change the prior while leaving the image fixed. Improved constraint satisfaction in the repair experiment therefore does not establish improved agreement with the image. The optional local geometry refinement is disabled in fixed-prior comparisons and is not evaluated separately.

## 6 Conclusion

RelArt addresses articulated-object geometry generation from an image and an executable structural prior, separating it from image-only inference of hidden articulation. Its construction module derives part relations from reusable templates, verifies proposed instances by execution, and repairs parameters that affect failed checks. Its autoregressive decoder conditions each part on relations to generated neighbors and their geometry. With fixed images and priors, RelArt improves F-score from 0.733 to 0.788 and motion success from 0.781 to 0.869 over the strongest compared generator. Separately, under equal budgets, repair within verified parameter scopes improves articulation validity from 0.74 to 0.82 and template coverage from 0.70 to 0.84. The motion results concern sampled states and do not establish collision-free motion throughout joint ranges.

## Ethical Statement

There are no ethical problems.

## Use of LLMs

An LLM was used to propose relation types and parameter dependencies in the construction module. A deterministic executor checked the proposed relations. LLM assistance was also used for language editing. The authors are responsible for the final content.

## References


Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. In *arXiv preprint arXiv:1512.03012*, 2015.

Yun-Chun Chen, Haoda Li, Dylan Turpin, Alec Jacobson, and Animesh Garg. Neural shape mating: Self-supervised object assembly with adversarial shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939--5948, 2019.

Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G. Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*, 2016.

Bi'an Du, Xiang Gao, Wei Hu, and Renjie Liao. Generative 3d part assembly via part-whole-hierarchy message passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. Meshart: Generating articulated meshes with structure-guided transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. GET3D: A generative model of high quality 3D textured shapes learned from images. In *Advances in Neural Information Processing Systems*, 2022.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mache approach to learning 3d surface generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, et al. Carto: Category and joint agnostic reconstruction of articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. Shapeassembly: Learning to generate programs for 3d shape structure synthesis. In *ACM SIGGRAPH Asia*, 2020.

Heewoo Jun and Alexander Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Jiahui Lei, Congyue Deng, William B. Shen, Leonidas J. Guibas, and Kostas Daniilidis. Nap: Neural 3d articulated object prior. In *Advances in Neural Information Processing Systems*, 2023.

Zhe Li, Xiang Bai, Jieyu Zhang, Zhuangzhe Wu, et al. URDF-Anything: Constructing articulated objects with 3D multimodal language model. *arXiv preprint arXiv:2511.00940*, 2025.

Di Liu, Anastasis Stathopoulos, Qilong Zhangli, Yunhe Gao, and Dimitris Metaxas. Lepard: Learning explicit part discovery for 3d articulated shape reconstruction. In *Advances in Neural Information Processing Systems*, 2023a.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023b.

Jiayi Liu, Hou In Ivan Tam, Ali Mahdavi-Amiri, and Manolis Savva. Cage: Controllable articulation generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Jiayi Liu, Denys Iliash, Angel X. Chang, Manolis Savva, and Ali Mahdavi-Amiri. SINGAPO: Single image controlled generation of articulated parts in objects. In *International Conference on Learning Representations*, 2025.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy J. Mitra, and Leonidas J. Guibas. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics*, 38(6):242:1--242:19, 2019a.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 909--918, 2019b.

Abhinav Narayan, Rajendra Nagar, and Shanmuganathan Raman. Rgl-net: A recurrent graph learning framework for progressive part assembly. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5762--5772, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaa El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Xiaoqian Ruan, Pei Yu, Dian Jia, Hyeonjeong Park, Peixi Xiong, and Wei Tang. Learning partonomic 3d reconstruction from image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jiayi Su, Youhe Feng, Zheng Li, Jinhua Song, et al. Artformer: Controllable generation of diverse 3d articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jianhua Sun, Yuxuan Li, Longfei Xu, Nange Wang, Jiude Wei, Yining Zhang, and Cewu Lu. Conceptfactory: Facilitate 3d object knowledge annotation with object conceptualization. In *Advances in Neural Information Processing Systems*, 2024.

Jianhua Sun, Yuxuan Li, Jiude Wei, Longfei Xu, Nange Wang, Yining Zhang, and Cewu Lu. Artipg: A toolbox for procedurally synthesizing large-scale and diverse articulated objects with rich annotations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6396--6405, 2025.

Xi Tian, Yong-Liang Yang, and Qi Wu. ShapeScaffolder: Structure-aware 3D shape generation from text. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2715--2724, 2023.

Penghao Wang, Siyuan Xie, Hongyu Yan, Xianghui Yang, Jingwei Huang, Chunchao Guo, and Jiayuan Gu. Artllm: Generating articulated assets via 3d llm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Ruiqi Wu, Xinjie Wang, Liu Liu, Chunle Guo, Jiaxiong Qiu, Chongyi Li, Lichao Huang, Zhizhong Su, and Ming-Ming Cheng. Dipo: Dual-state images controlled articulated object generation powered by diverse data. In *Advances in Neural Information Processing Systems*, 2025.

Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Zhuangzhe Wu, Yue Xin, Chengkai Hou, Minghao Chen, et al. URDF-Anything+: End-to-end generation for simulation-ready articulated assets. *arXiv preprint arXiv:2603.14010*, 2026.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Mandi Zhao, Yijia Weng, Dominik Bauer, and Shuran Song. Real2Code: Reconstruct articulated objects via code generation. In *International Conference on Learning Representations*, 2025.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, et al. Articraft: An agentic system for scalable articulated 3d asset generation. *arXiv preprint arXiv:2605.15187*, 2026.

# A Appendix

## A.1 Relation Illustration

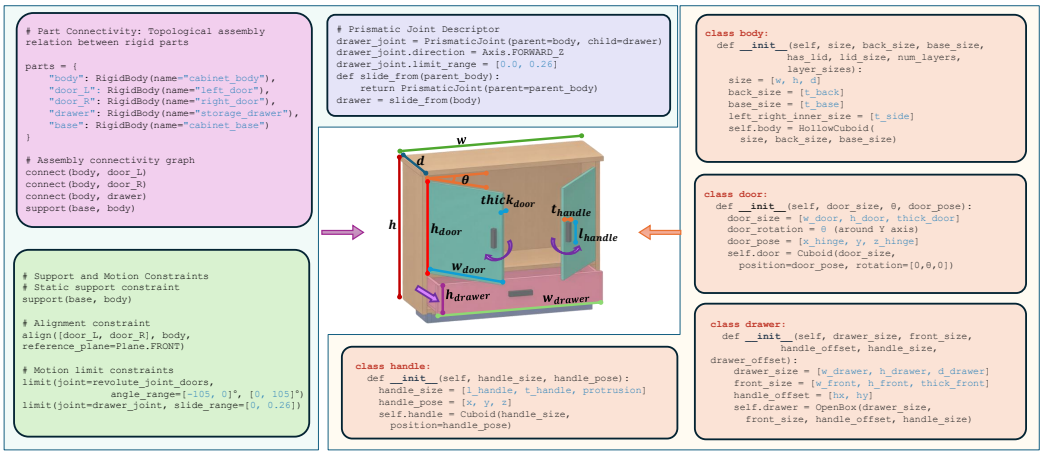



Figure 6: Example structural prior and parameterized part templates. The diagram specifies part connections, joint motion, and support constraints alongside schematic code for constructing each part. Labels on the cabinet identify dimensions and a door rotation angle. The snippets illustrate the representation and are not complete executable programs.

## A.2 Executor, Joint Adapter, and Checker Specification

**Executor interface.** The construction executor is a deterministic program runtime. Given a template $T$, parameters $x$, an optional joint adapter $J_T$, candidate relation checks $\mathcal{R}$, and construction states $Q_{\text{con}}$, it returns

$$\text{Exec}(T, x, J_T, \mathcal{R}; Q_{\text{con}}) = (C_0, \mathcal{M}, \mathbf{e}, \mathcal{V}, \mathcal{S}, \sigma)\,. \tag{1}$$

Here $C_0$ is the instantiated prior, $\mathcal{M} = \{M_i\}$ contains part meshes in their canonical frames, and $\mathbf{e} = (e_k)$ contains normalized relation residuals. The set $\mathcal{V} = \{k : e_k > \epsilon_{\rho_k}\}$ identifies failed relation checks, and $\mathcal{S} = \{S_k\}$ contains verified repair scopes. The status $\sigma$ records acceptance or a failure in parsing, geometry, joints, or relation checks. Thresholds $\epsilon_\rho$ are selected for each relation type on training templates and then held fixed. Each call records the template, parameter values, part identifiers, residuals, repair scopes associated with failed relation checks, sampled states, and status. These records allow the resulting prior to be reconstructed and checked without another LLM call. Before running $T(x)$, the executor checks parameter types, bounds, and required fields. The template produces part geometry and named coordinate frames. For articulated templates, $J_T$ associates joints with those frames and computes part transforms at the states in $Q_{\text{con}}$. The executor evaluates each relation check on the resulting geometry and transforms. A template without a joint adapter cannot be classified as articulation-valid. Before a candidate relation can enter $\mathcal{R}_{\text{agent}}$, its relation check must belong to the supported residual library, its participating parts must exist, and its numerical attributes must be computable.

**Joint adapter.** The adapter record for an edge contains the parent and child identifiers, joint type, home transform, pivot $p$, unit axis $a$, and range $[q^-, q^+]$. For a revolute joint its relative transform is

$$T_{ij}(q) = T_{ij}^0\,\text{Trans}(p)\,\text{Rot}(a, q)\,\text{Trans}(-p), \tag{2}$$

and for a prismatic joint it is $T_{ij}(q) = T_{ij}^0\,\text{Trans}(qa)$. Fixed joints use $T_{ij}^0$. Composing these transforms along the rooted kinematic tree gives $T_i(q; C)$. A template enters $\mathcal{T}_{\text{kin}}$ only if every referenced part and frame exists, each axis has unit length, each joint range has finite ordered limits, and the adapter reproduces the reference transforms at held-out articulation states. Templates failing any requirement are evaluated only for spatial relations.

Table 2: Residuals for the supported relation types. $[z_i^-, z_i^+]$ and $e_i$ are the axial interval and projected extent of part $i$; $m_{ij}$ stores margins $\delta$ and permitted ranges.

| Relation | Normalized residual $e_k$ | Required quantities |
|---|---|---|
| Attachment | $d_{\leftrightarrow}(P_i^k, P_j^k)$, restricted to the named contact patches | interface patches |
| Containment | $\lvert P_i\rvert^{-1} \sum_{p \in P_i} \max(0, S_{\mathrm{free}(j)}(p))$ | inner surface, parent free volume |
| Support | normalized vertical gap plus $1 - \mathrm{IoU}(\Pi_z P_i^k, \Pi_z P_j^k)$ | support patches, gravity axis |
| Alignment | normalized center/boundary discrepancy plus $1 - \lvert a_i^\top a_j\rvert$ when axes are specified | selected centers, axes, or boundaries |
| Symmetry | $d_{\leftrightarrow}(P_i, \mathrm{Reflect}_{a,\delta}(P_j))$ | plane normal, offset |
| Stacking | hinge penalties for incorrect order and gaps outside the permitted interval | stack axis, axial intervals |
| Size compatibility | $\sum_d \max(0, e_{i,d} - e_{j,d} - \delta_d)$ over constrained projected axes | projected extents, margins |
| Front-surface placement | normalized signed-depth error plus $1 - n_i^\top n_j$ | target surface, outward normals |
| Motion limitation | maximum non-contact penetration over $q \in Q_{\mathrm{con}}$ plus travel outside $[q^-, q^+]$ | joint transforms, contact masks |

**Static dependency scopes.** We construct a dependency graph from the template's abstract syntax tree. Its edges track how sampled parameters enter part-constructor arguments, coordinate frames, interfaces, and joint fields through assignments and function calls. We unroll loops whose iteration bounds are known statically and include dependencies from all conditional branches. For relation check $k$, we trace the geometric and joint quantities used by the check back to the source parameters; these parameters form $S_k$. A proposed repair scope is verified only if it contains every parameter found by this trace and every parameter it names can affect a quantity used by the check. If dynamic reflection, unresolved function calls, or data-dependent indexing prevent the trace, the relation has no verified repair scope. It can still condition generation and be checked, but its violation cannot initiate parameter repair.

**Typed residuals.** Let $P_i^k$ contain samples from the designated interface patch of part $i$, $S_j$ be the signed distance field of part $j$ with negative values inside, $d_{\leftrightarrow}$ be bidirectional mean surface distance, and $\Pi_a$ denote projection orthogonal to axis $a$. The operator $\mathrm{Reflect}_{a,\delta}$ reflects points across a plane with normal $a$ and offset $\delta$. Table 2 defines the residuals. Distances are divided by the object bounding-box diagonal; angular terms are dimensionless. A relation check passes when its residual is at or below its frozen type-specific threshold.

**Repair within verified parameter scopes.** Repair changes template parameter values within their normalized bounds. It does not change the part topology, part identifiers, or joint types. For failed relation checks with verified repair scopes, it forms $S = \bigcup_{k \in \mathcal{V}_S} S_k$, where $\mathcal{V}_S \subseteq \mathcal{V}$ contains the failed checks with verified scopes; parameters outside $S$ remain fixed. The algorithm uses projected coordinate search for continuous parameters in $S$, since template execution need not be differentiable, and enumerates discrete values from their allowed sets. Each proposed parameter change is evaluated by executing the template again and repeating the affected checks. Candidates are ranked first by the number of violated required constraints, then by the weighted residual $\sum_{k \in \mathcal{V}} w_k e_k$, and finally by the normalized change from the original parameter values. The algorithm keeps a candidate only if it improves this ranking and introduces no new required-constraint violation. Repair stops when the instance passes the checks, no improving candidate is found, the executor-call budget is exhausted, or three rounds have been completed. Failed relation checks without verified scopes still prevent acceptance, but they do not identify parameters that the repair algorithm may change.

**Independent evaluation checker.** The evaluation checker receives generated meshes, part identifiers, joint transforms checked against reference states, and metric thresholds fixed before testing. It does not receive LLM confidence values, inferred repair scopes, construction residuals, repair records, or executor decisions. It measures mesh gaps and intersections at the states in $Q_{\mathrm{eval}}$, which include the range endpoints and more interior states than $Q_{\mathrm{con}}$. Contact masks identify the surface regions where contact is permitted; intersections elsewhere on the same part pair are still checked. An instance that passes the construction checks may therefore fail evaluation. Evaluation results are not used to choose construction candidates or to accept geometry refinement.

## A.3 Executable Structural Prior

### A.3.1 Definition

An executable structural prior supplies part slots, coordinate frames, joints, interfaces, and motion constraints to the generator. A part slot is a record specifying one part to be generated; it is not the part's final mesh. The prior may be supplied for a target object or constructed from a template and verified relations. It specifies interface geometry and joint transforms while leaving the final part surfaces to the generator. We represent the prior as

$$C = (V, E, F, K, B, \Omega, \mathcal{R}), \tag{3}$$

where $V = \{v_i\}_{i=1}^{N}$ denotes the set of part slots, $E \subseteq V \times V$ denotes parent-child structural and articulation relations, $F = \{F_i\}_{i=1}^{N}$ denotes canonical part frames, $K = \{K_{ij}\}_{(i,j)\in E}$ denotes joint descriptors, $B = \{B_i\}_{i=1}^{N}$ denotes support descriptors, and $\Omega$ denotes interface, separation, and motion constraints. The relation graph $\mathcal{R}$ contains typed spatial and kinematic relations beyond the parent-child tree. Each part $v_i$ corresponds to a part slot. It specifies the role of the part, its canonical coordinate frame, its geometric prior, and its local support or attachment conditions:

$$c_i = (r_i, F_i, B_i, \eta_i), \tag{4}$$

where $r_i$ is the part role, such as body, door, drawer, lid, or handle; $F_i \in SE(3)$ is the canonical part frame; $B_i$ describes support regions or supporting surfaces; and $\eta_i$ contains local attributes such as symmetry, attachment side, coarse scale priors, or visibility indicators. For each articulated relation $(i, j) \in E$, the prior provides a joint-interface descriptor

$$e_{ij} = (\tau_{ij}, p_{ij}, a_{ij}, \ell_{ij}, R_{ij}), \tag{5}$$

where $\tau_{ij}$ is the joint type, e.g., revolute, prismatic, or fixed; $p_{ij} \in \mathbb{R}^3$ is the joint origin or pivot; $a_{ij} \in \mathbb{S}^2$ is the motion axis; $\ell_{ij}$ is the valid motion range; and $R_{ij}$ describes the interface region between the two parts. These quantities specify a transform and motion interval for each joint. Given an articulation state $q$, the prior determines the kinematic transforms

$$T_i(q; C) \in SE(3), \qquad i = 1, \ldots, N, \tag{6}$$

through $E$ and $K$. These transforms define joint states within the supplied limits but do not guarantee collision-free generated geometry. The model must produce surfaces that fit the specified interfaces throughout the motion range. Executable structural priors compose reusable part and relation programs. A storage furniture prior, for example, can connect two doors to a body through revolute joints, a drawer through a prismatic joint, and handles to the movable panels. This specifies the topology and joint conditions for geometry generation. RelArt takes the topology, joint type, pivot, axis, and motion range from this prior and generates the part surfaces.

### A.3.2 Relation Vocabulary and Scope

We use nine relation families between parts. Attachment specifies contact between an attachment region and a parent surface. Containment keeps an inner occupied region within the free volume of an outer part. Support aligns a supporting top surface with the bottom surface of the supported part. Alignment relates selected centers, axes, or boundaries in a common frame. Symmetry reflects paired parts across a specified symmetry plane. Stacking specifies order and separation along an axis. Size compatibility bounds a dependent part by the corresponding projection of its parent. Front-surface placement specifies a surface and outward normal for front-facing components. Motion limitation constrains depth, sweep, and travel to prevent a moving part from crossing a back panel or colliding

with non-contact parts. Relations with verified repair scopes identify parameters that may be changed when a relation is violated. Relations without verified scopes can still condition generation and be checked. Let $x$ collect the template parameters and let $S_k$ identify the parameters that may be changed to repair relation $k$. Let $\mathcal{V}_S(x) \subseteq \mathcal{V}(x)$ contain the failed checks with verified repair scopes. Within the ranking and acceptance rules in Appendix A.2, coordinate search seeks to reduce the residual objective

$$\mathcal{E}_{\text{repair}}(y_S; x^{(n)}) = \sum_{k \in \mathcal{V}(x^{(n)})} w_k \, \ell_k(y_S, x^{(n)}_{\bar{S}}), \qquad y_S \in \mathcal{B}_S, \tag{7}$$

where $S$ is the union of the scopes in $\mathcal{V}_S(x^{(n)})$, $\mathcal{B}_S$ contains the allowed parameter ranges, and $\ell_k$ is the normalized residual of check $k$. The objective includes all currently failed checks, while only checks with verified scopes determine which parameters may change. The search prioritizes reducing the number of violated required constraints, uses the residual to break ties, and then prefers smaller parameter changes; it rejects changes that introduce a new required-constraint violation. Parameters outside $S$ remain fixed, and discrete alternatives are enumerated separately. For each candidate change, the executor rebuilds the geometry and repeats all affected checks. Required constraints that remain violated after three rounds cause rejection, including constraints without verified scopes. Restricting changes to $S$ preserves parameters outside the verified scopes and records which parameters each repair changed. The construction partitions separately test unseen parameter regions, held-out programs, new combinations of supported relations, and categories composed of relation types seen in training. The few-shot results below use held-out programs; they do not test unsupported part or joint types. We report template coverage alongside validity to distinguish success across many templates from success concentrated in a small subset. Validity and yield use all proposals or attempts as their denominators, as specified in Appendix A.8.3. The synthesis model supports a variable number of parts:

$$p_\theta(\hat{G}_{1:N} \mid O, C) = \prod_{t=1}^{N} p_\theta\Big(\hat{G}_{\pi_t} \mid Z_g, z^C_{\pi_t}, z^R_{\pi_t}, H_{<t}\Big). \tag{8}$$

The decoder uses the same learned network across supported relation types and part counts. A new template supplies a different graph and generation order without changing the decoder architecture.

### A.4 Additional Related Work

Articulated reconstruction offers another source of structural priors. PARIS, DITTO, and CARTO infer parts or joints from observations and interactions Liu et al. (2023b); Jiang et al. (2022); Heppert et al. (2023). Their estimates could condition synthesis after validation; errors in the estimated structure would still need to be distinguished from errors in the generated geometry.

ShapeNet and PartNet provide large shape and part collections Chang et al. (2015); Mo et al. (2019b). 3D-R2N2, AtlasNet, Occupancy Networks, DeepSDF, and IM-NET study reconstruction and shape representations Choy et al. (2016); Groueix et al. (2018); Mescheder et al. (2019); Park et al. (2019); Chen & Zhang (2019). GET3D, SDFusion, Shap-E, and Point-E model shape generation Gao et al. (2022); Cheng et al. (2023); Jun & Nichol (2023); Nichol et al. (2022). Our task additionally requires generated parts to satisfy supplied articulation constraints.

Part-whole message passing models geometric assembly Du et al. (2024). GAPartNet and LEPARD study parts for manipulation and reconstruction Geng et al. (2023); Liu et al. (2023a); ShapeScaffolder and partonomic reconstruction use part structure for shape generation and reconstruction Tian et al. (2023); Ruan et al. (2025). These studies address related reconstruction and assembly tasks. Our controlled comparison measures how different generators satisfy the same supplied executable structure.

PartNet-Mobility supplies articulated objects and joint annotations, while SAPIEN supports articulated interaction studies Xiang et al. (2020). Neural articulated fields model pose-dependent appearance and geometry Noguchi et al. (2021); DIPO uses dual-state images Wu et al. (2025), and ArtLLM generates articulated assets from 3D observations using a multimodal language model Wang et al. (2026). Our experiments use a supplied structure whose joint transforms have been checked against reference articulation states.

### A.5 Structural Encoder Details

The structural encoder computes part and pair feature vectors by exchanging information through the graph. It is trained jointly with the geometry generator. The prior is supplied as input; the model is not trained to reconstruct it. For each part $v_i$, we first embed the part descriptor:

$$h_i^0 = \phi_{\text{part}}\left(E_r(r_i), \phi_F(F_i), \phi_B(B_i), \phi_\eta(\eta_i)\right),$$

where $E_r$ is a learnable role embedding, and $\phi_F$, $\phi_B$, and $\phi_\eta$ are MLP encoders for the canonical frame, support descriptor, and local attributes. For each typed relation between $i$ and $j$, we embed the extended descriptor and aggregate edges on the same part pair:

$$g_{ij}^0 = \phi_{\text{pair}}\left(\sum_{k \in \mathcal{R}_{ij}} \phi_{\text{rela}}(\tilde{e}_{ij}^{(k)}), |\mathcal{R}_{ij}|\right),$$

where $\mathcal{R}_{ij}$ is the set of typed edges for the part pair $(i, j)$. The edge encoder uses embeddings for discrete relation and joint types and MLPs for continuous attributes. The pair projection $\phi_{\text{pair}}$ takes the sum of the edge features and the number of edges, and produces one vector for the pair. Thus all relations for the pair contribute to the vector used by graph attention. A transformer that uses part and pair descriptors computes part feature vectors $z_i^C$ and pair feature vectors $g_{ij}^C$. The visual backbone encodes $O$, and cross-attention combines the image and structure features into the global context $Z_g$. This context represents the image and prior before any geometry is generated. At each decoding step, the retrieval module in Section 4.4 also combines pair features with representations of the corresponding generated neighbors. Appendix A.8.3 gives encoder dimensions and training settings.

### A.6 Relation-Guided Local Geometry Refinement

The decoded surface may violate a relation even when the decoder receives that relation as input. An optional refinement step therefore adjusts the current part's SDF latent vector after decoding. Let $u_t^0$ be the predicted latent vector and let $D_{\text{sdf}}(u_t, c_{\pi_t})$ map a candidate vector $u_t$ to the current part's SDF in its canonical frame. Refinement changes only $u_t$. It holds the image features, structural prior, decoder weights, joint parameters, and previously generated parts fixed. It therefore cannot change the part slots or joint specification supplied by the prior.

$$u_t^* = \underset{\|u - u_t^0\|_2 \le \epsilon}{\arg\min}\ \lambda_{\text{int}}\mathcal{L}_{\text{int}}(u; \mathcal{R}_{\pi_t,<t}, H_{<t}) + \lambda_{\text{col}}\mathcal{L}_{\text{col}}(u; C, H_{<t}) + \lambda_{\text{keep}}\mathcal{L}_{\text{keep}}(u, u_t^0).$$

The interface loss measures differentiable relation residuals between the current part and its generated neighbors. For templates in $\mathcal{T}_{\text{kin}}$, the collision loss evaluates sampled states within the joint limits and penalizes penetration outside the specified contact patches. Allowing contact at a hinge does not exempt the rest of the two parts from this loss. The preservation loss limits changes to the SDF away from the interface and limits the change from $u_t^0$. We optimize $u_t$ using a bounded number of projected gradient steps. We retain the refined vector only if the residuals targeted by refinement decrease and the geometric change stays within a fixed tolerance. Otherwise, we retain $u_t^0$. We then compute the history feature from the SDF produced by the retained vector. The independent evaluation checker is not used to choose between the two vectors. Relations to parts that have not yet been generated are evaluated when those parts are decoded. Local geometry refinement changes the current part's SDF latent vector; prior repair changes template parameters and requires another execution of the template. Local geometry refinement is disabled in fixed-prior decoder comparisons.

### A.7 Training Objective and Inference

#### A.7.1 Training Objective

RelArt is trained with four objectives:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda_{\text{art}}\mathcal{L}_{\text{art}} + \lambda_{\text{mot}}\mathcal{L}_{\text{mot}} + \lambda_{\text{rel}}\mathcal{L}_{\text{rel}}.$$

The reconstruction term supervises part-level geometry:

$$\mathcal{L}_{\text{rec}} = \sum_{t=1}^{N} \mathcal{D}\left(\hat{G}_{\pi_t}, G_{\pi_t}^*\right).$$

In the controlled implementation, $\mathcal{D}$ is the mean absolute error between predicted and reference truncated signed distances at the same query points in each part's canonical frame. Reference distances come from training meshes; the truncation radius and near-surface sampling ratio are set on training data and kept constant during evaluation. Chamfer Distance is used for evaluation. The base autoregressive generator and its fixed-prior variants use the same reconstruction target. The articulation-consistency term is

$$\mathcal{L}_{\text{art}} = \mathcal{L}_{\text{sup}} + \alpha\mathcal{L}_{\text{joint}} + \beta\mathcal{L}_{\text{sep}}.$$

Here $\mathcal{L}_{\text{sup}}$ penalizes deviations from supplied support descriptors, $\mathcal{L}_{\text{joint}}$ aligns generated interfaces with specified joints, and $\mathcal{L}_{\text{sep}}$ penalizes fusion or penetration between movable parts and generated neighbors. The motion consistency term evaluates generated geometry under sampled articulation states:

$$\mathcal{L}_{\text{mot}} = \frac{1}{T}\sum_{m=1}^{T} \text{Pen}\left(\hat{A}(q_m)\right), \qquad q_m \sim \ell^C,$$

Here $\ell^C$ denotes the supplied motion ranges and $\text{Pen}(\cdot)$ measures inter-part penetration across sampled states. The relation realization term applies a differentiable residual appropriate to each edge type:

$$\mathcal{L}_{\text{rel}} = \begin{cases} \dfrac{\sum_{r_{ij}\in\mathcal{R}} w_{\rho_{ij}}\varphi_{\rho_{ij}}(\hat{G}_i, \hat{G}_j; m_{ij})}{\sum_{r_{ij}\in\mathcal{R}} w_{\rho_{ij}}}, & |\mathcal{R}| > 0, \\ 0, & |\mathcal{R}| = 0. \end{cases} \tag{9}$$

Each relation type has its own residual. Attachment measures bidirectional surface distance on the specified interface. Containment penalizes occupancy outside the parent's free volume. Support measures vertical gap and projected overlap; alignment measures discrepancies in specified centers, axes, or boundaries. Symmetry measures surface distance after reflection. Stacking penalizes incorrect order and gaps outside the permitted range. Size compatibility penalizes projected dimensions that exceed their bounds. Front-surface placement measures signed depth and normal alignment. Motion limitation penalizes penetration across sampled articulation states. Distance residuals are normalized by object scale before aggregation. Positive relation-type weights are selected on training data and held fixed for both graph sources. An empty graph contributes zero relation loss. Early in training, the history encoder uses reference part geometry. As training proceeds, scheduled sampling replaces an increasing fraction of these reference parts with generated parts, so the decoder also learns to use imperfect geometry from earlier generation steps.

### A.7.2 Detailed Formulation of Motion Consistency Loss

We implement $\mathcal{L}_{\text{mot}}$ as a differentiable collision loss on the predicted SDF at sampled articulation states.

**Coordinate Projection via Kinematic Transforms** Let $\mathcal{P}_i$ be a set of points sampled near the surface of part $i$ in its canonical space. For a sampled articulation state $q_m \sim \ell^C$, the kinematic transform $T_i(q_m; C)$ determines the world-space position of a point $p \in \mathcal{P}_i$. To evaluate collisions between parts $i$ and $j$, we project $p$ into the local canonical frame of part $j$:

$$p_{i\to j}(q_m) = T_j(q_m; C)^{-1} T_i(q_m; C) p, \tag{10}$$

where $T_j(q_m; C)$ is the kinematic transform for part $j$.

**Penetration Penalty** Following the convention that $S_j(x) < 0$ denotes the interior of a shape, we define the asymmetric penetration penalty for the ordered pair $(i, j)$ at state $q_m$ as:

$$L_{pen}(i, j, q_m) = \frac{1}{|\mathcal{P}_i|}\sum_{p\in\mathcal{P}_i} \max(0, -S_j(p_{i\to j}(q_m))), \tag{11}$$

where $S_j$ is the SDF predicted by the geometry decoder. The total motion consistency loss is aggregated across $T$ sampled states and all unordered pairs of distinct parts without permitted contact:

$$\mathcal{L}_{\text{mot}} = \frac{1}{T}\sum_{m=1}^{T} \sum_{\substack{i<j \\ (i,j)\notin E_{contact}}} (L_{pen}(i, j, q_m) + L_{pen}(j, i, q_m)), \tag{12}$$

Here $E_{contact}$ contains part pairs designated as permitting contact. The generic penetration loss excludes each such pair entirely. If the graph includes a motion-limit relation for the pair, that relation supplies a separate penalty for penetration during motion. Without that relation, the pair may have no training penalty for penetration across articulation states. The evaluation checker still checks every pair, allowing contact only at specified surface regions. The optional local geometry refinement step also uses spatial contact masks instead of excluding an entire pair.

#### A.7.3 Inference

At inference, RelArt receives an image $O$ and a supplied or constructed prior $C$. The interface extracts the part descriptors, relations, and generation order. The model encodes the image and full relation graph once per generation pass. Starting with an empty history $H_0$, the decoder retrieves the current part's relations to generated neighbors, pairs those relations with the neighbors' geometry features, and predicts the current part's SDF. If local refinement is enabled, it adjusts the current part's latent vector and compares the resulting geometry with the original prediction. It uses the refined geometry only if the targeted residuals decrease and the geometry change remains within the prescribed tolerance; otherwise it uses the original prediction. The history encoder computes a feature vector from the chosen SDF and stores it for subsequent parts. After the pass, the executor checks the relation residuals. For constructed priors with parameter repair after generation enabled, it sends failed relation checks with verified repair scopes to the construction module. That module changes the relevant parameter values, executes the template again, and starts another generation pass. Failed relation checks without verified repair scopes are reported but cannot initiate a parameter change. After all parts are generated, articulation states are expressed as

$$\hat{A}(q) = \{T_i(q; C)\hat{G}_i\}_{i=1}^{N}.$$

Fixed-prior comparisons use one generation pass and disable local geometry refinement and parameter repair after generation. Table 11 evaluates parameter repair after generation separately, keeping the generator, initial constructed prior, and test set fixed while varying the number of repair rounds. Because that experiment changes $C$, its gains are not attributed to the relation-aware decoder.

### A.8 Additional Experiments and Implementation Details

#### A.8.1 Comparisons with Original Method Inputs

These results compare complete systems under their original inputs, including differences in available information. They complement the controlled shared-prior comparison in the main text. RelArt has the lowest State-CD (0.078) and collision rate (0.096) in Table 4; Figure 7 compares visible geometry under the original inputs.

Table 3: Paired geometry with each method's original inputs. CD is displayed in units of $10^{-3}$.

| Method | F-score ↑ | CD ↓ |
|---|---|---|
| NAP Lei et al. (2023) | 0.670 | 6.18 |
| ArtFormer Su et al. (2025) | 0.709 | 5.36 |
| SINGAPO Liu et al. (2025) | 0.695 | 5.63 |
| URDF-Anything+ Wu et al. (2026) | 0.718 | 5.18 |
| Base autoregressive generator | 0.732 | 4.96 |
| RelArt (ours) | 0.788 | 3.94 |

Table 4: Articulation consistency with each method's original inputs. State-CD compares geometry at synchronized articulation states; Coll. is the fraction of objects with a prohibited collision at any evaluated state. These results compare complete systems.

| Method | State-CD ↓ | Coll. ↓ |
|---|---|---|
| NAP Lei et al. (2023) | 0.161 | 0.258 |
| ArtFormer Su et al. (2025) | 0.130 | 0.214 |
| SINGAPO Liu et al. (2025) | 0.141 | 0.229 |
| URDF-Anything+ Wu et al. (2026) | 0.123 | 0.201 |
| Base autoregressive generator | 0.125 | 0.187 |
| RelArt (ours) | 0.078 | 0.096 |

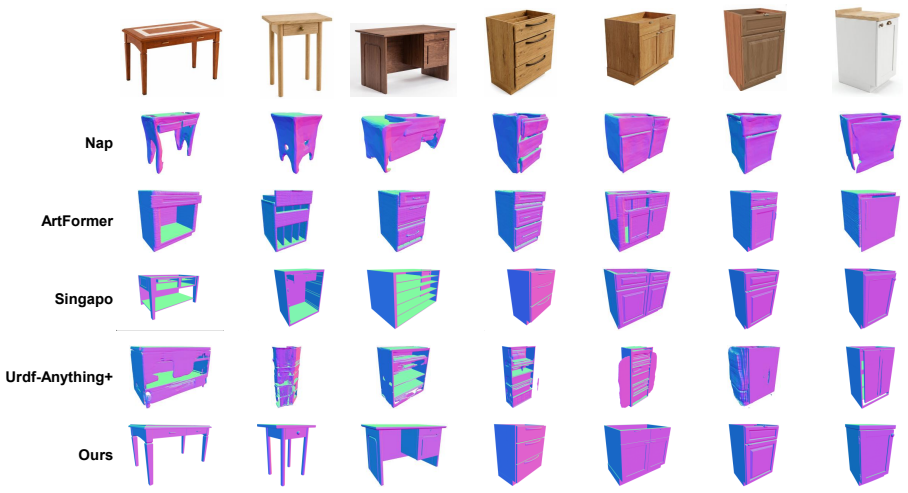



Figure 7: Visible geometry generated by different methods using the inputs of their original formulations. The rows compare complete systems, including differences in available input information. Figure 10 shows a separate comparison with a shared structural prior.

### A.8.2 Controlled Baselines and Ablations

Table 5 compares how variants of the base generator use a fixed structural prior. Every variant that receives a relation graph uses $\mathcal{R}_{\text{fix}}$. The label ``+ loss" denotes the same relation loss and coefficient in each variant. The global-fusion capacity control adds attention capacity comparable to the retrieval block, so the comparison tests more than the addition of an attention module. These experiments use no induced relations, local geometry refinement, or prior repair.

Table 5: Ablation of relation encoding, retrieval, and supervision under a fixed base prior. Graph-input variants receive $\mathcal{R}_{\text{fix}}$, whose relations are extracted from that prior. F-score and CD compare geometry with the reference; Intf. measures interface distance, Coll. is the fraction of objects with a prohibited collision at any evaluated state, and Motion is motion success.

| Method | F-score ↑ | CD ↓ | Intf. ↓ | Coll. ↓ | Motion ↑ |
|---|---|---|---|---|---|
| Base generator ($C_0$) | 0.732 | 4.96 | 0.037 | 0.187 | 0.742 |
| Base generator ($C_0$ + loss) | 0.740 | 4.81 | 0.033 | 0.172 | 0.761 |
| Flat graph | 0.746 | 4.69 | 0.031 | 0.161 | 0.773 |
| Flat graph + loss | 0.755 | 4.47 | 0.028 | 0.145 | 0.796 |
| Global fusion + loss | 0.762 | 4.33 | 0.026 | 0.136 | 0.812 |
| Global fusion + capacity control | 0.764 | 4.29 | 0.025 | 0.133 | 0.816 |
| Step-wise retrieval + loss (ours) | 0.788 | 3.94 | 0.019 | 0.096 | 0.869 |

Table 6: Fraction of supplied relations satisfied as structural fields are added. Methods receive the same information within each column.

| Method | Part roles | + Joints | + Interfaces | + Relation graph |
|---|---|---|---|---|
| Parallel decoder | 0.56 | 0.63 | 0.68 | 0.72 |
| Non-AR transformer | 0.58 | 0.66 | 0.71 | 0.76 |
| Base generator + flat relations | 0.61 | 0.71 | 0.77 | 0.81 |
| Step-wise relation retrieval (ours) | 0.61 | 0.71 | 0.77 | 0.90 |

The left panel of Figure 5 in the main text shows outputs from these variants. Table 5 measures interface distance and collision rate across the full test set. Table 7 evaluates distributional generation on one test split with the same number of generated objects from each method. Each baseline uses its original inputs. RelArt has the lowest ID-MMD (0.051) and POR (0.060), and the highest ID-COV (0.61). Table 1 separately compares generators given the same structural input.

Table 7: Distributional generation with each method's original inputs, a common test split, and equal numbers of generated objects. ID-MMD and ID-COV use Instantiation Distance; POR is averaged over ten synchronized articulation states.

| Method | ID-MMD ↓ | ID-COV ↑ | POR ↓ |
|---|---|---|---|
| NAP Lei et al. (2023) | 0.088 | 0.39 | 0.132 |
| CAGE Liu et al. (2024) | 0.081 | 0.42 | 0.118 |
| ArtFormer Su et al. (2025) | 0.070 | 0.48 | 0.096 |
| MeshArt Gao et al. (2025) | 0.065 | 0.51 | 0.089 |
| RelArt | 0.051 | 0.61 | 0.060 |

Figures 7 and 10 illustrate original-input and shared-prior comparisons, respectively. The right panel of Figure 5 in the main text shows three articulation states for RelArt outputs. The graph-source comparison uses the same target images, reference geometry, and base priors $C_0$ in all four conditions. To construct $\mathcal{R}_{\text{agent}}$, the construction module receives the target template, its base parameter values, and $K$ other valid instances. It can execute the target template at those parameter values to check the relations, but does not receive the benchmark reference mesh as a separate input. Comparing generators with the same graph source measures the decoder difference. Comparing graph sources for the same generator measures the effect of the additional relations.

Table 8: Two generators evaluated on the same target objects with either relations extracted from the base prior ($\mathcal{R}_{\text{fix}}$) or those relations plus induced relations ($\mathcal{R}_{\text{agent}}$).

| Relation source | Generator | IoU ↑ | Interface Dist. ↓ | State-CD ↓ | Coll. Rate ↓ |
|---|---|---|---|---|---|
| $\mathcal{R}_{\text{fix}}$ | Flat relations | 0.694 | 0.028 | 0.110 | 0.145 |
| $\mathcal{R}_{\text{fix}}$ | Step-wise retrieval | 0.725 | 0.019 | 0.078 | 0.096 |
| $\mathcal{R}_{\text{agent}}$ | Flat relations | 0.704 | 0.026 | 0.100 | 0.130 |
| $\mathcal{R}_{\text{agent}}$ | Step-wise retrieval | 0.733 | 0.017 | 0.071 | 0.084 |

With $\mathcal{R}_{\text{fix}}$, step-wise retrieval improves IoU by 0.031 and reduces collision rate by 0.049 relative to flat encoding. With $\mathcal{R}_{\text{agent}}$, the corresponding changes are 0.029 and 0.046. Interface distance decreases by 0.009 for either graph source. Adding induced relations also improves both generators, but by less than replacing flat encoding with step-wise retrieval in this experiment. Table 9 removes one training loss or the generated-part geometry history at a time. It keeps the prior and remaining training settings fixed and disables parameter repair after generation.

Removing $\mathcal{L}_{\text{mot}}$ raises collision rate from 0.096 to 0.170, the largest increase among the ablations in Table 9. Removing $\mathcal{L}_{\text{art}}$ produces the highest interface distance and State-CD. Removing generated-part history lowers F-score from 0.788 to 0.759 and raises interface distance from 0.019 to 0.029. These differences support the contribution of conditioning on previously generated geometry. The mechanism experiment compares step-wise retrieval with global fusion using attention blocks of comparable capacity. Figure 8 groups current parts by their relations to previously generated parts.

Table 9: Generator ablations under fixed priors, with parameter repair after generation disabled.

| Variant | F-score ↑ | CD ↓ | Interface Dist. ↓ | State-CD ↓ | Coll. Rate ↓ |
|---|---|---|---|---|---|
| w/o $\mathcal{L}_{\text{art}}$ | 0.751 | 4.52 | 0.033 | 0.132 | 0.152 |
| w/o $\mathcal{L}_{\text{mot}}$ | 0.777 | 4.11 | 0.022 | 0.115 | 0.170 |
| w/o $\mathcal{L}_{\text{rel}}$ | 0.771 | 4.20 | 0.025 | 0.102 | 0.124 |
| w/o generated history | 0.759 | 4.43 | 0.029 | 0.114 | 0.143 |
| Full relation-aware model | 0.788 | 3.94 | 0.019 | 0.078 | 0.096 |

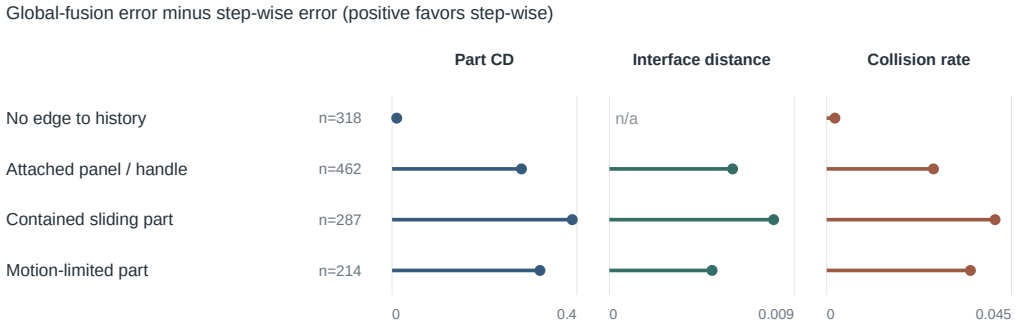



Figure 8: Error differences between global fusion with comparable capacity and step-wise retrieval on the same parts. Each point is the global-fusion error minus the retrieval error; a positive value favors retrieval. Part CD is shown in units of $10^{-3}$. Counts refer to parts. A part can appear in multiple relation groups. Interface distance is not defined for parts without a specified interface.

The errors differ little when the current part has no relation edge to a generated part, and differ more when such edges exist. A part can belong to more than one relation group, so these results do not isolate the contribution of each relation type.

### A.8.3 Implementation Details

We include a geometry template in $\mathcal{T}_{\text{kin}}$ only if a separate joint adapter can compute part transforms throughout the stated motion range. A static mesh, a rotation argument in a part constructor, or a joint label alone is insufficient. Adapter requirements are fixed before evaluation, and computed transforms are checked against held-out reference articulation states. The fixed-prior synthesis benchmark uses a 5:1 training-to-test split. Construction partitions separately hold out parameter regions, programs, combinations of relations, or categories composed of supported relation types. Programs are grouped by normalized constructor and joint-adapter signatures. In program, relation-combination, and category partitions, near-duplicate programs and all their instances stay in one partition. The parameter-region partition reuses a training template but uses disjoint parameter ranges. The construction module receives $K \in \{1, 3, 5, 10\}$ valid parameter instances per evaluation template. It allows 24 proposals and at most 120 executor calls per template in the construction comparisons. The reported few-shot results use held-out programs. For a held-out template, the joint adapter is authored from the template source and a category-level joint specification available to all methods. Authoring does not use target meshes, target images, or target articulation labels. The one-time cost of writing the adapter is excluded from every method's per-object runtime.

Part geometry, joint descriptors, support descriptors, and interfaces are expressed in canonical coordinates. To compute whole-object IoU, we take the union of the part occupancies on a common normalized grid for each object. Part IoU is the mean IoU between corresponding predicted and reference part slots within an object. F-score compares sampled mesh surfaces in both directions, using a distance threshold $\tau_F$ fixed on training objects. Chamfer Distance is the symmetric mean squared nearest-neighbor distance between equal-size sets of surface samples. Static CD is displayed in units of $10^{-3}$. State-CD computes the same distance after transforming predicted and reference parts to

ten synchronized states spanning the supplied joint ranges. It is averaged over states and objects and reported without the $10^{-3}$ display scaling. Interface distance is the bidirectional mean surface distance between the two specified interface patches, divided by the object diagonal.

We compute Instantiation Distance using the original NAP implementation Lei et al. (2023), with ten sampled articulation states and 2,048 surface points per state. For each pair of states, we express each whole-object point cloud in each of its part frames and take the minimum Chamfer Distance over all pairs of these frames. These distances form a $10 \times 10$ matrix. ID is the sum of the mean row minimum and the mean column minimum, allowing matches across different articulation states. ID-MMD is the mean distance from each reference object to its nearest generated object. ID-COV is the fraction of reference objects that are the nearest neighbor of at least one generated object. Each method contributes the same number of generated objects. POR averages volumetric intersection-over-union over part pairs where overlap is not permitted and over synchronized articulation states. ID-MMD, ID-COV, and POR are reported for distributional generation, not for the ablations comparing each generated object with a corresponding reference.

Collision rate across evaluated states is the fraction of objects with at least one prohibited part intersection whose penetration depth exceeds $0.005$ in normalized object coordinates at any state in $Q_{\mathrm{eval}}$. Each object contributes a binary collision indicator, whereas POR measures the amount of overlap. An object passes the motion-success test only if every joint remains within its supplied limits, every specified interface remains within its tolerance, and no prohibited collision occurs at any evaluated state. For orientation checks, the cosine deviation $1 - \cos\theta$ must not exceed $0.05$, corresponding to approximately $18^\circ$. Here $\theta$ is the angle between oriented normals or the smaller angle between undirected axes, consistent with the angular residuals in Table 2. Other interface tolerances are fixed on training data. The checker measures intersections and interface gaps even when the input graph omits the corresponding relation edge.

The definition of relation satisfaction depends on the experiment. In generation and parameter-repair experiments, it is the fraction of supplied relation constraints that the generated geometry satisfies under the independent checker. In construction ablations, it is the fraction of applicable executor relation checks whose residuals satisfy $e_k \leq \epsilon_{\rho_k}$; passing these checks does not establish final asset validity. Articulation validity is the fraction of all proposals from $\mathcal{T}_{\mathrm{kin}}$ that execute successfully and pass the independent motion test. A usable asset must execute successfully, meet the interface tolerance, and pass all motion checks. Template coverage is the fraction of eligible template signatures for which at least one usable asset is produced. Valid yield is the number of usable assets divided by the number of attempts. These construction measures differ from the nearest-neighbor coverage metric ID-COV. For object-level metrics, we average within each category and then take the mean across categories.

The relation-input variants of our base autoregressive generator share the procedure for aggregating edges on each part pair, visual backbone, part representation, SDF decoder, reconstruction targets, data split, and optimization budget. The parallel and non-autoregressive decoders in the structural-field experiment use the same flat relation input and relation loss. Parameter repair after generation is disabled in every fixed-prior comparison. The controlled benchmark renders images with fixed cameras and lighting. Each image depicts the object described by its reference geometry and supplied prior. Methods receive the same images and structural fields; target meshes are used only for training supervision and evaluation. The graph-source comparison uses these same images and keeps each generator's trained weights fixed when switching graph sources at test time. Construction templates come from ConceptFactory Sun et al. (2024). Joint descriptions that compute part motion are additional inputs; the time spent authoring them is excluded from per-object runtime.

The within-family controlled models are trained for 500 epochs with a global batch size of 32 on four NVIDIA RTX 4090 (24GB) GPUs. They use AdamW with a base learning rate of $10^{-4}$, 50 warmup epochs, and a cosine learning-rate schedule for the remaining epochs. The loss weights are $\lambda_{\mathrm{rec}} = 1.0$, $\lambda_{\mathrm{art}} = 0.5$, $\lambda_{\mathrm{mot}} = 0.5$, and $\lambda_{\mathrm{rel}} = 0.3$; the reconstruction weight is implicit in the objective in Section A.7. The image encoder is a pre-trained DINOv2 ViT-B/14 Oquab et al. (2023). A two-layer MLP maps its final-layer patch features to 256 dimensions. DINOv2 weights remain frozen in the controlled comparison. The structural encoder has four transformer layers with four attention heads and hidden dimension 256. Image--structure fusion uses cross-attention and a four-layer encoder. The autoregressive decoder has six layers with hidden dimension 256, and geometry prediction uses the base generator's part-wise SDF decoder. Reconstruction training uses 4096 query

points per part: half sampled near the reference surface and half sampled uniformly in the normalized bounding box.

**Generated-part feature and generation order.** After choosing the original or refined geometry as described in Section A.6, we evaluate the part's SDF at the decoder's canonical query points. For each query point $p$, we concatenate its normalized coordinates, truncated signed distance, an indicator of whether it is inside the part, and an indicator of whether its absolute signed distance is below the truncation threshold $\tau$. A shared point encoder maps each concatenated feature to a 256-dimensional vector. Elementwise max pooling over query points gives one geometry vector for the part. An MLP combines that vector with the part's role and frame embeddings to obtain the history feature $h_j$:

$$h_j = \phi_{\text{hist}}\left(\max_{p\in\mathcal{Q}_j} \phi_{\text{pt}}[p, \bar{S}_j(p), \mathbf{1}\{S_j(p) < 0\}, \mathbf{1}\{|S_j(p)| < \tau\}], E_r(r_j), \phi_F(F_j)\right). \tag{13}$$

Variants of the base autoregressive generator that use generated-part history share this encoder and the same number of query points. Numerical relation attributes are encoded in $g_{ij}^C$; $h_j$ encodes generated geometry, role, and frame. The generation order is a topological ordering of support and kinematic dependencies. Parent parts precede dependent parts. When multiple parts can be generated next, the algorithm selects them by template part index, using the same indices fixed before training for all compared models.

**Relation-feature normalization.** Discrete part roles, joint types, and relation types use learned embeddings. Continuous frame, joint, and relation attributes are expressed in canonical object coordinates. Distance attributes are normalized by the training-set object-diagonal scale, and joint limits by their declared range. A mask marks optional fields that are absent. Each continuous-field encoder is a two-layer MLP with GELU activation and 256-dimensional output. For each ordered part pair, we sum the encoded relation vectors, concatenate the number of relations, and project the result to a pair feature. Normalization statistics are computed on the training split and held fixed for both graph sources and all ablations.

### A.8.4 Independent Relation Audit

Two annotators assign reference relation types and parameter scopes to a stratified held-out set of template--role pairs before seeing the construction module's predictions. They inspect each template and the same valid seed instances supplied to the module, and resolve disagreements before scoring. A predicted relation is counted as correct only if its type, participating parts, and direction match the reference, the relation holds on independently sampled valid instances, and it rejects counterexamples created by changing relevant parameters. A predicted scope is counted as correct only if it exactly matches the minimal reference parameter set. Table 10 reports precision, recall, scope accuracy, audit coverage, and abstention. Passing the construction executor's checks is not the reference criterion for this audit.

Table 10: Independent review of predicted relations and parameter scopes on held-out templates. Audit coverage is the fraction of templates with at least one relation that passes the construction checks and matches the independent reference. ID-COV is a separate generation metric.

| Valid examples $K$ | Precision ↑ | Recall ↑ | Scope acc. ↑ | Audit cov. ↑ | Abstain ↓ |
|---|---|---|---|---|---|
| 1 | 0.78 | 0.51 | 0.68 | 0.56 | 0.34 |
| 3 | 0.85 | 0.66 | 0.77 | 0.72 | 0.24 |
| 5 | 0.88 | 0.73 | 0.82 | 0.80 | 0.18 |
| 10 | 0.90 | 0.78 | 0.85 | 0.86 | 0.13 |

From one to five valid instances, precision rises from 0.78 to 0.88, recall from 0.51 to 0.73, scope accuracy from 0.68 to 0.82, and audit coverage from 0.56 to 0.80; abstention falls from 0.34 to 0.18. Gains from five to ten instances are smaller.

### A.8.5 Parameter Repair after Generation

Table 11 evaluates parameter repair after generation separately because this procedure changes the structural prior. All conditions use the same initial priors that passed construction checks, the same

generator checkpoint, and the same images; only the maximum number of repair rounds changes. Each round executes the template again, and its cost is included in runtime. Relation satisfaction and motion success are evaluated against the repaired prior. The image remains fixed, and the experiment does not measure whether the repaired output better matches that image.

Table 11: Effect of repairing the prior after geometry generation. Conditions start from the same priors and differ in the allowed number of parameter-repair rounds.

| Setting | Rel. sat. ↑ | Motion success ↑ | Coll. rate ↓ | Sec./object ↓ |
|---|---|---|---|---|
| Constructed prior, no repair | 0.85 | 0.820 | 0.117 | 10.5 |
| One repair round | 0.89 | 0.853 | 0.093 | 14.2 |
| Up to three repair rounds | 0.91 | 0.871 | 0.081 | 18.9 |

Allowing up to three repair rounds raises relation satisfaction from 0.85 to 0.91 and motion success from 0.820 to 0.871; collision rate falls from 0.117 to 0.081. Runtime increases from 10.5 to 18.9 seconds per object. One round trades some of the validity gain for lower runtime.

### A.8.6 Construction-Stage Ablation

The construction ablation holds the synthesis model, seed instances, proposal budget, held-out templates, and executor-call limit fixed. The variants omit relation induction, execution checks, or parameter repair, or replace repair within verified scopes with resampling of all parameters. The experiment measures how these choices affect articulation validity and template coverage (Table 12). The few-shot experiment separately reports valid yield.

Table 12: Construction ablation with the same synthesis model and proposal and executor-call limits. Rel. sat. is relation satisfaction measured by the executor; articulation validity uses the independent checker.

| Construction variant | Artic. valid ↑ | Rel. sat. ↑ | Template cov. ↑ | Sec./valid ↓ |
|---|---|---|---|---|
| Without relation induction | 0.52 | 0.58 | 0.53 | 6.1 |
| Without executable checks | 0.57 | 0.63 | 0.59 | 7.0 |
| Filtering without repair | 0.57 | 0.72 | 0.59 | 7.3 |
| Full-object resampling | 0.74 | 0.81 | 0.70 | 14.6 |
| Repair within verified parameter scopes | 0.82 | 0.90 | 0.84 | 10.1 |

Filtering without parameter repair reaches 0.57 articulation validity and 0.59 template coverage. Resampling all parameters reaches 0.74 validity and 0.70 coverage at 14.6 seconds per valid asset. Repair restricted to verified scopes reaches 0.82 validity and 0.84 coverage at 10.1 seconds per valid asset. Under the same proposal and executor-call limits, restricting changes to parameters that affect failed checks improves both measures and reduces time per valid asset relative to full resampling. In Figure 9, some examples produced without relation induction have detached movable parts, and some produced without execution checks have incomplete interfaces. These selected outputs do not show the number of candidates rejected by filtering; Table 12 reports articulation validity and template coverage. The ``No generation history'' row changes the decoder rather than prior construction and is evaluated in Table 9.

### A.8.7 Few-Shot and Prior-Robustness Analysis

Table 13 varies the number of valid instances for held-out template programs and measures valid yield, template coverage, and time per valid asset. A separate robustness experiment changes relation inputs at test time while keeping the image, part slots, and joints fixed. Let $N_{\text{elig}}$ be the number of relation edges in $\mathcal{R}$ that are not marked as required. For each object with $N_{\text{elig}} > 0$, we select $\max(1, \lceil 0.3 N_{\text{elig}} \rceil)$ eligible edges, sampling uniformly within relation types. Each type with at least two eligible edges contributes at least one selected edge. We skip a candidate for removal if deleting it would leave a part without any relation edge to a previously generated neighbor at its decoding step, and select another eligible edge. Both test conditions use the same selected edges. In the omitted-edge condition, the selected edges are removed from the decoder input. In the attribute-perturbation

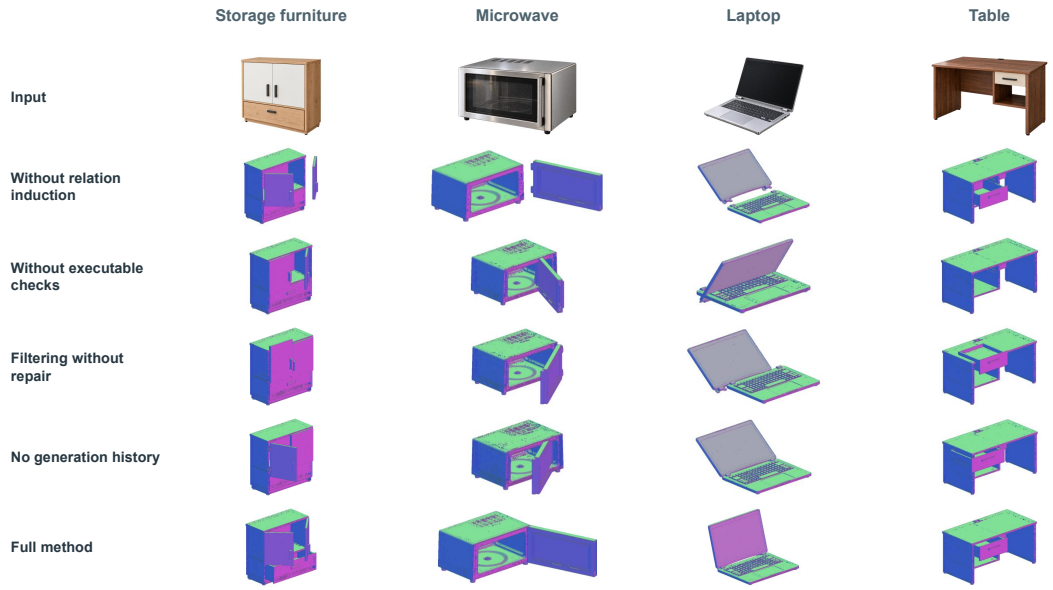



Figure 9: Examples from construction and decoder ablations. The Input row shows reference images. ``No generation history'' removes previously generated geometry from the decoder and corresponds to Table 9. The other ablation rows change the construction procedure and correspond to Table 12.

condition, every continuous numerical attribute $m_{ij}$ on a selected edge is first multiplied by $(1+\epsilon)$, where $\epsilon \sim \mathcal{U}[-0.15, 0.15]$, and then clipped as described below. These attributes include interface and support gaps, alignment offsets, symmetry offsets, stacking intervals, size margins, signed-depth tolerances, and non-contact penetration tolerances. Nonnegative distances and margins are clipped to $[0.005, 0.15]$ in normalized object-diagonal units. Stacking-interval endpoints are clipped separately and then sorted; signed offsets retain their sign and have absolute value at most $0.10$. Relation types, participating parts, edge directions, and joint pivots, axes, and ranges remain unchanged. The independent checker evaluates both conditions against the original prior $C^*$.

Table 13: Few-shot construction on held-out template programs with a fixed proposal and executor budget.

| Seed instances/template | Valid yield ↑ | Template cov. ↑ | Sec./valid ↓ |
|---|---|---|---|
| One | 0.56 | 0.54 | 15.3 |
| Three | 0.73 | 0.72 | 12.1 |
| Five | 0.82 | 0.84 | 10.1 |
| Ten | 0.85 | 0.88 | 9.7 |

Increasing the number of seed instances from one to five raises valid yield from 0.56 to 0.82 and template coverage from 0.54 to 0.84, while reducing time per valid asset from 15.3 to 10.1 seconds. Increasing the number from five to ten gives smaller gains in both rates and a smaller runtime reduction.

Table 14: Synthesis robustness with identical images, part slots, and joints. The independent checker uses the unperturbed reference prior.

| Condition | Rel. sat. ↑ | Interface dist. ↓ | Motion success ↑ |
|---|---|---|---|
| Unperturbed relations | 0.90 | 0.019 | 0.869 |
| Perturbed relation attributes | 0.84 | 0.025 | 0.816 |
| Omitted relation edges | 0.87 | 0.022 | 0.844 |

Perturbing numerical relation attributes reduces relation satisfaction by 0.06 and motion success by 5.3 percentage points. Omitting relation edges reduces these measures by 0.03 and 2.5 percentage points, respectively. Under the tested perturbation and omission settings, changing attribute values causes the larger degradation.

### A.8.8 Cross-Category Transfer of Shared Parts and Relations

We evaluate geometry generation on held-out categories composed of part roles and relation types found in other training categories. Each run excludes every instance, template signature, and parameter variant of the test category from training and model selection. Each test part role, joint type, and relation type occurs in training, but their combination in the test category does not. The test image and verified structural prior describe the same target object; the model receives neither its reference mesh nor reference geometry at articulated states. Each generator receives the same image and structural fields, converted to its conditioning format as in Table 1. Relation induction and prior repair are disabled. The base autoregressive generator, its flat-relation variant, and step-wise retrieval use the same training categories and optimization budgets. F-score compares generated and reference surfaces, interface distance measures attachment error, and motion success checks the geometry at synchronized articulation states. We average within held-out categories and then across categories. The test excludes unsupported part roles and joint types.

Table 15: Category-held-out synthesis with shared part roles and relation types. All generators receive the same test image and fixed structural prior.

| Method | F-score ↑ | Interface dist. ↓ | Motion success ↑ |
|---|---|---|---|
| Base autoregressive generator | 0.695 | 0.044 | 0.689 |
| Base generator + flat relations | 0.722 | 0.035 | 0.752 |
| Step-wise relation retrieval (ours) | 0.755 | 0.026 | 0.821 |

Compared with flat relation encoding, step-wise retrieval raises F-score by 0.033 and motion success by 6.9 percentage points, while reducing interface distance by 0.009. These results apply to new combinations of known part roles and relation types; they do not establish performance on unseen roles or joint types.

### A.8.9 Additional Qualitative Comparisons

Figures 3 and 7 show original-input comparisons. Figures 4, 10, and 11 show comparisons with a shared structural prior.

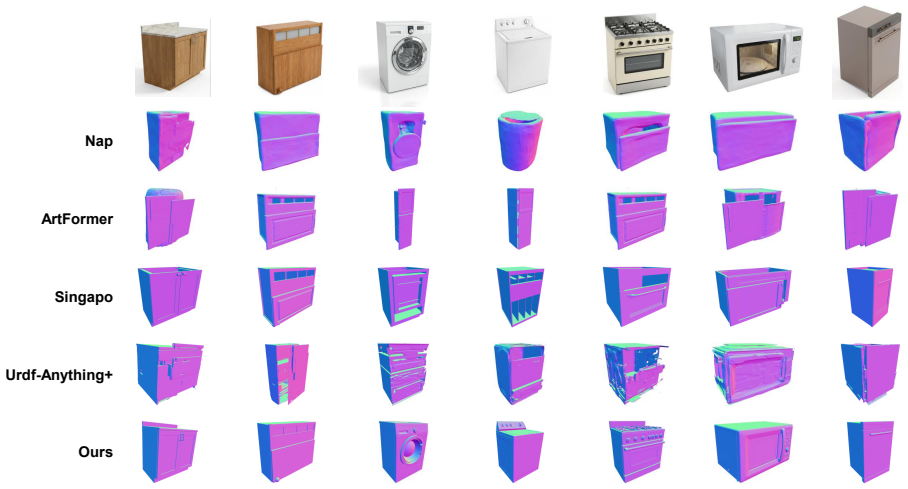



Figure 10: Generation with a shared executable structural prior. The top row shows reference images; the other rows show outputs after the common prior is converted to each method's input format.

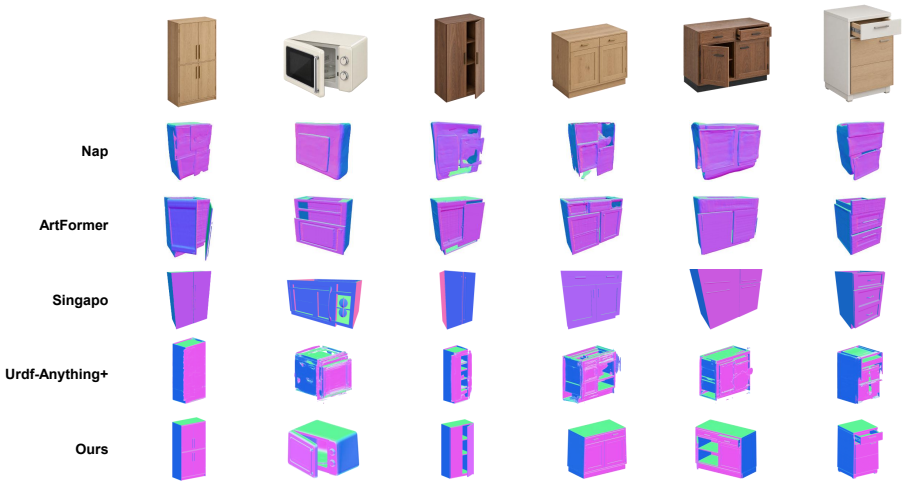



Figure 11: Comparisons with a shared structural prior across different part arrangements. The results illustrate differences in the subdivision of front surfaces and the placement of movable parts within the enclosing body.

In Figure 11, some baseline outputs merge distinct front sections, alter their proportions, or introduce fragmented surfaces. Our outputs more closely preserve the visible subdivision and relative placement of the parts. Fine details are still simplified, and the displayed opening angles do not always reproduce those of the references. These static views support comparisons of visible geometry and part arrangement; motion is evaluated separately.

### A.9 Discussion

**Scope of structural generalization.** Construction generates new parameter instances and combinations of supported relations within the executable template library. The few-shot experiment evaluates held-out programs that use those relations. We do not measure how reliably the system detects and rejects unsupported inputs.

**Impact statement.** Automated prior construction generates parameter instances from reusable templates for simulation or interactive environments. Generated assets can inherit errors from tem-

plates and joint adapters. They require additional validation before use in safety-critical simulation or physical deployment.

**Limitations.** Construction requires executable templates and can inherit errors in how those templates divide an object into parts. The generic motion loss excludes entire part pairs designated as permitting contact. During training, penetration penalties for those pairs depend on explicit motion-limit relations, although the evaluation checker still checks them. The experiments use supported joint types and images rendered with fixed cameras and lighting. Real images, unseen joint types, deformable motion, and substantially noisy priors remain untested.