DUALCRAFT: A DUAL-WORKSPACE AGENT FOR ARTICULATED ASSET GENERATION FROM IMAGES

Anonymous authors
Paper under double-blind review



Figure 1: **All articulated assets in this Isaac Sim scene are generated by DUALCRAFT.** Boards on the walls list the assets beside them, each with its moving parts counted by joint type and an arrow of its color.

ABSTRACT

Interactive articulated objects such as doors and home appliances are crucial in robotics, but creating their simulation-ready assets remains labor-intensive. Generating them from images is hard because part geometry and motion structure must be recovered jointly: refining geometry changes assembly positions and clearances, while motion errors may require geometric fixes. We present DUALCRAFT, a dual-workspace agent that models articulated objects from images without additional training. Its geometry workspace refines part shapes and materials against the images, and its program workspace defines the assembly and joints and checks their motion; shared part frames keep the two consistent. We further introduce REALPRODUCT, 114 assets of 63 product types, each modeled by hand after a specific real product, with physically based materials and the interiors that motion reveals, and combine it with two recent collections into ARTS-Bench, a 690-object benchmark that scores multiview appearance and part geometry at rest and in motion. DUALCRAFT achieves the best score on every ARTS-Bench metric against six baselines, reducing Chamfer distance in articulated states by 46% relative to a coding agent that uses the same model, and ablations show that removing its shared frames, visual feedback, or geometry checks each more than doubles Chamfer distance at rest. We further demonstrate that the generated assets can be used directly in simulated robot manipulation tasks that go beyond basic pick and place, and that trajectories planned with them in simulation transfer unchanged to a real robot. We will release the code, dataset, and benchmark.[1]

[1]Project page: `https://anonymous.4open.science/w/dualcraft-project-page-AD51`

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.

1 INTRODUCTION

Operating articulated objects such as cabinets, drawers, and home appliances is a fundamental capability for robots in homes and workplaces. Such skills are difficult to acquire through trial and error on real hardware, so simulation has become a primary training ground, which requires assets with accurate part geometry and joint definitions (Xiang et al., 2020; Jin et al., 2026). Modeling each asset and configuring its joints by hand is labor-intensive, whereas images are a ready source of input. Generating articulated assets from images is nevertheless hard, because part geometry and motion structure must be recovered jointly: refining a part's shape changes its attachment and clearances, while correcting a joint may in turn require geometric fixes.

Existing methods pursue fidelity to the input image and valid articulation separately. Agents that write the asset as a program make its articulation executable and checkable: Articulate-Anything (Le et al., 2025) refines joints with visual feedback, and Articraft (Zhou et al., 2026) checks motion through execution, but neither refines part shapes against the image. SINGAPO (Liu et al., 2025a) assembles retrieved parts to recover movable structures, and PhysX-Anything (Cao et al., 2026a) and PhysX-Omni (Cao et al., 2026b) generate detailed geometry with physical properties in a single pass, without correcting it against the image or checking its motion.

We present DUALCRAFT, an agentic system that generates simulation-ready articulated assets from one or more reference images without additional training (Fig. 1). Its key idea is to keep the asset in two representations, a free-form mesh scene for shape and appearance and an executable program for articulation, and to give every quantity a single owner that both read from, so that the two cannot drift apart. A planner first drafts a part blueprint that specifies which parts to build and how they move. An agent then works in a geometry workspace, where it refines shape and materials by comparing renderings with the reference images, and a program workspace, which derives the assembly and joints from the shared part frames and checks the asset in sampled motion states. Every geometric edit is thus rebuilt and checked as an articulated asset, and rendered comparisons and structural checks can trigger edits in either workspace.

Evaluating generated assets requires references that look and move like real objects. We build REALPRODUCT, a dataset in which every asset is modeled by hand after a specific real product, with physically based materials and the interiors that motion reveals, and combine it with REALAPPLIANCE (Gao et al., 2026) and ArtVIP (Jin et al., 2026) into ARTS-Bench, whose objects average 3.81 movable parts, compared with 2.85 in the PartNet-Mobility subset used to train SINGAPO. ARTS-Bench follows prior protocols for part geometry and adds multiview appearance at rest and in motion, which such references make possible. Motion exposes errors the rest state hides: once parts move, the two strongest baselines swap places in Chamfer distance (Sec. 4.3).

Our contributions are as follows:

1. **A dual-workspace agent system:** DUALCRAFT pairs a free-form mesh scene with an articulation program regenerated from shared part frames after every edit, so shape and articulation stay consistent by construction and visual and structural feedback can drive edits in either workspace.
2. **A high-fidelity articulated dataset:** REALPRODUCT has 114 assets of 63 product types with 410 annotated joints, each modeled by hand after a specific real product at its real size, with physically based materials and modeled interiors, and ready for simulation in URDF and USD.
3. **A unified benchmark:** ARTS-Bench covers 690 objects and 2,628 joints and scores multiview appearance and part geometry at rest and in motion, including interiors that open parts reveal.
4. **Extensive evaluation:** Against six baselines on 50 objects, DUALCRAFT achieves the best score on every geometry and appearance metric, reducing Chamfer distance in articulated states by 46% relative to a coding agent with the same model, and its assets load directly into Isaac Sim (NVIDIA, 2025) for robot manipulation. Ablations on objects not fully shown by one view show that removing the shared frames, visual feedback, or geometry checks each more than doubles Chamfer distance at rest.

2 RELATED WORK

Image-guided articulated asset generation. Learned generators predict parts and joints from an image and retrieve or generate part geometry (Liu et al., 2025a; Chen et al., 2024; Liu et al., 2026;

Wu et al., 2026); PhysX-Anything (Cao et al., 2026a) and PhysX-Omni (Cao et al., 2026b) also generate physical properties, and others add priors, optimization, or planning (Lu et al., 2025a; He et al., 2026; Yang et al., 2026). Further methods articulate existing meshes or point clouds (Li et al., 2026a; Qiu et al., 2025; Zhang et al., 2026; Wang et al., 2026; Li et al., 2025) or reconstruct from richer observations such as multiple articulation states (Liu et al., 2023; Jiang et al., 2022; Heppert et al., 2023; Li et al., 2026b; Mandi et al., 2025). Most learn structural priors from existing assets and produce the asset in a single pass, whereas DUALCRAFT needs no training, uses images of one state, and corrects the asset against them while checking its motion.

Agentic modeling with feedback. Blender-based agents write and refine modeling code with visual feedback (Huang et al., 2024; Hu et al., 2024; Lu et al., 2025b) but do not model articulation. Articulated agents express parts and joints as programs: Articulate-Anything (Le et al., 2025) places retrieved parts and refines joints through visual feedback, Articraft (Zhou et al., 2026) repairs code-built assets from execution checks, ArtiCAD (Shui et al., 2026) assembles CAD parts with cross-stage rollback, and, concurrently, Procedura (Lin et al., 2026) joins procedural parts from text with typed mates. None of them refines free-form part meshes and materials against reference images while checking the motion of the same asset.

Articulated datasets and benchmarks. Existing datasets annotate part motion on existing models (Xiang et al., 2020; Liu et al., 2022; Geng et al., 2023), provide professionally modeled interactive assets (Jin et al., 2026; Gao et al., 2026), or annotate them at scale (Iliash et al., 2026); REAL-PRODUCT models specific real products, including types that are absent or underrepresented. Prior benchmarks evaluate part geometry in rest and articulated states (Liu et al., 2025a), mesh articulation (Li et al., 2026a), or physical properties (Cao et al., 2026b); ARTS-Bench additionally evaluates multiview appearance in both states, including interior surfaces revealed by motion.

3 METHOD

DUALCRAFT converts reference images into an executable articulated asset in three stages (Fig. 2; Alg. 1 in Appendix B): it plans a part blueprint (Sec. 3.2), builds the asset in two coupled workspaces (Sec. 3.3), and refines it with visual and structural feedback until completion checks pass (Secs. 3.4 and 3.5).

3.1 PROBLEM FORMULATION

Given reference images $\mathcal{I} = \{I_v\}_{v=1}^{V}$ of an object in one articulation state ($V = 1$ in our main experiments), DUALCRAFT produces an asset

$$\mathcal{A} = \big(\{(M_i, \mu_i, F_i)\}_{i=1}^{N},\ \pi,\ \Theta\big), \tag{1}$$

where part i has a mesh M_i in its local frame, materials μ_i, and a frame pose $F_i \in SE(3)$ at rest in the object frame ($+x$ forward, $+z$ up). Each non-root part has a parent $\pi(i)$ and a joint in Θ with type $\tau_i \in \{\text{fixed}, \text{revolute}, \text{prismatic}, \text{continuous}\}$, an axis $\mathbf{a}_i$ unless it is fixed, and limits $[l_i, u_i]$ if it is revolute or prismatic. At joint state $\mathbf{q}$, a local vertex $\mathbf{x}$ of part i lies at $T_i(\mathbf{q})\,\mathbf{x}$, with

$$T_i(\mathbf{q}) = T_{\pi(i)}(\mathbf{q})\, F_{\pi(i)}^{-1} F_i\, J_{\tau_i}(\mathbf{a}_i, q_i), \qquad T_{\text{root}} = F_{\text{root}}, \tag{2}$$

where J_τ is the identity for a fixed joint and otherwise rotates about, or translates along, $\mathbf{a}_i$. Eq. 2 couples geometry and motion: reshaping a part changes M_i and often its pivot F_i, and with them the volume the part sweeps, while changing a joint may require new geometry to keep clearances.

3.2 PART BLUEPRINT

A planner built on a vision-language model first anchors each reference image to a camera viewpoint in the object frame, from the side facing the camera, the azimuth, the elevation, and a coarse distance (Appendix B.2). The render camera is later placed at these viewpoints to compare the asset with the references.

The planner then drafts the blueprint: the object's estimated size, mass, materials, and use, and a list of parts, each with a parent, a movable flag, a description, a visibility label, and an appearance record. Motion is filled in as slots instead of free text: a type (hinge, slide, turn, spin, or press), the

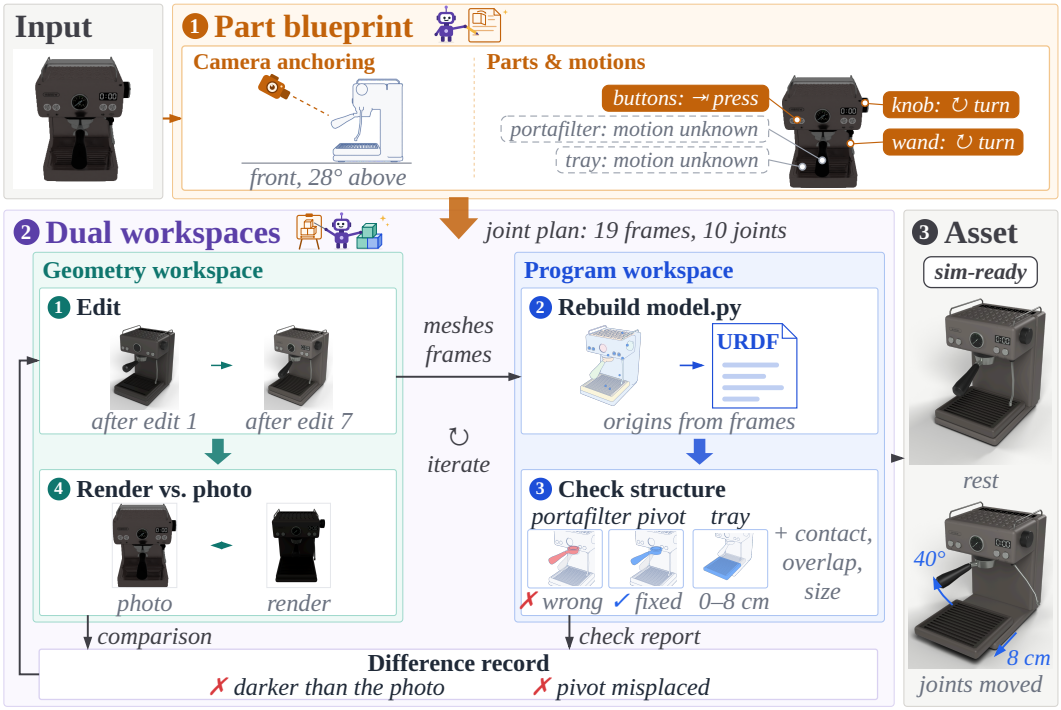


Figure 2: **Method overview.** A planner built on a vision-language model (VLM) anchors each reference image to a camera viewpoint and drafts a part blueprint, which is compiled into a joint plan. The geometry workspace edits part meshes, materials, and pivot frames in Blender; the program workspace regenerates the assembly program from the shared frames and compiles it. Visual comparison and structural checks drive edits in both workspaces, and the output is a URDF with part meshes.

hinge edge or axis, the direction, and the range, with an end-state sentence for bounded motions; a motion or range the planner cannot judge reliably is left open for the agent to set. A linter checks the draft without ground truth.

The blueprint is compiled deterministically into a joint plan. Parents define the tree, and each joint type follows from the motion slot: fixed for non-movable parts, revolute for hinges and turns, prismatic for slides and presses, and continuous for spins, with default axis signs under which positive values open a part outward. For the espresso machine in Fig. 2, the buttons become prismatic joints and the knob and wand revolute joints; the photo does not show how the portafilter and drip tray move, so the agent later sets them as a $40°$ turn and an 8 cm slide. The plan fixes part identifiers for both workspaces, and each part adds checklist items, from presence to the end pose, that the agent must answer before finishing.

3.3 Dual workspaces

Consistency by construction. Both workspaces follow one principle: each quantity in Eq. 1 has a single owner. The Blender scene of the geometry workspace owns $\{M_i, \mu_i, F_i\}$, the joint plan owns π and the joint semantics $(\tau_i, \mathbf{a}_i, l_i, u_i)$, starting from the blueprint, and the assembly program derives every joint origin from the shared frames instead of storing its own copy. The workspaces therefore cannot diverge: moving a door's hinge frame moves its joint origin at the next rebuild, a reversed opening direction or a wrong range is fixed in the joint plan rather than by editing coordinates, and every check runs on the rebuilt assembly rather than on a stale copy.

Geometry workspace. The agent edits a persistent Blender scene with Blender Python. Each part owns a frame, an axis-aligned empty object placed at the part's pivot, and the part's meshes are parented to it. Beyond shape, the agent assigns materials and redraws printed marks, such as dials and labels, as metric panel textures. After every successful edit, the workspace exports each part's

meshes in its frame and rebuilds and checks the assembly, so no geometry edit goes structurally unchecked.

Program workspace. The assembly program is Python code over an articulation SDK, compiled and probed with tools adapted from Articraft (Zhou et al., 2026), and regenerated by the system rather than written by the agent. Each part references its exported mesh, and each joint takes its origin $F_{\pi(i)}^{-1} F_i$ from the frames and its type, axis, and limits from the joint plan. The agent contributes structural tests, such as a closed drawer lying inside its cabinet, which may be evaluated at chosen joint states, and can set a joint's type, axis, and range in the joint plan, for example when the blueprint left a motion open or a pose check shows it is wrong; compilation yields the URDF.

3.4 Visual and structural feedback

The asset is judged in two ways: visual comparison tests whether it matches the reference images, and structural checks test whether it forms a valid mechanism.

Visual comparison. DualCraft renders the asset from each anchored viewpoint in one fixed virtual studio shared by all objects and places the rendering beside the reference. Because the viewpoint is estimated, the agent judges parts, sides, and proportions rather than pixel overlap. It can also render close-ups of named parts and crop the reference, and each whole-asset rendering is checked against the reference in color (Appendix B.4).

Structural checks. Each rebuild reports the assembled size against the blueprint estimate, the flushness of paired panels, and the compiler's checks: a valid tree with unit axes and consistent limits, every part connected to the root through contact at rest, and no interpenetration at rest beyond allowances that the agent must justify. The agent's tests add contact, gap, and containment constraints at chosen joint states. For motion, a pose check renders each revolute and prismatic joint at its far limit from two views, labeled with the blueprint's end state, so the agent can verify the direction and extent of each motion.

Difference record. Each discrepancy the agent observes becomes an entry naming the affected parts and its kind, such as missing, placement, motion, or appearance. Entries other than appearance block completion, turning visual critique into tracked obligations.

3.5 Closed-loop refinement and export

Alg. 1 (Appendix B) organizes the feedback into two phases. In the geometry phase, the agent alternates edits, automatic checks, and observations that include the reference view, reviewing the difference record after each observation, until the last edit has passed its checks with no blocking entry open. The assembly phase validates and commits the scene, which exports textured part meshes and recompiles the program, and runs pose checks and the checklist review. A failed compilation or validation, a rejected completion check, or a new blocking discrepancy reopens geometry editing for a short rework window, so structural and motion errors flow back into the scene. The run finishes when the completion checks pass (Appendix B.6), and the output is a URDF with per-part meshes and baked textures. Runs are capped at 20 model turns (Appendix B).

4 RealProduct dataset and ARTS-Bench

Evaluating image-to-asset generation requires references that are faithful to real products, inside and out, and a protocol that scores what simulation needs: the parts, their motion, and their appearance once they move. We build RealProduct (Sec. 4.1) and unify it with two recent sources in ARTS-Bench (Secs. 4.2 and 4.3).

4.1 RealProduct dataset

RealProduct contains 114 articulated assets of 63 product types across nine categories, with 410 movable joints (247 revolute and 163 prismatic). It is designed for fidelity and breadth rather than volume. For fidelity, each asset is modeled by hand after a specific commercial product at its real size, with separate part meshes, physically based materials, and the interior surfaces that motion



Figure 3: **REALPRODUCT at a glance.** Center: all 114 assets at true metric scale. Surrounding panels: six assets at rest (left) and in an articulated state (right) from the same viewpoint.

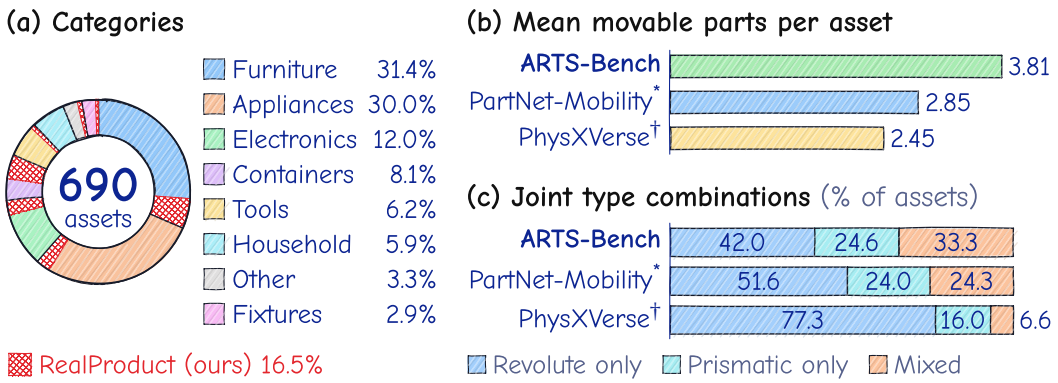


Figure 4: ARTS-Bench statistics. (a) Categories, with REALPRODUCT marked in red hatching. (b) Mean movable parts. (c) Joint type combinations. *Augmented PartNet-Mobility training subset used by SINGAPO. †PhysXVerse subset with revolute or prismatic joints.

reveals, such as cavities behind doors and the insides of drawers, so that appearance can be scored once parts move (Sec. 4.3). For breadth, 46 of the 63 product types occur only once, including items that articulated datasets rarely cover, such as display freezers, egg cartons, and folding phones, and REALPRODUCT supplies 30 of the 56 containers in ARTS-Bench. Mechanisms range from a laptop's single screen hinge to an 18-drawer chest, and the largest asset has 36 movable parts (Fig. 3).

Each movable part is annotated with its joint type, axis, anchor, and limits in the object frame at rest, and every asset is provided in USD for Isaac Sim and in URDF after automatic checks (Appendix C).

4.2 ARTS-BENCH

Unified sources. ARTS-Bench combines the 114 REALPRODUCT assets with 100 from REALAPPLIANCE (Gao et al., 2026) and 476 from ArtVIP (Jin et al., 2026), for 690 assets and 2,628 movable joints. All assets are converted into one representation with validated joints and their materials, and source categories are mapped to eight shared categories (Fig. 4a; Appendix D).

Complexity beyond existing datasets. Assets have 1 to 36 movable parts, 3.81 on average, compared with 2.85 in the augmented PartNet-Mobility subset used to train SINGAPO (Liu et al., 2025a) and 2.45 in a comparable PhysXVerse subset (Cao et al., 2026b). A third of the assets (33.3%) combine revolute and prismatic joints, against 24.3% and 6.6% (Fig. 4b,c).

Table 1: Comparison on 50 evaluation objects. Every metric is reported in the rest state (RS) and in sampled articulation states (AS). * marks methods trained for asset generation. Best results in bold.

	Geometry						Appearance					
	$d_{\mathrm{GIoU}}\downarrow$		cDist ↓		CD ↓		LPIPS ↓		PSNR ↑		FG IoU ↑	
Method	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS
SINGAPO*	1.2256	1.2308	0.2182	0.2550	0.0748	0.0990	0.3325	0.3280	16.96	17.28	0.6853	0.6367
PhysX-Anything*	1.1271	1.1350	0.3043	0.3550	0.0970	0.1538	0.2524	0.2622	18.52	18.38	0.7202	0.6528
PhysX-Omni*	1.1120	1.1198	0.2869	0.3283	0.1014	0.1286	0.2535	0.2617	18.21	18.19	0.7126	0.6516
Articulate-Anything	1.2307	1.2402	0.2068	0.3387	0.0935	0.2480	0.2861	0.2862	17.89	18.15	0.6973	0.6286
Articraft	0.9995	1.0078	0.0929	0.1142	0.0327	0.0462	0.2063	0.2133	21.69	21.11	0.7950	0.7506
Codex-Native	0.8253	0.8379	0.0648	0.1080	0.0258	0.0582	0.1814	0.1924	22.08	21.44	0.8301	0.7750
DUALCRAFT	**0.7961**	**0.8077**	**0.0599**	**0.0875**	**0.0203**	**0.0313**	**0.1677**	**0.1840**	**22.54**	**21.47**	**0.8460**	**0.7917**

Means over three runs per object. d_{GIoU}: bounding box error ($1 - \mathrm{GIoU}$, range $[0, 2]$); FG IoU: image foreground IoU. PhysX-Anything and PhysX-Omni each have 49 evaluated assets; all other methods have 50.

Evaluation subset. Every ARTS-Bench asset can be used for evaluation. For cost reasons, the experiments in this paper use a fixed subset of 50 objects (20 from REALPRODUCT, 15 from REALAPPLIANCE, and 15 from ArtVIP), drawn with a fixed seed within each source category and adjusted to cover every joint trait (Appendix D.3).

4.3 INPUT PROTOCOL AND EVALUATION METRICS

Inputs. In the single-image comparison, every method receives the same 512×512 RGB image of the asset at rest, rendered in Isaac Sim from the front over a white background, and no category, part count, or joint description. The multiple-image protocol uses nested sets of 1, 4, and 6 views of the same state. Reference geometry and joints are used only to render inputs and score outputs.

Geometry at rest and in motion. Following SINGAPO (Liu et al., 2025a), we center both assets, scale the generated asset along each axis to the reference bounds, and match parts with the Hungarian algorithm on bounding-box centers, keeping the matches fixed across states. We report the bounding-box error $d_{\mathrm{GIoU}} = 1 - \mathrm{GIoU}$ (Rezatofighi et al., 2019), the part-center distance (cDist), and the Chamfer distance (CD), at rest (RS) and averaged over five articulation states including rest (AS); lower is better.

Appearance at rest and in motion. We render the generated and reference assets in Isaac Sim with path tracing, with the same cameras and lighting, from eight viewpoints in five articulation states. States are indexed by each asset's own motion range, so opening a door exposes the interior on both sides of the comparison. We report LPIPS (Zhang et al., 2018), PSNR, and foreground IoU over the eight rest views (RS) and all 40 images (AS; Appendix D.6).

Why articulated states matter. Evaluating in motion changes the conclusions: for every method in Table 1, CD increases and foreground IoU decreases from RS to AS, and baseline rankings shift: the two strongest baselines swap places in CD, and Articulate-Anything falls from third to fifth in cDist.

5 EXPERIMENTS

5.1 COMPARISON WITH EXISTING METHODS

Evaluation setting and compared methods. We evaluate generation from a single image on the 50 evaluation objects (Sec. 4.2) with the protocol of Sec. 4.3; all methods that query a language model use `gpt-6-astra` with high reasoning effort. SINGAPO, PhysX-Anything, and PhysX-Omni are trained for asset generation; the other methods prompt the model with tools. Codex-Native runs the Codex coding agent with the same model, building the asset directly in Blender within the same 20-turn budget as DUALCRAFT (Appendix E).



Figure 5: Generation from single product photographs, which have no ground truth. Each prediction is shown at rest (top) and articulated (bottom) under one lighting setup; the line below lists its movable joints, and red marks predictions without movable parts. Two photographs show the object partly opened.

Quantitative comparison. DUALCRAFT achieves the best score on every metric in Table 1 (per-object results and examples in Appendices F.1 and G.1). Codex-Native is the strongest baseline. It comes close to DUALCRAFT in appearance (LPIPS within 0.014, PSNR within 0.5 dB) but falls behind in part geometry, and more so once parts move and geometry and joints must agree: DUALCRAFT lowers cDist by 8% at rest and 19% in articulated states, and CD by 21% and 46%.

Scene or program alone. With the same model, Articraft keeps the asset as a program and Codex-Native as a Blender scene; Codex-Native leads at rest and in appearance, Articraft in CD once parts move. DUALCRAFT, which couples both through shared part frames, beats both on every metric; since all three use the same model, the gains come from the system design rather than the model, and Sec. 5.2 isolates the shared frames.

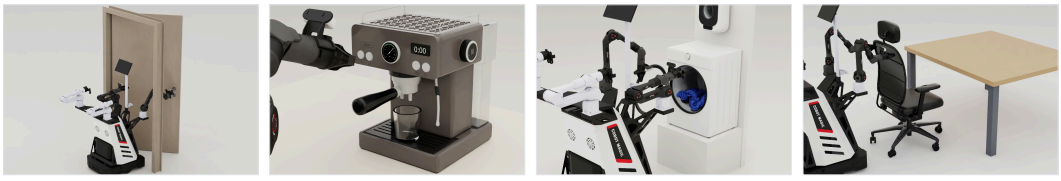
Product photographs. Fig. 5 applies DUALCRAFT and the five prior baselines to three product photographs (Appendix G.2). Several baselines return no movable parts or a different object, whereas DUALCRAFT recovers the drawers, the folding frame, and the lid and wheels.

5.2 ABLATION STUDIES

We ablate the number of input views (1, 4, and 6) and three components, geometry checks, visual feedback, and shared frames, each removed alone (Appendix B.7), on 20 objects we selected because a single view does not fully show them, some also in the 50-object subset (Table 2; Appendix G.3).

Table 2: DUALCRAFT ablations on 20 objects not fully shown by one view, means of three runs. Overlap: undeclared interpenetrating part pairs at rest (Obj.: % of assets with any; Pairs: mean).

	Geometry						Appearance						Structure	
	$d_{\mathrm{GIoU}}\downarrow$		cDist ↓		CD ↓		LPIPS ↓		PSNR ↑		FG IoU ↑		Overlap ↓	
Setting	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS	Obj.	Pairs
DUALCRAFT (1 view)	1.3339	1.3353	0.2520	0.3075	0.2555	0.3340	0.1850	0.2088	20.55	19.97	0.7267	0.6765	0	0.0
DUALCRAFT (4 views)	0.8304	0.8368	0.0461	0.0577	0.0040	0.0109	0.1204	0.1233	22.75	22.51	0.8416	0.8319	0	0.0
DUALCRAFT (6 views)	0.8606	0.8726	0.0499	0.0785	0.0059	0.0192	0.1142	0.1289	23.26	22.55	0.8493	0.8199	0	0.0
w/o geometry checks	1.4049	1.4044	0.4695	0.5307	0.5569	0.6690	0.1963	0.2319	20.70	19.33	0.6576	0.5817	100	9.3
w/o visual feedback	1.4295	1.4324	0.4600	0.5201	0.6009	0.6645	0.2207	0.2553	17.99	17.14	0.6327	0.5549	0	0.0
w/o shared frames	1.4067	1.4093	0.4848	0.5340	0.7026	0.7142	0.2231	0.2541	20.38	19.12	0.5897	0.5188	0	0.0



(a) **Interior door** (b) **Coffee machine** (c) **Washer** (d) **Office chair**

Figure 6: DUALCRAFT assets operated by a bimanual ALOHA robot in Isaac Sim (Appendix H.1).

Number of views. From one to four views, every geometry and appearance metric improves (CD at rest by 98%); six views mainly improve rest-state appearance, so four capture most of the gain.

Components. Removing any component from the single-view setting worsens every geometry metric and more than doubles CD at rest. Without geometry checks, every asset has undeclared interpenetrating part pairs at rest, 9.3 on average, and no other setting has any. Without visual feedback, the agent never sees its scene rendered, and PSNR is the lowest in the table, 2.56 dB below the full method at rest. Without shared frames, the agent writes every joint origin itself, and CD and cDist are the highest in the table in both states. Each component thus guards against a different error. In the first run on each object of Table 1, frames moved after the first rebuild in 17 runs, with joint origins following, and structural or motion checks reopened geometry editing in 8 (Appendix F.3).

6 APPLICATIONS IN ROBOTICS

Doors and levers need joints that work under contact. In Isaac Sim, a bimanual ALOHA robot operates DUALCRAFT's assets in eight household tasks, from pulling open a fridge door to twisting a portafilter into place (Fig. 6; Appendix H.1); with open-loop scripted trajectories, contacts and grasps resolved by PhysX, and randomized object poses, each task succeeds in 50/50 trials. Trajectories planned in simulation with a generated microwave and laptop transfer unchanged to a real dual-arm robot, succeeding in 10/10 trials on each task (Appendix H.2).

7 CONCLUSION

We presented DUALCRAFT, an agentic system that generates simulation-ready articulated assets from images without additional training. Its two workspaces share part frames, so geometry and joints stay consistent; more generally, our results suggest that giving each quantity one owner and regenerating the rest after every edit helps keep an agent's representations from drifting apart. We also introduced REALPRODUCT and ARTS-Bench, which scores appearance and geometry at rest and in motion; on it, DUALCRAFT scores best on every metric against six baselines, including same-model agents that keep the asset in only one of the two representations. Its assets support robot manipulation in simulation, with plans made on them carrying over to a real robot. Its main limitations are that hidden parts must be inferred and physical properties are approximated (Appendix G.5).

AI USE STATEMENT

In this work, we used generative AI tools to implement methods, through coding assistants during software development, to provide feedback on the design of experiments, to propose and refine hypotheses, and to help interpret results. We have not used generative AI tools to generate synthetic data sets, to assist with translation, or to clean and reformat datasets, and theoretical models, mathematical claims and proofs, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, to draft and edit parts of the paper, and to check and format references. Large language models are also part of the method studied here: DUALCRAFT and every compared method that queries a language model use `gpt-6-astra` (Section 5.1; Appendices B.1 and E). REALPRODUCT was modeled by hand, and no language model was used for data annotation. We have reviewed all AI-assisted work: the authors designed the method and the experiments, tested the code, ran all experiments, and verified every reported result. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Appendix B describe DUALCRAFT: the model and run configuration, blueprint generation, the workspaces and every agent tool, the visual feedback, the assembly and motion checks, the completion checks, and the settings of each ablation. Appendix C describes the composition, format, and release checks of REALPRODUCT. Appendix D gives the preparation and validation of the ARTS-Bench assets, the evaluation subset, the input images, and the metric definitions. Appendix E states how each compared method was run and converted for evaluation, and Appendix H describes the manipulation tasks and their trials. We will release the code of DUALCRAFT, the REALPRODUCT assets, and the ARTS-Bench conversion scripts, evaluation code, and evaluation subset; the REALAPPLIANCE and ArtVIP assets are obtained from their original releases.

REFERENCES

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Conference on Robot Learning (CoRL)*, 2025.

Ziang Cao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-Anything: Simulation-Ready Physical 3D Assets from Single Image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Ziang Cao, Yinghao Liu, Haitian Li, Runmao Yao, Fangzhou Hong, Zhaoxi Chen, Liang Pan, and Ziwei Liu. PhysX-Omni: Unified Simulation-Ready Physical 3D Generation for Rigid, Deformable, and Articulated Objects. *arXiv preprint arXiv:2605.21572*, 2026b.

Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Dieter Fox, and Abhishek Gupta. URDFormer: A Pipeline for Constructing Articulated Simulation Environments from Real-World Images. In *Robotics: Science and Systems (RSS)*, 2024.

Yuzheng Gao, Yuxing Long, Lei Kang, Yuchong Guo, Ziyan Yu, Shangqing Mao, Jiyao Zhang, Ruihai Wu, Dongjiang Li, Hui Shen, and Hao Dong. RealAppliance: Let High-fidelity Appliance Assets Controllable and Workable as Aligned Real Manuals. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yumeng He, Ying Jiang, Jiayin Lu, Yin Yang, and Chenfanfu Jiang. SPARK: Sim-ready Part-level Articulated Reconstruction with VLM Knowledge. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Nick Heppert, Muhammad Zubair Irshad, Sergey Zakharov, Katherine Liu, Rares Andrei Ambrus, Jeannette Bohg, Abhinav Valada, and Thomas Kollar. CARTO: Category and Joint Agnostic Reconstruction of ARTiculated Objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A. Ross, Cordelia Schmid, and Alireza Fathi. SceneCraft: An LLM Agent for Synthesizing 3D Scenes as Blender Code. In *International Conference on Machine Learning (ICML)*, 2024.

Ian Huang, Guandao Yang, and Leonidas Guibas. BlenderAlchemy: Editing 3D Graphics with Vision-Language Models. In *European Conference on Computer Vision (ECCV)*, 2024.

Denys Iliash, Jiayi Liu, Egor Fokin, Qirui Wu, Ali Mahdavi-Amiri, Manolis Savva, and Angel X. Chang. Artiverse: A Diverse and Physically Grounded Dataset for Articulated Objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building Digital Twins of Articulated Objects from Interaction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Zhao Jin, Zhengping Che, Tao Li, Zhen Zhao, Kun Wu, Yuheng Zhang, Yinuo Zhao, Zehui Liu, Qiang Zhang, Xiaozhu Ju, Jing Tian, Yousong Xue, and Jian Tang. ArtVIP: Articulated Digital Assets of Visual Realism, Modular Interaction, and Physical Fidelity for Robot Learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Long Le, Jason Xie, William Liang, Hung-Ju Wang, Yue Yang, Yecheng Jason Ma, Kyle Vedder, Arjun Krishna, Dinesh Jayaraman, and Eric Eaton. Articulate-Anything: Automatic Modeling of Articulated Objects via a Vision-Language Foundation Model. In *International Conference on Learning Representations (ICLR)*, 2025.

Ruining Li, Yuxin Yao, Chuanxia Zheng, Christian Rupprecht, Joan Lasenby, Shangzhe Wu, and Andrea Vedaldi. Particulate: Feed-Forward 3D Object Articulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Zhe Li, Xiang Bai, Jieyu Zhang, Zhuangzhe Wu, Che Xu, Ying Li, Chengkai Hou, and Shanghang Zhang. URDF-Anything: Constructing Articulated Objects with 3D Multimodal Language Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zizhang Li, Cheng Zhang, Zhengqin Li, Henry Howard-Jenkins, Zhaoyang Lv, Chen Geng, Jiajun Wu, Richard Newcombe, Jakob Engel, and Zhao Dong. ART: Articulated Reconstruction Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

Youtian Lin, Yikang Yang, Zhanpeng Hu, Mengqi Zhou, Feihu Zhang, Xun Cao, Jiaheng Liu, and Yao Yao. Procedura: Agentic 3D Modeling with Procedural Control. *arXiv preprint arXiv:2608.26238*, 2026.

Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. PARIS: Part-level Reconstruction and Motion Analysis for Articulated Objects. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

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 (ICLR)*, 2025a.

Jiayi Liu, Manolis Savva, and Ali Mahdavi-Amiri. Survey on Modeling of Human-made Articulated Objects. *Computer Graphics Forum*, 44(2):e70092, 2025b.

Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. AKB-48: A Real-World Articulated Object Knowledge Base. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Qingming Liu, Xinyue Yao, Shuyuan Zhang, Yueci Deng, Guiliang Liu, Zhen Liu, and Kui Jia. PAct: Part-Decomposed Single-View Articulated Object Generation. *arXiv preprint arXiv:2602.14965*, 2026.

Ruijie Lu, Yu Liu, Jiaxiang Tang, Junfeng Ni, Yuxiang Wang, Diwen Wan, Gang Zeng, Yixin Chen, and Siyuan Huang. Generating Objects with Part-Articulation from a Single Image. In *ACM SIGGRAPH Asia Conference Papers*, 2025a.

Sining Lu, Guan Chen, Nam Anh Dinh, Itai Lang, Ari Holtzman, and Rana Hanocka. LL3M: Large Language 3D Modelers. *arXiv preprint arXiv:2508.08228*, 2025b.

Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2Code: Reconstruct Articulated Objects via Code Generation. In *International Conference on Learning Representations (ICLR)*, 2025.

NVIDIA. Isaac Sim. `https://github.com/isaac-sim/IsaacSim`, 2025. Version 5.1.0.

Xiaowen Qiu, Jincheng Yang, Yian Wang, Zhehuan Chen, Yufei Wang, Tsun-Hsuan Wang, Zhou Xian, and Chuang Gan. Articulate AnyMesh: Open-Vocabulary 3D Articulated Objects Modeling. In *Conference on Robot Learning (CoRL)*, 2025.

Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Yuan Shui, Yandong Guan, Zhanwei Zhang, Juncheng Hu, Jing Zhang, Dong Xu, and Qian Yu. ArtiCAD: Articulated CAD Assembly Design via Multi-Agent Code Generation. *arXiv preprint arXiv:2604.10992*, 2026.

Penghao Wang, Siyuan Xie, Hongyu Yan, Xianghui Yang, Jingwei Huang, Chunchao Guo, and Jiayuan Gu. ArtLLM: Generating Articulated Assets via 3D LLM. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zhuangzhe Wu, Yue Xin, Chengkai Hou, Minghao Chen, Yaoxu Lyu, Jieyu Zhang, and Shanghang Zhang. 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, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas Guibas, and Hao Su. SAPIEN: A SimulAted Part-based Interactive ENvironment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yunhan Yang, Chunshi Wang, Junliang Ye, Yang Li, Zanxin Chen, Zehuan Huang, Yao Mu, Zhuo Chen, Chunchao Guo, and Xihui Liu. PhysForge: Generating Physics-Grounded 3D Assets for Interactive Virtual World. In *International Conference on Machine Learning (ICML)*, 2026.

Chuanrui Zhang, Minghan Qin, Yuang Wang, Baifeng Xie, Hang Li, and Ziwei Wang. SimArt: Decomposing Monolithic Meshes into Sim-ready Articulated Assets via MLLM. In *ACM SIGGRAPH Conference Papers*, 2026.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Matt Zhou, Ruining Li, Xiaoyang Lyu, Zhaomou Song, Zhening Huang, Chuanxia Zheng, Christian Rupprecht, Andrea Vedaldi, and Shangzhe Wu. Articraft: An Agentic System for Scalable Articulated 3D Asset Generation. *arXiv preprint arXiv:2605.15187*, 2026.

Appendix overview. Appendix A extends the related work. Appendix B gives the implementation of DUALCRAFT: the refinement loop, the run configuration, blueprint generation, the workspaces and the agent's tools, the feedback computations, the completion checks, and the ablation settings. Appendix C describes the composition and release of REALPRODUCT and shows the complete collection. Appendix D gives the ARTS-Bench statistics, asset preparation, the evaluation subset, the input images, and the metric definitions. Appendix E describes how the compared methods were run and converted for evaluation. Appendices F and G report additional quantitative and qualitative results, including runtime and failure cases, and Appendix H shows generated assets in simulated manipulation and repeats two of them on a real robot.

A EXTENDED RELATED WORK

This appendix extends Section 2. Liu et al. (2025b) survey the broader modeling of human-made articulated objects.

A.1 IMAGE-GUIDED ARTICULATED ASSET GENERATION

Image-guided articulated generation requires inferring part geometry, connectivity, and motion parameters. SINGAPO (Liu et al., 2025a) generates part attributes conditioned on a connectivity graph and image features, then retrieves meshes to complete the assembly. PAct (Liu et al., 2026) jointly models geometry and articulation properties in a part latent representation, while PhysX-Anything (Cao et al., 2026a) and PhysX-Omni (Cao et al., 2026b) further generate geometry, motion structures, and physical properties. URDFormer (Chen et al., 2024) predicts articulated kinematic structures from real-world images with a model trained on images synthesized by text-to-image models, and URDF-Anything+ (Wu et al., 2026) generates URDF models from a single image with an autoregressive diffusion framework. These methods reduce the need for manual asset creation by leveraging existing geometry or learned generative priors. Another line of work constrains geometry and motion with additional priors, optimization, or planning. DreamArt (Lu et al., 2025a) exploits part completion and video motion priors, SPARK (He et al., 2026) combines structural information from vision-language models with differentiable kinematic optimization, and PhysForge (Yang et al., 2026) organizes generation through hierarchical physical attribute planning. Particulate (Li et al., 2026a) takes 3D meshes as input and predicts parts and motion structures, and can also be chained with image-to-3D generation models; Articulate AnyMesh (Qiu et al., 2025), SimArt (Zhang et al., 2026), and ArtLLM (Wang et al., 2026) likewise infer parts and joints from existing meshes, and URDF-Anything (Li et al., 2025) from point clouds.

Reconstruction methods instead recover articulation from richer observations, such as multi-view images of two articulation states (Liu et al., 2023), observations before and after an interaction (Jiang et al., 2022), a stereo observation (Heppert et al., 2023), or sparse multi-state images (Li et al., 2026b); Real2Code (Mandi et al., 2025) reconstructs parts from multi-view images and predicts joints as code with a fine-tuned language model. Most of these methods learn structural priors from existing articulated assets, so their predictions remain constrained by the object categories and joint structures covered in training (Liu et al., 2026), and the feed-forward generators among them produce the asset in a single pass, without correcting it against the input or checking its motion. DUALCRAFT requires no additional training for articulated asset generation and only images of the object in a single state; it corrects the asset against these images and checks its motion before export.

A.2 AGENTIC MODELING WITH FEEDBACK

Agent systems can organize multi-round 3D modeling through tool calls, program execution, and observation of results. BlenderAlchemy (Huang et al., 2024) evaluates program edits with rendered results, supporting 3D scene editing. SceneCraft (Hu et al., 2024) writes Blender code for text-described scenes and refines it through visual self-critique, and LL3M (Lu et al., 2025b) coordinates multiple agents that write and refine Blender scripts for geometry and appearance from text prompts. Articulate-Anything (Le et al., 2025) accepts inputs such as images and videos, expresses part placement and joints in programs, and refines them through visual feedback. Articraft (Zhou et al., 2026) designs a dedicated programmatic interface and runtime environment that describes parts, geometry, joints, and check conditions, and repairs assets based on structured execution feedback. ArtiCAD

(Shui et al., 2026) determines assembly relations at the design stage, uses multiple agents to generate and assemble CAD parts, and fixes design and code errors through cross-stage rollback. Procedura (Lin et al., 2026) plans an assembly graph from text, writes procedural code whose named parts are joined by typed, machine-checkable mates, and refines the result with a vision critic.

The Blender-based agents edit shape and appearance with visual feedback but do not model articulation, while the articulated systems retrieve part meshes, start from text, or build geometry as code or parametric CAD parts, without refining free-form part meshes and materials against reference images. These works have established the foundations of program generation, validation, and cross-stage correction. We focus on how mesh refinement and joint modeling can collaborate under image constraints: the geometry workspace keeps part meshes that are directly editable and renderable for observation, while the program workspace references these meshes and defines their motion. The two types of operations share a common part organization and coordinate convention, and motion feedback can trigger geometric edits. The geometry workspace also refines materials and surface details against the reference images, so appearance is corrected on the same asset whose motion is checked.

A.3 Articulated asset datasets and evaluation benchmarks

PartNet-Mobility (Xiang et al., 2020) provides part motion annotations and simulation assets for articulated objects, AKB-48 (Liu et al., 2022) covers the structure, physical properties, and semantics of real objects, and GAPartNet (Geng et al., 2023) builds perception and manipulation tasks around actionable parts. ArtVIP (Jin et al., 2026) offers assets with modular interactive behaviors through professional manual modeling, texturing, and dynamics configuration, while RealAppliance (Gao et al., 2026) implements the physical mechanisms, electronic functions, and operational logic of home appliances based on product specifications. Artiverse (Iliash et al., 2026) aggregates 5.4K human-authored objects from static 3D repositories and annotates their functional parts, interior structures, multi-DoF joints, and physical attributes. Our RealProduct complements these efforts with 63 product types, including types that are absent or underrepresented in existing datasets; each asset is modeled by hand after a specific real product at its real size, with separate parts and joints, physically based materials, and the interiors that motion reveals, and is ready for direct use in simulation.

In terms of evaluation, SINGAPO (Liu et al., 2025a) measures part reconstruction quality in both rest and articulated states, Particulate (Li et al., 2026a) establishes a benchmark for mesh articulation estimation, and PhysX-Bench (Cao et al., 2026b) covers properties such as geometry, scale, material, affordance, and kinematics. ARTS-Bench brings three high-quality articulated data sources under a unified input, coordinate, and part-scoring protocol. Beyond part geometry in rest and articulated states, it evaluates appearance from matched multiview renderings in both states, including interior surfaces revealed by motion.

B DualCraft implementation

This appendix details Section 3. Alg. 1 summarizes the refinement loop of Sec. 3.5; the subsections below follow the order in which a run uses each component.

Algorithm 1 DualCraft

Input: reference images $\mathcal{I}$. **Output:** URDF and part meshes.

$\mathcal{C} \leftarrow$ AnchorCameras$(\mathcal{I})$; $B \leftarrow$ DraftBlueprint$(\mathcal{I}, \mathcal{C})$
$P \leftarrow$ JointPlan(B); place a frame per part in scene S
repeat ▷ geometry phase
  edit S; export meshes; rebuild the program from (P, S); run checks
  render S at $\mathcal{C}$ beside $\mathcal{I}$; update the difference record D
until checks pass and D is reviewed with no blocking entry, or 8 edits
repeat ▷ assembly phase
  validate S; commit meshes; compile the program and tests
  pose-check joints and correct their type, axis, or range; answer the checklist of B
  if a check fails **then** reopen edits on S for a rework window
until completion checks pass or the turn budget is spent

B.1 RUN CONFIGURATION

All DUALCRAFT results in Section 5.1 use one configuration. The planner, for camera anchoring and blueprint drafting, and the agent both call `gpt-6-astra` through its API with high reasoning effort. Part geometry and materials are edited in Blender 4.4. Each run is limited to 20 model turns, and the geometry phase ends after at most eight geometry edits. The input is the single 512×512 image described in Section 4.3. The object category passed to the agent is the blueprint's own estimate (Appendix B.2); no benchmark annotation is given to the agent. Turn counts, token usage and runtime over the 50 evaluation objects are reported in Appendix F.2.

B.2 BLUEPRINT GENERATION

Camera anchoring. One planner call covers all reference images, with one retry when a reply is rejected or two images map to the same view. For each image the model reports the direction the object's front faces in the image (toward or away from the camera, left, or right), which side face is visible, the azimuth magnitude (0–180°), the elevation, a distinguishing feature, a confidence (high, medium, or low), and the camera distance (near, normal, or far). The signed azimuth and the nearest of the twelve canonical views are derived in code. The drafting call receives, for each image, which faces its camera can and cannot show.

Drafting. The object-level estimates are composed into a short text stating the outer envelope at rest, the mass, the materials, the style, the use, and the numbers of parts and movable parts. Each part has an identifier, a parent listed before its children, a movable flag, a one-sentence description that states its location first, a visibility label, the images in which it is seen, and an appearance record with a color name, a hexadecimal color, a finish from a fixed vocabulary (matte, glossy, brushed metal, polished metal, transparent glass, rubber, fabric, or wood), printed content, and optional color zones. Movable identifiers use one of 46 mechanism families and are numbered from the bottom up or from the object's left to right. A hinge names its edge (left, right, top, bottom, front, or rear) and axis (vertical or horizontal); a slide names a direction (forward, backward, left, right, up, or down); ranges are given in degrees for hinges and turns, centimeters for slides, and millimeters for presses; and a turn may be symmetric. The rendered clause is appended to the part's description, for example "swings open 90° about a left-edge vertical hinge". The linter requires 2–64 parts, a single root, unique identifiers, part counts that match the text, end states for bounded motions, and descriptions and identifiers free of color and print words. A failing draft is retried with the errors attached, up to three attempts in total. No step reads ground truth, and the run receives the images without view labels. Figure 7 shows the blueprint of the espresso machine of Figure 2.

Visibility. The planner labels each part as visible, partial, or inferred; an inferred part is seen in no image. The agent's task text marks inferred parts as built from what such an object must have, and their checklist items test consistency with the agent's own renderings rather than with the reference.

Joint plan. Every non-root part receives one joint, a fixed joint if it is not movable. Hinges become revolute joints with range $[0, \theta]$; turns become revolute joints about the front–back axis, or about the vertical axis when described as a twist, with range $[-\theta, \theta]$ when symmetric; slides become prismatic joints along the stated direction; presses become prismatic joints pointing into the object; and spins become continuous joints. A motion or range the planner leaves open starts as a placeholder range, and the agent sets the joint's type, axis, and range during the run, as for the portafilter and the drip tray in Figure 7; the agent can also change the type and axis of a planned joint. Default signs make positive values open a part outward; for example, a vertical hinge on the left edge turns about $+z$ and one on the right edge about $-z$. The agent moves each frame to the pivot it chooses for the part, leaving the part's meshes in place; frames keep the object's orientation, and the agent cannot rotate, scale, create, or delete them. Each part's mass is its bounding-box volume times $400\ \mathrm{kg\,m^{-3}}$, with a minimum of 0.05 kg; the blueprint's mass estimate is applied when the asset is loaded into simulation (Appendix H.1).

Checklist. For each part in blueprint order, the checklist asks about presence, placement relative to the parent, stated proportions and counts, color and print, motion, and the end pose, which is judged on pose-check renderings. The agent answers every item as true, false, or not visible, giving one sentence of evidence for any answer other than true. A false answer blocks completion except on color and print items, which are advisory. Each part with printed content also opens a blocking print entry in the difference record at the start of the run.

Input



reference image

Camera anchor



front, 28° above shows the front and top

Part blueprint *as the agent receives it, abridged*

hibrew countertop espresso machine (coffee machine)

0.25 m wide, 0.30 m deep, 0.33 m high · 5.5 kg · 19 parts, 10 movable

- housing
  - cup_warming_plate
  - cup_guard_left
  - cup_guard_right
  - knob_1 — *turns about a left-right axis*
  - button_1–4 — *presses in*
  - pressure_gauge
  - brew_group
    - part_1 — *moves in a way the photos do not show*
      - portafilter_handle
      - filter_basket *(not visible)*
  - wand_1 — *turns in place*
  - tray_1 — *moves in a way the photos do not show*
    - drip_grille
  - part_2 *(not visible)* — *moves in a way the photos do not show*
    - lid_1 *(not visible)* — *moves in a way the photos do not show*

One entry in full

knob_1 [MOVABLE: turns about a left-right axis] parent=housing: Left side of the upper housing near its front edge, a short outward-projecting cylindrical knob with a broad circular grip and a narrow mounting neck — appearance: black matte (#111211); no print — end state: Grip at a different angular position beside housing's left face.

Figure 7: Blueprint of the espresso machine of Figure 2, as the agent receives it (abridged), with the camera anchor of its reference image.

B.3 Workspaces, part frames and agent tools

Part frames. Each part of the joint plan (Appendix B.2) is represented in the Blender scene by an empty frame object, and its meshes are parented to that frame. The frame's position is the part's pivot in the shared object frame of Section 3.1; frames keep the object's orientation, and a rotated frame is rejected at commit.

Assembly program. After every successful edit and at every commit, each part's meshes are exported in the part's local frame, and the assembly program is generated from the joint plan and the frame transforms and then compiled. Each part's inertia is approximated from its bounding box. The agent does not write the assembly program itself; it can edit only the program's test function.

Statement-wise execution. Blender edits are executed one top-level statement at a time, with a viewport update between statements. If a statement fails, the preceding statements remain in effect and the error names the failed statement, so the agent repeats only the rest of the edit. Each successful edit returns an automatic geometry check of the current scene.

Difference record. An entry records the scene revision at which it was opened, the view, the affected parts, its kind (missing, extra, placement, proportion, orientation, motion, appearance, or print), and a note; the agent resolves an entry or drops it with a reason, and the harness retires entries that stall so that a run does not loop on one discrepancy. The geometry phase ends when the last edit has passed its geometry check, the observations since that edit have been reviewed, and no blocking entry is open; otherwise it ends after eight successful edits.

Table 3 lists the tools available to the agent.

Table 3: Tools available to the agent. The reference tools are available when an input image is given. `compile_model` and `probe_model` are adapted from Articraft (Zhou et al., 2026).

Tool	Function
Geometry workspace	
`blender_edit`	Runs Blender Python statements in the live scene and returns a geometry check of the scene.
`blender_move_frame`	Moves a non-root part's frame to the pivot the agent specifies, leaving the part's meshes in place; the assembly program is then regenerated, so the joint origin follows.
`blender_observe`	Renders the whole asset, or up to three named parts for a local view.
`view_image`	Reopens an earlier rendering.
`reference_zoom`	Crops a region of the input image.
`reference_trace`	Rectifies a printed face of the input image onto a labeled grid to read the position and size of marks.
Program workspace	
`blender_validate`	Checks the export specification and returns renderings and a validation token.
`blender_commit`	Exports the parts, then regenerates and compiles the assembly program.
`blender_set_joint_axis`	Sets a joint's type, axis, and range in the joint plan.
`pose_check`	Renders selected joints at their far limits.
`compile_model`	Compiles the program and runs the assembly checks; in finalize mode it also runs the completion checks and ends the run.
`probe_model`	Inspects the compiled asset without modifying it.
Review	
`blueprint_checklist_review`	Records answers to the blueprint checklist.
`blender_diff_review`	Opens, updates or closes entries in the difference record.

B.4 VISUAL FEEDBACK

Rendering. Observations are rendered in Blender Cycles with ACES tone mapping at 512×512 by default, in one fixed virtual studio for every object and every input: an environment map rotated with the view azimuth, camera-relative key and fill lights that do not appear in reflections, and a backdrop wall that the camera does not see. A reference view renders the asset from the anchored azimuth and elevation with a 24 mm lens (50 mm when the camera is near) and places the result to the right of the reference, scaled to the same height, both on the reference backdrop color. Because the anchored camera only approximates the reference camera, the panel is labeled to compare parts, sides, and proportions rather than outlines, and no pixel-overlap score is computed.

Close-ups and reference tools. A close-up aims the camera at the center of the combined bounding box of up to three named parts along the view direction, at the distance where the box corners fill 0.859 of the half-frame, and credits each named part toward the completion check that every part is inspected. The agent can crop a region of the reference covering at least 0.5% of the image, which is resized to a longer side of 1024 pixels, and can rectify a printed face by giving its four corners and physical size; the face is returned in color and in high contrast with a centimeter grid, so marks are redrawn rather than pasted. Drawing helpers create metric panel textures, text decals, and wood materials.

Color check. Each whole-asset observation compares the rendering with the reference in color. The foreground is taken from the alpha channel when present and otherwise consists of the pixels that differ from the backdrop by more than 4 levels in any channel; at least 64 foreground pixels are required, and the reference is resized to the rendering with nearest-neighbour alpha and bilinear color. Mean foreground colors are compared over the whole object and over a 3×3 grid. A luminance ratio below 0.5 or above 2.0 is reported as darker or lighter, and a chroma distance above 0.06 or a saturation gap above 0.05 as a tint. A darker or lighter result blocks completion once; tints are advisory.

B.5 Assembly and motion checks

After each edit. The checkpoint exports one part-local OBJ per frame and lists meshes that belong to no frame as well as frames that are missing or rotated. It reports the assembled size along each axis against the blueprint estimate (tolerance 15%, since the sizes are estimated from the image; deviations within 1 cm are ignored) and the flushness of paired sibling panels, such as `door_1` and `door_2` under one parent (tolerance $\max(5\ \mathrm{mm},\ 10\%$ of the panel thickness)). It then regenerates and compiles the program; nothing is committed.

Compiler checks. Compilation runs the agent's tests and a fixed baseline: unique names; a tree with a single root; non-zero axes; limits present exactly for bounded joints, with lower $\leq$ upper and positive effort, velocity, and mass; existing mesh files; connectivity, where the contact graph at rest (FCL, contact tolerance 10^{-6} m) must link every part to the root; and interpenetration at rest. For the latter, each part's visual mesh is split into connected components with one bounding-volume hierarchy each, and every unordered part pair is tested except declared allowances. Pairs whose bounding boxes overlap by more than 5 mm along all three axes and by more than $5 \times 10^{-7}\ \mathrm{m}^3$ are passed to an exact FCL collision query, and a collision fails the check. Declaring an allowed overlap requires a stated reason.

Agent tests. The SDK checks contact (an FCL collision or a minimum distance of at most 10^{-6} m), the gap along a world axis (bounds on the gap and on penetration), containment of one part's extent within another's up to a margin, and overlap of projected extents. A pose context sets joint values and places each child as in Eq. 2. Across the first main-table run on each object, the tests contain 144 contact checks, 51 gap checks, 26 containment checks, 12 overlap checks, 40 posed evaluations, and 97 declared overlap allowances.

Pose check. On request, a separate headless Blender loads the compiled URDF and sets each revolute and prismatic joint to its far limit, the bound of larger magnitude, with all other joints at rest; continuous joints are skipped. It renders two views (45° azimuth at 28° elevation and 90° azimuth at 10° elevation, 640×640 each) side by side, labeled with the blueprint's end state. The agent answers the checklist's pose items from these images and corrects a joint's type, axis, or range when its motion is wrong, flipping an axis sign at commit when a motion only points the wrong way.

B.6 Completion checks and export

Validation and commit. Validation rejects shadow, grain, and texture-proxy objects, requires parts to be meshes and frames to be empty objects, compares the union of the mesh bounding boxes with the blueprint's width, depth, and height with the tolerance of Appendix B.5, renders front and front-left views, and issues a token tied to the scene revision. Committing with the token exports one triangulated OBJ per part in its frame, with UVs, normals, and physically based material values, bakes procedural materials into textures of 256 to 2048 pixels according to surface area, and regenerates and compiles the program.

Rework windows. After the geometry phase, scene edits are refused except in a rework window. A window opens after a failed compilation or validation, after a rejected completion check, or once for a new blocking difference entry, and it closes at the next commit or after three turns.

Completion checks. Finalizing requires a committed, unchanged scene and an up-to-date compilation. It then checks, in order, that the URDF uses the committed meshes; that link frames at $\mathbf{q} = \mathbf{0}$ equal the scene frames within 10^{-5}, with unit scale and identity visual origins; that every committed part's parent is committed and every blueprint part is committed; the assembled size; that every part has had a close-up; that a reference view has been observed since the last edit; that no blocking difference entry is open; that the checklist is current, complete, and free of non-advisory failures; and that the appearance has been attempted. The flushness of paired panels and the main color are also reported before finalizing and open a rework window when they fail.

Final turns. In the last two turns, once a compilation has succeeded, the harness relaxes the blocking checks so that a compiled asset is still delivered, and a run whose budget runs out with a committed scene is completed by the system. Every run of Table 1 finished within the 20-turn budget, and none relied on the relaxed checks.

Output. A run returns the assembly program, the compiled URDF, whose collision geometry reuses the visual meshes, and the part meshes with baked textures. For evaluation, the benchmark adds a part tree with 2,048-point surface samples per part and a USD conversion that keeps the kinematic hierarchy (Appendix E).

B.7 Ablation settings

Each component ablation of Section 5.2 removes one component and leaves everything else as in the full method. The end of the agent's prompt states which components are disabled and instructs the agent not to look for substitutes. Every setting of Table 2 is run three times on each of the 20 objects, and the table reports the means.

Without geometry checks. No geometric check runs at any point of a run. After each edit, the parts are still exported so that their pivots are tracked, but the program is not compiled and no signal is returned: interpenetration, floating parts, parts broken into disconnected pieces, and the flushness and coplanarity of paired panels are never reported. Every compilation runs without checks, including those at commit and finalization and those the system runs itself to rescue or complete a run: neither the compiler's checks nor the agent's tests are executed, and a compilation succeeds whenever a URDF can be exported. The size gate at validation and finalization and the structural validation at export (a single root part and complete mesh files) remain, and the pose check remains available after a successful commit.

Without visual feedback. The agent never sees a rendering of its own scene. The tools that render the asset beside the reference (`blender_observe`), render joints at their limits (`pose_check`), and maintain the difference record (`blender_diff_review`) are removed, and validation returns no renderings. The input image remains in the first message, together with the tools that crop it and rectify its printed faces. The completion checks that depend on observation are disabled: the close-up of every part, the comparison with the reference view, the difference record, and the color check. Print entries are not opened, and changing a joint axis no longer requires a prior pose check.

Without shared frames. Joint origins are no longer derived from the part frames. The agent writes each joint origin into the program, in its parent's frame, with a `set_joint_origin` tool; a joint without one stays at the world origin and is rejected at commit. `blender_move_frame` is removed, and the blueprint, the other tools, the checks, the visual feedback, and the budget are unchanged. Meshes are still exported at their positions in the scene, so the written origins do not move parts at rest. Because the agent must keep the origins consistent with its own edits over many iterations, however, errors accumulate, and the rest-state geometry degrades as well (Table 2).

C RealProduct dataset

C.1 Composition

By mechanism family, the 410 movable joints comprise 136 doors, 119 drawers, 47 pivots, 34 lids, 27 knobs, 26 sliders, 18 buttons and 3 flaps; laptop screens are counted as doors. Every family has a single joint type except lids (31 revolute, 3 prismatic) and buttons (3 revolute, 15 prismatic). Figure 8 shows the number of movable parts per asset and the joints by family.

C.2 Format and release

Each asset is provided in URDF and USD with separate part meshes and their materials. A ground-truth record gives, for every movable joint, the joint type, axis, anchor point and limits, expressed in the object frame at rest. Automatic checks verify that each USD joint maps to one URDF joint, that axes and limits are valid, and that all material and texture files resolve within the asset bundle.

C.3 Gallery

Figures 9 and 10 show every RealProduct asset at rest and in an articulated state.

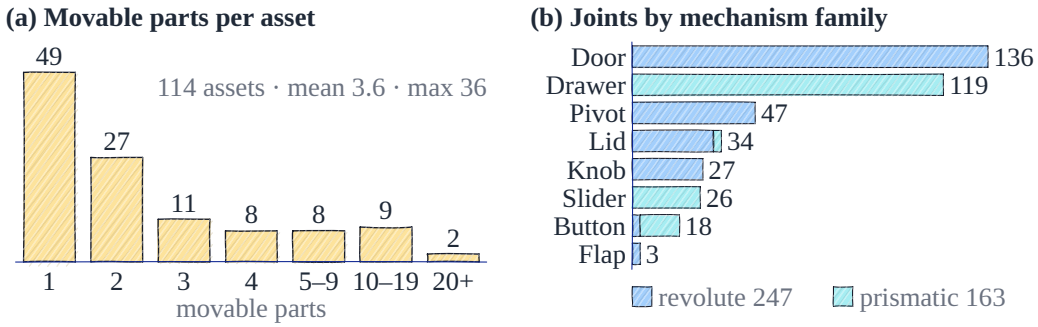


Figure 8: REALPRODUCT statistics. (a) Movable parts per asset. (b) Movable joints by mechanism family, split into revolute and prismatic.

D ARTS-BENCH DETAILS

D.1 DATASET STATISTICS AND COMPARISON SUBSETS

Table 4 lists the assets, joints, and movable parts that each source contributes to ARTS-Bench.

Table 4: Sources of ARTS-Bench. Joints are movable joints; "mixed" is the share of assets with both revolute and prismatic joints.

Source	Assets	Joints	Mean movable parts	Mixed (%)	Max movable parts
REALPRODUCT (ours)	114	410	3.60	17.5	36
REALAPPLIANCE	100	428	4.28	64.0	13
ArtVIP	476	1,790	3.76	30.7	24
ARTS-Bench	690	2,628	3.81	33.3	36

Categories. All statistics in Fig. 4 count assets rather than mesh objects. The eight shared categories are furniture, appliances, electronics, containers, tools, household items, fixtures and other objects. They contain 217, 207, 83, 56, 43, 41, 20 and 23 assets, respectively, of which REALPRODUCT contributes 36, 17, 18, 30, 5, 0, 3 and 5. The nine original category labels of REALPRODUCT are mapped to these categories; in particular, laptops are included in electronics and small appliances in appliances. Product types are counted from the distinct product labels within REALPRODUCT and are not added to the shared category count.

Joints. Movable part counts refer to movable joints; fixed parts are excluded. ARTS-Bench contains 290 assets with only revolute joints, 170 with only prismatic joints and 230 with both types. For the source comparison, continuous rotation is grouped with revolute joints.

Comparison subsets. The PartNet-Mobility comparison uses the 3,077 assets in the augmented training subset released with SINGAPO (Liu et al., 2025a). The PhysXVerse comparison uses 2,931 assets whose original motion definitions contain only revolute or prismatic joints, including objects with both types. Static objects and objects with free, ball or combined joints are excluded. These comparisons describe the specified subsets, not the complete source datasets.

D.2 ASSET PREPARATION AND VALIDATION

The processed asset uses its original rest pose as $q = 0$. Joint axes and mounting positions are expressed in the object coordinate system at rest ($+x$ forward, $+z$ up), in meters and radians. We check that axes and positions contain finite values, normalize nonzero axes and reject reversed joint limits. We also check the part connections against the URDF structure and reject unreadable meshes or geometry that cannot be scored. For a part whose bounding box has zero extent along an axis, the geometry scorer uses two copies of its surface shifted by ± 0.5 mm along that axis. This gives a 1 mm extent while preserving the bounding box center. This treatment is used only in the geometry evaluation representation.



Figure 9: REALPRODUCT gallery (1 of 2). Each asset is shown at rest and articulated, with its product type and number of movable joints.



Figure 10: REALPRODUCT gallery (2 of 2), in the layout of Figure 9.

Each run records its input image and configuration. The evaluator records the output status and the reason when conversion or scoring fails. Metric averages use valid outputs, with their sample counts reported separately from completion over all test objects. No error score is assigned to an output that cannot be scored.

Release. We will release the conversion scripts and the evaluation code for all three sources, together with the list of the evaluation subset. The REALPRODUCT assets will be released with this paper (Appendix C.2); the REALAPPLIANCE and ArtVIP assets are obtained from their original releases under their own licenses and converted with our scripts.

D.3 EVALUATION SUBSET

Every ARTS-Bench asset can be used for evaluation; for cost reasons, the comparison in Section 5.1 uses a fixed subset of 50 objects: 20 from REALPRODUCT, 15 from REALAPPLIANCE and 15 from ArtVIP. Each source has a fixed quota that is filled stratum by stratum, by category for REALPRODUCT and ArtVIP and by appliance type for REALAPPLIANCE, with a seeded random draw inside each stratum. Assets with the same label, joint signature and overall dimensions form a twin group and are drawn together, so the subset and the remaining 640 assets share no near duplicates. After the draw, a few whole twin groups were exchanged by hand so that the subset covers each joint trait, such as buttons and knobs, without changing the per-source totals. The list of the 50 objects will be released with the benchmark.

D.4 INPUT IMAGES

Figure 11 shows the input image of each evaluation object, as given to every method in Section 5.1.



Figure 11: Input images of the 50 evaluation objects, grouped by source.

D.5 GEOMETRIC ALIGNMENT AND PART MATCHING

We follow the SINGAPO geometry protocol (Liu et al., 2025a). Both assets are centered, and the generated asset is scaled along each axis to match the overall bounding box of the reference. This

alignment removes differences in overall size and proportions before part geometry is scored. The distance between part bounding box centers is used as the cost for Hungarian matching. When one asset has more parts, its remaining parts are assigned to the closest center in the other asset. We compute mappings in both directions and average the two directional scores. The mappings remain fixed during motion evaluation.

For matched part boxes $\hat{B}$ and B with centers $\hat{\mathbf{c}}$ and $\mathbf{c}$, the pairwise errors are

$$d_{\mathrm{GIoU}}(\hat{B}, B) = 1 - \mathrm{GIoU}(\hat{B}, B), \qquad d_{\mathrm{center}} = \|\hat{\mathbf{c}} - \mathbf{c}\|_2. \tag{3}$$

The reported cDist averages d_{center} over the part matches. For sampled surfaces P and Q, the pairwise CD is

$$D_2(P, Q) = \frac{1}{|P|} \sum_{\mathbf{p} \in P} \min_{\mathbf{q} \in Q} \|\mathbf{p} - \mathbf{q}\|_2^2 + \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} \min_{\mathbf{p} \in P} \|\mathbf{q} - \mathbf{p}\|_2^2. \tag{4}$$

We average each pairwise error over source parts for each mapping direction, then average the two directional scores. Bounding box error and cDist exclude the base part, while CD includes it. CD uses 2,048 points per part. cDist has units of length and CD has units of squared length, using the reference asset scale. RS uses the initial state. AS averages five states at normalized values 0, 0.25, 0.5, 0.75 and 1, including the initial state. Prismatic joints follow their stored evaluation ranges. The geometry scorer uses a signed 90° span for revolute joints and a full turn for continuous joints. Both assets are moved by the same joint values from rest, so their geometry and joint axes are compared under identical motion; errors in the generated limits appear in the appearance evaluation, which samples each asset's own range (Appendix D.6). Each joint moves its part and all parts below it in the assembly tree.

D.6 Appearance sampling and rendering

Appearance evaluation uses the motion limits of each asset. For a finite range on one side of rest, we sample five evenly spaced values from rest to the endpoint. Revolute motion is capped at 90°; prismatic motion uses the stored endpoint. When a finite range extends to both sides of rest, the five values are rest, the midpoint and endpoint on the negative side, and the midpoint and endpoint on the positive side. Continuous joints sample one full turn. All joints use the same state index. Thus, a state index refers to the corresponding stage of each asset's own motion, not an identical joint value shared by the generated and reference assets.

Each state is rendered from the front, front left, front right, back, back left, back right, top and bottom. The six side views have an elevation of 28°, and the top and bottom views use $\pm 89.9^\circ$. We align the generated asset to the reference using their rest state bounds and keep this transform fixed across states. For each view and state, the generated asset uses the camera pose and lighting recorded for the reference. It is not framed independently. Rendering uses Isaac Sim at 512×512 resolution. Figure 12 shows the five states and the eight views for one object.

The renderer records the object region in the alpha channel. RGB images are placed over the same gray background before LPIPS and PSNR are computed. Foreground masks use an alpha threshold of 128 on the 8 bit scale. Foreground IoU is the overlap of the two masks divided by their combined area. LPIPS uses AlexNet features. PSNR is computed from the mean squared RGB difference and capped at 100 dB. Each score is first averaged over the eight rest views or all 40 images for one asset, then averaged across valid assets. These image scores reflect both surface appearance and geometry, since shape and pose also change the rendered image. Reference motion images and benchmark scores are not returned to the generating method.

Let I and $\hat{I}$ denote a pair of RGB images with values in $[0, 255]$, and let N count their scalar color entries. We compute

$$\mathrm{MSE} = \frac{1}{N} \sum_{u=1}^{N} (I_u - \hat{I}_u)^2, \qquad \mathrm{PSNR} = \min\left(100, 10 \log_{10} \frac{255^2}{\mathrm{MSE}}\right). \tag{5}$$

Identical images receive 100 dB. With foreground regions S and $\hat{S}$, the image foreground score is

$$\text{FG IoU} = \frac{|S \cap \hat{S}|}{|S \cup \hat{S}|}. \tag{6}$$

If both regions are empty, the implementation returns one. For LPIPS, RGB values are first normalized to $[0, 1]$. The score uses spatially averaged differences between normalized AlexNet features with the learned channel weights from Zhang et al. (2018):

$$d_{\mathrm{LPIPS}}(I, \hat{I}) = \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} \left\| \mathbf{w}_l \odot \left(\phi_l(I)_{hw} - \phi_l(\hat{I})_{hw} \right) \right\|_2^2 . \tag{7}$$

Here, ϕ_l denotes normalized features at layer l, $H_l W_l$ is the number of spatial locations, and $\mathbf{w}_l$ contains the learned weights. The operator $\odot$ denotes elementwise multiplication.

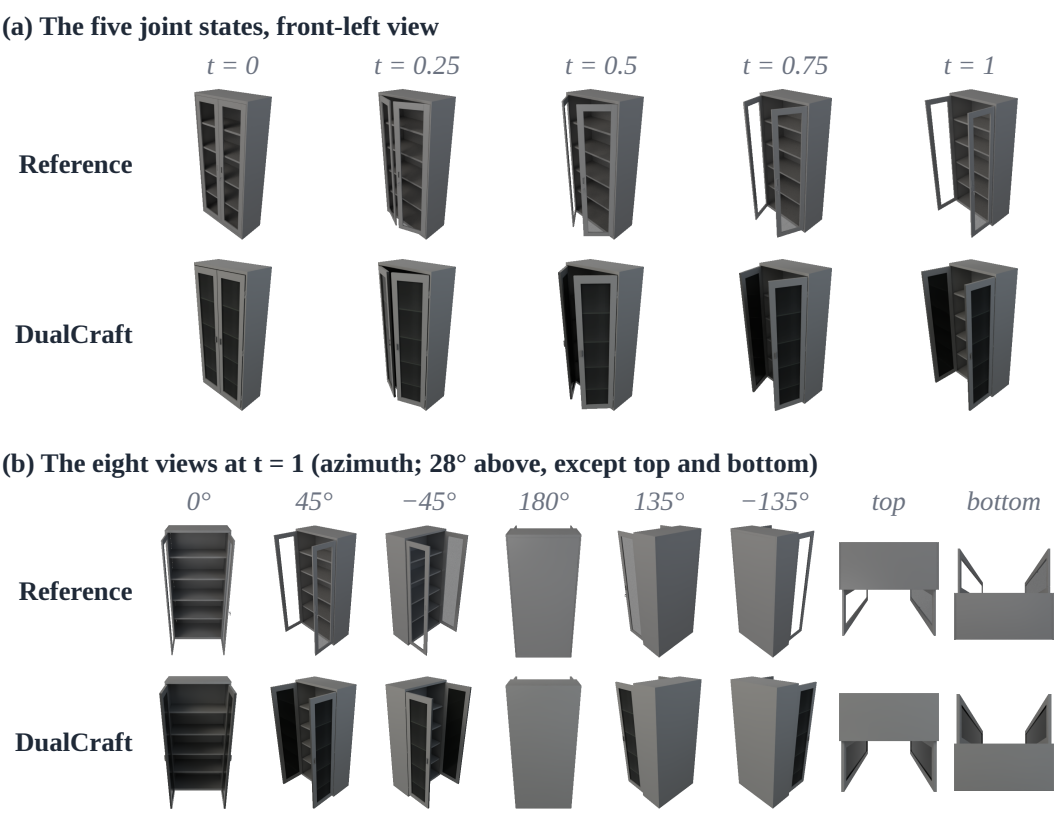


Figure 12: Appearance evaluation of one object, reference and DUALCRAFT. (a) The five joint states from the front-left view. (b) The eight views in the last state.

E BASELINE CONFIGURATIONS

All methods receive the same 512×512 front view of each object. Each baseline from prior work is run on the 50 evaluation objects from the latest version of its official code repository at the time of our experiments, with its released weights and default parameters. The only change is the language model: every method that queries one uses `gpt-6-astra` with high reasoning effort. We did not tune any baseline. Codex-Native runs the Codex coding agent with the same model on the same image: it builds the asset in Blender with at most 20 turns, as DUALCRAFT does, and its Blender output is converted to URDF. Every method is run three times on each object, and Table 1 reports the means. Table 5 summarizes the configurations.

Outputs are converted to one evaluation format: a part tree with each part's rest-state bounding box and joint parameters, and a surface sample of 2,048 points per part for the geometry metrics. Coordinates are converted to the benchmark frame, with z up and the object's front facing $+x$. For appearance evaluation, each prediction is converted to USD: predictions that include a URDF are converted from it directly, and SINGAPO predictions are assembled from the part tree and the

Table 5: Configurations of the compared methods. Trained: trained for articulated asset generation. Every method that queries a language model uses the same model with the same reasoning effort.

Method	Trained	Part geometry	Language/vision model	Evaluated
SINGAPO	✓	Retrieved (PartNet-Mobility, 567 objects)	`gpt-6-astra` (high)	50
PhysX-Anything	✓	Generated	Released model	49
PhysX-Omni	✓	Generated	Released model	49
Articulate-Anything		Retrieved (PartNet-Mobility)	`gpt-6-astra` (high)	50
Articraft		Generated by program	`gpt-6-astra` (high)	50
Codex-Native		Generated in Blender	`gpt-6-astra` (high)	50
DUALCRAFT		Generated in Blender	`gpt-6-astra` (high)	50

retrieved meshes. PhysX-Anything produced no output for one REALPRODUCT object and PhysX-Omni for one REALAPPLIANCE object; these are the generation failures in Table 1.

All six baselines run on an NVIDIA RTX 4090 D GPU with 48 GB of memory.

F ADDITIONAL QUANTITATIVE RESULTS

F.1 PER-OBJECT RESULTS

Table 6 counts, for each metric and state, the evaluation objects on which Articraft, Codex-Native, and DUALCRAFT score best among all seven methods. Ties count for every tied method.

Table 6: Number of the 50 evaluation objects on which each method is best among all seven.

	d_{GIoU}		cDist		CD		LPIPS		PSNR		FG IoU	
Method	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS	RS	AS
Articraft	4	5	6	15	7	12	2	5	13	13	9	8
Codex-Native	6	12	9	9	10	13	9	11	14	15	10	10
DUALCRAFT	31	24	25	16	25	16	37	29	19	15	29	28

F.2 RUNTIME

Over the 50 evaluation objects, a construction run of DUALCRAFT takes 16.3 minutes on average (median 16.1, range 5.7–29.2), measured from its first to its last logged step. It uses 13.6 model turns on average (median 13, range 7–20) and a median of 0.84M tokens. Camera anchoring and blueprint drafting precede the construction run and are not included in these figures; they take a median of about 15 and 58 seconds per object, estimated from file timestamps.

F.3 FEEDBACK ACROSS WORKSPACES

Table 7 counts how often feedback crossed the two workspaces in the first of the three runs on each of the 50 objects of Table 1. In the geometry phase, every edit is rebuilt and checked automatically: 120 of the 248 checks failed, in 42 runs, on interpenetration (105 checks) or floating parts (39; some checks report both), and the agent kept editing until they passed. In the assembly phase, geometry editing was reopened 14 times in 12 runs. Ten of these reopenings, in 8 runs, followed structural or motion checks: the compiler found floating and interpenetrating parts twice in one run, parts were left uncommitted once, failed motion or end-pose checklist items reopened editing five times in four runs, and pose-check renderings revealed a wrong motion twice, a lamp head tilting inward and the temples of a pair of glasses unfolding forward. The other four followed a control panel placed too low, two renderings darker than the reference, and an unreviewed checklist. At commit, 12 of the 376 joints of the final assets, in 7 runs, had their axis flipped (knobs, lamp-head joints, the temples of the glasses, a blade, a lever, and a scale arm), and all 12 flips remain in the final assets. In 47 runs the agent placed the part frames in its first edit. After the first rebuild, 30 edits in 17 runs moved 39 part frames, and each move updated the dependent joint origins at the next rebuild.

Table 7: Feedback across workspaces in the first run on each object of Table 1.

Event	Count	Runs (of 50)
Failed automatic checks in the geometry phase	120 of 248 checks	42
Geometry editing reopened in the assembly phase	14	12
after structural checks	3	2
after motion checks	7	6
for other reasons	4	4
Joint axes flipped at commit	12 of 376 joints	7
Part frames moved after the first rebuild	39 in 30 edits	17

G ADDITIONAL QUALITATIVE RESULTS

G.1 BENCHMARK EXAMPLES

Rendering of Fig. 13. Each generated asset is first scaled uniformly so that its largest rest-state dimension is the same, because predicted absolute scale varies by method. Within each object, the six generated columns are rendered in Blender with the same orthographic camera, lighting, and camera scale, so their sizes are comparable. The reference assets are rendered in Isaac Sim from their USD files with their own perspective framing, so that column differs in rendering and scale and is shown for shape and articulation only. The input column applies a display curve for legibility.

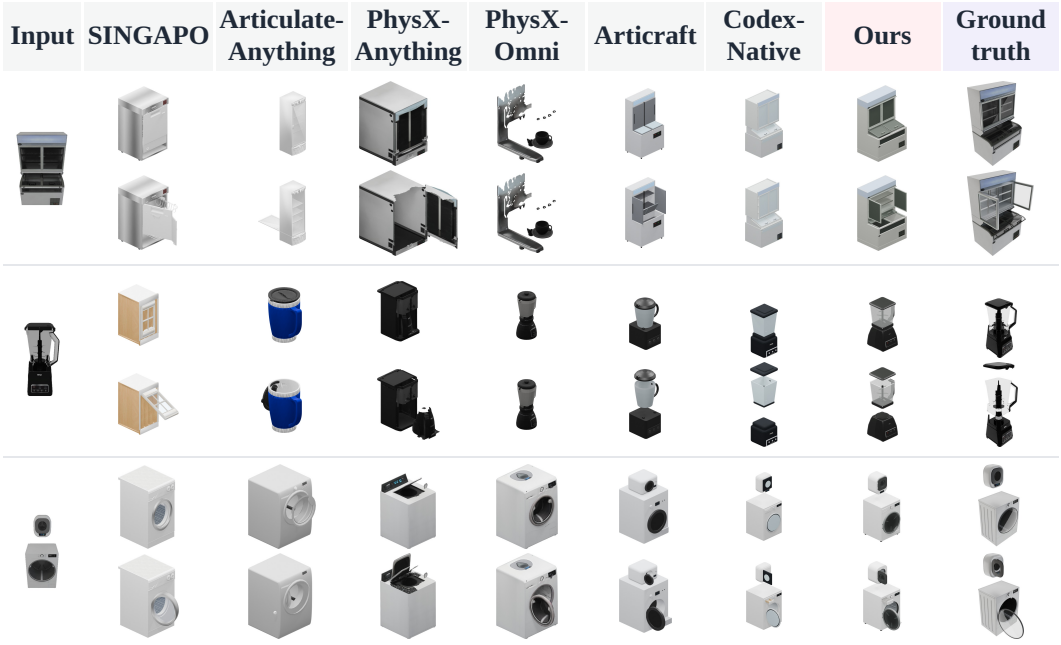


Figure 13: Qualitative comparison on three evaluation objects, a display freezer (REALPRODUCT), a blender (REALAPPLIANCE), and a washer-dryer (ArtVIP), each at rest (top) and in an articulated state (bottom). Generated assets are scaled to a common size and rendered with one camera and lighting per object; references are rendered in Isaac Sim (Appendix G.1).

Figure 14 shows six further evaluation objects, chosen at random, in the layout of Figure 13 without the reference column.

G.2 PRODUCT PHOTOGRAPHS

Figure 5 applies DUALCRAFT and the five baselines from prior work to three product photographs. The photographs have plain backdrops; each is padded to a square with the backdrop color sampled at its corners and resized to 512×512, without background removal. Predictions are rendered under



Figure 14: Qualitative comparison on six further evaluation objects, chosen at random, each at rest (top) and articulated (bottom).

one lighting setup and scaled for display. No ground truth exists for these objects, so no scores are computed.

G.3 Ablation examples

Figure 15 compares the view, geometry-check, and visual-feedback settings of Table 2 on example objects.

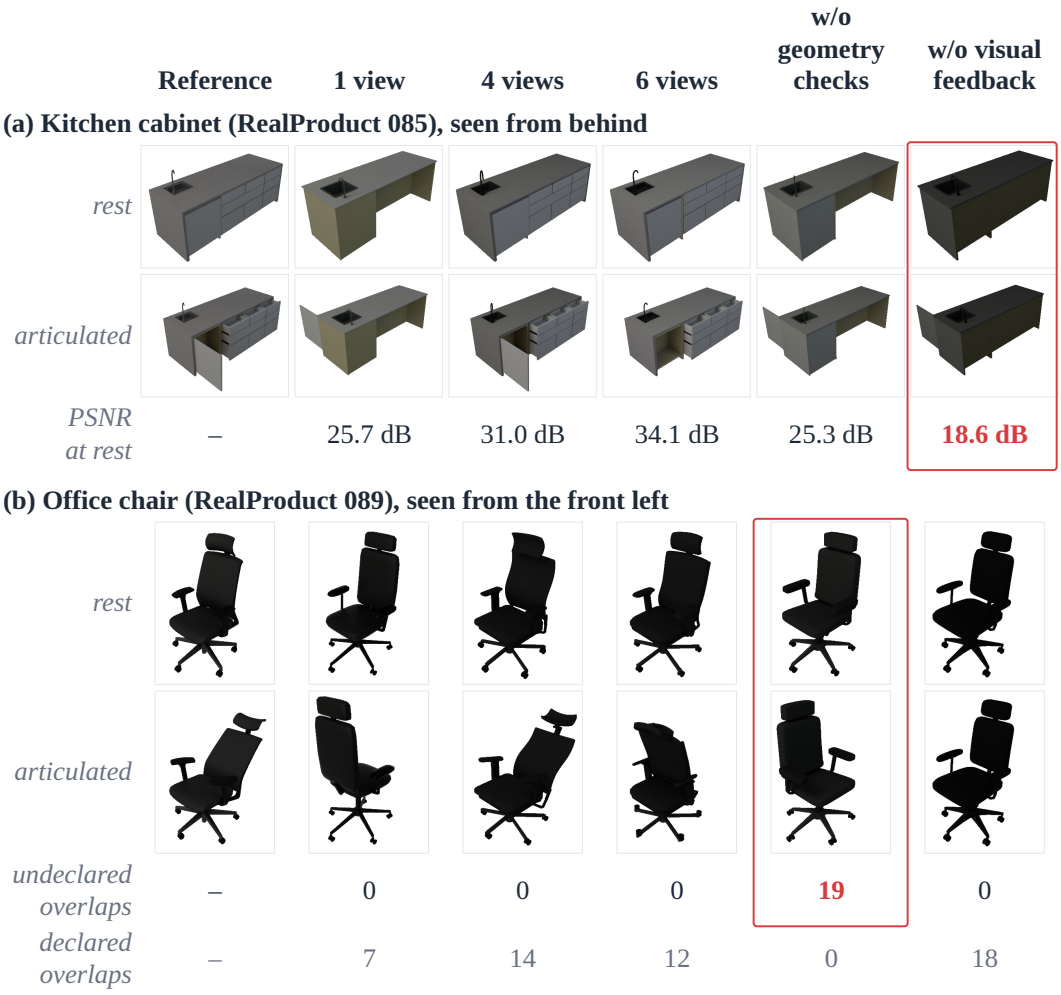


Figure 15: Visual comparison of the view, geometry-check, and visual-feedback settings in Table 2 on two ablation objects, at rest and articulated.

G.4 Refinement trajectories

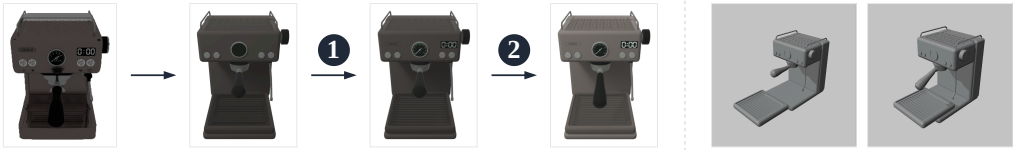
Figure 16 follows single runs through the loop of Alg. 1.

G.5 Failure cases and limitations

Figure 17 shows typical failures. Most start in the blueprint: a motion drafted with the wrong type (a deep drawer read as a hinged door, a lift-off lid as a twist cap), parts or joints missed (two of four doors, a telescoping pole), or an interior that no view shows left empty.

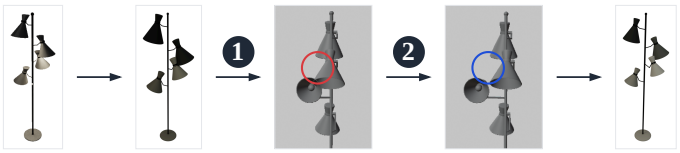
More generally, parts that are hidden in every input view must be inferred: the planner labels them as inferred, and the agent builds them from what such an object must have (Appendix B.2); additional views mitigate this (Section 5.2). Physical properties are approximated: the object's mass is

(a) Espresso machine: the run of Figure 2



1. • geometry check: two part pairs overlap by 5.5 mm.
   • difference record: the grate's front margin is wider than in the photo.
2. • difference record: darker than the photo; the grip should slope down; the slots are too sparse.
   • motion check: at its placeholder range the tray leaves the machine (right); its range is set to 8 cm.
   • geometry check: edit 6 sinks the lowered grip 5.1 mm into its holder; edit 7 trims it.

(b) Floor lamp (ArtVIP 425): a motion check sends the agent back to the geometry



1. • difference record: the shades are too large and too dark; a socket cuts through a shade.
2. • motion check: at its limit the tilted head hits the head above it (red); edit 3 moves the two apart in depth (blue).

Figure 16: Refinement over two runs: (a) the espresso machine of Figure 2; (b) a floor lamp, where a motion check sends the agent back to the geometry. Numbered notes give the feedback behind each change.

estimated from the images, part masses follow their bounding-box volumes (Appendix B.2), and inertia is approximated from bounding boxes (Appendix B.3).

H USE IN SIMULATION

H.1 SIMULATED MANIPULATION

Each generated asset is imported into an Isaac Sim scene as USD, with the mass estimated in its blueprint, and simulated with PhysX. The robot is a bimanual ALOHA platform; in the juicing task its grippers are replaced by Revo2 dexterous hands. The robot's trajectories are generated by automated scripts and executed open loop, and every contact and grasp is computed by the physics simulation alone. For each task, the object's position and orientation are randomized within ± 10 cm and $\pm 25°$ of its initial pose, and the task is run 50 times; a trial succeeds when the task in Table 8 is completed. All eight tasks succeed in 50/50 trials. Fig. 18 shows four stages of each task.

H.2 SIMULATION AND REAL-ROBOT EXECUTION

Figure 19 shows that manipulation planned on generated assets transfers to hardware. The microwave and the laptop are generated by DUALCRAFT from smartphone photographs of the physical objects and scaled to their measured dimensions, and the simulation is calibrated against the real setup so that the two match closely. In simulation, the robot pulls the microwave door open and closes the laptop lid, and the trajectories planned there are replayed on a real dual-arm robot operating the physical objects, open loop and without feedback correction, and succeed in 10/10



Dressing table (ArtVIP 031) *front view, articulated*
Wrong motion type: the deep side drawer was read as a hinged door.

Media cabinet (RealProduct 077) *front left view, articulated*
Missing parts: the blueprint drafted 2 of the 4 doors and read the other two as open cubbies.

Vanity cabinet (RealProduct 066) *front left view, articulated*
Unseen interior: the space behind the closed doors was left empty; the reference has an inner shelf.

Jar (RealProduct 034) *front view, articulated*
Wrong motion type: the lift-off lid was drafted as a twist cap.

Floor lamp (RealProduct 050) *front view, articulated*
Missed joint: the telescoping pole, the lamp's only joint, was missed; two turning knobs were drafted instead.

Figure 17: Failure cases of DUALCRAFT, with the input, the result, the reference, and the cause of each failure.

Table 8: Simulated manipulation tasks with generated assets. Each asset was generated from the reference images of the listed benchmark asset. Success: successful trials out of 50, with randomized object poses.

Task	Generated from	Interaction with the articulated asset	Success
Coffee making	REALAPPLIANCE 011	Mount the portafilter and twist it into place	50/50
Juicing	REALAPPLIANCE 086	Lift and replace the lid with dexterous hands; press the switch	50/50
Toasting	REALAPPLIANCE 097	Press the lever; take out the toast	50/50
Fridge retrieval	REALPRODUCT 015	Pull open the upper door; take out a bottle	50/50
Laundry unloading	ArtVIP 307	Open the door with both arms in turn; close it again	50/50
Pouring milk	REALPRODUCT 045	Lift the lid, pour, and replace the lid	50/50
Door opening	REALPRODUCT 064	Press the handle and swing the door open	50/50
Chair pushing	REALPRODUCT 089	Push the chair so that it rolls on its casters	50/50

real trials on each task. Its wrist and top cameras show the same progress in simulation and on the real robot at the same moment of a run.

Policy learning. We also fine-tune $\pi_{0.5}$ (Black et al., 2025) from its official pretrained weights on the two tasks. For each task, 100 successful executions of the planned trajectories are recorded in simulation as training data, and failed executions are discarded; in simulation, the fine-tuned policy succeeds in 10/10 trials on each task.

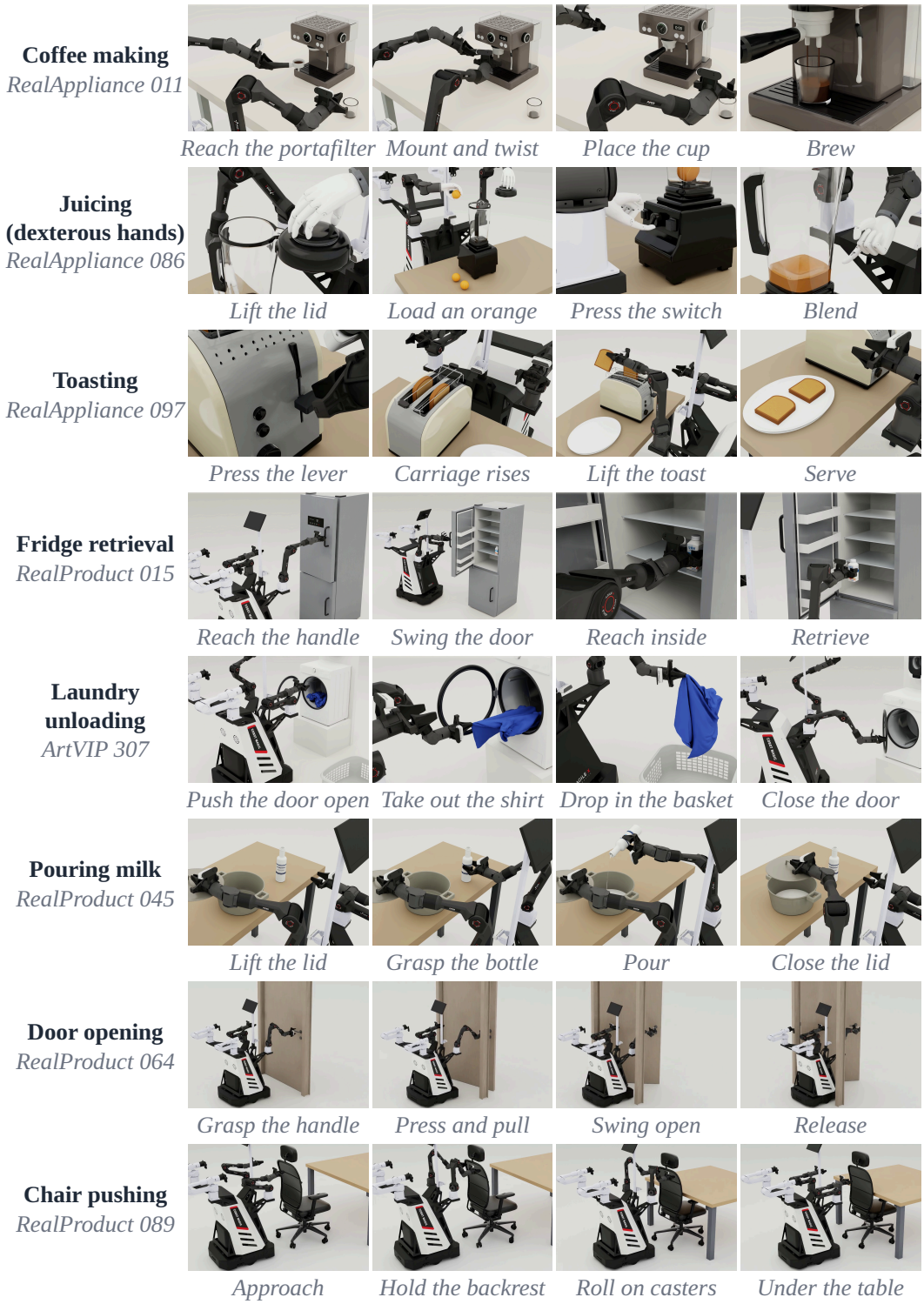


Figure 18: Generated assets in simulated manipulation. Each row is one task in Isaac Sim with PhysX, performed by a bimanual ALOHA robot (with Revo2 dexterous hands for juicing); the label under the task names the benchmark asset from whose reference images the asset was generated.

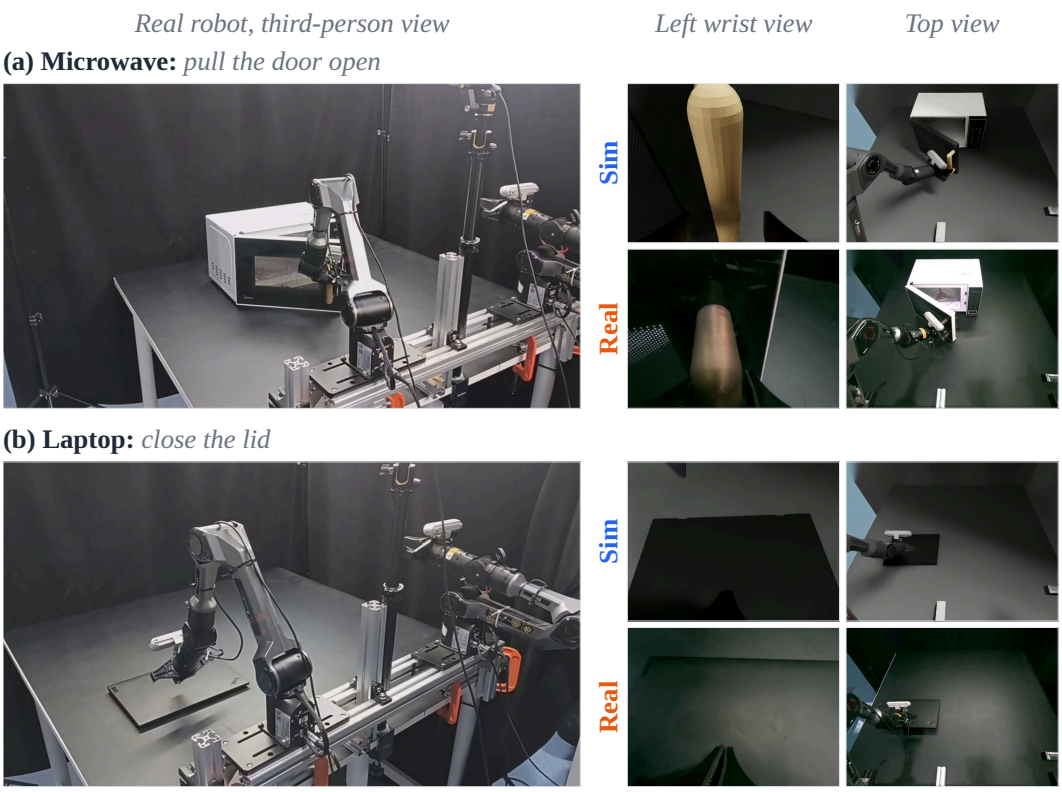


Figure 19: The same manipulation in simulation and on the real robot: (a) pulling a microwave door open, (b) closing a laptop lid. Left: the real robot seen from a third-person camera. Right: the robot's left-wrist and top cameras in simulation (Sim) and on the real robot (Real), at the same moment of one run. In simulation, both objects are generated by DUALCRAFT, and the real robot executes the trajectories planned there without adjustment.