2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Kinematify: Open-Vocabulary Synthesis of High-DoF Articulated Objects

Jiawei Wang[1,3] Dingyou Wang[1,2] Jiaming Hu[3] Qixuan Zhang[1,2 †] Jingyi Yu[2 *] Lan Xu[2 *]

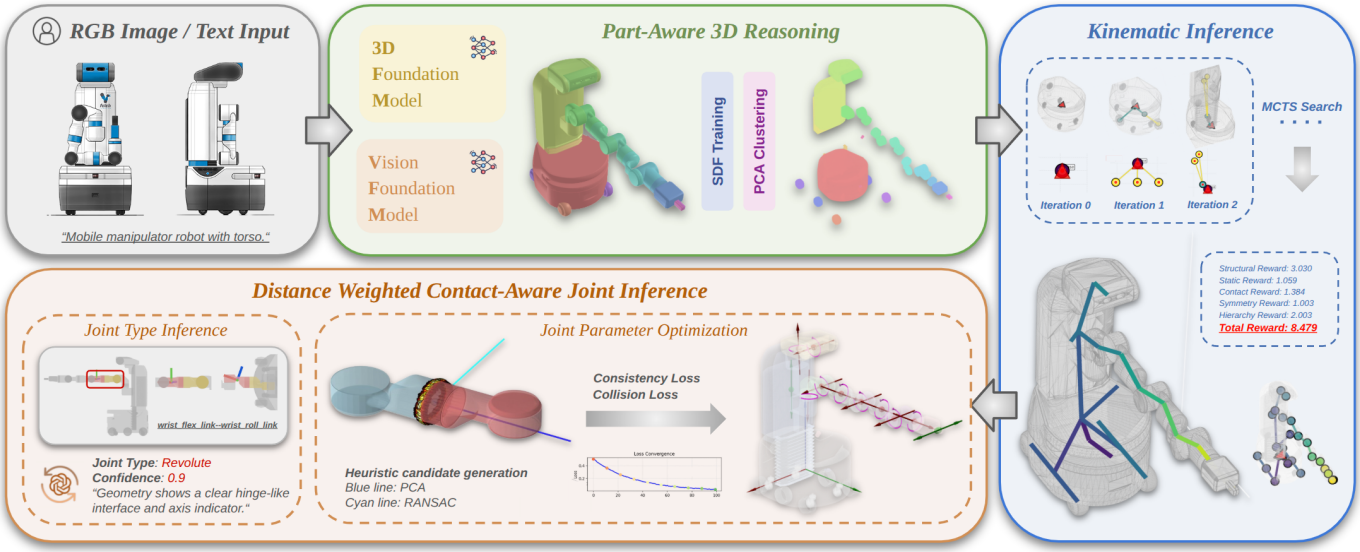

Fig. 1. **Overview of Kinematify.** A part-aware 3D foundation model first reconstructs a segmented digital twin. Then, the kinematic tree is recovered via Monte Carlo Tree Search (MCTS) driven by rewards for structure, stability, contact, symmetry, and hierarchy. Finally, joint types are predicted by a vision language model (VLM), and joint parameters are optimized on the parent link's signed distance field (SDF) to enforce contact consistency and avoid collisions.

***Abstract*—A deep understanding of kinematic structures is essential for robot motion and interaction with the environment. Such understanding is captured through articulated objects, which are essential for physical simulation, motion planning, and policy learning. However, creating these models, particularly for objects with high degrees of freedom (DoF), remains a significant challenge. Existing methods typically rely on motion sequences or strong assumptions from hand-curated datasets. In this paper, we introduce Kinematify, an automated framework that synthesizes articulated objects from arbitrary RGB images or textual descriptions. Our method addresses two core challenges: (i) inferring kinematic topologies for high-DoF objects and (ii) estimating joint parameters from static 3D geometry. To achieve this, we combine MCTS search for structural inference with geometry-driven optimization for joint reasoning, producing physically consistent and functionally valid models. We evaluate Kinematify on diverse inputs from both synthetic environments and real-world, demonstrating improvements in registration and kinematic topology accuracy over prior work. https://sites.google.com/deemos.com/kinematify**

[1]Deemos Corporation, Wilmington, DE 19801, USA. Emails: {joel.wang, dingyou, zhangqx}@deemos.com.
[2]ShanghaiTech University, Shanghai, China. Emails: {wangdy2024, zhangqx1, yujingyi, xulan1}@shanghaitech.edu.cn.
[3]Contextual Robotics Institute, UC San Diego, La Jolla, CA 92093, USA. Emails: {jiw179, jih189}@ucsd.edu.
[†]Project lead: Qixuan Zhang (zhangqx@deemos.com).
[*]Corresponding authors: Jingyi Yu (yujingyi@shanghaitech.edu.cn), Lan Xu (xulan1@shanghaitech.edu.cn).

I. Introduction

Enabling robots to effectively interact with objects, as well as to model their own articulated structures for self-perception and adaptation, requires an accurate understanding of kinematic topologies and joint parameters. Articulated robot descriptions capture this understanding by encoding geometry, kinematic dependencies, and dynamic constraints in standard formats like the Unified Robot Description Format (URDF) [1]. These descriptions are essential for robotic tasks such as manipulation, locomotion, and policy learning. However, customizing such descriptions for articulated objects remains a significant challenge, demanding substantial manual effort, especially for high degrees of freedom (DoF) systems like humanoids, quadrupeds, and arms. This difficulty arises from the labor intensive processes of part-aware 3D modeling, resolving intricate kinematic dependencies, and inferring precise joint parameters.

While recent advances in part-aware 3D generation [2]–[5] now enable the on-demand creation of high-quality segmented meshes from RGB images or textual descriptions, the bottleneck of kinematics inference remains. This challenge has driven robotics researchers to explore high-DoF articulated objects generation approaches.

Prior work has followed two main directions. Geometry-

first approaches infer parts and joints from dense 4D sequences or multi-scan data, which achieve high fidelity but rely on controlled capture settings [6], [7]. Program-synthesis pipelines, in contrast, predict executable descriptions directly from visual inputs [8]–[10]. While effective, these systems mainly target everyday objects such as laptops, bottles, and drawers, which typically contain only a few moving parts and relatively simple kinematic dependencies. In the context of self-modeling, related work such as AutoURDF [11] extends this idea to robots, recovering topology and joint types from point-cloud sequences. However, it presumes motion data and is largely limited to serial-chain structures, whereas high-DoF objects often exhibit multi-branched linkages.

To address these challenges, we introduce **Kinematify**, a framework that generates articulated 3D objects from RGB images or texts. An overview of Kinematify is shown in Fig. 1. Kinematify generates the segmented mesh with a part-aware 3D foundation model, such as BANG [2], then infers the kinematic tree using an MCTS [12], [13] objective that balances hierarchy and structural regularity. Subsequently, it estimates joint parameters via **DW-CAVL**, a novel **D**istance-**W**eighted **C**ontact-**A**ware **V**irtual **L**inkage optimization approach. This approach preserves near-contact regions while penalizing collisions under virtual motion. The resulting description is exported to URDF and is readily convertible to formats like MJCF [14] or USD [15]. Our contributions are:

- *An open-vocabulary articulated object generation framework.* Kinematify generates physics-aware articulated objects directly from arbitrary RGB images or textual descriptions, without requiring motion data, training, or pre-defined articulation priors.
- *A MCTS-based kinematic tree inference approach.* We propose a search objective that encodes structural priors like hierarchy and regularity to resolve ambiguous attachments for complex, high-DoF articulated objects with multiple branches.
- *A SDF-driven joint parameter estimation approach.* The DW-CAVL algorithm accurately infers revolute and prismatic joint parameters from static geometry by optimizing an SDF-based, contact-aware objective under virtual motions.

II. Related Work

A. 3D Articulated Object Modelling

A significant body of work focuses on reconstructing the kinematic structure of everyday objects from visual data. The most common paradigm leverages motion to reveal articulation [6], [7], [11], [16], [17]. By observing an object over time, these methods can directly infer which parts move together and identify the axes of rotation or translation. For example, MultiBodySync [7] registers multiple 3D scans of an object in different states, using spectral synchronization to jointly solve for part segmentation and motion. Similarly, ReArt [6] fits a rearticulable model to a 4D point cloud sequence by jointly optimizing segmentation, topology, and joint parameters. These methods achieve high fidelity but depend critically on the availability of multi-view or temporal data, which requires a controlled capture setup. Another trend frames articulation modelling as a program synthesis problem [8]–[10], [18]–[30]. URDFormer [10] trains a transformer to predict a URDF from a single image, relying on a large-scale synthetic dataset of image-URDF pairs. Real2Code [8] and Articulate-Anything [9] leverage large language models to generate code-based representations of articulated objects, with the latter using a reinforcement learning loop to refine the model through simulation feedback. While powerful in their open-vocabulary capabilities, these methods often struggle with the multi-branch kinematics, which are common in high-DoF objects.

B. Robot Self-Modeling

Distinct from general everyday object modeling, robot self-modeling is the online process by which a robot autonomously discovers its own body plan [11], [31]–[34]. This is typically achieved by correlating motor actions with sensory feedback. The foundational concept of task-agnostic self-modeling [31] involves a robot performing random motions and building a self-representation from the resulting data. This has been realized in various ways. Ledezma et al. [33] used IMU sensors on each link, applying machine learning to the sensor data to explicitly solve for the robot's topology and kinematic parameters. These methods are powerful but require an embodied agent with access to its own motor and sensory signals. AutoURDF [11] represents a purely visual approach to this problem. It operates on time-series point cloud frames of a robot in motion, but without access to the underlying motor commands. By tracking the 6-DoF transformations of point clusters, it segments moving parts, infers the kinematic tree using a minimum spanning tree, and estimates joint parameters. It demonstrated superior performance in topology inference for serial-chain objects.

III. Kinematify

We introduce Kinematify, a framework that generates kinematics-aware articulated objects directly from RGB images or textual descriptions in a zero-shot context.

A. Preliminaries

Assemblies and parts. An *assembly* $\mathcal{A}$ is a set of parts $P = \{P_i\}_{i=1}^{N}$. Part P_i has a triangulated surface mesh $M_i = (V_i, F_i)$ with vertices $V_i \subset \mathbb{R}^3$ and triangular faces $F_i \subset \{1, \dots, |V_i|\}^3$. Each part stores a world transform $T_i \in \mathrm{SE}(3)$, an intrinsic rotation $R_i \in \mathrm{SO}(3)$ used for alignment, a centroid $c_i \in \mathbb{R}^3$, a robust volume $v_i > 0$, and axis-aligned bounding-box extents $e_i \in \mathbb{R}^3_{>0}$.

Graphs and kinematic tree. We build an undirected connection graph $G = (V, E)$ with $V \leftrightarrow P$, where an edge indicates geometric contact. A directed kinematic tree $T = (V, E_T)$, rooted at the base link $b \in V$, orients G and annotates joints.

Joints. For a directed edge $(u \to v) \in E_T$, a joint stores a type $J_{uv} \in \{fixed, revolute, prismatic\}$, a parent-to-child

origin $o_{uv} \in \mathbb{R}^3$, and if movable, an axis $\mathbf{a}_{uv} \in \mathbb{S}^2$ and optionally a pivot $\mathbf{p}_{uv} \in \mathbb{R}^3$ when revolute.

B. Part-Aware 3D Representations

We generate part-level 3D meshes with a part-aware 3D foundation model [2] from the input RGB images or textual description, and discard meshes with too few vertices or a degenerate spatial spread. For each prospective parent part, we train a continuous SDF [35]–[37] $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ on (i) noisy surface points, (ii) near-surface offsets, and (iii) far samples in a bounding AABB.

Afterward, we build a connection graph G with the trained SDF, as shown in the middle left panel of Fig. 2. Given two candidate parts A and B with respective SDFs f_{θ_A} and f_{θ_B}, we evaluate mutual distances between their sampled surfaces under f_θ. Pairs of parts whose minimum bidirectional distance falls below a tolerance ϵ are declared in contact, and an undirected edge is added between them.

C. Kinematic Topology Inference

We orient the graph G into a directed kinematic tree T with root b. For any directed tree X, let $V(X)$ and $E(X)$ denote its node and oriented edge sets. Node positions $c_i \in \mathbb{R}^3$ are reference points of part i. For an oriented edge $(u \to v) \in E(X)$, its edge-origin vector is $o_{uv} := c_v - c_u$. For node-wise functions $f : V(\tilde{T}) \to \mathbb{R}$, the average is $\overline{f} := |V(\tilde{T})|^{-1} \sum_{i \in V(\tilde{T})} f(i)$. Depth $d(i)$ is the graph distance from the root b to node i. The out-degree in the directed tree is $\deg^+(i)$. All edge-wise sums $\sum_{(u \to v)}$ are over $(u \to v) \in E(\tilde{T})$ unless stated otherwise. A positive distance threshold $d_{\max} > 0$ is used when attaching disconnected components.

a) Base selection and BFS orientation: We choose the base link b as any node in G with the highest undirected degree. Starting from b, we run BFS on G as a warm start for the MCTS search. During BFS, when a new neighbour v is first reached from an already visited node u, we define u as the parent and v as the child, orient the edge as $u \to v$, and set its origin $o_{uv} \leftarrow c_v - c_u$. Any edge whose addition would create a cycle is not inserted into T and is recorded as broken. If G is disconnected, each remaining component is attached to the current tree by adding a virtual edge from its nearest neighbor, provided the Euclidean distance is at most $d_{\max}$.

1) State, actions, constraints: A search state is $S = (T_S, V_S, B_S)$, where T_S is the current partial directed tree, $V_S \subseteq V(G)$ the visited-node set, and $B_S \subseteq E(G)$ the set of broken undirected edges discovered so far. An action adds a feasible oriented edge $u \to v$ with $u \in V_S$ and $v \notin V_S$. To respect discovered symmetries, we form part clusters $\{C_k\}$ by the Chamfer distance between segmented meshes. During expansion, we forbid connecting two nodes that belong to the same multi-member cluster to avoid spurious intra-cluster links.

2) Transition: Applying an action $u \to v$ updates the edge origin $o_{uv} \leftarrow c_v - c_u$, provisionally treats the joint as fixed until later typing, inserts v into V_S, and appends to B_S any undirected edge $(v, w) \in E(G)$ with $w \in V_S \setminus \{u\}$ that would otherwise form a cycle.

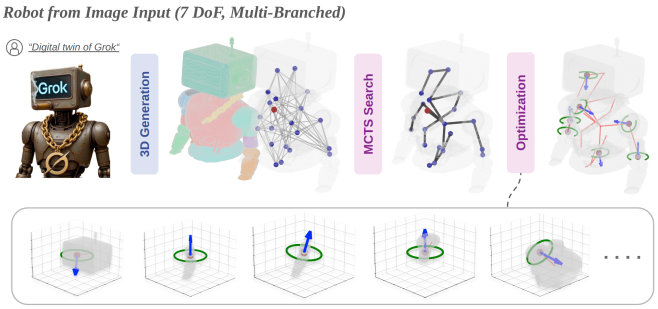

Fig. 2. Pipeline of Kinematify for recovering articulated robots from a single RGB image. **Step 1**: A 3D foundation model generates a segmented mesh of the robot. **Step 2**: A contact graph is constructed over mesh parts, capturing candidate relations between components. **Step 3**: Infer the kinematic tree using MCTS, resolving ambiguous connections by leveraging structural priors such as hierarchy and symmetry. **Step 4**: Refine joint parameters using the DW-CAVL optimization approach while preserving near-contact geometry. **Bottom row**: Examples of inferred revolute joints with optimized axes.

3) Reward: For a completed tree $\tilde{T}$, the terminal reward is a weighted sum of five terms:

a) R_{struct}: This term penalizes large depth variance and high degree deviation:

$$R_{\text{struct}} = \frac{1}{\overline{d^2} + \overline{(\deg^+ - k)^2}}, \tag{1}$$

where k is the preferred out-degree and $\lambda > 0$ is a hyperparameter.

b) R_{static}: This term favours centre-of-mass support to reduce gravitational torque about joint frames. Let $v_i > 0$ denote the estimated volume of part i and $m_i = \rho\, v_i$ its mass for a density parameter $\rho > 0$. With subtree mass $M(i)$ and subtree centre $\mathbf{c}_{\text{sub}}(i)$,

$$M(i) = m_i + \sum_{j \in \text{ch}(i)} M(j), \tag{2}$$

$$\mathbf{c}_{\text{sub}}(i) = \frac{m_i\, c_i + \sum_{j \in \text{ch}(i)} M(j)\, \mathbf{c}_{\text{sub}}(j)}{M(i)}, \tag{3}$$

where $\text{ch}(i)$ is the set of children of i in $\tilde{T}$. Let $\hat{\mathbf{z}}^- = [0, 0, -1]^\top$ denote downward unit gravity and $g > 0$ the gravitational constant. The total gravitational torque is

$$\tau = \sum_{i \neq b} \left\| \left(\mathbf{c}_{\text{sub}}(i) - c_i\right) \times \left(M(i)\, g\, \hat{\mathbf{z}}^-\right) \right\|_2, \tag{4}$$

$$R_{\text{static}} = \frac{1}{1 + \tau/\sigma_\tau}, \tag{5}$$

with $\sigma_\tau > 0$ a robust per-assembly normaliser based on MAD scale.

c) R_{contact}: We quantify contact strength from SDF-based bidirectional proximity. Let $s(u, v) \in [0, 1]$ denote the contact strength of a physical contact on edge $(u \to v)$. We reward higher average strength:

$$R_{\text{contact}} = \frac{1}{|E(\tilde{T})|} \sum_{(u \to v) \in E(\tilde{T})} s(u, v). \tag{6}$$

d) R_{sym}: Within each discovered symmetry cluster C_k ($|C_k| \geq 2$), we prefer equal depths and a shared parent, such as legs attached to the same torso, fingers to the same palm. Let $P_k = \{\text{parent}(i) : i \in C_k,\, i \neq b\}$. We define

$$S_k = \frac{1}{1 + \text{Var}\big(d(i) : i \in C_k\big)} \;+\; \Big[1 - \frac{|P_k| - 1}{|C_k| - 1 + \varepsilon}\Big], \qquad R_{\text{sym}} = \text{mean}_k\, S_k. \tag{7}$$

The second term equals 1 when all parts in C_k share the same parent ($|P_k| = 1$) and decreases linearly as parents diversify. $\varepsilon > 0$ avoids division by zero.

e) R_{hier}: We discourage children much larger than their parents by estimated volume. With a small $\varepsilon > 0$ to avoid divide-by-zero,

$$R_{\text{hier}} = \frac{1}{1 + \sum_{(u \to v)} \max\Big\{0,\; \frac{v_v}{v_u + \varepsilon} - 1\Big\}}. \tag{8}$$

4) Search: We use Monte Carlo Tree Search (MCTS) with UCT. Each state is $S = (T_S, V_S, B_S)$. From a state S, each child c corresponds to applying one feasible action $u \to v$. Let $Q(c)$ be the cumulative return backed up through child c, $N(c)$ its visit count, and $N(\text{parent})$ the visit count of its parent state. With exploration constant $C > 0$, selection chooses

$$\arg\max_c \; \frac{Q(c)}{N(c)} \;+\; C\sqrt{\frac{\ln N(\text{parent})}{N(c)}}. \tag{9}$$

Rollouts greedily complete the tree by repeatedly choosing any available action with the highest immediate score, and the terminal return $R(\tilde{T})$ is backed up along the simulation path. This objective helps resolve symmetric attachments and multi-branch ambiguities at scale. The middle right panel of Fig. 2 shows the kinematics structure after MCTS search.

Algorithm 1: KINEMATIFY

Input: Connection graph $G = (V, E)$, base b, SDFs $\{f_\theta\}$, samples $\{P_v\}$
Output: Kinematic tree $T = (V, E_T)$, joints $\{J_{uv}\}$

$S_0 \leftarrow (\emptyset, \{b\}, \emptyset)$; init stats $Q, N \leftarrow 0$;
for $t = 1$ **to** $N_{\max}$ **do**
  $S \leftarrow S_0$; path $\mathcal{P} \leftarrow \emptyset$;
  while $|V_S| < |V|$ **do**
    if *untried edge exists* **then**
      $a \leftarrow$ pick untried; $S \leftarrow \text{Transition}(S, a)$;
      break
    else
      $S \leftarrow \arg\max_c \frac{Q(c)}{N(c)} + C\sqrt{\ln N(S)/N(c)}$
    $\mathcal{P}$.append(S)
  $\tilde{T} \leftarrow$ greedy rollout from S ;
  $R \leftarrow \text{Reward}(\tilde{T})$; backprop R along $\mathcal{P}$;
$T \leftarrow$ best cached tree ;
for *edge* $(u \to v) \in E_T$ **do**
  candidates $\leftarrow$ generate from contact stats ;
  for *candidate* (p, u) **do**
    optimize $\mathcal{J}_{rev}$ or $\mathcal{J}_{pri}$
  select best class: revolute if $s_{rev} > \zeta s_{pri}$, else prismatic;
  store J_{uv}
return $(T, \{J_{uv}\})$

D. Joint Reasoning

We render orthographic viewsets for the whole assembly and for joint close-ups. For each $(u \to v)$, we query VLM on the joint viewset and adopt a decision with abstention. If the VLM successfully identifies the joint type, we group joints by child clusters C_k and select a representative by majority and correct outliers.

Let $P_A = \{\mathbf{a}_i\}$ and $P_B = \{\mathbf{b}_j\}$ be surface samples of two parts A and B in a common frame. We first extract a contact region $\mathcal{C}$ as the union of points on either part whose nearest neighbour on the other part lies within a small threshold. To downweight spurious pairs, each $\mathbf{x} \in \mathcal{C}$ is assigned a weight that decays with its nearest-neighbour distance, using these weights we compute a weighted contact centroid $\boldsymbol{\mu}_c$ and a weighted covariance $\boldsymbol{\Sigma}$. The principal direction with the smallest variance provides a hinge-axis estimate $\widehat{\mathbf{u}}_{\text{PCA}}$. In parallel, we obtain a contact normal $\widehat{\mathbf{n}}$ by averaging nearest-point difference vectors across the two surfaces.

We then form a diverse set of revolute candidates $(\mathbf{p}, \mathbf{u})$ as follows. Pivots are initialised at the contact centroid ($\mathbf{p} = \boldsymbol{\mu}_c$). Axes are drawn from a compact pool that includes $\widehat{\mathbf{u}}_{\text{PCA}}$, the contact normal $\widehat{\mathbf{n}}$, an orthogonal completion $\widehat{\mathbf{u}}_{\perp}$, the principal axes of $\boldsymbol{\Sigma}$, and a few random unit directions. Along each candidate axis $\mathbf{u}$ we place a handful of pivot samples by sliding $\mathbf{p}$ slightly along $\mathbf{u}$ around $\boldsymbol{\mu}_c$.

1) Differentiable rigid motions: For a revolute motion $(\mathbf{p}, \mathbf{u}, \theta)$ with $\|\mathbf{u}\|_2 = 1$,

$$\mathbf{y} = \mathbf{p} + \mathbf{R}(\mathbf{u}, \theta)\,(\mathbf{x} - \mathbf{p}), \tag{10}$$

$$\mathbf{R}(\mathbf{u}, \theta) = \cos\theta\,\mathbf{I} + \sin\theta\,[\mathbf{u}]_\times + (1 - \cos\theta)\,\mathbf{u}\mathbf{u}^\top, \tag{11}$$

where $[\mathbf{u}]_\times$ is the 3×3 cross-product matrix. For a prismatic motion with displacement $t \in \mathbb{R}$, $\mathbf{y} = \mathbf{x} + t\,\mathbf{u}$. We parametrise $\mathbf{u}$ by an unconstrained $\mathbf{a} \in \mathbb{R}^3$ via $\mathbf{u} = \mathbf{a}/\|\mathbf{a}\|_2$, whose Jacobian is

$$\frac{\partial \mathbf{u}}{\partial \mathbf{a}} = \frac{1}{\|\mathbf{a}\|_2}\left(\mathbf{I} - \frac{\mathbf{a}\mathbf{a}^\top}{\|\mathbf{a}\|_2^2}\right). \tag{12}$$

2) DW-CAVL objective: Let child samples be $\{\mathbf{b}_i\}$. A virtual motion parameter $\delta \in \Theta$ denotes either a revolute angle θ or a prismatic displacement t. Let Φ_δ be the corresponding rigid transform of the child, and define

$$s_0(\mathbf{x}) = f_\theta(\mathbf{x}), \qquad s_\delta(\mathbf{x}) = f_\theta\big(\Phi_\delta(\mathbf{x})\big).$$

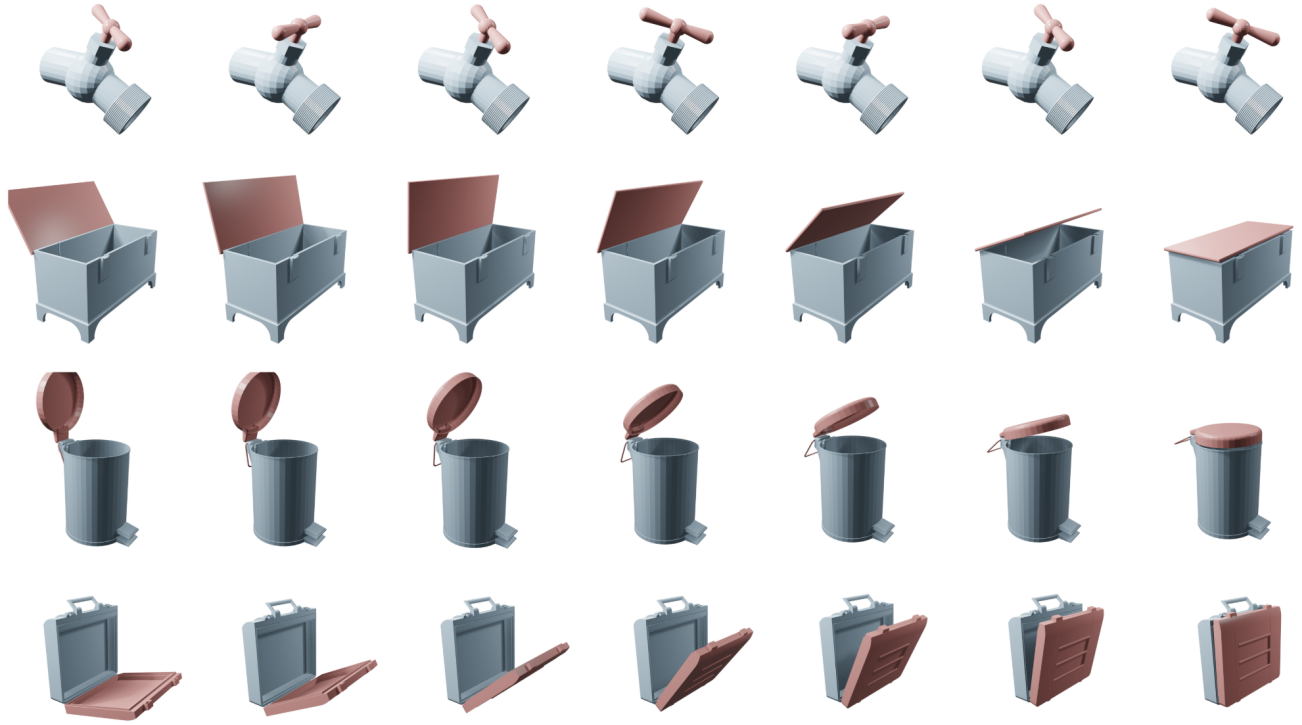

Fig. 3. **Examples of articulated objects generated by Kinematify.** Each row shows different objects across a sequence of joint configurations.

With hyperparameters volumetric margin $m_{\text{vol}} > 0$, logistic sharpness $k > 0$, contact band width $\sigma_c > 0$, and $\varepsilon_{\text{small}} > 0$,

$$w_{\text{vol}}(s_0) = \sigma\big(-k(s_0 - m_{\text{vol}})\big), \quad \sigma(z) = \frac{1}{1+e^{-z}}, \tag{13}$$

$$w_{\text{dist}}(s_0) = \exp\left(-\frac{s_0^2}{2\sigma_c^2}\right), \qquad w_i = w_{\text{vol}}(s_0(\mathbf{b}_i))\, w_{\text{dist}}(s_0(\mathbf{b}_i)) \tag{14}$$

The consistency term penalizes separation near contact after motion,

$$\mathcal{L}_{\text{cons}}(\delta) = \frac{\sum_i w_i \big[\max\{0,\ s_\delta(\mathbf{b}_i) - m_{\text{vol}}\}\big]^2}{\sum_i w_i + \varepsilon_{\text{small}}}. \tag{15}$$

Using inverse volumetric weights $\tilde{w}_i = \sigma\big(+k(s_0(\mathbf{b}_i) - m_{\text{vol}})\big)$, the collision term penalises penetration,

$$\mathcal{L}_{\text{coll}}(\delta) = \frac{\sum_i \tilde{w}_i \big[\max\{0,\ -s_\delta(\mathbf{b}_i) - m_{\text{vol}}\}\big]^2}{\sum_i \tilde{w}_i + \varepsilon_{\text{small}}}. \tag{16}$$

For revolute joints we regularise the pivot toward $\boldsymbol{\mu}_c$:

$$\mathcal{L}_{\text{reg}} = \lambda_p \|\mathbf{p} - \boldsymbol{\mu}_c\|_2^2, \qquad \lambda_p \geq 0. \tag{17}$$

Aggregating over $\delta \in \Theta$ yields

$$\mathcal{J}(\mathbf{p}, \mathbf{u}) = \frac{1}{|\Theta|} \sum_{\delta \in \Theta} \big(\lambda_c \mathcal{L}_{\text{cons}}(\delta) + \lambda_{\text{coll}} \mathcal{L}_{\text{coll}}(\delta)\big) + \mathcal{L}_{\text{reg}}, \tag{18}$$

with nonnegative weights $\lambda_c, \lambda_{\text{coll}}$.

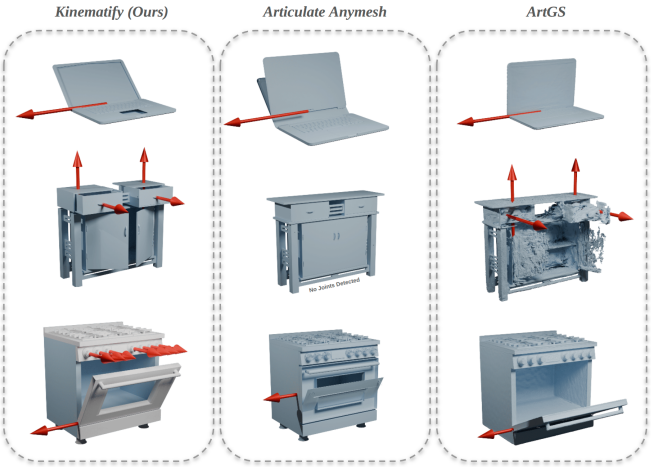

Fig. 4. **Qualitative comparison of articulation recovery on everyday objects across three methods:** Kinematify (ours), Articulate Anymesh, and ArtGS. The red line indicates the joint direction.

3) Candidate selection: We rank many candidates $(\mathbf{p}, \mathbf{u})$ on subsampled points using $s(\mathbf{p}, \mathbf{u})$, refine the top-K on full points by minimising $\mathcal{J}$, and output scores $1/(1+\mathcal{J})$.

In summary, we infer joint parameters from static geometry and select the candidate with the highest score. The bottom panel in Fig. 2 shows the optimized results for each joint for the input.

TABLE I

UNIFIED BASELINE COMPARISON ACROSS ROBOTS SORTED BY DEGREES OF FREEDOM (DOF). WE REPORT AXIS ANGLE ERROR (°), AXIS POSITION ERROR (M), AND TREE EDIT DISTANCE, WHERE LOWER VALUES ARE BETTER.

Metric	Method	Everyday Object 1-8 DoF	UR10e 6 DoF	Franka Panda 7 DoF	Unitree Go2 12 DoF	Fetch 13 DoF	Allegro 16 DoF	Unitree H1 19 DoF	Mean
Axis Angle Error↓	Articulate Anymesh	35.80	39.67	42.10	53.23	75.60	78.77	79.35	57.79
	ArtGS	13.80	25.52	21.30	22.32	53.81	65.59	41.29	34.80
	Ours	**2.92**	**5.34**	**10.42**	**9.97**	**23.10**	**31.39**	**29.31**	**16.06**
Axis Pos Error↓	Articulate Anymesh	**0.19**	**0.25**	0.31	0.41	0.89	0.32	0.74	0.44
	ArtGS	0.75	0.97	0.68	1.13	1.93	0.67	1.32	1.06
	Ours	0.23	0.27	**0.15**	**0.30**	**0.71**	**0.21**	**0.68**	**0.36**
TED↓	AutoURDF	0.27	**0.87**	1.28	2.21	3.21	4.83	8.13	2.97
	Ours	**0.13**	1.03	**0.89**	**1.97**	**1.78**	**1.22**	**2.23**	**1.32**

IV. EXPERIMENTS

We evaluate Kinematify in two settings: (i) everyday articulated objects and (ii) robotic platforms. We follow prior protocols of Articulate Anymesh [19], which use ground-truth segmented meshes from PartNet-Mobility [38], [39] to isolate the impact of 3D segmentation. This ensures fair, direct comparisons to baselines that do not accept raw images or texts. We erase the provided kinematic graphs and joint parameters, and reconstruct the mesh. Under this configuration, we compare against the baselines Articulate AnyMesh [19] and ArtGS [16].

For the robotics platforms, we evaluate Kinematify on six commonly used robot models spanning a range of DoF. Because ArtGS and Articulate AnyMesh do not expose explicit kinematic tree structure, we additionally compare against AutoURDF [11] for kinematics reconstruction performance.

We also conduct experiments on the end-to-end pipelines, starting from RGB images, and evaluate the output quality with ground-truth to quantify the discrepancies.

A. Metrics

We report three metrics evaluating both joint parameter and kinematics tree quality.

Axis Angle Error: The angular deviation between the predicted and ground-truth joint-axis directions, and opposite directions are treated as equivalent.

Axis Position Error: The Euclidean distance between predicted and ground-truth pivot positions in the dataset coordinate frame.

Tree Edit Distance: The Tree Edit Distance [40] between the predicted and ground-truth kinematic trees, for instance, the minimal number of node insertions, deletions, or relabelings needed to match the trees.

B. Quantitative results

Everyday Objects. Table I reports a comparison of Kinematify against Articulate Anymesh and ArtGS on the PartNet-Mobility benchmark. Our method achieves the lowest axis angle error among all approaches, indicating superior accuracy in joint orientation estimation. In terms of axis position error, Kinematify also performs competitively, with values close to the best baseline. A qualitative visualization of these comparisons is provided in Fig. 4. Together, these results demonstrate that Kinematify produces precise joint axes and stable pivot placements for everyday objects.

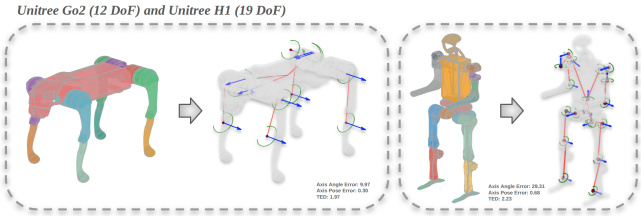

Fig. 5. **Demonstration of Kinematify on two high-DoF robots:** Unitree Go2 (12 DoF, left) and Unitree H1 (19 DoF, right). For each case, the pipeline starts from a segmented mesh, followed by kinematic tree inference and joint parameter optimization.

Robots. We further evaluate performance on six robotic systems by measuring both registration quality and body topology reconstruction. As shown in Table I, Kinematify reduces the Tree Edit Distance by a substantial margin on average, reflecting more faithful recovery of kinematic structures. Representative results on Unitree H1 and Go2 are visualized in Fig. 5. These findings highlight the effectiveness of our MCTS-based objective in reasoning about high-DoF, multi-branched kinematic structures, surpassing prior methods in structural consistency.

C. End-to-end evaluation

We further evaluate Kinematify in a full end-to-end setting that starts from single RGB images. A part-aware 3D foundation model [2] first produces a segmented mesh. Then, we apply the kinematic reasoning stack unchanged. Because existing baselines do not natively support image to articulated objects at comparable scope, we report absolute performance rather than head to head comparisons.

As shown in Table II, compared to the geometry-only track in Table I, end-to-end errors increase modestly on EO and more noticeably on Fetch and Panda, consistent with their tighter kinematic tolerances.

TABLE II
END-TO-END RESULTS. NUMBERS ARE ABSOLUTE.

Metric	Everyday Objects	Fetch	Panda
Axis Angle Error↓	3.78	32.84	14.08
Axis Position Error (m)↓	0.28	0.95	0.22
TED↓	0.67	2.95	1.17

TABLE III
ABLATION STUDY ON THE PROPOSED METHOD, WHERE EO DENOTES EVERYDAY OBJECT.

Metric	Variant	EO	Fetch	Panda
Axis Angle Error↓	w/o MCTS	4.32	28.30	10.92
	w/o DW-CAVL	13.94	42.30	29.39
	Ours	**2.92**	**23.10**	**10.42**
Axis Pos Error↓	w/o MCTS	0.59	0.97	0.30
	w/o DW-CAVL	1.34	1.82	0.97
	Ours	**0.23**	**0.71**	**0.15**
TED↓	w/o MCTS	0.39	3.32	2.97
	w/o DW-CAVL	0.14	1.93	0.98
	Ours	**0.13**	**1.78**	**0.89**

D. Ablation study

We quantify the contribution of each core component by comparing the full method against two ablations: (i) removing the DW-CAVL anchor term so that optimization considers only collision avoidance; and (ii) replacing the MCTS-based kinematic inference with a BFS strategy.

As shown in Table III, substituting MCTS with BFS consistently yields larger TED across robots. BFS greedily attaches along local contacts and lacks long range regularization, leading to incorrect parent choices in symmetric substructures and unbalanced trees. In contrast, removing the DW-CAVL anchor does not drastically change the tree but significantly degrades joint parameters. Without an attraction to the contact centroid and near-surface band, the optimizer favors axes that quickly reduce interpenetration yet drift from true pivots. Overall, the full model achieves the best balance.

E. Real-world robot manipulation

We export the recovered kinematics to URDF and deploy the models in simulation and on a real robot, shown in Fig. 6. From the segmented mesh, Kinematify generates a Fetch URDF and a simple cabinet URDF. For planning, we load both URDFs into a single MoveIt [41] planning scene and derive an SRDF group for the arm.

We use a constraint motion planner [42], [43] as the backend. The task is executed in two stages: (1) reach-to-grasp and (2) constrained pull. In addition to this drawer-opening scenario, we also demonstrate an online planning task of pouring water from a cup into a container, using the same URDF pipeline and motion-planning setup. The same URDFs are used in Isaac Sim and on hardware. In both cases, the arm follows the planned trajectories without collision, showing that the recovered kinematics are physically consistent and directly usable for online planning in ROS and MoveIt [41].

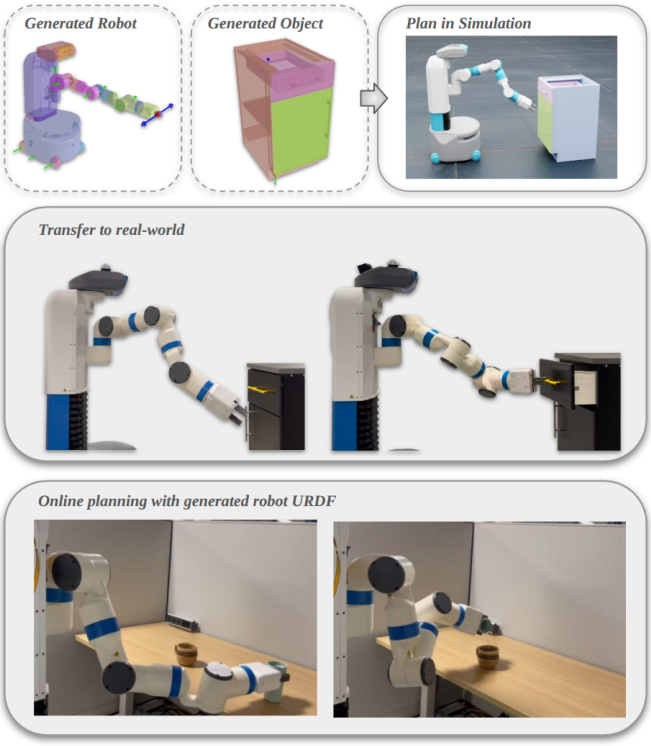

Fig. 6. **Real world experiment.** Kinematify generates URDFs for both the Fetch robot and the drawer, enabling a demonstration of the robot opening the drawer in Isaac Sim and transferring to real, with the same models usable for online planning with MoveIt [41].

V. CONCLUSION

We presented **Kinematify**, an automated pipeline that synthesizes articulated object and robot descriptions from RGB images or text. Across everyday objects and multiple robot platforms, Kinematify improves joint estimation accuracy and kinematic tree fidelity over prior work.

Kinematify assumes accurate part segmentation and a reliable contact graph. In practice, spurious seams, missed contacts, and decorative geometry can mislead topology inference. Future work includes jointly refining segmentation and contact reliability during structure inference, and exploring a learning-based model trained on data generated by Kinematify that directly predicts kinematic topology and joint parameters from input, with physics-based constraints to ensure validity.

Overall, we view Kinematify as a step toward open-vocabulary synthesis of high-DoF articulated structures.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant W2431046, National Key R&D Program of China 2025YFA1309603, Central Guided Local Science and Technology Foundation of China YDZX20253100001001,and by MoE Key Lab of Intelligent Perceptionand Human-Machine Collaboration (ShanghaiTech University), the Shanghai Frontiers Science Center of Human-centered Artificial Intelligence.

REFERENCES

[1] ROS Wiki, "Xml robot description format (urdf)." ROS Wiki. Accessed: 2026-02-26.

[2] L. Zhang, Q. Zhang, H. Jiang, Y. Bai, W. Yang, L. Xu, and J. Yu, "Bang: Dividing 3d assets via generative exploded dynamics," *ACM Transactions on Graphics (TOG)*, vol. 44, no. 4, pp. 1–21, 2025.

[3] Y. Yang, Y.-C. Guo, Y. Huang, Z.-X. Zou, Z. Yu, Y. Li, Y.-P. Cao, and X. Liu, "Holopart: Generative 3d part amodal segmentation," *arXiv preprint arXiv:2504.07943*, 2025.

[4] J. Tang, R. Lu, Z. Li, Z. Hao, X. Li, F. Wei, S. Song, G. Zeng, M.-Y. Liu, and T.-Y. Lin, "Efficient part-level 3d object generation via dual volume packing," *arXiv preprint arXiv:2506.09980*, 2025.

[5] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler, "Get3d: A generative model of high quality 3d textured shapes learned from images," *Advances in neural information processing systems*, vol. 35, pp. 31841–31854, 2022.

[6] S. Liu, S. Gupta, and S. Wang, "Building rearticulable models for arbitrary 3d objects from 4d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21138–21147, 2023.

[7] J. Huang, H. Wang, T. Birdal, M. Sung, F. Arrigoni, S.-M. Hu, and L. J. Guibas, "Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7108–7118, 2021.

[8] Z. Mandi, Y. Weng, D. Bauer, and S. Song, "Real2code: Reconstruct articulated objects via code generation," in *The Thirteenth International Conference on Learning Representations*.

[9] L. Le, J. Xie, W. Liang, H.-J. Wang, Y. Yang, Y. J. Ma, K. Vedder, A. Krishna, D. Jayaraman, and E. Eaton, "Articulate-anything: Automatic modeling of articulated objects via a vision-language foundation model," *arXiv preprint arXiv:2410.13882*, 2024.

[10] Z. Chen, A. Walsman, M. Memmel, K. Mo, A. Fang, K. Vemuri, A. Wu, D. Fox, and A. Gupta, "Urdformer: A pipeline for constructing articulated simulation environments from real-world images," *arXiv preprint arXiv:2405.11656*, 2024.

[11] J. Lin, L. Zhang, K. Lee, J. Ning, J. Goldfeder, and H. Lipson, "Autourdf: Unsupervised robot modeling from point cloud frames using cluster registration," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 27628–27637, 2025.

[12] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*, pp. 72–83, Springer, 2006.

[13] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*, pp. 282–293, Springer, 2006.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.

[15] Pixar Animation Studios, "Introduction to USD," 2021. Accessed: 2025-09-28.

[16] Y. Liu, B. Jia, R. Lu, J. Ni, S.-C. Zhu, and S. Huang, "Building interactable replicas of complex articulated objects via gaussian splatting.," in *ICLR*, 2025.

[17] L. Shen, S. Zhang, H. Li, P. Yang, Z. Huang, Z. Zhang, and H. Zhao, "Gaussianart: Unified modeling of geometry and motion for articulated objects," *arXiv preprint arXiv:2508.14891*, 2025.

[18] R. Luo, H. Geng, C. Deng, P. Li, Z. Wang, B. Jia, L. Guibas, and S. Huang, "Physpart: Physically plausible part completion for interactable objects," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12386–12393, IEEE, 2025.

[19] X. Qiu, J. Yang, Y. Wang, Z. Chen, Y. Wang, T.-H. Wang, Z. Xian, and C. Gan, "Articulate anymesh: Open-vocabulary 3d articulated objects modeling," *arXiv preprint arXiv:2502.02590*, 2025.

[20] J. Zhang, M. Wu, and H. Dong, "Generative category-level object pose estimation via diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 54627–54644, 2023.

[21] M. Chen, R. Shapovalov, I. Laina, T. Monnier, J. Wang, D. Novotny, and A. Vedaldi, "Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5881–5892, 2025.

[22] C.-H. Yao, A. Raj, W.-C. Hung, M. Rubinstein, Y. Li, M.-H. Yang, and V. Jampani, "Artic3d: Learning robust articulated 3d shapes from noisy web image collections," *Advances in Neural Information Processing Systems*, vol. 36, pp. 48173–48184, 2023.

[23] J. Liu, D. Iliash, A. X. Chang, M. Savva, and A. Mahdavi-Amiri, "Singapo: Single image controlled generation of articulated parts in objects," *arXiv preprint arXiv:2410.16499*, 2024.

[24] H. Wang, X. Yuan, F. Zhang, R. Jian, Y. Zhu, X. Qiao, and Y. Huang, "Artgen: Conditional generative modeling of articulated objects in arbitrary part-level states," *arXiv preprint arXiv:2512.12395*, 2025.

[25] K. Kotar and R. Mottaghi, "Interactron: Embodied adaptive object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14860–14869, 2022.

[26] D. Gao, Y. Siddiqui, L. Li, and A. Dai, "Meshart: Generating articulated meshes with structure-guided transformers," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 618–627, 2025.

[27] L. Liu, W. Xu, H. Fu, S. Qian, Q. Yu, Y. Han, and C. Lu, "Akb-48: A real-world articulated object knowledge base," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14809–14818, 2022.

[28] P. Wang, Y. He, X. Lv, Y. Zhou, L. Xu, J. Yu, and J. Gu, "Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding," *arXiv preprint arXiv:2510.20155*, 2025.

[29] A. Raj, J. Tanke, J. Hays, M. Vo, C. Stoll, and C. Lassner, "Anr: Articulated neural rendering for virtual avatars," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3722–3731, 2021.

[30] V. Ramanathan, A. Kalia, V. Petrovic, Y. Wen, B. Zheng, B. Guo, R. Wang, A. Marquez, R. Kovvuri, A. Kadian, *et al.*, "Paco: Parts and attributes of common objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7141–7151, 2023.

[31] R. Kwiatkowski and H. Lipson, "Task-agnostic self-modeling machines," *Science Robotics*, vol. 4, no. 26, p. eaau9354, 2019.

[32] B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson, "Fully body visual self-modeling of robot morphologies," *Science Robotics*, vol. 7, no. 68, p. eabn1944, 2022.

[33] F. Díaz Ledezma and S. Haddadin, "Machine learning–driven self-discovery of the robot body morphology," *Science Robotics*, vol. 8, no. 85, p. eadh0972, 2023.

[34] Z. Jiang, C.-C. Hsu, and Y. Zhu, "Ditto: Building digital twins of articulated objects from interaction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5616–5626, 2022.

[35] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

[36] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.

[37] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.

[38] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11097–11107, 2020.

[39] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 909–918, 2019.

[40] M. Pawlik and N. Augsten, "Efficient computation of the tree edit distance," *ACM Transactions on Database Systems (TODS)*, vol. 40, no. 1, pp. 1–40, 2015.

[41] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE robotics & automation magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[42] J. Hu, S. R. Iyer, J. Wang, and H. I. Christensen, "Motion planning in foliated manifolds using repetition roadmap," in *Robotics: Science and Systems*, 2024.

[43] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.