Building Rearticulable Models for Arbitrary 3D Objects from 4D Point Clouds

Shaowei Liu[1] Saurabh Gupta[1*] Shenlong Wang[1*]
[1]University of Illinois Urbana-Champaign
https://stevenlsw.github.io/reart/

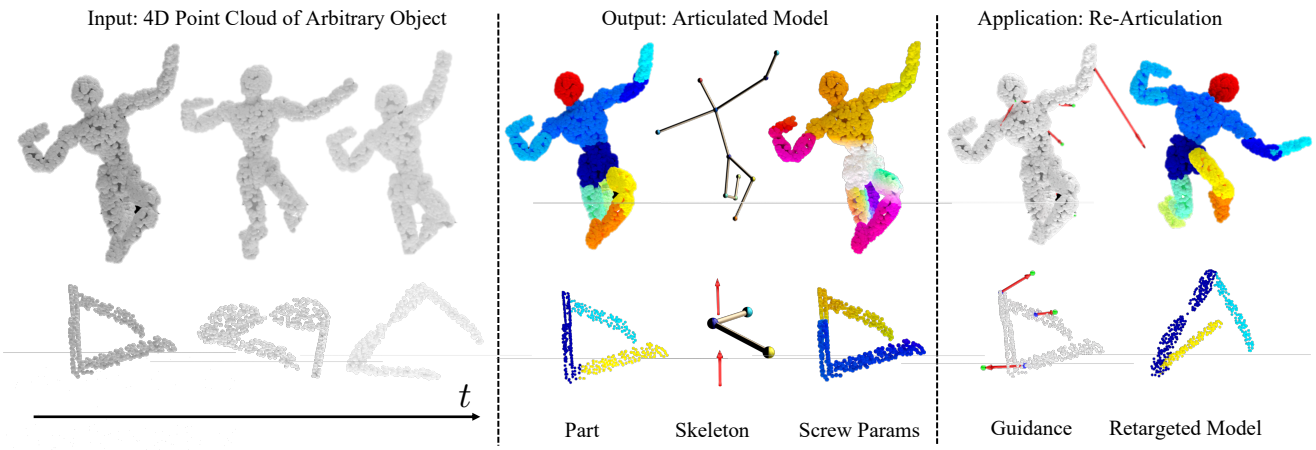


Figure 1. Given a short point cloud sequence of arbitrary articulated object **(left)**, our method outputs an animatable 3D model **(middle)**, which can be retargeted to novel poses with only a few sparse point correspondences **(right)**.

Abstract

We build rearticulable models for arbitrary everyday man-made objects containing an arbitrary number of parts that are connected together in arbitrary ways via 1 degree-of-freedom joints. Given point cloud videos of such everyday objects, our method identifies the distinct object parts, what parts are connected to what other parts, and the properties of the joints connecting each part pair. We do this by jointly optimizing the part segmentation, transformation, and kinematics using a novel energy minimization framework. Our inferred animatable models, enables retargeting to novel poses with sparse point correspondences guidance. We test our method on a new articulating robot dataset, and the Sapiens dataset with common daily objects, as well as real-world scans. Experiments show that our method outperforms two leading prior works on various metrics.

1. Introduction

Consider the sequence of points clouds observations of articulating everyday objects shown in Figure 1. As humans, we can readily infer the *kinematic structure* of the underlying object, *i.e.* the different object parts and their connectivity and articulation relative with one another [23]. This paper develops computational techniques with similar abilities. Given point cloud videos of *arbitrary* everyday objects (with an *arbitrary* number of parts) undergoing articulation, we develop techniques to build animatable 3D reconstructions of the underlying object by a) identifying the distinct object parts, b) inferring what parts are connected to what other parts, and c) the properties of the joint between each connected part pair. Success at this task enables

Figure 2. Many man-made everyday objects can be explained with rigid parts connected in a kinematic tree with 1DOF joints.

*equal advising, alphabetic order

	Arbitrary Parts	Realistic Joint Constraints	Arbitrary Kinematics
Category-specific *e.g.*			
people [34]	no	yes	no
quadrupeds [55, 66]	no	yes	no
cartoons [57]	no	yes	no
DeepPart [61]	yes	no	no
NPP [13]	yes	no	no
ScrewNet [19]	yes	yes	no
UnsupMotion [46]	no	yes	no
Ditto [22]	yes	yes	no
MultiBodySync [17]	yes	no	no
WatchItMove [39]	yes	no	yes
Ours	yes	yes	yes

Table 1. Most past work on inferring rearticulable models is category specific. Building rearticulable models for arbitrary everyday man-made objects requires reasoning about arbitrary part geometries, arbitrary part connectivity, and realistic joint constraints (1DOF w.r.t. parent part). We situate past work along these 3 dimensions, and discuss major trends in Sec. 2.

rearticulation of objects. Given just a few user clicks specifying what point goes where, we can fill in the remaining geometry as shown on the right side in Figure 1.

Most past work on inferring how objects articulate tackles it in a *category-specific* manner, be it for people [34, 40, 44], quadrupeds [55, 66], or even everyday objects [38]. Category-specific treatment allows the use of specialized shape models (such as the SMPL model [34] for people), or defines the output space (*e.g.* location of 2 hinge joints for the category eye-glasses). This limits applicability of such methods to categories that have a canonical topology, leaving out categories with large intra-class variation (*e.g.* boxes that can have 1-4 hinge joints), or in-the-wild objects which may have an arbitrary number of parts connected in arbitrary ways (*e.g.* robots).

Only a very few past works tackle the problem of inferring rearticulable models in a category-agnostic manner. Huang *et al*. [17] only infer part segmentations, which by itself, is insufficient for rearticulation. Jiang *et al*. [22] only consider a single 1-DOF joint per object, dramatically restricting its application (think about a humanoid robot with four limbs, but the articulable model can only articulate one). Noguchi *et al*. [39] present the most complete solution but instead work with visual input and don't incorporate the 1DOF constraint, *i.e*. a part can only rotate or translate about a fixed axis on the parent part, common to a large number of man-made objects as can be seen in Fig. 2. Inferring 3DOF / 6DOF joints leads to unrealistic rearticulation and is thus undesirable (consider the leg of eyeglasses can freely move or rotate). Our work fills this gap in the literature. Our method extracts 3D rearticulable models for arbitrary everyday objects (containing an arbitrary number of parts that are connected together in arbitrary ways via 1DOF joint) from point cloud videos. To the best of our knowledge, this is the first work to tackle this specific problem.

Our proposed method jointly reasons about part geometries and their 1-DOF inter-connectivity with one another. At the heart of our approach is a novel continuous-discrete energy formulation that seeks to jointly learn parameters of the object model (*i.e*. assignments of points in the canonical view to parts, and the connectivity of parts to one another) by minimizing shape and motion reconstruction error (after appropriate articulation of the inferred model) on the given views. As it is difficult to directly optimize in the presence of continuous and discrete variables with structured constraints, we first estimate a *relaxed* model that infers parts that are free to follow an arbitrary 6DOF trajectory over time (*i.e*. doesn't require parts to be connected in a kinematic tree with 1DOF joints). We *project* the estimated relaxed model to a kinematic model and continue to optimize with the reconstruction error to further finetune the estimated joint parameters. Our joint approach leads to better models and improved rearticulation as compared to adaptations of past methods [17, 39] to this task.

2. Related Work

Building rearticulable models for arbitrary objects requires a) identifying the distinct parts, b) the kinematic topology that connects the different parts, and c) any constraints that there may be on the joints. This makes inferring articulable models from raw observation inputs challenging. Consequently, there are very few past works that tackle all these challenges to produce an end-to-end system for this task. A large body of work has focused on tackling subsets of these problems, or they operate in settings where some of these aspects are simplified. We overview past works with respect to these aspects in Tab. 1 and discuss major lines of work in more detail below.

A large body of work tackles this problem in a **category-specific** manner. This leads to a well specified definition of parts and their kinematic connectivity, and allows the use of sophisticated modeling. A good example is modeling for humans [34], human hands [40, 44], cartoon characters [57], and quadrupeds [55, 60, 66]. However, doing this for each new category requires manual effort and this approach doesn't scale to arbitrary objects that we consider in our work. Many recent works have pursued extending such approaches to other categories making assumptions about the number of parts [1, 38, 52], kinematic topology [2, 12, 58], articulation type or common geometric features [14, 30, 53]. Researchers have also tackled intermediate problems that are useful for eventual rearticulation in a category-specific manner, *e.g*. articulated pose estimation [11, 28, 30], motion prediction [16, 21, 29, 45], and 3D reconstruction [18, 26, 38].

Category-agnostic modeling, that is necessary to enable modelling of arbitrary objects, is considerably more

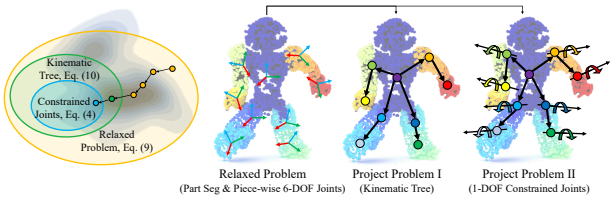


Figure 3. Overview: We formulate the problem of rearticulable object modeling from 4D point cloud as an energy minimization problem. The optimization is divided into a *relaxation* stage that reasons about a 6-DOF piece-wise rigid model without kinematic constraints and a *project* stage that casts the solution to a valid kinematic tree (all joints satisfy 1-DoF constraints).

challenging, and most past work only tackles subsets of the problem. A body of work focuses on unsupervised part segmentation, *i.e*. parsing articulated objects into multiple rigid moving parts. Early work addressed the problem by clustering and co-segmentation [15, 46, 48, 49, 62] by relying on motion and geometry, while more recent approaches [7, 17, 32, 50, 51, 59, 61, 64] use more expressive features extracted using a neural network. Recent work has also tackled pose estimation for arbitrary 1DOF joints [19, 22, 43] or for given kinematic trees [31].

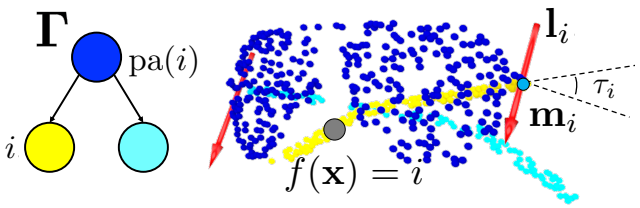
Closest to us are works from Huang *et al*. [17], Jiang *et al*. [22], and Noguchi *et al*. [39]. Huang *et al*. [17] build upon part segmentation work from Yi *et al*. [61] to output temporally-consistent part segmentation by solving a joint synchronization problem. This works very well for part segmentation. However, Huang *et al*. do not infer the kinematic tree connecting these parts together and parts can undergo 6DOF transformation relative to one another. Thus, their output, as is, falls short of the rearticulation goal. Jiang *et al*. [22] tackle rearticulation of generic objects and study the 1DOF nature of the joint. However, their formulation is limited to only infering a *single* 1DOF joint (*e.g*. laptops), and is thus incapable of analysing more complex objects that our approach is able to handle. Lastly, Noguchi *et al*. [39] fit articulated objects models to multi-view RGB video sequences and demonstrate impressive reanimation results. In contrast, we a) work with point-cloud input, b) model fine-grained shape for parts (as opposed to approximating them as ellipsoids), and c) additionally incorporate 1DOF constraint for joints as is common to everyday objects (as opposed to a general 3DOF spherical joints for [39]). To the best of our knowledge, our work is the first to simultaneously achieves all three desiderata for reanimation: inferring parts, kinematic connectivity, and joint constraints.

3. Approach

Our goal is to infer the articulated structures, parts, and joint parameters jointly, given a point cloud video. Given as input a point cloud sequence $\mathcal{P} = \{\mathbf{P}^t\}_{t\in 1,\dots T}$, our goal is to produce an animatable kinematic model with part definitions, part connectivity, joint parameters, and joint states at each time step.

We first introduce the parameterized articulated model $\mathbf{\Gamma}$ in Sec. 3.1. We then cast estimation of this articulated model as an energy minimization problem (Sec. 3.2). It turns out that directly optimizing this energy is difficult, thus we design a two-stage *relax and project* approach to optimizing this energy (Sec. 3.3), as shown in Fig. 3. We first solve a more tractable relaxed problem (estimating a 6DOf piece-wise rigid model without any kinematic constraints, Sec. 3.3.1), project the solution to a valid kinematic tree (Sec. 3.3.2) and further optimize the 1DOF joint parameters (Sec. 3.3.3).

3.1. Articulated Model



Our kinematic model $\mathbf{\Gamma}$ comprises of n parts that are connected to one another. Each part $i \in [1 \dots n]$ (except the root joint) is connected to its parent part $\mathrm{pa}(i) \in \mathbf{\Gamma}$ via a 1-DOF joint. Each joint, is either be a revolute joint or a prismatic joint parameterized by screw parameters [36], $\mathbf{s}_i = \{\mathbf{l}_i, \mathbf{m}_i\}$, specifying where the joint is located ($\mathbf{m}_i$) and its axis ($\mathbf{l}_i$, rotation axis for revolute joints or translation axis for prismatic joints) with respect to the parent part.

The articulation is controlled by a set of joint state parameters $\boldsymbol{\theta}^t = \{\boldsymbol{\theta}_i^t\}_{i\in 1,\dots n}$ where each joint parameter $\boldsymbol{\theta}_i^t = (\tau_i^t, d_i^t)$ represents the rotation τ or translation d along its axis for joint i at time step t.

Parts are represented as a partition of Euclidean space, $\mathbb{R}^3$, at a *canonical* time step c, *i.e*. a coordinate-based semantic field $f(\mathbf{x})$ that maps points $\mathbf{x} \in \mathbb{R}^3$ to a part label in $[1 \dots n]$. We use a coordinate-based neural network to parameterize the part segmentation field. This completes the definition of the articulable model.

Given a set of state parameters $\boldsymbol{\theta}^t$, we can deform a point cloud in canononical frame to the target pose through piece-wise rigid transform:

$$M(\boldsymbol{\theta}^t; \mathbf{\Gamma}, f) = \{\mathbf{T}^t_{f(\mathbf{x})} \cdot \mathbf{x}\}_{\mathbf{x}\in\mathbf{P}^c} \tag{1}$$

where the rigid transformation of each part is computed through rigid pose composition along the kinematic chain from the part to root:

$$\mathbf{T}_i^t = \prod_{i'\in \mathrm{ans}(i)} \mathbf{T}(\mathbf{s}_{i'}, \boldsymbol{\theta}_{i'}^t) \tag{2}$$

and $\mathrm{ans} = \{i, \mathrm{pa}(i), \mathrm{pa}(\mathrm{pa}(i)), ...\}$ denotes the ordered set of ancestor nodes of i, which forms a kinematic chain.

Each joint's rigid transformation to its parent $\mathbf{T}(\mathbf{s}_i, \boldsymbol{\theta}_i^T)$

can be computed in closed-form using its screw parameters $(\mathbf{s}_i, \boldsymbol{\theta}_i^T)$ and Rodriguez formula [10]:

$$\mathbf{T}(\mathbf{s}, \boldsymbol{\theta}) = \text{Exp}([\boldsymbol{\omega}, \mathbf{v}]) = \text{Exp}([\tau\mathbf{l}; \tau\mathbf{m} \times \mathbf{l} + d\mathbf{l}]^T) \quad (3)$$

where $\boldsymbol{\omega} = \tau\mathbf{l} \in \mathbb{R}^3$ is the Euler vector representing rotation; $\mathbf{v} = \tau\mathbf{l} \times \mathbf{m} + d\mathbf{l}$ the translational vector; $\text{Exp} : \mathbb{R}^6 \mapsto \mathbb{SE}(3)$ maps the minimal 6-DoF representation to an $\mathbb{SE}(3)$ rigid transform; $\times$ is the cross product between vectors. There exists two special cases. A joint is called **prismatic joint** if its rotation angle is always zero, *i.e.* $\tau = 0$. In this case, the rigid body can only slide along the axis $\mathbf{l}$ without rotation (think about a drawer). A joint is called **revolute joint** if translational component always equals to 0, *i.e.* $d = 0$. In this case, the rigid body can only rotate along the rotation axis $\mathbf{l}$ (think about a door);

3.2. Energy Formulation

Our approach takes as input a point cloud sequence $\mathcal{P} = \{\mathbf{P}^t\}_{t\in 1,\dots T}$. The output of our approach is the number of parts n, the part labeling function f, a kinematic tree connecting the different parts $\boldsymbol{\Gamma}$, and screw parameters $\{\mathbf{s}_i\}$ for each non-root joint.

We estimate these parameters using an analysis-by-synthesis approach and find model parameters M, such that after appropriate per-time articulation $\boldsymbol{\theta}^t$, $M(\boldsymbol{\theta}^t; \boldsymbol{\Gamma}, f)$ matches the given point cloud $\mathbf{P}^t$ well.

$$\min_{n,f,\boldsymbol{\Gamma},\{\mathbf{s}_i\}} \sum_t \min_{\boldsymbol{\theta}_t} E_{\text{recons}}\left(M(\boldsymbol{\theta}^t; \boldsymbol{\Gamma}, f), \mathbf{P}^t\right), \quad (4)$$

where $M(\boldsymbol{\theta}^t; \boldsymbol{\Gamma}, f)$ denotes the canonical point cloud $\mathbf{P}^c$ transformed using the model parameters and the articulation $\mathbf{x}^t$ at time t.

E_{recons} measures the compatibility between the point cloud articulated according to the inferred model and the observed point cloud at each step. Specifically, the energy consists of three terms:

$$E_{\text{recons}} = \lambda_{\text{CD}} E_{\text{CD}} + \lambda_{\text{EMD}} E_{\text{EMD}} + \lambda_{\text{flow}} E_{\text{flow}} \quad (5)$$

Chamfer distance term: E_{CD} measures agreement between two point clouds using the Chamfer distance:

$$E_{\text{CD}}(\mathbf{X}, \mathbf{Y}) = \sum_{\mathbf{x}\in\mathbf{X}} \min_{\mathbf{y}} \|\mathbf{x}-\mathbf{y}\|_2^2 + \sum_{\mathbf{y}\in\mathbf{Y}} \min_{\mathbf{x}} \|\mathbf{x}-\mathbf{y}\|_2^2. \quad (6)$$

Earth-mover distance term: The earth-mover distance E_{EMD} also measures geometric compatibility. It finds the optimal assignment between the two points sets by solving a bipartite matching problem [24] and measure the residual loss:

$$E_{\text{EMD}}(\mathbf{X}, \mathbf{Y}) = \min_{\mathbf{S}} \|\mathbf{SX} - \mathbf{Y}\|_F^2, \quad (7)$$

where $\mathbf{S}$ is a permutation matrix that represents the bipartite assignment. Compare against E_{CD}, E_{EMD} could better capture details while E_{CD} only focus on point coverage. We compute the assignment via linear assignment solver [9].

Flow energy term: This term encourages the estimated 3D motion to be similar to observed point-wise 3D motion obtained from a scene flow network,

$$E_{\text{flow}} = \sum_{\mathbf{x}} \|\mathbf{x}^t - \mathbf{x}^{t-1} - g(\mathbf{P}^t, \mathbf{P}^{t-1}; \mathbf{x}^t)\|_2^2. \quad (8)$$

where $\mathbf{x}^t - \mathbf{x}^{t-1} = (\mathbf{T}^t_{f(\mathbf{x})} - \mathbf{T}^{t-1}_{f(\mathbf{x})}) \cdot \mathbf{x}$ is the predicted motion flow using the state for point $\mathbf{x}$ at time t and $t-1$. $g(\mathbf{P}^t, \mathbf{P}^{t-1}; \mathbf{x}^t)$ is the observed 3D motion flow between two point clouds $\mathbf{P}^t$ and $\mathbf{P}^{t-1}$, at the query point location $\mathbf{x}^t$. Since there is no guaranteed one-to-one mapping between $M(\boldsymbol{\theta}^t)$ and $\mathbf{P}^t$, we use trilinear interpolation to estimate the inferred flow at an arbitrary point $\mathbf{x}^t$:

$$g(\mathbf{P}^t, \mathbf{P}^{t-1}; \mathbf{x}^t) = \frac{\sum_{\mathbf{x}_k^t \in \text{knn}(\mathbf{x}^t; \mathbf{P}^t)} \|\mathbf{x}^t - \mathbf{x}_k^t\|^{-1} F(\mathbf{x}_k^t)}{\sum_{\mathbf{x}_k^t \in \text{knn}(\mathbf{x}^t; \mathbf{P}^t)} \|\mathbf{x}^t - \mathbf{x}_k^t\|^{-1}},$$

where $\mathbf{F}^t$ is the predicted flow between observation $\mathbf{P}^t$ and $\mathbf{P}^{t-1}$. Given the established flow $\mathbf{F}^t$ at locations in $\mathbf{P}^t$, $g(\cdot)$ takes query prediction $\mathbf{x}^t$ as input and returns an interpolated motion estimation at point $\mathbf{x}^t$. $\text{knn}(\mathbf{x}^t; \mathbf{P}^t)$ retrieves the K-nearest neighbors of $\mathbf{x}^t$ from $\mathbf{P}^t$.

3.3. Inference via Relax and Project

Directly optimizing over the set of model parameters as defined in Eq. (4) is difficult. It involves both discrete and continuous optimization variables as well as structured constraints such as the tree-structure of $\boldsymbol{\Gamma}$, making it hard for both numerical approaches and combinatorial methods. Hence we pursue an alternate "relax-and-project" approach.

3.3.1 Fitting a Relaxed Model

Our method first fits a *relaxed* model $\hat{M}$. This relaxed model doesn't constrain the parts to form a kinematic tree and thus lets individual parts to follow their own independent 6DOF trajectory over time. This relaxed model is parameterized via the number of parts n (same as for M), the part labeling function f (also same as for M). This model is articulated via a 6DOF pose for each part at each time step t: $\hat{\mathbf{T}}^t = \{\hat{\mathbf{T}}_i^t\}_{i\in[1\dots n]}$.

We first optimize this relaxed model using the same cost function as in Eq. (4) but evaluated for reposing under the relaxed model at each time step t via $\hat{\mathbf{T}}^t$:

$$\min_{n,f} \sum_t \min_{\mathbf{T}^t} E_{\text{recons}}\left(\hat{M}(\mathbf{T}^t), \mathbf{P}^t\right), \quad (9)$$

where $M(\hat{\mathbf{T}}^t)$ denotes the canonical point cloud $\mathbf{P}^c$ transformed using the model parameters and the per-time step part transformations $\mathbf{T}^t$ at time t.

The energy function defined in Eq. (5) involves a joint optimization over 6-DOF transformations $\hat{\mathbf{T}}_i^t$ and a neural segmentation field f. For each point, the part segmentation field outputs a discrete-valued index $f(\mathbf{x})$, which is later used to query the corresponding rigid transform to warp the point cloud as defined in Eq. (1). Optimizing through this

discrete index $f(\mathbf{x})$ is hard. We leverage Gumbel-softmax trick [20] along with the straight-through gardient estimator [3] to overcome this challenge. This makes the energy function end-to-end differentiable w.r.t. f and $\hat{\mathbf{T}}^t$, allowing us to use gradient descent to minimize the energy.

3.3.2 Projecting to the Kinematic Model

The estimated relaxed model parameters are projected onto a valid kinematic model by inducing a tree structured connectivity between the parts and projecting absolute 6DOF transformations into child-parent screw parameters. This is done using a cost function that measures the discrepancy between kinematic parameters $\mathbf{\Gamma}, \{\mathbf{s}_i\}, \{\boldsymbol{\theta}^t\}$ and relaxed 6DOF transformations $\{\hat{\mathbf{T}}^t\}$:

$$\min_{\mathbf{\Gamma}, \{\mathbf{s}_i\}, \{\boldsymbol{\theta}^t\}} E_{\text{project}} \left(\left(\mathbf{\Gamma}, \{\mathbf{s}_i\}, \{\boldsymbol{\theta}^t\} \right), \{\hat{\mathbf{T}}^t\} \right). \quad (10)$$

Given individual 6DOF part trajectories for all parts, we want to infer a kinematic model that is as similar to the relaxed model while obeying the kinematic constraints. The kinematic constraints consists of two aspects: the tree structure (*i.e.* the connectivity of parts with one another) and the 1DOF constraint for the joint between each pair of connected parts. We tackle the projection problem defined in Eq. (10) by minimizing the cost over different trees topologies $\mathbf{\Gamma}$ and associated screw parameters $\mathbf{s}_i = \{\mathbf{l}_i, \mathbf{m}_i\}$:

$$E_{\text{project}} = \lambda_{\text{spatial}} E_{\text{spatial}} + \lambda_{\text{1-DOF}} E_{\text{1-DOF}}. \quad (11)$$

This loss function evaluates the total fitness of parent-child pairs in $\mathbf{\Gamma}$. E_{spatial} measures the spatial proximity of parent-child pairs, $E_{\text{spatial}}(\mathbf{\Gamma}) = \sum_i \min_{\mathbf{x} \in \mathbf{p}_i} \min_{\mathbf{y} \in \mathbf{p}_{\text{pa}(i)}} \|\mathbf{x} - \mathbf{y}\|_2^2$, where $\mathbf{p}_i = \{\mathbf{x} \in \mathbf{P}^c | f(\mathbf{x}) = i\}$ is the set of points corresponding to part i. The $E_{\text{1-DOF}}$ term measures how close to a 1DOF motion is the motion of the child part relative to the parent part. $E_{\text{1-DOF}}$ is computed as the error in approximating the temporal sequence of relative transformation of the child part with respect to the parent part as a 1DOF transformation:

$$E_{\text{1-DOF}} = \sum_i \sum_t \text{trace}((\hat{\mathbf{T}}^t_{pa(i)} \ominus \hat{\mathbf{T}}^t_i) \ominus \mathbf{T}(\mathbf{s}_i, \boldsymbol{\theta}^t_i)), \quad (12)$$

where $\oplus$ is the rigid pose composition operator: $\mathbf{T}_a \oplus \mathbf{T}_b = \mathbf{T}_b \cdot \mathbf{T}_a$ and $\ominus$ is the inverse rigid pose composition operator $\mathbf{T}_a \ominus \mathbf{T}_b = \text{inv}(\mathbf{T}_b) \cdot \mathbf{T}_a$.

Solving the projection problem defined in Eq. (11) is also challenging: it's a joint optimization between discrete tree structure $\mathbf{\Gamma}$ and continuous screw parameters $\mathbf{s}_i, \boldsymbol{\theta}^t_i$; the desire for a valid tree topology $\mathbf{\Gamma}$ introduces a complicated structured constraint. We observe that screw parameters $\mathbf{s}_i, \boldsymbol{\theta}^t_i$ are only involved in exactly 1 1-DOF energy term under any valid tree. Thus, they can be independently optimized. For all part pairs (i, j), we compute screw parameters from $\hat{\mathbf{T}}_i$ and $\hat{\mathbf{T}}_j$ under the assumption that $j = \text{pa}(i)$ using screw theory [47]. This let us compute the 1DOF energy for all part pairs (i, j). The spatial term can also be similarly computed for all part pairs. The minimization in Eq. (11) then reduces to finding the minimum spanning tree which can be done efficiently [8].

Merging. Before the projection, we iteratively merge parts that are spatially close and don't have relative motion. The latter falls out directly from Eq. (12) if the approximation against the identity transform is below ϵ_m.

3.3.3 Final Fitting

We further finetune $\{\mathbf{s}_i\}$, $\{\boldsymbol{\theta}^t_i\}$ over the original problem in Eq. (4) with gradient descent while keeping f and $\mathbf{\Gamma}$ fixed. This gives the final estimation of tree structure $\mathbf{\Gamma}$, part segmentation f, screw parameters $\{\mathbf{s}_i\}$, and joint states $\{\boldsymbol{\theta}^t_i\}$.

Canonical frame selection. Given the non-convexity of the optimization, we run the optimization multiple times by choosing different time steps c as *canonical* frame to build neural part field. We pick the best c that has the lowest E_{project} in Eq. (10).

4. Experiments

We performing experiments on two 3D asset datasets and real-world setting, demonstrates our method could be applied on arbitrary articulated objects both qualitatively and quantitatively.

4.1. Experimental Setup

Datasets. We conduct experiments on two datasets: the *RoboArt* dataset that we introduce in this work, and the *Sapiens* dataset from [17]. The RoboArt dataset consists of 18 popular robots which span manipulator like panda robot to humanoid robots like nao. These robots have large variation in number of parts (7–15), part geometries (barrett robot's fingers to atlas robot's trunk), and part connectivity (linear chains to complex trees). We split the dataset in training (4 robots), validation (4 robots), and test (10 robots). For each robot we articulate them using [37] to record a 10 time-step points cloud sequence, each containing 4096 points. Crucially, we make sure that these 4096 points are randomly resampled at each time step to prevent any leakage of correspondence information between time steps. Supplementary material shows visualizations for different robots and summarizes more statistics. The Sapiens dataset from [17] contains 720 test sequences across 20 different object categories from Sapien [56]. The dataset provides 4 point cloud frames as input for each object. The 4 frames all have a different global coordinate frame.

Metrics. Our experiments measure the final reanimation error, intermediate metrics that evaluate part segmentation and inferred kinematics, and reconstruction metrics that evaluate how well our inferred models explain the input

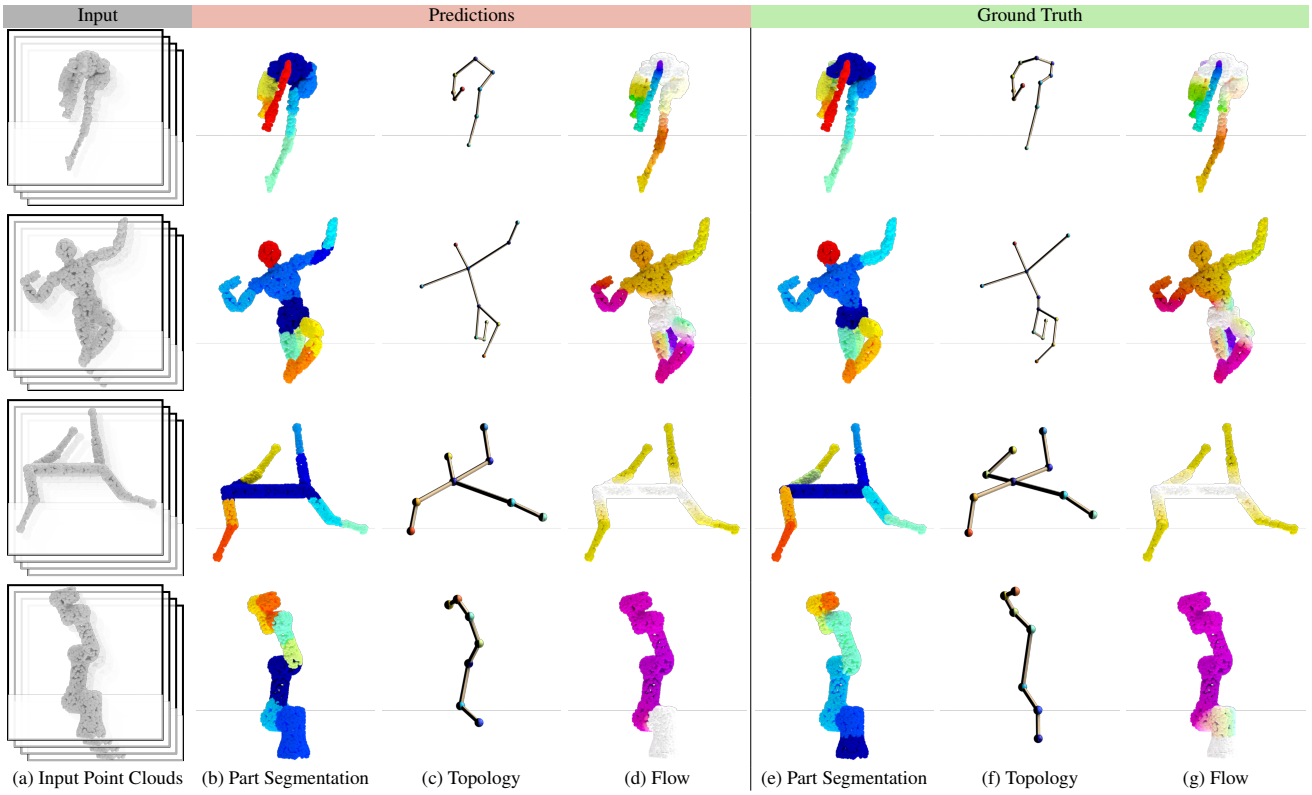


Figure 4. Qualitative Results on RoboArt Dataset. Given the input point cloud sequence (shown in (a)), we show the part segmentation, part connectivity, and implied flow using our inferred articulated model (in (b, c, d)) and the ground truth articulated model (in (e, f, g)) across 4 different type of robots. Our method can successfully deal with various parts connected in complex ways.

point clouds. We detail them next. **Reconstruction metrics** include a) Reconstruction Error that measures the mean end-point-error between the object articulated using the inferred articulation parameters *vs*. with the ground truth parameters, b) Flow Error that measures the error between the flow between consecutive frames implied by the predicted model *vs*. the ground truth model, and c) Flow Accuracy that measures the %age of points for which flow implied by the predictions is within a δ threshold from ground truth flow. **Intermediate metrics** include a) Random Index (RI) [6] that measures overlap between the partition induced by predicted part labels *vs*. the partition induced by ground truth part labels and b) Tree Edit Distance [41,65] that measures the similarity between predicted kinematic tree and the ground truth kinematic tree. These metrics have been used in past works, [17] and [57] respectively. **Reanimataion Metric:** Our inferred articulated model can be reanimated given new locations for a sparse set of points (1 per part). We measure the quality of such reanimation via the mean end-point-error between the ground truth reanimated model and the predicted reanimated model. Precise metric definition is provided in the supplementary.

Implementation Details. Our training requires us to compute the flow between pairs of frames. We train a Siamese correspondence network using PointNet++ [42]. At test time, we compute the similarity between each pair of points in the two point clouds and use correspondences that are mutually the best. The correspondence network is trained on the Sapiens and the RoboArt datasets and does not include any objects or robots that we evaluate upon (neither for validation nor for testing). **Part Segmentation Field** function f is realized using an MLP with one hidden layer of 128 dimensions. All optimizations are done using PyTorch. We use the standard Adam optimizer [27] for all optimization with a learning rate of 1e-3 for the MLP and 1e-2 for transformations. During **relaxed model estimation stage**, we set maximum number of parts to 20 and all 6DOF transformations are initialized to be identity. We found it helpful to only optimize with E_{CD} and E_{flow} for the first 5000 iterations, and replace E_{CD} by E_{EMD} and optimizing for another 10000 iterations. For efficiency reasons, we apply E_{EMD} on $4\times$ downsampled point clouds and only update the optimal assignment every 5 iterations. During **projection stage**, we first merge parts that are spatially close and don't move relative to one another with $\epsilon_m = 3e - 2$. During the final optimization stage, we only optimize the screw parameters and their states. We keep the number of parts and part segmentation fixed. Merging details and visualization can be found in supplementary.

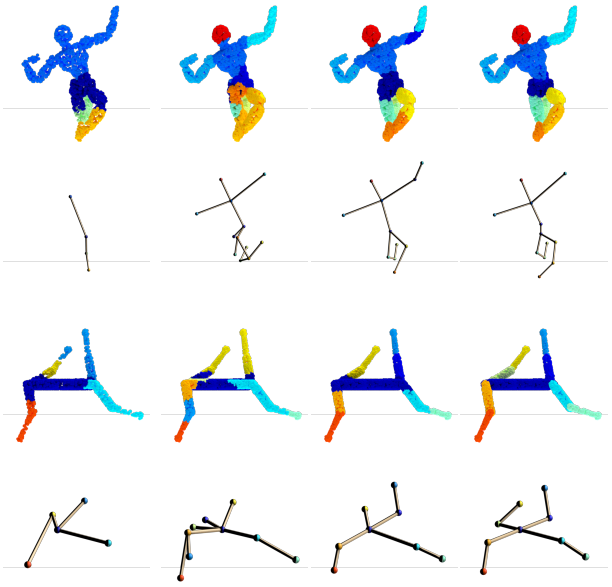


Figure 5. Qualitative comparison against MultiBodySync [17] and WatchItMove [39] on the RoboArt dataset. Note, a) MultiBodySync by itself doesn't produce a kinematic tree, we use our method on top of their output to generate one, and b) we provide WatchItMove [39] with the ground truth SDFs and number of parts (which are not used by our method). Even after these modifications, the past methods cannot solve the task as well as ours.

4.2. Articulation Object Modeling Results

Comparison on the RoboArt Dataset. As discussed in Sec. 2, to the best of our knowledge, there isn't a past work that exactly solves the entire task we tackle. We adapt and extend MultiBodySync [17] and WatchItMove [39] for our task and compare them on the RoboArt dataset. MultiBodySync [17] uses pairwise flow prediction and sets up a joint synchronization problem to obtain a temporally-consistent part segmentation. We use the part segmentation and pairwise flows to produce a kinematic tree and associated screw parameters using the projection part of our algorithm. WatchItMove [39] was designed for building articulable models from RGB videos. We modify it to work with point clouds. For point clouds, there is no rendering loss but only a loss on the predicted SDF. We provide ground truth SDFs and ground truth numbers of parts to their method.

Qualitative results on different types of robots are shown in Fig. 4. Our method can successfully deal with various parts connected in complex ways. Tab. 2 presents the quantitative results on the RoboArt. We outperform both these past methods by a large margin across all metrics. We looked into the poor performance of these past methods. MultiBodySync relies on accurate *pairwise* flow predictions. For objects with a long chain of articulation, *e.g.* a robot arm, over time, the object deforms quite a bit limiting the performance of the correspondence network. For WatchItMove, we believe poor performance comes from some their use of *soft* assignment of points to segments during training. This leads to incorrect assignment of points to parts at test time. In our own ablations (presented in

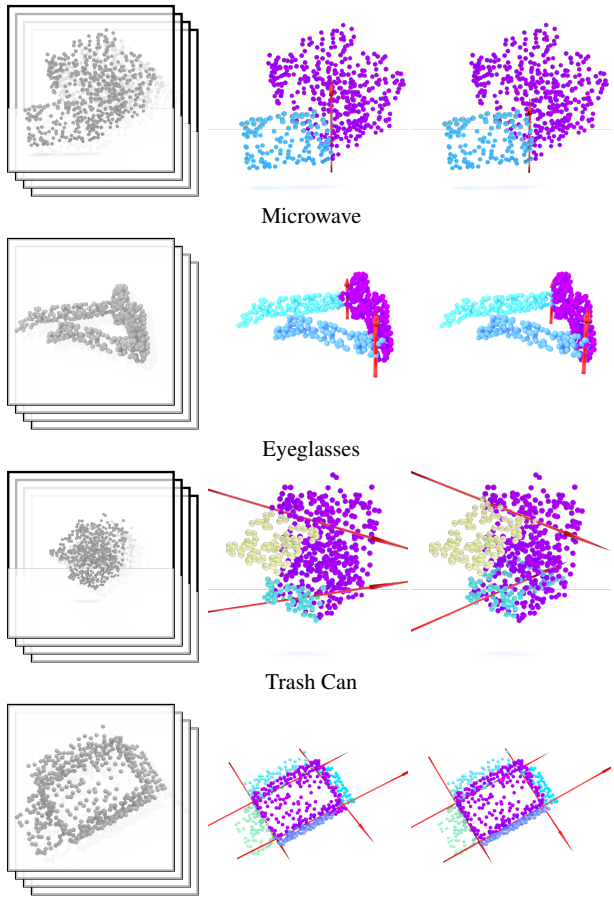


Figure 6. Qualitative results on Sapiens dataset from [17]. We visualize the predicted and ground truth articulated models. We show results on 4 daily objects with different number of parts and articulation types, a microwave (single revolute), eyeglasses (two revolute), trash can (a prismatic and a revolute) and a box (4 revolute). Different parts are in different colors, and we also show the screw parameters (in red) for the inferred joints.

Table 2. Comparison to past methods on RoboArt test set. We outperform past methods on all metrics that measure input reconstruction quality (shape reconstruction error, flow error, flow accuracy), model accuracy (part segmentation rand index, and kinematic tree edit distance), and reanimiation error.

Method	Recons. error ↓	Flow error ↓	Flow acc. ↑	Rand Index ↑	Tree Edit Distance ↓	Reanimate Error ↓
MultiBodySync [17]	4.76	3.42	0.26	0.70	6.6	9.72
WatchItMove [39]	12.77	8.42	0.06	0.78	6.6	7.43
Ours	**1.26**	**0.57**	**0.59**	**0.86**	**2.9**	**3.66**

Table 3. Comparison to past methods on Sapiens dataset [17]. We do better in both flow estimation and segmentation under two different evaluation settings.

Method	Flow Error ↓	Multi-scan RI↑	Per-scan RI↑
PWC-Net [54]	6.20	-	-
PointNet++ [42]	-	0.62	0.65
MeteorNet [33]	-	0.59	0.60
Deep Part [61]	5.95	0.64	0.67
NPP [13]	21.22	0.63	0.66
MultiBodySync [17]	5.03	0.76	0.77
Ours	**4.79**	**0.79**	**0.79**

Table 4. Role of different energy terms in Eq. (5) evaluated on the RoboArt validation set. To isolate the direct impact of the energy terms, we conduct this experiment w/o the canonical frame selection (we always use the middle frame) and w/o screw parameter finetuing after projection to a kinematic model. All three terms are important with the flow term being the most important.

E_{CD}	E_{flow}	E_{EMD}	Recons. error ↓	Flow error ↓	Flow acc. ↑	Rand Index ↑	Tree Edit Distance ↓
✓			6.97	8.05	0.31	0.76	6.50
	✓		2.97	1.64	0.31	0.83	5.00
		✓	3.49	3.03	0.13	0.28	6.75
✓	✓		1.47	0.88	**0.48**	0.83	4.00
✓		✓	1.93	1.99	0.17	**0.86**	4.25
	✓	✓	1.54	0.97	0.33	0.83	5.25
✓	✓	✓	**1.31**	**0.86**	0.40	**0.86**	**3.25**

Table 5. Evaluation of other design choices on the RoboArt validation set. Neural segmentation field (**f**), Gumbel-softmax (**Gumbel**), projection to the kinematic model (**Project**), canonical frame selection (**Cano**) all contribute to the final performance.

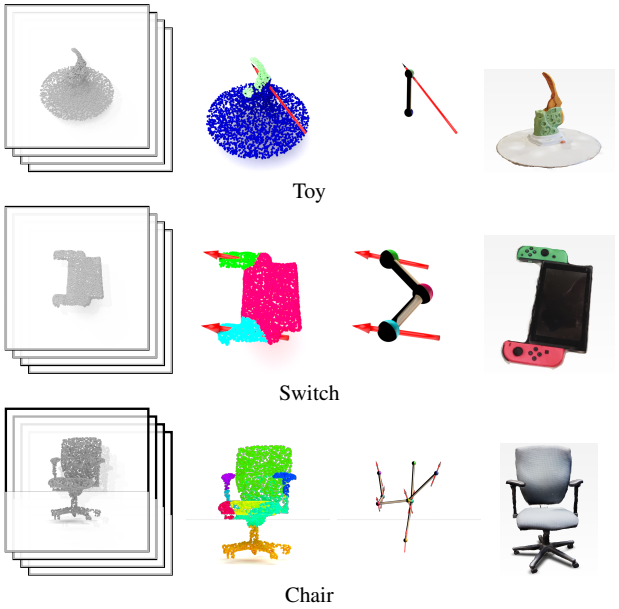
Designs	Recons. error ↓	Flow error ↓	Flow acc. ↑	Rand Index ↑	Tree Edit Distance ↓	Reanimate Error ↓
w/o **f**	2.63	1.26	0.40	0.76	7.25	11.68
w/o **Gumbel**	3.54	1.65	0.34	0.74	7.25	11.69
w/o **Project**	1.25	0.75	0.45	**0.88**	**3.00**	8.86
w/o **Cano**	1.54	0.74	0.47	0.85	3.33	6.83
ours	**1.19**	**0.64**	**0.49**	**0.88**	**3.00**	6.50

Table 6. Testing performance between prismatic *vs*. revolute joints on Sapiens dataset. Prismatic joints could be harder to predict than revolute joints.

Method	Flow Error ↓	Multi-scan RI↑	Per-scan RI↑
Prismatic	4.80	0.64	0.64
Revolute	4.78	0.80	0.80

(a) Input (b) w/o kinematics (c) w/ kinematics (d) GT

Figure 7. Reanimation Results on the RoboArt Dataset. Given new locations for a sparse set of points on the object(shown in (a)), our full method (shown in (c)) is able to generate a reasonable reanimation to match the specified points. (b) shows reanimation results from an ablated version of our model where we don't restrict the parts to form a kinematic tree. This results in poor reanimation. Thus, correctly inferring the connectivity of different parts with one another is crucial for high-quality reanimation.



(a) Input (b) Segmentation (c) Kinematics (d) Real scan

Figure 8. Qualitative results in real-world setting. We verify our method on three daily objects in each row, a toy (single revolute), a switch (two prismatic) and a chair (multiple revolute and prismatic). Each row shows in the input, part segmentation, part connectivity and screw parameters (in red) for the inferred joints, and the real scan captured by scanning apps of iPhone. Our method is robust to noise and partial observations. Chair cushion gets slightly over-seg due to noisy surface.

Sec. 4.3) we note that our hard assignment of points to segments during training is crucial to the final performance. Fig. 5 show qualitative comparisons.

Comparison on Sapiens Dataset. We also compare performance on specific sub-tasks (part segmentation and flow prediction) useful for re-animation as studied in past work [17] on their Sapiens dataset in Tab. 3. Here, we also compare a number of past methods, including PWC-Net [54], PointNet++ [42], *etc*. Please refer to [17] for details about these methods. We outperform all these past methods. We believe our strong performance on this dataset is because of the additional joint and kinematic constraints enforced by our method, which are not used by these past

methods. See qualitative results in Fig. 6 on objects from Sapiens dataset with different number of parts and articulation types (revolute and prismatic). For a fair comparison, we use the flow network from [17] for our method.

4.3. Ablations

Tab. 4 evaluates the importance of the different energy terms in our formulation on RoboArt validation set. We note that all terms are important, but particularly E_{flow} has the largest effect. We also find that just the chamfer distance term, E_{CD}, is insufficient and works poorly. We believe this is because E_{CD} doesn't provide good gradients (it tries to snap a point to the closest point in the shape, which may not necessarily be the correct location for it).

Tab. 5 reports the impact of our other design choices. Representing the part segmentation as a neural field induces a smoothness prior on the parts. Removing this prior by optimizing each point' segmentation logits independently works worse (Row 1: w/o **f**). Taking Gumbel-softmax as hard assignment during training is important. Soft assignment (softmax) of points to transformations provide a lot more freedom for optimization. This prevents learning of the correct transformations, which don't work when used with hard assignments at test time (Row 2, w/o **Gumbel**). Projecting relaxed model to valid kinematic one leads to better reconstruction quality and lower re-animation error by further constraint the transformations and regularize the motions (Row 3, w/o **Project**). We also found that the choice of frame for building the neural part field is important. Sometimes two otherwise far away parts can be close to one another in some frame (*e.g.* left and right arms of a humanoid) making it difficult for the part segmentation field to separate them. Searching for which frame to use as the canonical frame helps in over-coming the non-convexity in the optimization (Row 4, w/o **Cano**).

Re-animation Results. We now showcase how the reconstructed rearticulatable model can be retargeted to new poses. Given an inferred model consisting of part segmentation, kinematic tree, and 1DoF joint parameters, we use inverse kinematics (IK) to compute the transformation between our canonical point frame and a target pose, using new locations for a sparse set of points on the object. We then re-animate the point cloud through the inferred joint parameters. Fig. 7 shows the results. We see that the re-animation quality significantly improves by inferring the kinematic linking of parts.

Prismatic joints. We compare testing performance between prismatic vs. revolute joints on Sapiens dataset. The results are shown in Tab. 6. Compare against revolute joints, prismatic joints could be harder to predict. The reasons include 1) less training samples; 2) segmentation is hard between base part and cluttered prismatic part. Qualitative comparisons of prismatic joints against revolute joints are shown in Fig. 6 and Fig. 8 under real-world setting.

Run-time. On RoboArt sequences, ours takes 45 minutes on a RTX 3090 GPU while WatchItMove takes 2 hours. On Sapiens, ours takes 5min (same as MultiBodySync).

4.4. Real-World Experiments

We verified our method on real world scans with diverse articulations and kinematic structures. We choose three daily objects, a toy (single revolute), a switch (two prismatic) and a chair (multiple revolute and prismatic) and reconstruct their geometry. Each object has five articulation states. The scans are gathered using scanning apps on iPhone. The results are shown in Fig. 8. Our method is robust to noise and partial observations. Chair cushion gets slightly over-segmentation due to noisy surface.

4.5. Limitation.

Our method heavily relies on motion cues. It might fail to distinguish two rigid parts if they undergo the same motion in the entire sequence (*e.g.* a humanoid robot moves both arms synchronously). We leave this to future work and plan to tackle it by incorporating appearance information.

5. Conclusion

We presented a novel method for building arbitrary rearticulable models from a point cloud sequence. Our approach jointly infers part segmentation, screw-parametric joints, and kinematic tree structures in a category-agnostic manner. We validated our method's efficacy on two challenging datasets, showing superior results over previous leading works. We further showed that the inferred model could be retargeted at any novel pose, demonstrating the potential for reanimation and manipulation.

Acknowledgements: This material is based upon work supported by an NSF CAREER Award (IIS-2143873) and the USDA/NSF AIFARMS National AI Institute (USDA #2020-67021-32799). SW is supported, in part, by Amazon research award and Insper innovation grant.

References

[1] Ben Abbatematteo, Stefanie Tellex, and George Konidaris. Learning to generalize kinematic models to novel objects. In *CoRL*, 2019. 2

[2] Hameed Abdul-Rashid, Miles Freeman, Ben Abbatematteo, George Konidaris, and Daniel Ritchie. Learning to infer kinematic hierarchies for novel object instances. In *ICRA*. IEEE, 2022. 2

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, 2013. 5

[4] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. 14

[5] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic faust: Registering human bodies in motion. In *CVPR*, pages 6233–6242, 2017. 14, 15

[6] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3d mesh segmentation. *TOG*, 28(3), 2009. 6

[7] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *ICCV*, 2019. 3

[8] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022. 5

[9] David F Crouse. On implementing 2d rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 52(4), 2016. 4

[10] Jian S Dai. Euler–rodrigues formula variations, quaternion conjugation and intrinsic connections. *Mechanism and Machine Theory*, 92, 2015. 4

[11] Andrea F Daniele, Thomas M Howard, and Matthew R Walter. A multiview approach to learning articulated motion models. In *Robotics Research*. Springer, 2020. 2

[12] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. Factored pose estimation of articulated objects using efficient nonparametric belief propagation. In *ICRA*. IEEE, 2019. 2

[13] David S Hayden, Jason Pacheco, and John W Fisher. Nonparametric object and parts modeling with lie group dynamics. In *CVPR*, 2020. 2, 8

[14] Eric Heiden, Ziang Liu, Vibhav Vineet, Erwin Coumans, and Gaurav S Sukhatme. Inferring articulated rigid body dynamics from rgbd video. *arXiv*, 2022. 2

[15] Ruizhen Hu, Lubin Fan, and Ligang Liu. Co-segmentation of 3d shapes via subspace clustering. In *Computer graphics forum*, volume 31. Wiley Online Library, 2012. 3

[16] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. Learning to predict part mobility from a single static snapshot. *TOG*, 36(6), 2017. 2

[17] Jiahui Huang, He Wang, Tolga Birdal, Minhyuk Sung, Federica Arrigoni, Shi-Min Hu, and Leonidas J Guibas. Multibodysync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *CVPR*, 2021. 2, 3, 5, 6, 7, 8, 9, 13, 15, 16, 20

[18] Xiaoxia Huang, Ian Walker, and Stan Birchfield. Occlusion-aware reconstruction and manipulation of 3d articulated objects. In *ICRA*. IEEE, 2012. 2

[19] 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)*. IEEE, 2021. 2, 3

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *ICLR*, 2017. 5

[21] Hanxiao Jiang, Yongsen Mao, Manolis Savva, and Angel X Chang. Opd: Single-view 3d openable part detection. *arXiv*, 2022. 2

[22] Zhenyu Jiang, Cheng-Chun Hsu, and Yuke Zhu. Ditto: Building digital twins of articulated objects from interaction. In *CVPR*, 2022. 2, 3

[23] Gunnar Johansson. Visual perception of biological motion and a model for its analysis. *Perception & psychophysics*, 14, 1973. 1

[24] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4), 1987. 4

[25] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 14

[26] Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. Unsupervised pose-aware part decomposition for 3d articulated objects. *arXiv*, 2021. 2

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014. 6

[28] Suren Kumar, Vikas Dhiman, Madan Ravi Ganesh, and Jason J Corso. Spatiotemporal articulated models for dynamic slam. *arXiv*, 2016. 2

[29] Hao Li, Guowei Wan, Honghua Li, Andrei Sharf, Kai Xu, and Baoquan Chen. Mobility fitting using 4d ransac. In *Computer Graphics Forum*, volume 35. Wiley Online Library, 2016. 2

[30] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *CVPR*, 2020. 2

[31] Qihao Liu, Weichao Qiu, Weiyao Wang, Gregory D Hager, and Alan L Yuille. Nothing but geometric constraints: A model-free method for articulated object pose estimation. *arXiv*, 2020. 3

[32] Xueyi Liu, Xiaomeng Xu, Anyi Rao, Chuang Gan, and Li Yi. Autogpart: Intermediate supervision search for generalizable 3d part segmentation. In *CVPR*, 2022. 3

[33] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, 2019. 8

[34] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *TOG*, 34(6), 2015. 2

[35] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 12

[36] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017. 3

[37] Matthew Matl. Urdfpy. `https://github.com/mmatl/urdfpy`, 2019. 5

[38] 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 *ICCV*, 2021. 2

[39] Atsuhiro 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 *CVPR*, 2022. 2, 3, 7, 13, 15, 16

[40] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 2

[41] Mateusz Pawlik and Nikolaus Augsten. Efficient computation of the tree edit distance. *ACM Transactions on Database Systems (TODS)*, 40(1), 2015. 6

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30, 2017. 6, 8, 12

[43] Shengyi Qian, Linyi Jin, Chris Rockwell, Siyi Chen, and David F Fouhey. Understanding 3d object articulation in internet videos. In *CVPR*, 2022. 3

[44] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *SIGGRAPH Asia*, 2017. 2

[45] Andrei Sharf, Hui Huang, Cheng Liang, Jiapei Zhang, Baoquan Chen, and Minglun Gong. Mobility-trees for indoor scenes manipulation. In *Computer Graphics Forum*, volume 33. Wiley Online Library, 2014. 2

[46] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *SIGGRAPH Asia*, 2011. 2, 3

[47] Stefano Stramigioli and Herman Bruyninckx. Geometry and screw theory for robotics. *Tutorial during ICRA*, 2001, 2001. 5

[48] Dimitrios Tzionas and Juergen Gall. Reconstructing articulated rigged models from rgb-d videos. In *ECCV*. Springer, 2016. 3

[49] Oliver Van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. Co-hierarchical analysis of shape structures. *TOG*, 2013. 3

[50] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, 2018. 3

[51] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *CVPR*, 2019. 3

[52] Fangyin Wei, Rohan Chabra, Lingni Ma, Christoph Lassner, Michael Zollhöfer, Szymon Rusinkiewicz, Chris Sweeney, Richard Newcombe, and Mira Slavcheva. Self-supervised neural articulated shape and appearance models. In *CVPR*, 2022. 2

[53] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J Guibas. Captra: Category-level pose tracking for rigid and articulated objects from point clouds. In *ICCV*, 2021. 2

[54] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *ECCV*, 2020. 8

[55] Yuefan Wu, Zeyuan Chen, Shaowei Liu, Zhongzheng Ren, and Shenlong Wang. Casa: Category-agnostic skeletal animal reconstruction. In *NeurIPS 35 (NeurIPS)*, 2022. 2

[56] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *CVPR*, 2020. 5

[57] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv*, 2020. 2, 6, 14

[58] Han Xue, Liu Liu, Wenqiang Xu, Haoyuan Fu, and Cewu Lu. Omad: Object model with articulated deformations for pose estimation and retrieval. *arXiv*, 2021. 2

[59] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. Rpm-net: recurrent prediction of motion and parts from point cloud. *arXiv*, 2020. 3

[60] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. Banmo: Building animatable 3d neural models from many casual videos. In *CVPR*, 2022. 2

[61] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *arXiv*, 2018. 2, 3, 8

[62] Qing Yuan, Guiqing Li, Kai Xu, Xudong Chen, and Hui Huang. Space-time co-segmentation of articulated point cloud sequences. In *Computer Graphics Forum*, volume 35. Wiley Online Library, 2016. 3

[63] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, pages 1592–1599, 2015. 12

[64] Vicky Zeng, Tabitha Edith Lee, Jacky Liang, and Oliver Kroemer. Visual identification of articulated object parts. In *IROS*. IEEE, 2021. 3

[65] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6), 1989. 6

[66] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

Appendix

The supplementary material provides implementation details, additional results and details of RoboArt dataset to support the main paper. In summary, we include

- Appendix A. **Implementation details.** Implementation details include flow prediction network, kinematic model projection, Merging, final fitting, canonical frame selection, optimization details, rearticulation, baseline implementations and evaluation metrics.
- Appendix B **Details of RoboArt dataset.**
- Appendix C. **Additional results.** Additional quantitative and qualitative results on Sapiens and RoboArt datasets, and results beyond 1 DOF joints.

A. Implementation details

A.1. Flow Prediction Network

The flow $\mathbf{F}^t$ is computed between pairs of frames $\mathbf{P}^t$ and $\mathbf{P}^{t-1}$ and used in the flow energy term in Sec. 3.2, where $\mathbf{F}^t = F(\mathbf{P}^t)$. To ensure good generalization, we first establish correspondence between input frames $\mathbf{P}^t$ and $\mathbf{P}^{t-1}$ via a correspondence network and induce the flow by correspondence.

Built upon PointNet++ [42] MSG segmentation network, the correspondence network takes point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$ as input and outputs a point-wise feature map $\mathbf{f}(\mathbf{P})$. Given two features map $\mathbf{f}(\mathbf{P}^{t-1})$ and $\mathbf{f}(\mathbf{P}^t) \in \mathbb{R}^{N\times d}$, where $d = 64$ is the feature dimension. We compute matching score matrix $\mathbf{S}(\mathbf{P}) \in \mathbb{R}^{N\times N}$:

$$\mathbf{S}(\mathbf{P}) = \text{softmax}(\frac{1}{\sqrt{d}}\mathbf{f}(\mathbf{P}^{t-1}) \cdot \mathbf{f}(\mathbf{P}^t)^T)$$

Each column of $\mathbf{S}(\mathbf{P})$ represents the probability of matching point $\mathbf{x}^t \in \mathbf{P}^t$ into a point in $\mathbf{P}^{t-1}$. At inference, Given the query $\mathbf{x}^t$, we find the matching point $\mathbf{x}^{t-1}_{match} \in \mathbf{P}^{t-1}$ by taking the argmax position in corresponding column of $\mathbf{S}(\mathbf{P})$. Then the induced flow at location $\mathbf{x}^t$ is given by $F(\mathbf{x}^t) = \mathbf{x}^t - \mathbf{x}^{t-1}_{match}$. To ensure the induced flow achieves high quality, we filter out spurious correspondences by applying mutual nearest neighbors (MNN) criteria, which guarantees the match falls into each other's nearest neighbor.

We train the correspondence network by minimizing the the contrastive cross-entropy loss [35, 63]. Each point in $\mathbf{P}^{t-1}$ is treated as one class, and the ground truth label is computed as the nearest neighbor of a point $\mathbf{x}^t$ in $\mathbf{P}^{t-1}$ when $\mathbf{P}^t$ and $\mathbf{P}^{t-1}$ are aligned. We train the correspondence network under cross-entropy loss between predicted scores $\mathbf{S}(\mathbf{P})$ and ground-truth labels $\mathbf{S}^*(\mathbf{P})$:

$$\mathcal{L}_{\text{corr}} = -\sum_{j=1}^{N} \mathbf{S}^*(\mathbf{P}_j) log \mathbf{S}(\mathbf{P}_j)$$

A.2. Projecting to the Kinematic Model

The projection from estimated relaxed model to the valid kinematic model is achieved by minimizing the cost over E_{project}, which consists of a spatial term E_{spatial} and the 1-DOF motion term $E_{\text{1-DOF}}$, we explain each term in details.

E_{spatial}. If two parts are linked, they should be close in 3D space. The E_{spatial} measures the spatial proximity of the parent-child pair $\text{pa}(i)$ and i in canonical frame $\mathbf{P}^c$. We query the part segmentation field f and extract the corresponding part segmentation points of parent and child $\mathbf{p}_{\text{pa}(i)} = \{\mathbf{x} \in \mathbf{P}^c | f(\mathbf{x}) = \text{pa}(i)\}$ and $\mathbf{p}_i = \{\mathbf{x} \in \mathbf{P}^c | f(\mathbf{x}) = i\}$. The $E_{\text{spatial}}(i, \text{pa}(i)) = \min_{\mathbf{x}\in\mathbf{p}_i} \min_{\mathbf{y}\in\mathbf{p}_{\text{pa}(i)}} \|\mathbf{x} - \mathbf{y}\|_2^2$. To improve efficiency, we do farthest point sampling from $\mathbf{p}_{\text{pa}(i)}$ and $\text{pa}(i)$ and sample 20 points per part to represent the set. We compute the $E_{\text{spatial}}(i, \text{pa}(i))$ for all part pairs in parallel.

$E_{\text{1-DOF}}$. In articulated objects, if two parts are linked, their relative transformation should be explained by a 1-DOF screw joint. The $E_{\text{1-DOF}}$ in Eq. (12) computed the approximation error for the temporal sequence of relative transformation between parent $\text{pa}(i)$ and child i treating as a 1DOF transformation. The relative transformation sequence is computed as $\{\hat{\mathbf{T}}^t_{pa(i)} \ominus \hat{\mathbf{T}}^t_i\}_{t\in 1,\dots T}$ between parent $\text{pa}(i)$. We compute the approximated screw parameters $\mathbf{s}_i, \{\boldsymbol{\theta}^t\}$ by the following objective:

$$\mathbf{s}_i, \{\boldsymbol{\theta}^t\} = \underset{\mathbf{s}_i, \{\boldsymbol{\theta}^t\}}{\arg\min} \left(\sum_t \text{trace}((\hat{\mathbf{T}}^t_{pa(i)} \ominus \hat{\mathbf{T}}^t_i) \ominus \mathbf{T}(\mathbf{s}_i, \boldsymbol{\theta}^t_i)) \right).$$

We solve the above for all part pairs i and $\text{pa}(i)$. The residual error is taken as $E_{\text{1-DOF}}(i, \text{pa}(i))$.

A.3. Merging.

To make the kinematic topology compact, we merge parts that are close in space with small relative motion. The static joint is a special case of the 1-DOF screw joint, where the rotation and translation component both equal to 0. Similar to $E_{\text{1-DOF}}$, we define $E_{\text{merge}}(i, \text{pa}(i)) = \sum_t \text{trace}((\hat{\mathbf{T}}^t_{pa(i)} \ominus \hat{\mathbf{T}}^t_i) \ominus \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. We merge pair $\text{pa}(i)$ and i if $E_{\text{merge}}(i, \text{pa}(i)) < \epsilon_m$, meaning their relative motion is small. The merging is done iteratively. We start from the part pair $\text{pa}(i)$ and i with the lowest E_{spatial} and stop merging until all remained part pairs have $E_{\text{merge}} \geq \epsilon_m$. In Fig. 9, we show segmentation of real-world switch before and after merging step.

A.4. Final Fitting.

After projection and merging, we obtain a valid kinematic model $\boldsymbol{\Gamma}, \{\mathbf{s}_i\}, \{\boldsymbol{\theta}^t\}$, we infer the joint type (revolute or prismatic) for part i and its parents by check $\{\boldsymbol{\theta}^t_i\}_{t\in 1,\dots T}$,

where $\boldsymbol{\theta}_i^t = (\tau_i^t, d_i^t)$. As dicussed in Sec. 3.1, the rotation angle of a prismatic joint is always zero, *i.e.* $\{\tau_i^t = 0\}_{t\in 1,\dots T}$, while the translational component always be 0 for a revolute joint, *i.e.* $\{d_i^t = 0\}_{t\in 1,\dots T}$. We compute the mean $\bar{\tau}_i = \sum_{t=1}^{T} \tau_i^t$ and $\bar{d}_i = \sum_{t=1}^{T} d_i^t$ for part i. If $\bar{\tau}_i < \bar{d}_i$, we treat the joint between i and its parent pa_i as a prismatic joint, otherwise as a revolute joint. In final fitting stage, we ensure all the joints fall into these two classes and keep $\{\tau_i^t = 0\}_{t\in 1,\dots T}$ for prismatic joint and $\{d_i^t = 0\}_{t\in 1,\dots T}$ for revolute joint during optimization.

A.5. Canonical Frame Selection.

Our algorithm is flexible in taking arbitrary frames in the input sequence as the canonical frame c. Certain frames could make part segmentation field more easily separating different parts, *e.g.* if two rigid parts undergo some similar motion throughout the entire sequence, certain frames could better capture those subtle differences and gives better segmentation result. Thus, we develop a criteria for selecting best canonical frame within the input sequence. We pick the canonical frame by selecting the one with lowest E_{project} + E_{group}. E_{project} is the same defined in Eq. (11). E_{group} is used to measure the deviation of each cluster in the segmentation field. For each part $i \in [1 \dots n]$, point segmentation cluster $\mathbf{p}_i = \{\mathbf{x} \in \mathbf{P}^c | f(\mathbf{x}) = i\}$, we compute the cluster center $\mathbf{c}_i = \frac{1}{|\mathbf{p}_i|} \sum_{\mathbf{x}\in\mathbf{p}_i} \mathbf{x}$, the E_{group} is computed as:

$$E_{\text{group}} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathbf{p}_i|} \sum_{\mathbf{x}\in\mathbf{p}_i} (\mathbf{x} - \mathbf{c}_i)^2$$

A.6. Optimization Details

We set $\lambda_{\text{CD}} = 1.0$, $\lambda_{\text{EMD}} = 0.3$, and $\lambda_{\text{flow}} = 1.0$ for E_{recons} in Eq. (5). Those parameters are tuned on validation set and fixed for all testing samples. We use knn $= 3$ for flow trilinear interpolation. We set $\lambda_{\text{spatial}} = 100$ and $\lambda_{\text{1-DOF}} = 1.0$ in Eq. (11), merging threshold $\epsilon_m = 3e-2$. In relaxed model estimation stage, We optimize the model for 15,000 iterations, E_{EMD} is applied on $4\times$ downsampled point clouds and updated every 5 iterations. We use a cosine annealing schedule anneal for Gumbel-softmax temperature. It start from 5.0 and decay 1.0 in the last iteration. In final optimization stage, we optimize the model for 200 iterations, E_{EMD} is applied on $2\times$ downsampled point clouds and updated every iteration.

A.7. Rearticulation

We can re-articulate our predicted model to a given target pose by only given a sparse set of point locations (Figure. 1). Given the source points, We use the part segmentation field to infer part labels and use forward kinematics in Eq. (2) of the model $M(\boldsymbol{\theta}^t; \boldsymbol{\Gamma}, f)$ to deform those points to match target points. We fix $\boldsymbol{\Gamma}, f$ and only optimize the joint state parameters $\boldsymbol{\theta}$ for 200 iterations with learning rate 0.1. We optimize the MSE loss between the deformed points and provided target points $\mathbf{P}'$.

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{mse} \left(M(\boldsymbol{\theta}; \boldsymbol{\Gamma}, f), \mathbf{P}'\right)$$

A.8. Implementation of the Baselines

We describe the implementation of baselines MultiBodySync [17] and WatchItMove [39].

MultiBodySync. MultiBodySync synchronizes between all $\binom{T}{2}$ states in the input sequence of length T and iteratively refinem the motion prediction and segmentation. The method requires the eigen-decomposition of a Laplacian matrix with size $\mathbb{R}^{NT\times NT}$, N is the number of points at each state. When input sequence becomes long, the matrix computation becomes the bottleneck of the method and could very hard to fit into the memory. To this end, we $2\times$ downsampled input point clouds to 2048 points as input. MultiBodySync estimates the number of parts by analysing the spectrum of predicted motion segmentation matrices and counting the number of eigenvalues larger than a cutting threshold. We found out this strategy works well on Sapiens dataset, but performs poorly on RoboArt dataset given the more complicated part motions controlled by the kinematic tree. We choosing the cutting threshold among $[0.05, 0.005, 0.001]$ and choose the best one which is 0.001 on the validation set. We also increase the number of iterations from 4 to 6 for better iterative refinement. However, we found out the method still severely suffer from missing parts and wrong motion prediction as shown in Fig. 12. The method requires pairwise flow prediction, this could be extremely challenging in robot case with large deformations between the start and the end of a long sequence.

WatchItMove. The original WatchItMove takes as input multi-view RGB videos with strong cues on both geometry and appearance. To apply WatchItMove to our setting with the 4D point cloud, we adjust their implementation * with two major changes: 1) Replace the photometric reconstruction loss with SDF $\mathcal{L}_1$ loss, where the label comes from ground-truth signed distance field; 2) We use the ground-truth # of rigid components. Both changes give certain levels of advantage to WatchItMove. However, the results demonstrated that motion cue is indispensable. Without motion cues, there is no constraint to regularize the ellipsoids motion. Though the overall shape could match to the input and SDF loss could be minimized, those ellipsoids could move with random motion internally. The result also justify the importance of hard assignment of points to segments during training. Instead of using hard assignment,

*https://github.com/NVlabs/watch-it-move

(a) Real scan (b) Before merging (c) After merging

Figure 9. Visualization of merging. We show the segmentation of real-world switch before and after merging step.

WatchItMove uses soft assignment during training. The motion of a certain point is blended by all ellipsoid motions. At inference, we require each point follow one corresponding ellipsoid motion by taking the argmax of segmentation weights from all parts. The inconsistency between training and testing hinders the motion estimation performance. We also note that it is crucial to incorporate the 1-DOF constraint when building kinematic tree. Without considering the constraint could result in unmeaningful linkage as shown in Fig. 12.

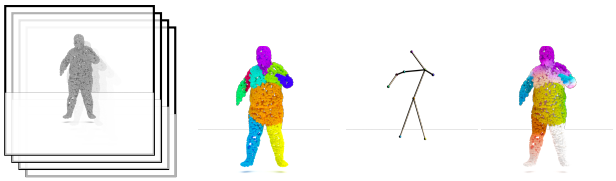
A.9. Evaluation Metrics

We discuss the reconstruction metrics, intermediate metrics and reanimataion metric in more details.

Reconstruction Metrics. We reconstruct the input sequence using our built animatable model $M(\boldsymbol{\theta}^t; \boldsymbol{\Gamma}, f)$. We measure the per-point reconstruction error across all time steps T. The flow is computed between the canonical frame and all reconstructions in the sequence. For flow accuracy, we set the treshold $\delta = 0.005$ on RoboArt dataset and $\delta = 0.05$ on Sapiens dataset.

Intermediate Metrics. The tree edit distance is the minimal-cost sequence of node edit operations to turn the predicted tree into ground-truth. The three allowed operations are delete, insert and rename. Follow [57], we set rename cost to be 0 and all other two operations cost to be 1. Given the predicted kinematic tree is undirected, we traverse all possible orders of the tree and select the minimum one as the final metric.

Reanimataion Metric. Given the ground-truth point cloud in a novel frame, we sample one pair of correspondence per-part between the canonical frame and novel frame, which guarantees the novel part poses is impossible to recovered from ICP [4] or Kabsch [25] algorithm. We use the provided sparse correspondences and algorithm described in Appendix A.7 to deform the canonical frame into novel frame, and measures the per-point error against the ground truth.



(a) Input (b) Segmentation (c) Topology (d) Flow

Figure 10. Qualitative results beyond 1DOF joints. We apply our method to model spherical joints of human on D-FAUST dataset [5]. From left to right, we show the input human point cloud sequence, part segmentation, part connectivity, and implied flow. This demonstrates that our framework is general and can tackle other joint types beyond 1DOF joints.

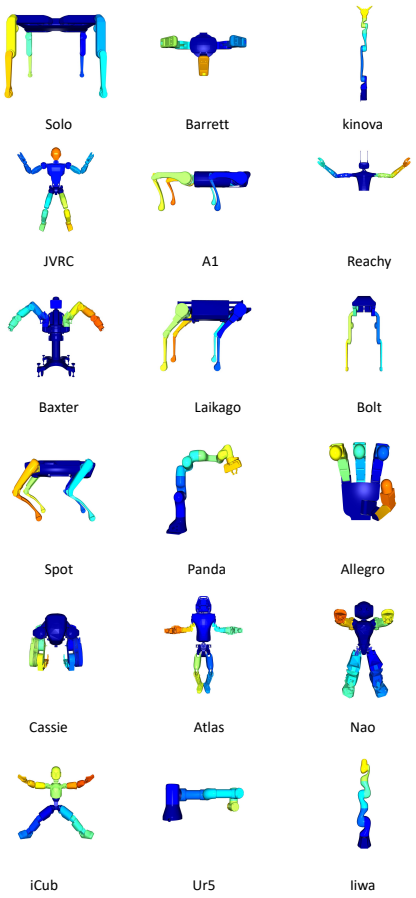


Figure 11. Robot categories visualization on RoboArt dataset. Different parts are in different colors.

B. RoboArt Dataset

The RoboArt dataset consists of 18 robots, including 6 different robot type: arms, bipeds, hands, mobile manipulators, humanoids, and quadrupeds. The robots, train-validation-test split are shown in Tab. 8 and Tab. 9. For each category, the points cloud sequence contain 10 frames with 4096 points independently sampled in each frame. Vi-

Table 7. Per-category performance on Sapiens dataset. We report flow error ↓ and Multi-scan RI ↑.

Box	Dishwasher	Display	Furniture	Eyeglasses	Faucet	Kettle	Knife	Laptop	Lighter
6.4/0.84	6.7/0.82	3.6/0.68	4.2/0.84	2.9/0.85	2.9/0.71	5.5/0.76	4.2/0.72	5.7/0.79	3.0/0.88
Oven	Phone	Washer	Pliers	Safe	Stapler	Door	Toilet	TrashCan	Microwave
7.0/0.75	3.5/0.66	3.6/0.76	2.19/0.77	3.7/0.84	7.5/0.78	2.9/0.74	3.0/0.77	6.5/0.82	5.8/0.83

Table 8. Robot type and categories on RoboArt dataset.

Robot Type	Robot Categories
Arms	Panda, UR5, Baxter, Kinova, iiwa
Bipeds	Bolt, Cassie
Hands	Allegro, Barrett
Mobile Manipulators	Reachy
Humanoids	Nao, Atlas, iCub, JVRC
Quadrupeds	A1, Laikago, Solo, Spot

Table 9. RoboArt dataset train, validation, and test split.

Split	Robot Categories
Train	Atlas, Baxter, Laikago, iiwa
Validation	Panda, Cassie, Spot, Panda
Test	Kinova, UR5, Bolt, Allegro, Barrett Reachy, iCub, JVRC, A1, Solo

sualization of different categories are shown in Fig. 11.

C. Additional Results

Beyond 1DOF joints. While our focus is on everyday objects, many of which have a piece-wise rigid structure with 1DOF joints, our framework is general and can tackle other joint types by modifying the *project* and *final fitting* steps. As a concrete example, spherical joints, which are a better model human and animals, can be tackled by a) replacing $E_{\text{1-DOF}}$ with $E_{\text{3-DOF}}{}^{\dagger}$ in E_{project} in Eq. (11), and b) optimizing over spherical joint parameters (*vs*. screw params) during final fitting. We show results on a 10 time-step human point cloud sequence from D-FAUST dataset [5] in Fig. 10. This demonstrates that our framework can tackle objects with more general joints.

Per-category performance on Sapiens dataset. We report per-category (20 category in total) performance including both flow error and Multi-scan RI on Sapiens daatset in Tab. 7.

Comparison against MultiBodySync and WatchItMove on RoboArt dataset. We provide additional comparison results on remaining test set categories besides those shown in Fig. 5 in the main paper. We compare our method on all RoboArt test set robot categories against MultiBodySync [17] and WatchItMove [39], the qualitative comparison is shown in Fig. 12. As can be seen MultiBodySync severely suffer from missing parts and and WatchItMove suffer from incorrect topology given it only takes spatial closeness into account when constructing the topology.

$^{\dagger}E_{\text{3-DOF}}$ measures how well the child part motion (relative to the parent) is explained by rotation around a fixed center.

Qualitative Results Visualization on RoboArt dataset. We provide additional qualitative visualization results in Fig. 13 and Fig. 14 on robot categories of validation set and remaining test set besides those shown in Fig. 4 in the main paper. We visualize part segmentation, topology and implied flow against ground-truth in each column. It can be seen our method work well for all robots with arbitrary topologies and geometries.

Reanimation Results on RoboArt dataset. We provide additional reanimation results in Fig. 15 and Fig. 16 on robot categories of validation set and remaining test set besides those shown in Fig. 7 in the main paper. As shown in the figure, we observe the results looks reasonable and match to the sparse guidance input. This demonstrates our animatable models's rearticulation ability.

Qualitative Results Visualization on Sapiens dataset. We provided common Sapien categories prediction results in Fig. 17 besides those shown in Fig. 6 in the main paper. As shown in the figure, our method works well on arbitrary daily articulated objects with different geometries and number of parts. We also show some inaccurate results in the last row of Fig. 17. We found out those inaccuracy are mostly caused by the noisy flow estimation provided by [17].

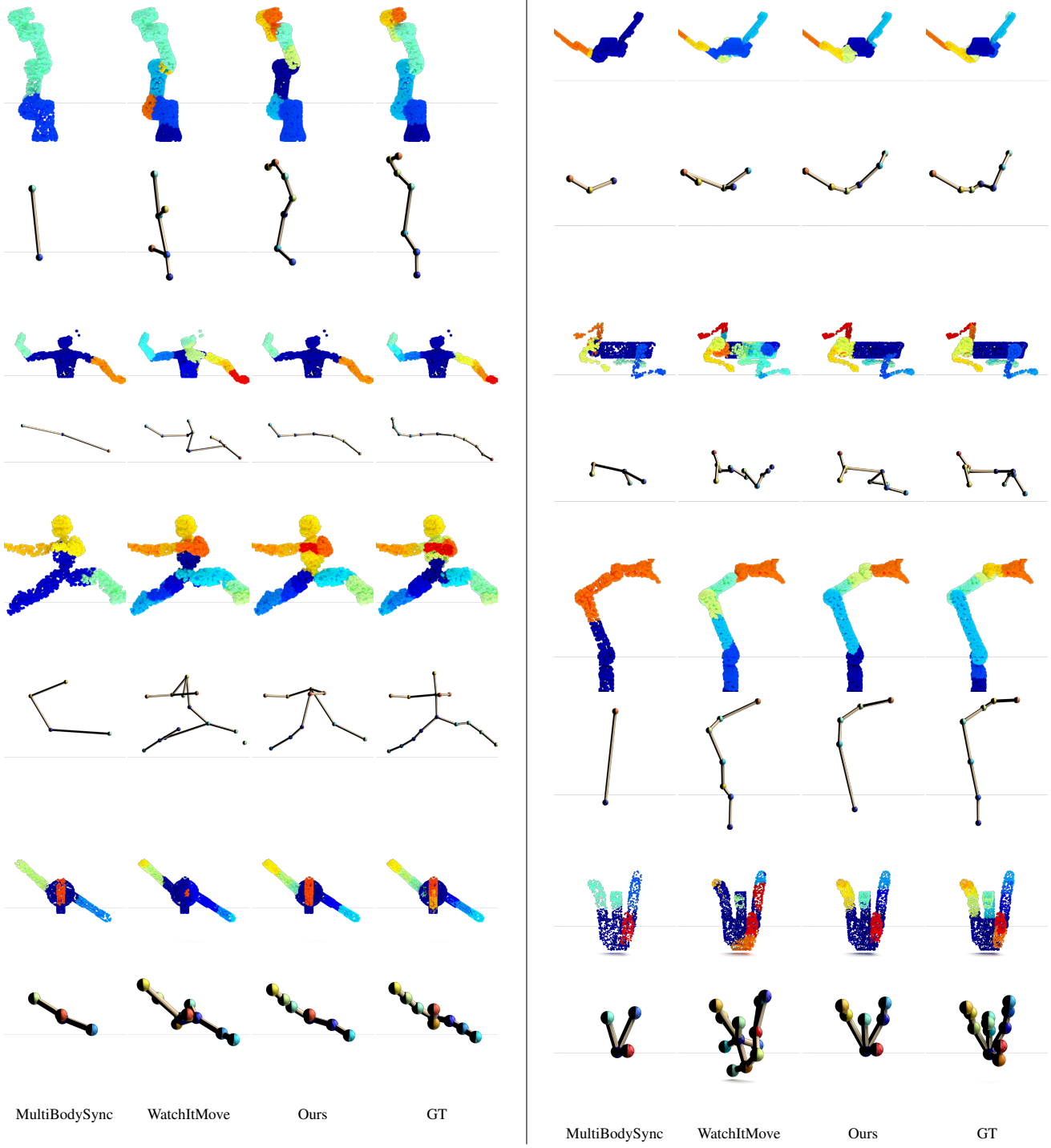


Figure 12. Qualitative comparison against MultiBodySync [17] and WatchItMove [39] on the RoboArt dataset test set. Note, a) MultiBodySync by itself doesn't produce a kinematic tree, we use our method on top of their output to generate one, and b) we provide WatchItMove [39] with the ground truth SDFs and number of parts (which are not used by our method). Even after these modifications, the past methods cannot solve the task as well as ours.

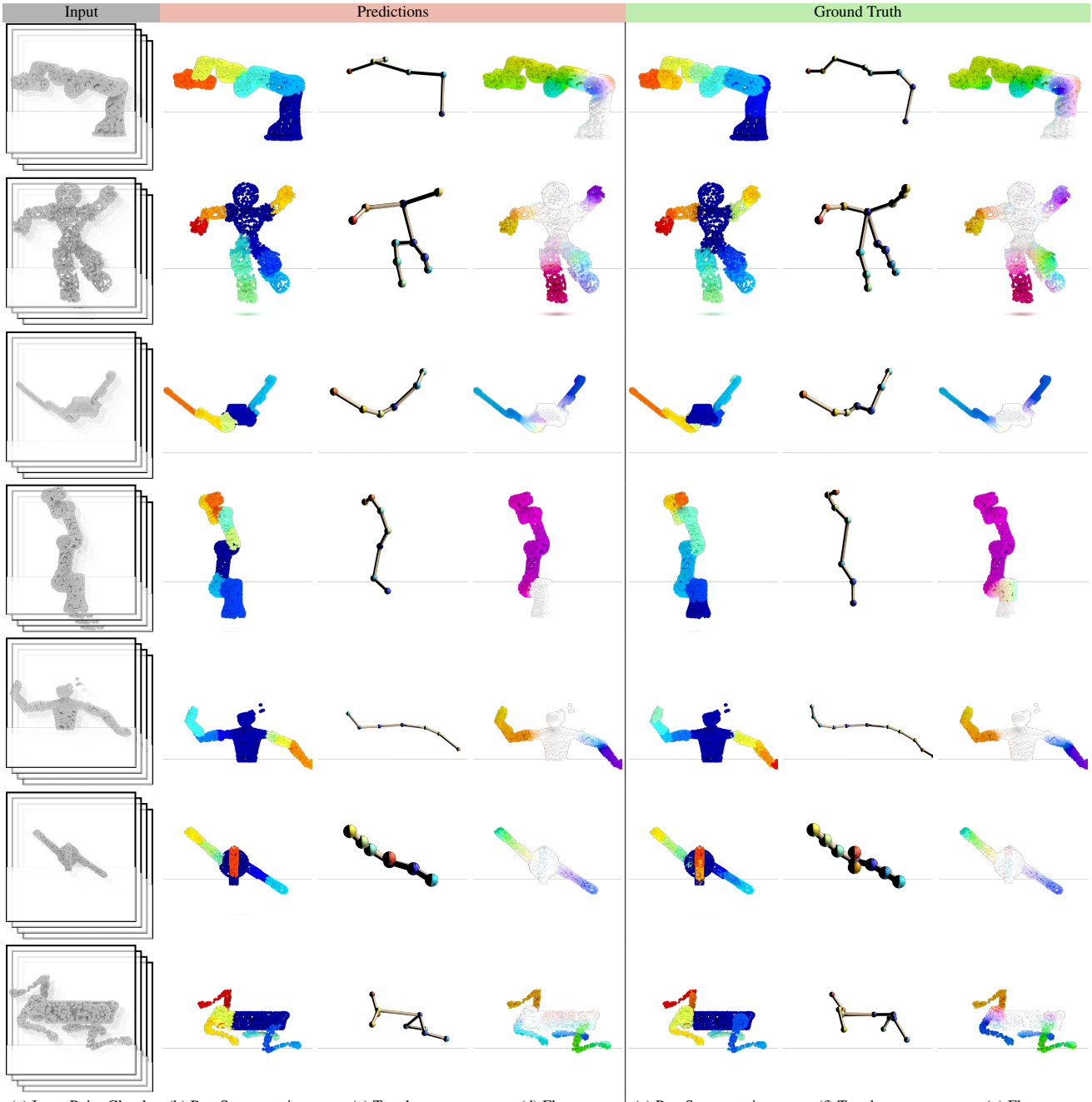


Figure 13. Qualitative Results on RoboArt Dataset (1/2). Given the input point cloud sequence (shown in (a)), we show the part segmentation, part connectivity, and implied flow using our inferred articulated model (in (b, c, d)) and the ground truth articulated model (in (e, f, g))

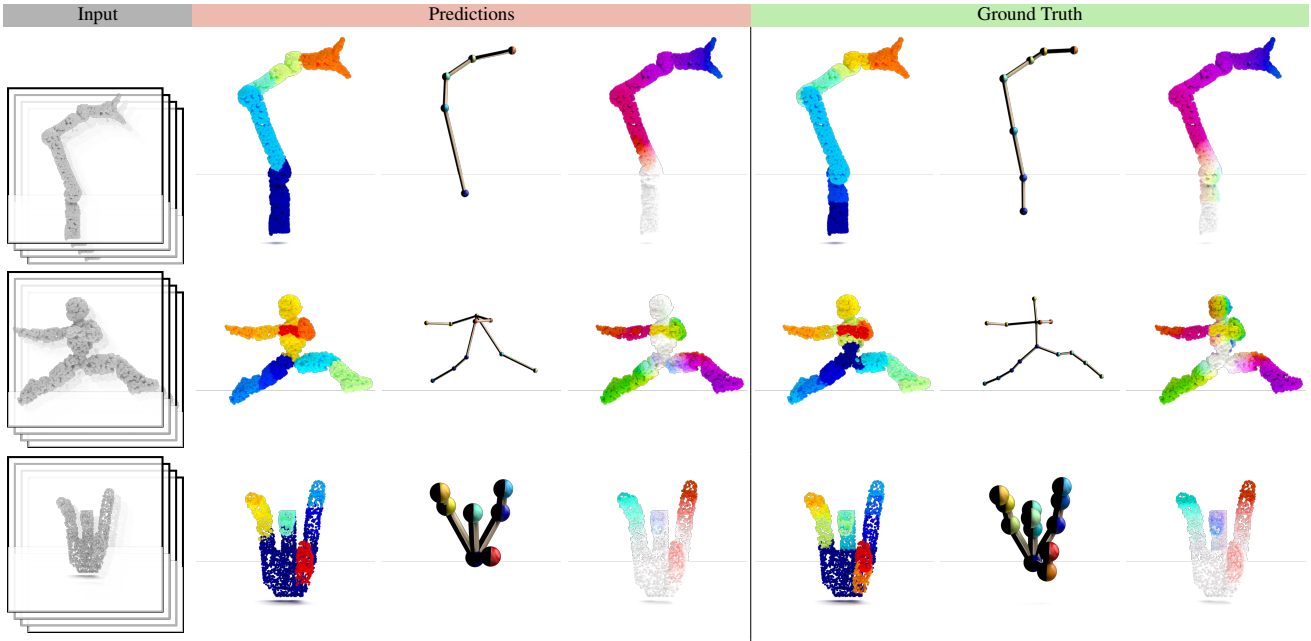


Figure 14. Qualitative Results on RoboArt Dataset (2/2).

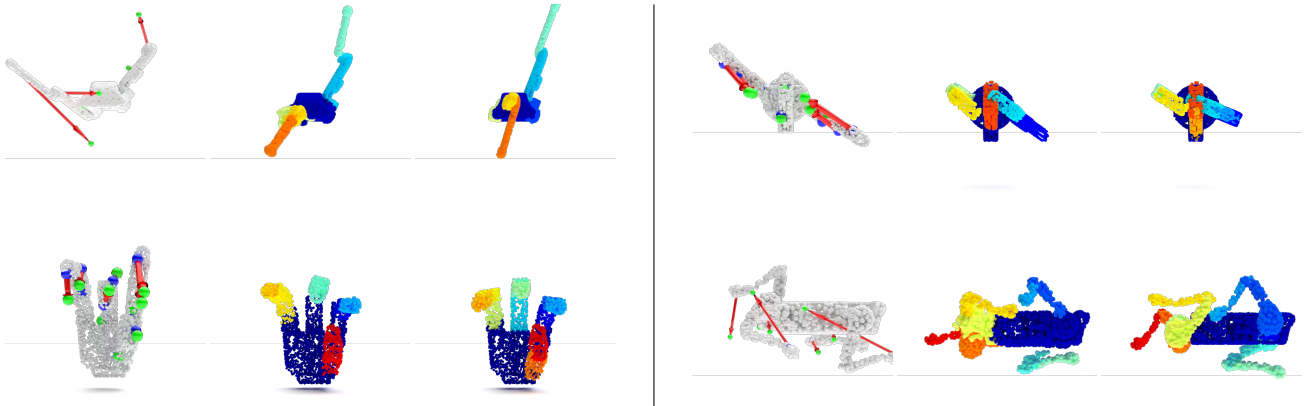


Figure 15. Reanimation Results on the RoboArt Dataset (1/2). Given new locations for a sparse set of points on the object(shown in (a)), our method (shown in (b)) is able to generate a reasonable reanimation to match the specified points.

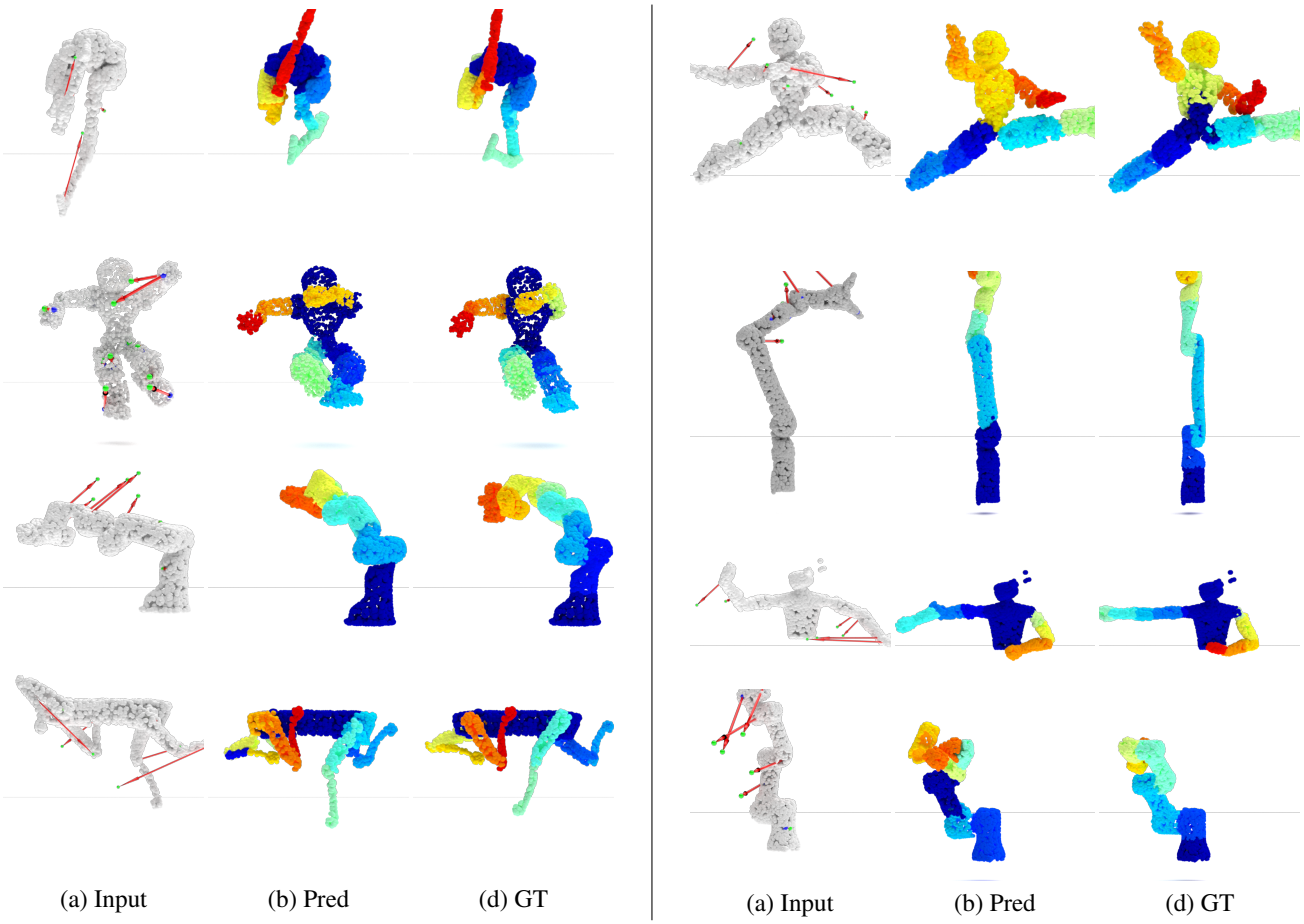


Figure 16. Robot reanimation Results on the RoboArt Dataset (2/2).

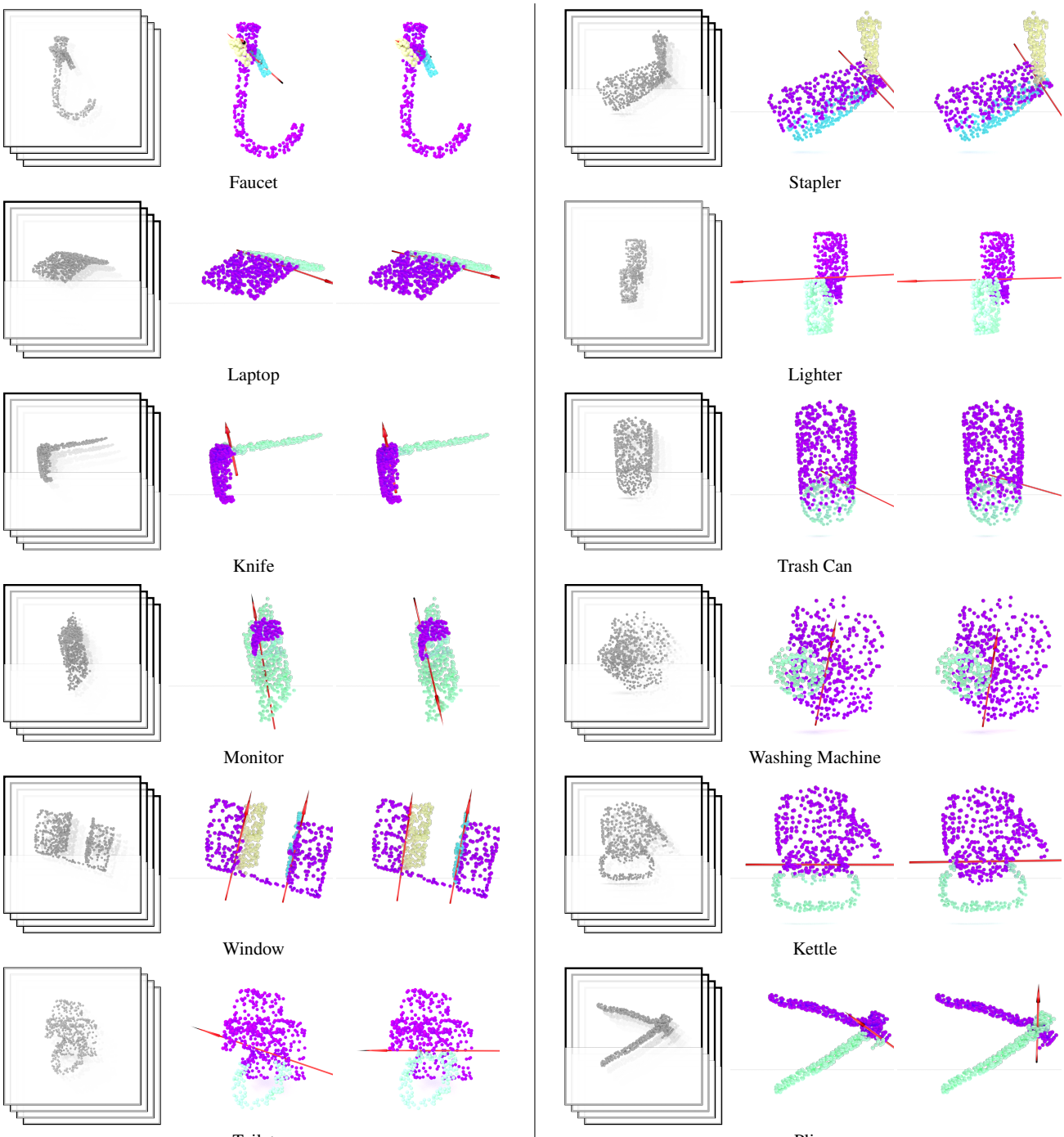


Figure 17. Qualitative results of common categories on Sapiens dataset from [17]. We visualize the predicted and ground truth articulated models. Different parts are in different colors, and we also show the screw parameters (in red) for the inferred joints. We use the provided flow estimation model [17]. Last row show some inaccurate results mostly caused by the noise in flow estimation.