

# AutoURDF: Unsupervised Robot Modeling from Point Cloud Frames Using Cluster Registration

Jiong Lin Lechen Zhang Kwansoo Lee Jialong Ning Judah Goldfeder Hod Lipson  
 Columbia University, Creative Machines Lab

<https://jl6017.github.io/AutoURDF/>

## Abstract

*Robot description models are essential for simulation and control, yet their creation often requires significant manual effort. To streamline this modeling process, we introduce AutoURDF, an unsupervised approach for constructing description files of unseen robots from point cloud frames. Our method leverages a cluster-based point cloud registration model that tracks the 6-DoF transformations of point clusters. Through analyzing cluster movements, we hierarchically address the following challenges: (1) moving part segmentation, (2) body topology inference, and (3) joint parameter estimation. The complete pipeline produces robot description files that are fully compatible with existing simulators. We validate our method across a variety of robots, using both synthetic and real-world scan data. Results indicate that our approach outperforms previous methods in registration and body topology estimation accuracy, offering a scalable solution for automated robot modeling.*

## 1. Introduction

Accurate and structured representations of robots are essential for applications such as real-time control, motion planning, and physics-based simulation. Robot description files, such as the Unified Robot Description Format (URDF) [45], are among the most widely used representations and explicitly capture robot geometry, kinematics, and dynamic properties. Over time, new formats such as MJCF for MuJoCo [49] and USD for Isaac Gym [29] have been introduced to allow scene descriptions and parallel simulations. Despite these advancements, customizing basic robot models still requires manual effort, often involving CAD model conversions [4] or tedious XML file modifications. This challenge has driven robotics and computer vision researchers to explore data-driven methods to automate the robot modeling process.

*Robot self-modeling* [20] has emerged as a vital approach in enabling robots to autonomously discover their

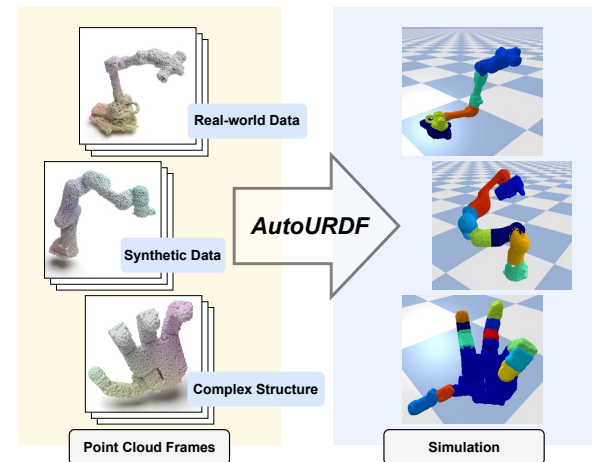


Figure 1. We present AutoURDF, a novel framework that derives robot description files from time-series point cloud frames. We validate our method across a diverse range of robots, including both synthetic and real-world scan data.

own body kinematics. Previous works in this field utilize sensorimotor data, integrating depth images [6], RGB images [13, 24] or IMU sensors [21] with motor signals to achieve accurate self-modeling. Another related field, *Articulated Object Modeling* [22], aims to reconstruct the kinematic structure of articulated rigid bodies from visual data. Prior research has primarily concentrated on everyday objects [14, 16, 30, 52], such as scissors, laptops, and drawers, which typically have only a few moving parts and relatively simple kinematic structures. Robots, however, consist of serially connected joints and, in some cases, multi-branched links. Additionally, large-scale datasets capturing diverse robot morphologies are still lacking. These factors make it challenging to effectively apply supervised learning methods designed for articulated objects to more complex robotic structures.

Here, we present AutoURDF, an unsupervised approach that constructs robot description files, specifically URDFs, from time-series point cloud frames. Rather than relying on

sensorimotor data, our method derives the robot structure purely from visual data. Furthermore, it does not require ground-truth annotations or manual intervention, making it scalable to a wide range of robots. The key objective of this work is illustrated in Figure 1.

Our method follows a hierarchical process, beginning with clustering the point cloud from the first frame based on point positions. Next, we compute a series of 6-DoF transformations across all time steps by registering the initial point clusters to each frame. From the motion patterns of these clusters, we segment them into distinct moving parts. Using the minimum spanning tree (MST) [19] and segmented parts, we identify link connections and infer the body topology. Transformation matrices are then calculated for each child part relative to its parent, allowing us to derive the joint parameters. Finally, we generate the URDF file and validate the reconstructed robots in the PyBullet [9] simulator. Our contributions are as follows:

- We propose a novel framework, AutoURDF, that builds robot description models purely from point cloud data.
- We design a neural network for cluster-based point cloud registration in complex articulated objects.
- We implement a description file generation pipeline that outputs URDF files, including XML formatting and mesh file creation.

## 2. Related Work

**Robot self-modeling.** Task-agnostic self-modeling [20] enables robots to autonomously construct and update their own kinematic models from interaction data, without requiring specific task information. Recent advances in neural implicit representations have shown promising results in reconstructing dynamic 3D shapes [27, 41]. By applying spatial implicit functions [17, 31, 37] to robot modeling, Chen et al. [6] demonstrated effective shape prediction by modeling robot morphology as a spatial query model conditioned on motor commands. Liu et al. [24] further extended this approach with a fully differentiable robot body rendering model. However, both methods require robot control information for model supervision, limiting their autonomy. Additionally, implicit function-based robot models are powerful for visual supervision and shape prediction, but they are not directly compatible with physics-based simulators, and accurately and efficiently calculating physical interactions with them remains challenging. On the other hand, Ledezma et al. [21] take a data-driven approach to robot modeling, explicitly representing the robot’s morphology through topology and kinematic parameters. However, their method requires IMU sensors on each moving part, which greatly increases setup complexity as the robot’s degrees of freedom (DoF) increase. Our method is motivated by similar goals to task-agnostic self-modeling approaches, aiming to

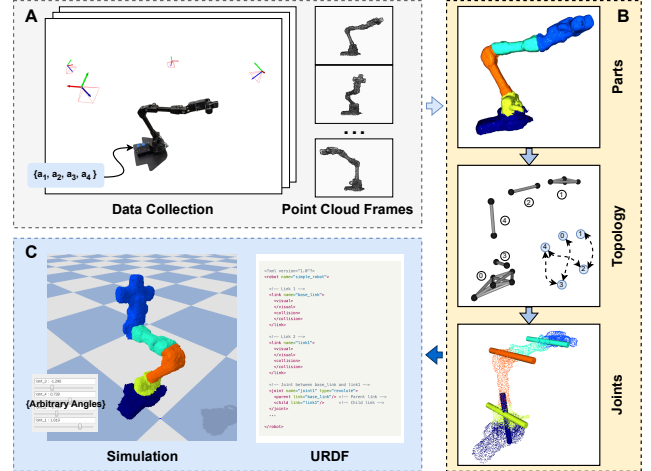


Figure 2. **AutoURDF overview.** Our method provides a complete pipeline for the automated construction of robot description files. **(A). Data Collection:** By commanding robots with randomly sampled motor angle sequences, we capture the corresponding time-series point cloud frames. **(B). Three Substeps:** We tackle the problem in three substeps: 1) part segmentation, 2) body topology inference, and 3) joint parameter estimation. **(C). Description File Generation:** The final output is a URDF file that defines the robot’s links, joints, and collision properties. We successfully build and simulate the description model for a WX200 robot arm from real-world scan data.

enable autonomous robot modeling without specific task constraints. Unlike these approaches, our method relies solely on raw point cloud frames, without needing forward kinematics, degrees of freedom (DoF), or motor control data. Additionally, our approach maintains an explicit representation of the robot’s kinematics and geometry, ensuring direct compatibility with robot simulators.

**Articulated objects modeling.** *PartNet-Mobility* dataset [32] has advanced research in supervised learning for articulated objects modeling. Most existing studies focus on everyday objects with only one or a few degrees of freedom [8, 12, 15, 16, 23, 33, 35, 47, 50, 53]. Few works [7, 25, 30, 36, 51] have explored the challenging task of reconstructing complex kinematic structures with more than 10 degrees of freedom. Among these, *Real2Code* [30] and *URDFormer* [7] are recent work that reconstructs the kinematic structure of actuated objects through code generation. However, these works assume moving parts are connected to a single parent part, which is not applicable to robotic structures with a series of joints for each kinematic branch. *Watch-It-Move* [36] and *Reart* [25] are two notable works that reconstruct the kinematic structure for multiple kinds of robots. However, these methods use custom interaction code: *Watch-It-Move* repositions the output robot models by applying rototranslation to each joint, and *Reart* reani-

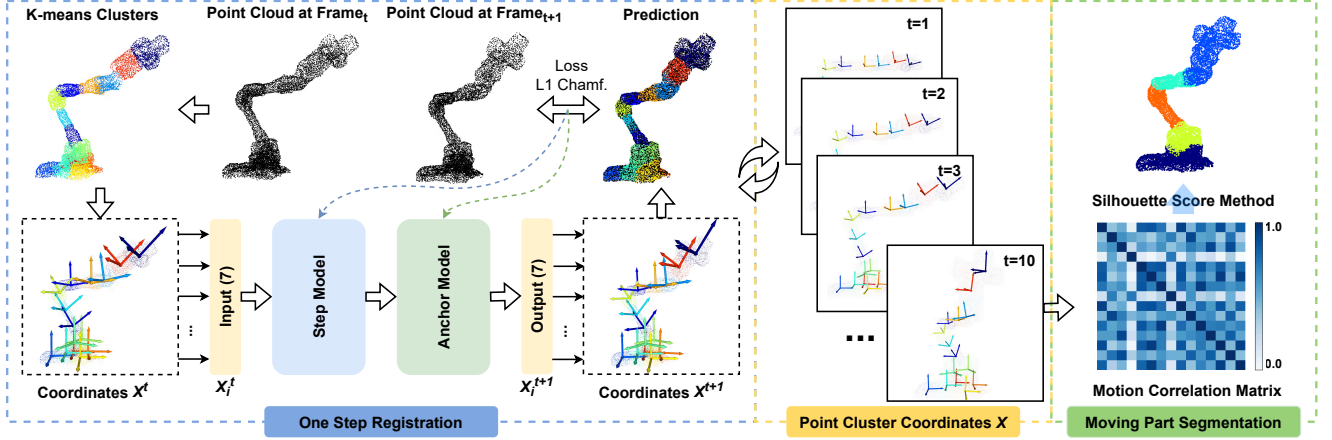


Figure 3. **Point Cluster Registration and Part Segmentation.** We tackle the part segmentation problem by extracting 6-DoF transformations from the registration of a set of point clusters. For each pair of consecutive frames, we begin by initializing point clusters using the K-means clustering algorithm. Each point cluster is assigned a 6-DoF coordinate,  $X_i^t$ , predicted from the previous registration step. Next, we employ a shared neural network as the Step Model to predict the target frame’s point cluster coordinates,  $X_i^{t+1}$ . Following the rotation representation study [5, 10, 54], we represent it as a 7-dimensional vector, combining Cartesian coordinates and quaternion orientation. The predicted coordinates are passed through an Anchor Model that computes cluster transformations, starting from the initial coordinates  $X^1$ . This process is repeated across frames in the sequence, tracing the motion of a set of 6-DoF coordinate frames. Finally, based on the time-series 6-DoF transformations, we compute a correlation matrix between the point clusters to segment distinct moving parts.

mates the model by specifying target points for each part. In contrast to the aforementioned methods, our approach does not rely on ground-truth annotations for training supervision and generates standard URDF files that integrate seamlessly with kinematics computation in simulators.

To the best of our knowledge, our method is the first to construct functional description files for complex robotic structures (with up to 18 degrees of freedom and serially connected links) from unlabeled point cloud data.

### 3. Methodology

Our goal is to reconstruct a robot description file from a time series of point cloud frames by extracting the necessary structural and motion information. This process involves segmenting the robot links, identifying the connections between them, determining joint parameters, and capturing the geometry of each link as mesh files. In this paper, we focus on revolute joints, the most common type in robotic structures, and assume a tree-like structure without closed kinematic loops.

#### 3.1. Problem Formulation and Method Overview

For each robot reconstruction task, we collected a time series of whole-body point cloud frames,  $\mathcal{P} = \{P^t \in \mathbb{R}^{3 \times N}\}_{t \in [1, T]}$ , where  $P^t$  is the point cloud at time  $t$  and  $N$  is the number of points in each frame. Our goal is to extract three components:

- **Segmented parts:** denoted as link point clouds  $\mathcal{L} = \{L_i \in \mathbb{R}^{3 \times N_i}\}_{i \in [1, M]}$ , where  $L_i$  is the point cloud of the

$i$ -th link, and  $M$  is the total number of links.

- **Body topology:** represented as graph  $\mathcal{G} = (I, E)$ , where  $I = \{i \in [1, M]\}$  is the set of link indices and  $E = \{(i_p, i_c)\}_{i_p, i_c \in I, i_p \neq i_c}$  is the set of edges connecting the parent and child links.
- **Joint parameters:** represented as  $\mathcal{J} = \{J_i\}_{i \in [1, K]}$ , where  $J_i$  contains the 6-DoF parameters of the  $i$ -th joint, and  $K$  is the number of joints. For the tree-like structure, we have  $K = M - 1$ .

To achieve this goal, we design a rigid body registration algorithm to track the motion of a set of point clusters, initialized at time step  $t = 1$ . We define the set of point clusters as  $\mathcal{C} = \{C_i^t \in \mathbb{R}^{3 \times N_i}\}_{t \in [1, T], i \in [1, S]}$ , where  $S$  is a hyperparameter that determines the number of clusters. The motion of the clusters is represented by a set of position and orientation coordinates,  $\mathcal{X} = \{X_i^t = (x_i^t, q_i^t)\}_{t \in [1, T], i \in [1, S]}$ , where  $x \in \mathbb{R}^3$  represents the Cartesian center of the point cluster, and  $q \in \mathbb{R}^4$  is the quaternion orientation of the cluster. These coordinates are used to compute a correlation matrix between point clusters. Based on this correlation matrix,  $\mathcal{M}^{S \times S} = \{\rho(X_i, X_j)\}_{i, j \in [1, S]}$ , we segment the clusters into distinct moving parts, denoted as the predicted links  $\hat{\mathcal{L}}$ . We then apply the MST algorithm on the position components of  $\mathcal{X}$  to infer the body topology  $\hat{\mathcal{G}}$ . Finally, we estimate the joint parameters  $\hat{\mathcal{J}}$  by computing the homogeneous transformation matrices for pairs of clusters associated with connected links.

In summary, we derive  $\hat{\mathcal{L}}, \hat{\mathcal{G}}, \hat{\mathcal{J}}$  from  $\mathcal{P}$  by solving for transformation coordinates  $\mathcal{X}$  and correlation matrix  $\mathcal{M}$ .

### 3.2. Cluster Registration and Part Segmentation

Traditional point cloud registration algorithms [2] typically focus on rigid body alignment, while non-rigid registration methods handle more complex deformation scenarios, such as those involving the human body, cloth, or soft objects. In these cases, 3D coordinates (e.g., x, y, z) and graph representations are often used to model deformation. Robots, however, are articulated rigid bodies with 1-DoF joints, which can be viewed as a simplified case of non-rigid structures. We leverage the coherent motion of rigid parts to extract accurate 6-DoF transformations for each point cluster.

As shown in Figure 3, The **Step Model** registers point clusters from time step  $t$  to the ground truth at  $t + 1$ , while the **Anchor Model** registers clusters from the first time step directly to the ground truth at  $t + 1$ . The initial coordinates represent the 6-DoF poses (translation and rotation) of the initially segmented clusters, serving as the starting point for registration. Inspired by PointNet [42, 43], we designed a new lightweight neural network that computes coordinates using a shared MLP module, leveraging the permutation-invariant property to ensure that the input cluster order does not affect the output. Details on the model architecture and 6-DoF pose comparisons are provided in the appendix B.

The loss function Eq. 1 is a bi-directional L1 Chamfer distance [3, 44] between the predicted point clusters and the target whole-body point cloud. This loss function serves to optimize the network parameters, which in turn updates the output coordinates.

$$\mathcal{L}_{\text{Chamfer}}^1(\hat{P}^t, P^t) = \sum_{x=1}^N \min_y \left\| \hat{P}_x^t - P_y^t \right\|_1 + \sum_{y=1}^N \min_x \left\| \hat{P}_x^t - P_y^t \right\|_1 \quad (1)$$

$\hat{P}^t = \bigcup_{i=1}^S {}^w\hat{C}_i^t$  is the predicted point cloud for the entire body, integrated from the set of point clusters in the *world coordinate frame* at time step  $t$ ,  $\{{}^w\hat{C}_i^t\}_{i \in [1, S]}$ .  $P^t$  is the target whole body point cloud at the same time step. Step registration includes a resampling module, where the predicted coordinates  $X^{t+1}$  are used as the center for clustering the next frame's point cloud  $P^{t+1}$ . Consequently, the step matching model performs registration over one frame interval. The Anchor Model takes as input the predicted coordinates from the Step Model and the initial point cluster from the first frame. It refines the coordinates to capture the transformation from the initial step to the target frame.

We demonstrate our registration pipeline, whose input is a whole body point cloud video  $\mathcal{P}$  and output are the clustered point cloud set  $\mathcal{C}$  and the correlated 6D coordinates  $\mathcal{X}$ , in algorithm 1. In the clustering and Chamfer distance calculations, we use point clusters in the world coordinate frame. For optimization in the Step Model  $\mathcal{F}_S$  and Anchor

Model  $\mathcal{F}_A$ , we work with point clusters in the local coordinate system, represented by the tracked 6-DoF coordinates associated with each cluster. We employ two clustering algorithms: K-means[28] in the resampling module, where cluster centers are provided as input, and K-means++[1] in the initialization step, where the hyperparameter  $S$  (the number of clusters) is specified.  $H$  represents the set of homogeneous transformations in  $SE(3)$ , and  $\theta$  denotes the parameters of the neural network.

---

#### Algorithm 1 Point Cluster Registration

---

**Input:** Point Cloud Frames  $\mathcal{P}$

**Output:** Point Clusters  $\mathcal{C}$ , Coordinates  $\mathcal{X}$

**Initialize:** at step 1,  ${}^wC^1 = \text{Clustering}_1(P^1, N_{\text{seg}} = S)$   
 $X^1 = \{X_i^1 = (x = \text{mean}(C_i^1), \alpha = \text{zeros}(3))\}_{i \in [1, S]}$   
 Add  $X^1, {}^lC^1$  to  $\mathcal{X}, \mathcal{C}$  ( ${}^lC$  in local coordinate frame)

**for**  $t = 1, \dots, T$  **do**

    Load  $X^t, {}^lC^t$  from  $\mathcal{X}, \mathcal{C}$

**Step Registration:**

${}^w\hat{H}_i^{t+1} \leftarrow \hat{X}^{t+1} = \mathcal{F}_S(X^t, {}^lC^t, \theta_S)$

        where  ${}^wH_i = \{H_{i,i} \in SE(3)\}_{i \in [1, S]}$

${}^w\hat{C}^{t+1} \leftarrow \{{}^w\hat{H}_{i,i}^{t+1} \cdot {}^lC_i^t\}_{i \in [1, S]}$

$\hat{P}^{t+1} = \bigcup_{i=1}^S {}^w\hat{C}_i^{t+1}$

        Find  $\theta_S^* = \arg \min_{\theta} \mathcal{L}_{\text{Chamfer}}^1(\hat{P}^{t+1}, P^{t+1})$

        Update  $\hat{X}^{t+1}, \hat{C}^{t+1}$  with  $\theta_S^*$

**Resample:**

${}^w\hat{C}^{t+1} = \text{Clustering}_2(P^{t+1}, \text{Centers} = \hat{x}^{t+1})$

        where the  $\hat{x}$  is the position term of  $\hat{X}$

        Transfer  ${}^w\hat{C} \rightarrow {}^l\hat{C}$

**Anchor Alignment:**

$\bar{X}^{t+1} = \mathcal{F}_A(\hat{X}^{t+1}, {}^lC^1, P^{t+1})$

    Add  $\bar{X}^{t+1}, {}^l\hat{C}^{t+1}$  to  $\mathcal{X}, \mathcal{C}$

**end for**

---

As shown in equation 2 and equation 3, we compute the correlation matrix  $\mathcal{M}$  between the point clusters based on the transformation coordinates  $\mathcal{X}$ . The distance is calculated as the Euclidean distance between positions and the geodesic distance between orientations. Here,  $\alpha$  is a scaling parameter automatically calculated from the bounding box of the point cloud. The result is normalized to a range between 0 and 1 to facilitate alignment across multiple sequences.

$$\rho(X_i, X_j)_{i,j \in [1, S]} = \frac{\sum_{t=1}^T \mathcal{D}(X_i^t, X_j^t)}{\max_{i,j} \sum_{t=1}^T \mathcal{D}(X_i^t, X_j^t)} \quad (2)$$

$$\mathcal{D}(X_i^t, X_j^t) = \alpha \cdot \mathbf{d}_{\text{Euc}}(x_i^t, x_j^t) + \mathbf{d}_{\text{Geo}}(q_i^t, q_j^t) \quad (3)$$

Clusters with highly correlated coordinate sequences (indicating minimal pose differences) are grouped into the same moving parts. We then merge the point clusters within each

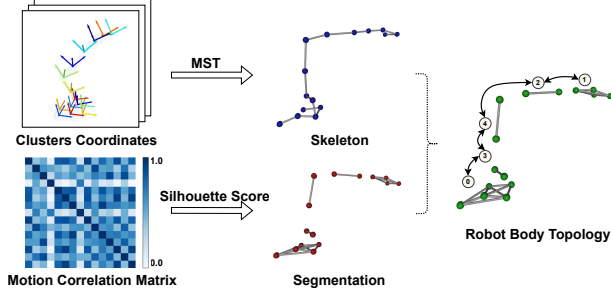


Figure 4. **Topology Inference.** We infer the robot’s body topology by merging the result of the MST algorithm on position terms  $x$  and the segmented point clusters. In the segmentation graph, the nodes represent the center positions of the clusters; the edges indicate the motion correlated between the connected nodes.

moving part to form the predicted links  $\hat{\mathcal{L}}$ , which represent sets of point clouds that capture the shape of the robot links.

We use the Silhouette Score method [46] to determine the optimal number of segments based on the Motion Correlation Matrix.

### 3.3. Topology Inference

We infer the body topology using two graphs: (1) the graph of segmented point clusters, where nodes represent cluster indices and edges represent inter-cluster connections; and (2) the minimum spanning tree constructed using the accumulated positional distance between point clusters as edge weights. We demonstrate the process in Figure 4, and detailed algorithm in Appendix B. The MST algorithm is applied to the point clusters, where the weight of each edge is determined by the accumulated position distance between the point clusters. We identify edges in the MST that connect point clusters from different moving parts, which correspond to the connections between robot links. The resulting graph  $\hat{\mathcal{G}}$  represents the body topology of the robot, with the nodes as the link index and the edges as the connections between the links. To comply with the URDF format, we assign the robot’s base link as the root node. We rank all links by their total pose variation across frames and select the one with the least variation as the base link. Starting from this base link, we traverse the graph as a directed tree to establish the parent-child relationships between the links.

### 3.4. Joint Estimation

Referring to the merged point clusters, we compute the average 6D coordinates of each link. In joint estimation, we convert the pose coordinates  $X = (x, q)$  into a homogeneous transformation matrix  $H \in \text{SE}(3)$ . The base link is assigned the identity matrix  $I$ , and each link’s pose in the world frame is denoted by  $H_i$ . The relative transformation from the parent link to the  $i$ -th link is then computed as  $H_i^p = H_i \cdot H_p^{-1}$  where  $H_p$  is the pose of the parent link.

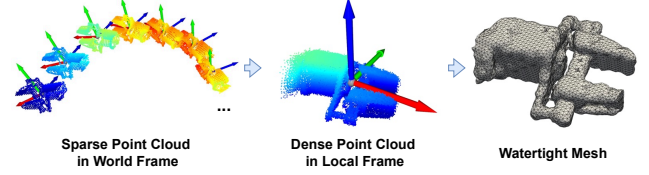


Figure 5. **Point Cloud to Mesh.** For each segmented link, we integrate the sparse point cloud data from the world coordinate system at each time step to form a dense point cloud in the local frame. The dense point cloud is constructed by combining 10 frames of data. We then apply the marching cubes[26] algorithm to convert this dense point cloud into a mesh file. The resulting mesh files are high-quality and watertight, as demonstrated by the example of the WX200 robot arm’s end-effector shown in the figure.

According to the body topology graph, we locate every parent-child pair of links and compute the time-series transformation matrices of each child link with respect to the parent link’s coordinate system. For each 1-DoF revolute joint connecting a pair of links, the relative transformation yields a rotation matrix  $R \in \text{SO}(3)$  and a static translation vector  $t \in \mathbb{R}^3$ . We extract the joint parameters  $\hat{\mathcal{J}}$  as the rotation axis and joint center, derived from the rotation matrix and translation vector, respectively. Our method constrains  $\text{SE}(3)$  transformations to 1-DoF joint motions using Euler’s rotation theorem, parameterizing motion with a fixed point, rotation axis, and angle, making it naturally compatible with URDF.

### 3.5. URDF Generation

The URDF contains an XML file that defines the properties of the robot’s links and joints, along with a set of mesh files representing the geometric details of each link. Based on the predicted body topology and joint parameters, we generate the XML file specifying the links and joints of the robot. The pose coordinates of each link, which represent the transformation from the link’s local coordinate system to the world coordinate system, allow us to express all link point clouds in a consistent world frame. This enables integration of sparse point clouds across frames into a dense point cloud in the local frame. As shown in Figure 5, we combine data from 10 frames to construct a dense point cloud, which we then process using the marching cubes algorithm[26] to convert the dense representation into a high-quality, watertight mesh. The resulting mesh files are used to define the geometric properties of each link in the URDF file. In this paper, we focus on kinematics and morphology, leaving the modeling of dynamic properties for future work.

In summary, we demonstrate the derivation of  $\hat{\mathcal{L}}$ ,  $\hat{\mathcal{G}}$ ,  $\hat{\mathcal{J}}$  in Section 3.2, 3.3, and 3.4, which together enable URDF generation.

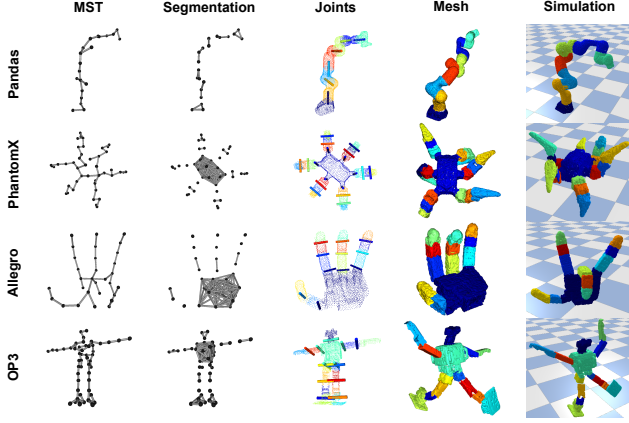


Figure 6. **Qualitative Results.** We illustrate the core stages of our method for various robots, including the minimum spanning tree, part segmentation, joint estimation results, mesh, and constructed URDFs for simulation. The colors of the joints indicate the parent link they are connected to.

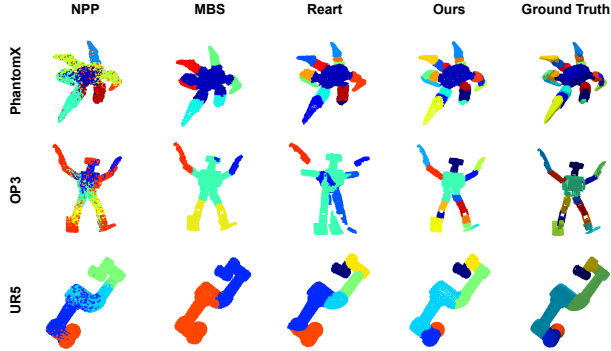


Figure 7. **Qualitative Comparison with Baseline Methods.** Given the same input sequence (10 point cloud frames), we compare our method with NPP [11], MBS [14], and Reart [25] for part segmentation and point cloud registration on the final frame.

## 4. Experiments

### 4.1. Experiments Setup

**Dataset:** The validation dataset includes a variety of robots, including single-branch link robot arms (WX200, Franka Panda, UR5e) and multi-branch link robots (Bolt, Solo8, PhantomX, Allegro Hand, OP3 Humanoid), with degrees of freedom ranging from 5 to 18. We evaluate our method on one point cloud video of a real-world scan of the WX200 robot arm and five point cloud videos of eight robots in simulated settings. Each video consists of 10 frames of point cloud data, down-sampled to 5,000 points per frame. For data collection, we fixed certain joints on the Allegro Hand and OP3 Humanoid and removed the end-effectors for the Franka Panda and UR5e. Further details on the data collection pipeline are provided in the Appendix A.

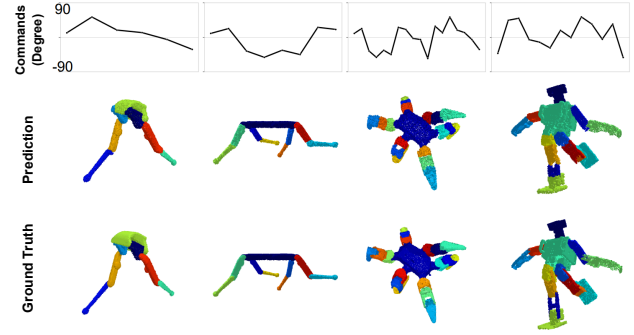


Figure 8. **Qualitative Results for Simulating New Configurations.** We compare the predicted URDFs with ground truth by applying the same motor commands. AutoURDF generates accurate description files, producing control results that closely match those of the ground-truth models. For visualization, we use the same colormap for both the predicted and ground-truth URDFs.

**Metrics:** We evaluate our method across four metrics: registration, body topology inference, joint estimation, and repose. **Registration** is measured using the average L1 Chamfer Distance (in millimeters), denoted as  $CD$  in our tables, which quantifies the alignment between the transformed whole-body point cloud from the first frame and each subsequent frame. **Body topology inference** is evaluated by comparing the predicted kinematic tree with the ground-truth kinematic tree using the tree editing distance [39, 40], denoted as  $TED$ , to assess the accuracy of the inferred body topology. We measure **joint estimation** by calculating the average angular difference between joints (in degrees) and the average normal distance between the predicted and ground-truth joint axes (in millimeters). We denote these two metrics as  $E_{JD}$  and  $E_{JA}$ , unit in millimeters and degrees, respectively. **Repose** accuracy, denoted as  $CD_r$ , is evaluated by commanding two URDFs with random new motor angles and computing the surface point cloud L1 Chamfer distance. It is important to note that Chamfer distance depends on the robot’s scale and the density of the point cloud. In this work, we calculate the L1 Chamfer distance using the original scale of the robots, with 5,000 points sampled for alignment in all experiments.

### 4.2. Qualitative Experiments

Figure 6 illustrates the core stages of AutoURDF, including the MST, segmentation results represented as cluster skeletons, joint prediction results, mesh quality, and simulation examples in the PyBullet simulator. In Figure 7, we qualitatively compare our segmentation and registration results with NPP [11], MultibodySync (MBS)[14], and Reart [25] using the same point cloud sequences. Our method produces more accurate segmentation and matching results across all three robot sequences. Figure 8 provides a direct

| Metrics          | Methods     | WX200       | Panda        | UR5e         | Bolt        | Solo        | PhantomX     | Allegro     | OP3         | Mean $\pm$ Std                    |
|------------------|-------------|-------------|--------------|--------------|-------------|-------------|--------------|-------------|-------------|-----------------------------------|
| $CD \downarrow$  | Reart[25]   | 9.33        | 18.81        | 15.86        | 10.39       | 11.14       | 14.73        | 6.38        | 44.95       | $16.45 \pm 12.18$                 |
|                  | <b>Ours</b> | <b>7.49</b> | <b>13.56</b> | <b>12.84</b> | <b>8.41</b> | <b>9.77</b> | <b>10.88</b> | <b>5.80</b> | <b>8.30</b> | <b><math>9.63 \pm 2.67</math></b> |
| $TED \downarrow$ | MBS[14]     | 3.33        | 5.00         | 3.40         | 3.80        | 4.40        | 14.60        | 8.60        | 10.00       | $6.64 \pm 4.07$                   |
|                  | Reart[25]   | 0.83        | 2.40         | 4.40         | 3.20        | 4.00        | 13.00        | 6.00        | 11.60       | $5.68 \pm 4.37$                   |
|                  | <b>Ours</b> | <b>0.33</b> | <b>1.40</b>  | <b>0.60</b>  | <b>1.75</b> | <b>0.00</b> | <b>4.00</b>  | <b>4.00</b> | <b>6.00</b> | <b><math>2.26 \pm 2.16</math></b> |

Table 1. Baseline Comparison.

| Metrics          | Methods      | WX200       | Panda        | UR5e         | Bolt        | Solo        | PhantomX     | Allegro     | OP3         | Mean $\pm$ Std                    |
|------------------|--------------|-------------|--------------|--------------|-------------|-------------|--------------|-------------|-------------|-----------------------------------|
| $CD \downarrow$  | w/o anchor m | 7.95        | 14.67        | 14.19        | 9.74        | 10.73       | 12.31        | 6.38        | 9.86        | $10.73 \pm 2.89$                  |
|                  | w/o step m   | <b>7.52</b> | 13.59        | 12.90        | 8.42        | 9.80        | 11.03        | <b>5.79</b> | 8.42        | $9.68 \pm 2.69$                   |
|                  | <b>Ours</b>  | 7.55        | <b>13.52</b> | <b>12.84</b> | <b>8.39</b> | <b>9.78</b> | <b>10.93</b> | <b>5.79</b> | <b>8.32</b> | <b><math>9.64 \pm 2.69</math></b> |
| $TED \downarrow$ | w/o ori      | 3.17        | 3.20         | 2.20         | 3.00        | 7.20        | 20.60        | 8.80        | 9.80        | $7.25 \pm 6.14$                   |
|                  | w/o pos      | 1.33        | <b>0.60</b>  | 0.80         | <b>1.40</b> | 6.20        | 17.40        | 7.60        | 12.00       | $5.92 \pm 6.18$                   |
|                  | <b>Ours</b>  | <b>0.33</b> | 1.40         | <b>0.60</b>  | 1.75        | <b>0.00</b> | <b>4.00</b>  | <b>4.00</b> | <b>6.00</b> | <b><math>2.26 \pm 2.16</math></b> |

Table 2. Ablation Experiment.

| Metrics             | Methods     | WX200        | Panda        | UR5e         | Bolt         | Solo         | PhantomX     | Allegro     | OP3          | Mean $\pm$ Std                      |
|---------------------|-------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|--------------|-------------------------------------|
| $CD_r \downarrow$   | 1 Sequence  | 11.51        | <b>42.82</b> | 35.76        | 17.86        | 13.41        | 14.76        | 8.66        | 32.25        | $22.13 \pm 12.87$                   |
|                     | 5 Sequences | <b>11.10</b> | 45.60        | <b>21.24</b> | <b>12.39</b> | <b>13.03</b> | <b>13.85</b> | <b>8.30</b> | <b>21.85</b> | <b><math>18.42 \pm 11.96</math></b> |
| $E_{JD} \downarrow$ | 1 Sequence  | <b>1.16</b>  | 4.46         | 3.32         | 5.44         | 5.34         | 4.39         | 6.20        | 11.60        | $5.24 \pm 3.00$                     |
|                     | 5 Sequences | 1.17         | <b>3.95</b>  | <b>2.17</b>  | <b>4.94</b>  | <b>1.61</b>  | <b>2.50</b>  | <b>3.97</b> | <b>7.37</b>  | <b><math>3.46 \pm 2.04</math></b>   |
| $E_{JA} \downarrow$ | 1 Sequence  | 1.91         | 7.37         | 7.39         | 4.13         | 3.59         | 3.91         | 7.85        | 15.81        | $6.50 \pm 4.34$                     |
|                     | 5 Sequences | <b>1.13</b>  | <b>4.16</b>  | <b>2.88</b>  | <b>2.44</b>  | <b>1.86</b>  | <b>1.74</b>  | <b>3.68</b> | <b>7.11</b>  | <b><math>3.13 \pm 1.90</math></b>   |

Table 3. Multi-Sequence Merging Experiment.

comparison between the generated URDFs and the ground truth URDFs under the same new motor configurations. We also demonstrate the Silhouette Score with respect to DoF in Figure 9, based on the Motion Correlation matrices. The highest score indicates the optimal number of distinct moving parts, and therefore the number of degrees of freedom. Finally, we tested our method on real scanned point cloud data of a WX200 robot arm, commanding on all 5 servo motors, shown in Figure 10. Despite the high noise in the scan and misalignment of the coordinate systems across frames, our method successfully derives a functional URDF file for this robot with only 10 frames data, achieving relatively accurate kinematics and morphology computation.

### 4.3. Quantitative Results and Ablation Study

Table 1 shows a quantitative comparison between our method and two previous works [14, 25] on our validation dataset. We compare our method with Reart for point cloud registration, using average end-point error measured in L1 Chamfer distance ( $CD$ ). For robot body topology inference, we compare our method with both *Reart* [25] and *MultiBodySync* (MBS) [14] by computing the tree editing distance ( $TED$ ) between the ground truth kinematic tree and the predicted ones. Our method consistently achieves

lower scores on both metrics, indicating improved registration accuracy and body topology inference accuracy across all types of robots.

To validate the contributions of the Step Model and Anchor Model in our point cluster registration algorithm, we compare the matching results of our complete pipeline with results using only the Step Model and only the Anchor Model. The complete method achieves superior performance, as shown in Table 2. Additionally, to validate the contributions of the two distance terms in Equation 3, we conduct ablation experiments comparing the complete equation with versions that include only the position term or only the orientation term. As shown in Figure 11, using only the position distance struggles to produce accurate segmentation, while using only the orientation term is easily misled by links with similar rotational motions. Multiple links are segmented into the same group due to their similar rotational motion within this point cloud sequence.

Table 3 presents a comparison of our method, extending the input point cloud data from 1 sequence (10 frames) to 5 sequences (50 frames). We evaluate the reposed robot shape error using the  $CD_r$  metric, the joint axis normal distance with the  $E_{JD}$  metric, and the joint angle error with the  $E_{JA}$  metric. As the dataset scales, AutoURDF improves perfor-

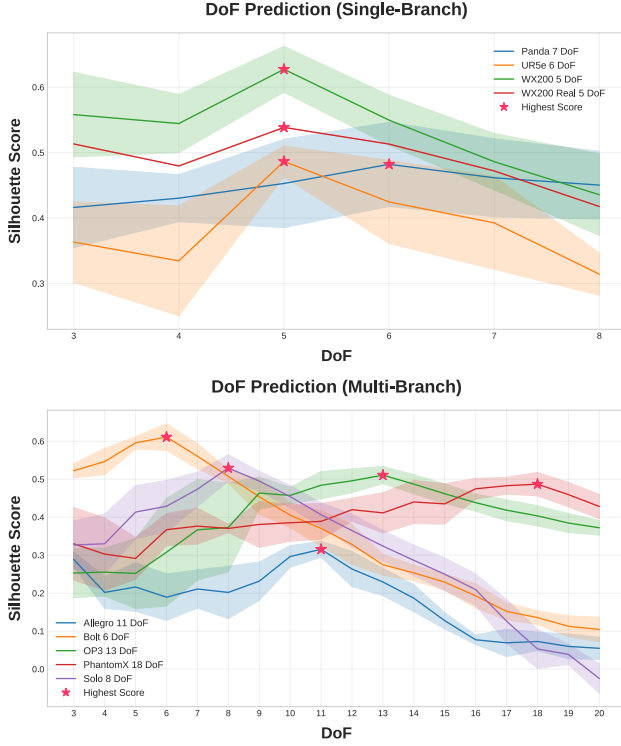


Figure 9. **DoF Prediction.** We plot the Silhouette Score for eight different robots across five point cloud sequences. The highest score indicates the optimal number of distinct moving parts, and thus the number of DoF. By averaging the Silhouette Scores across sequences, our method accurately predicts the correct number of DoF for all eight robots.

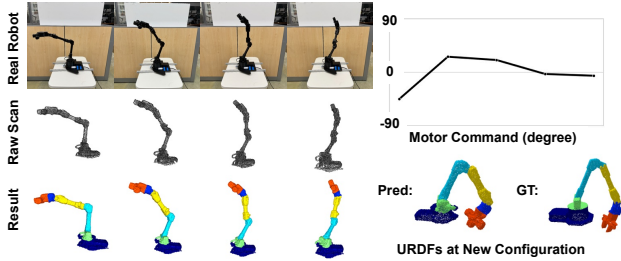


Figure 10. **Real World Experiment.** The first row shows the WX200 robot arm at four different time steps under random motion commands. The second row displays the corresponding whole-body point cloud frames. The third row presents alignment results with meshes. The right column compares the predicted URDF with the ground-truth model under a new configuration.

mance in 7 out of 8 robots for repose and joint distance and in all robots for joint angle accuracy. With 50 frames of point cloud data from the WX200 robot arm, we achieve an average joint angle difference of 1.13 degrees and a joint normal distance of 1.16 mm across its 5 joint axes.

Among all these improvements, AutoURDF stands out

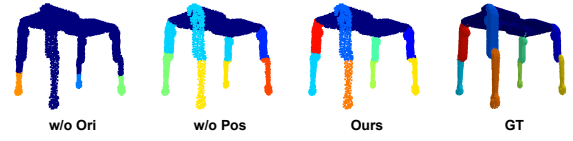


Figure 11. **Ablation study on Correlation Matrix.** We compare the distance equation 3 under three conditions: without the orientation term  $\mathbf{d}_{Geo}$ , without the position term  $\mathbf{d}_{Euc}$ , and with the complete calculation.

for its speed as an unsupervised method. On the WX200 robot data, registration takes **50** seconds, and URDF construction takes **12** seconds on an NVIDIA 3090 GPU. In comparison, *Reart* [25] relax model and projection model training, with the flow model pre-trained, takes 35.5 minutes (2,120 seconds) on the same machine.

## 5. Discussion and Conclusion

**Limitations and future work:** Our method has three key limitations. First, we collected collision-free robot motion sequences, there is no self-collision or interaction with the environment, and thus our method does not learn dynamic information, and the resulting URDF files lack mass and moment of inertia data. Second, segmenting more complex structures requires longer point cloud sequences. For example, with the OP3 humanoid robot, our method struggles to produce clean segmentation due to structural complexity. Extending the length of the point cloud video could improve segmentation accuracy for such complex robots. Third, our method is limited to robots with revolute joints and tree-like kinematic structures. Future work could extend the approach to handle non-revolute joints—such as prismatic or spherical joints—as well as robots with parallel or closed-loop kinematic chains.

**Conclusion:** In this paper, we present a novel approach for constructing robot description files directly from point cloud data. Our method employs 6-DoF cluster registration to segment moving parts, infer body topology, and estimate joint parameters, achieving accurate joint direction estimation and whole-body shape reposing. The output URDF-formatted robot description file is compatible with widely used robot simulators. Experimental results show that our method outperforms previous approaches in the accuracy of robot point cloud registration and body topology estimation, offering an efficient and scalable solution for automated robot modeling from visual data.

**Acknowledgements:** This work was supported in part by the US National Science Foundation AI Institute for Dynamical Systems (DynamicsAI.org) (grant no. 2112085). The authors thank Prof. Changxi Zheng and Ruoshi Liu for their invaluable feedback.

## References

- [1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006. 4
- [2] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 4
- [3] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 10(6):849–865, 1988. 4
- [4] Stephen Brawner. Solidworks to urdf exporter. [https://github.com/ros/solidworks\\_urdf\\_exporter](https://github.com/ros/solidworks_urdf_exporter), 2024. Accessed: 2024-11-13. 1
- [5] Romain Brégier. Deep regression on manifolds: a 3d rotation case study. In *2021 International Conference on 3D Vision (3DV)*, pages 166–174. IEEE, 2021. 3
- [6] Boyuan Chen, Robert Kwiatkowski, Carl Vondrick, and Hod Lipson. Fully body visual self-modeling of robot morphologies. *Science Robotics*, 7(68):eabn1944, 2022. 1, 2
- [7] Zoey Chen, Aaron Walsman, Marius Memmel, Kaichun Mo, Alex Fang, Karthikeya Vemuri, Alan Wu, Dieter Fox, and Abhishek Gupta. Urdformer: A pipeline for constructing articulated simulation environments from real-world images. *arXiv preprint arXiv:2405.11656*, 2024. 2
- [8] Ruihang Chu, Zhengzhe Liu, Xiaoqing Ye, Xiao Tan, Xiaojuan Qi, Chi-Wing Fu, and Jiaya Jia. Command-driven articulated object understanding and manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8813–8823, 2023. 2
- [9] Erwin Coumans. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, page 1. ACM, 2015. 2, 1
- [10] A René Geist, Jonas Frey, Mikel Zobro, Anna Levina, and Georg Martius. Learning with 3d rotations, a hitchhiker’s guide to so (3). *arXiv preprint arXiv:2404.11735*, 2024. 3
- [11] David S Hayden, Jason Pacheco, and W Fisher John. Non-parametric object and parts modeling with lie group dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7426–7435, 2020. 6
- [12] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21201–21210, 2023. 2
- [13] Yuhang Hu, Jiong Lin, and Hod Lipson. Teaching robots to build simulations of themselves. *arXiv preprint arXiv:2311.12151*, 2023. 1
- [14] Jiahui Huang, He Wang, Tolga Birdal, Minhyuk Sung, Federica Arrigoni, Shi-Min Hu, and Leonidas Guibas. Multi-body sync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7104–7114. IEEE, 2021. 1, 6, 7
- [15] Ajinkya Jain, Rudolf Lioutikov, Caleb Chuck, and Scott Niekum. Screwnet: Category-independent articulation model estimation from depth images using screw theory. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13670–13677. IEEE, 2021. 2
- [16] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 1
- [18] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2
- [19] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956. 2
- [20] Robert Kwiatkowski and Hod Lipson. Task-agnostic self-modeling machines. *Science Robotics*, 4(26):eaau9354, 2019. 1, 2
- [21] Fernando Díaz Ledezma and Sami Haddadin. Machine learning-driven self-discovery of the robot body morphology. *Science Robotics*, 8(85):eadh0972, 2023. 1, 2
- [22] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3706–3715, 2020. 1
- [23] Jiayi Liu, Ali Mahdavi-Amiri, and Manolis Savva. Paris: Part-level reconstruction and motion analysis for articulated objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 352–363, 2023. 2
- [24] Ruoshi Liu, Alper Canberk, Shuran Song, and Carl Vondrick. Differentiable robot rendering, 2024. 1, 2
- [25] Shaowei Liu, Saurabh Gupta, and Shenlong 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*, pages 21138–21147, 2023. 2, 6, 7, 8, 4
- [26] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’87)*, pages 163–169. ACM, 1987. 5
- [27] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2
- [28] J MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967. 4
- [29] Viktor Makovychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. 1
- [30] Zhao Mandi, Yijia Weng, Dominik Bauer, and Shuran Song. Real2code: Reconstruct articulated objects via code generation. *arXiv preprint arXiv:2406.08474*, 2024. 1, 2

- [31] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2
- [32] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 5
- [33] Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. A-sdf: Learning disentangled signed distance functions for articulated shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13001–13011, 2021. 2
- [34] Inc. Niantic. Scaniverse. <https://scaniverse.com/>, 2024. Mobile app. 1
- [35] Neil Nie, Samir Yitzhak Gadre, Kiana Ehsani, and Shuran Song. Structure from action: Learning interactions for articulated object 3d structure discovery. *arXiv preprint arXiv:2207.08997*, 2022. 2
- [36] Atsuhiko Noguchi, Umar Iqbal, Jonathan Tremblay, Tatsuya Harada, and Orazio Gallo. Watch it move: Unsupervised discovery of 3d joints for re-posing of articulated objects. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3667–3677. IEEE, 2022. 2
- [37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 2
- [38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA, 2019. 2
- [39] Mateusz Pawlik and Nikolaus Augsten. Efficient computation of the tree edit distance. *ACM Transactions on Database Systems (TODS)*, 40(1):1–40, 2015. 6
- [40] Mateusz Pawlik and Nikolaus Augsten. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56:157–173, 2016. 6
- [41] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2
- [42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 4
- [43] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 4
- [44] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 4
- [45] ROS Wiki. Unified robot description format (urdf), 2023. Accessed: 2024-09-24. 1
- [46] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. 5, 2
- [47] Chaoyue Song, Jiacheng Wei, Chuan Sheng Foo, Guosheng Lin, and Fayao Liu. Reacto: Reconstructing articulated objects from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5384–5395, 2024. 2
- [48] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 2
- [49] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 1
- [50] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinpeng Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 2
- [51] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020. 2
- [52] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Trans. Graph.*, 37(6), 2018. 1
- [53] Harry Zhang, Ben Eisner, and David Held. Flowbot++: Learning generalized articulated objects manipulation via articulation projection. In *Conference on Robot Learning*, pages 1222–1241. PMLR, 2023. 2
- [54] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5745–5753, 2019. 3, 2