Supplementary Material – 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/

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++ [7] 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 [5,9]. 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 $\mathbf{S}^*(\mathbf{P})$:

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

*equal advising, alphabetic order

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 $E_{\text{merge}}(i, \text{pa}(i)) =$

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

Figure 1. Visualization of merging. We show the segmentation of real-world switch before and after merging step.

$\sum_t \text{trace}((\hat{\mathbf{T}}^t_{pa(i)} \ominus \hat{\mathbf{T}}^t_i) \ominus \mathbf{I}$, 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. 1, 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 $\mathbf{\Gamma}, \{\mathbf{s}_i\}, \{\boldsymbol{\theta}^t\}$, 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 $\boldsymbol{\theta}^t_i = (\tau^t_i, d^t_i)$. As dicussed in Sec. 3.1, the rotation angle of a prismatic joint is always zero, *i.e.* $\{\tau^t_i = 0\}_{t\in 1,\dots T}$, while the translational component always be 0 for a revolute joint, *i.e.* $\{d^t_i = 0\}_{t\in 1,\dots T}$. We compute the mean $\bar{\tau}_i = \sum_{t=1}^T \tau^t_i$ and $\bar{d}_i = \sum_{t=1}^T d^t_i$ 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 $\{\tau^t_i = 0\}_{t\in 1,\dots T}$ for prismatic joint and $\{d^t_i = 0\}_{t\in 1,\dots T}$ 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 $M(\boldsymbol{\theta}^t; \mathbf{\Gamma}, f)$ to deform those points to match target points. We fix $\mathbf{\Gamma}, f$ 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 $\mathbf{P}'$.

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

A.8. Implementation of the Baselines

We describe the implementation of baselines MultiBodySync [3] and WatchItMove [6].

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. 5. 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, 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. 5.

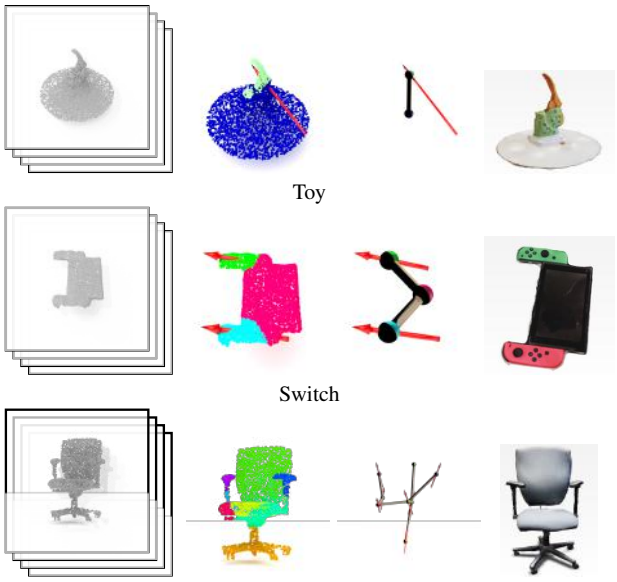
A.9. Evaluation Metrics

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 [8], 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.

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



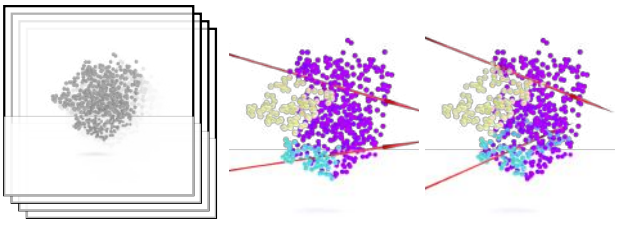
(a) Input (b) Segmentation (c) Kinematics (d) Real scan

Figure 2. 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.

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

Figure 3. Qualitative results beyond 1DOF joints. We apply our method to model spherical joints of human on D-FAUST dataset [2]. 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.

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 [1] or Kabsch [4] 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) Predicted Model (c) Ground Truth

Figure 4. Qualitative visualization of prismatic joints on Sapiens dataset. From left to right, We visualize the input point cloud sequence, 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.

Table 1. 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

B. Additional Results

Real-world setting. 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. 2. Our method is robust to noise and partial observations. Chair cushion gets slightly over-segmentation due to noisy surface.

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 [2] in Fig. 3. This demonstrates that our framework can tackle objects with more general joints.

Prismatic joints. 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. A qualitative result on Sapiens dataset is shown in Fig. 4. We also show prismatic joints (switch and chair) under real-world setting in Fig. 2. The testing performance between prismatic vs. revolute joints on Sapiens dataset are shown in Tab. 1.

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. 2.

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 [3] and WatchItMove [6], the qualitative comparison is shown in Fig. 5. 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.

Qualitative Results Visualization on RoboArt dataset. We provide additional qualitative visualization results in Fig. 6 and Fig. 7 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. 8 and Fig. 9 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. 10 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. 10. We found out those inaccuracy are mostly caused by the noisy flow estimation provided by [3].

C. 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-

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

Table 2. 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 3. 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 4. 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

validation-test split are shown in Tab. 3 and Tab. 4. For each category, the points cloud sequence contain 10 frames with 4096 points independently sampled in each frame. Visualization of different categories are shown in Fig. 11.

References

[1] 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. 3

[2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic faust: Registering human bodies in motion. In *CVPR*, pages 6233–6242, 2017. 3, 4

[3] 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, 4, 6, 10

[4] 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. 3

[5] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 1

[6] 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, 4, 6

[7] 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. 1

[8] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv*, 2020. 3

[9] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, pages 1592–1599, 2015. 1

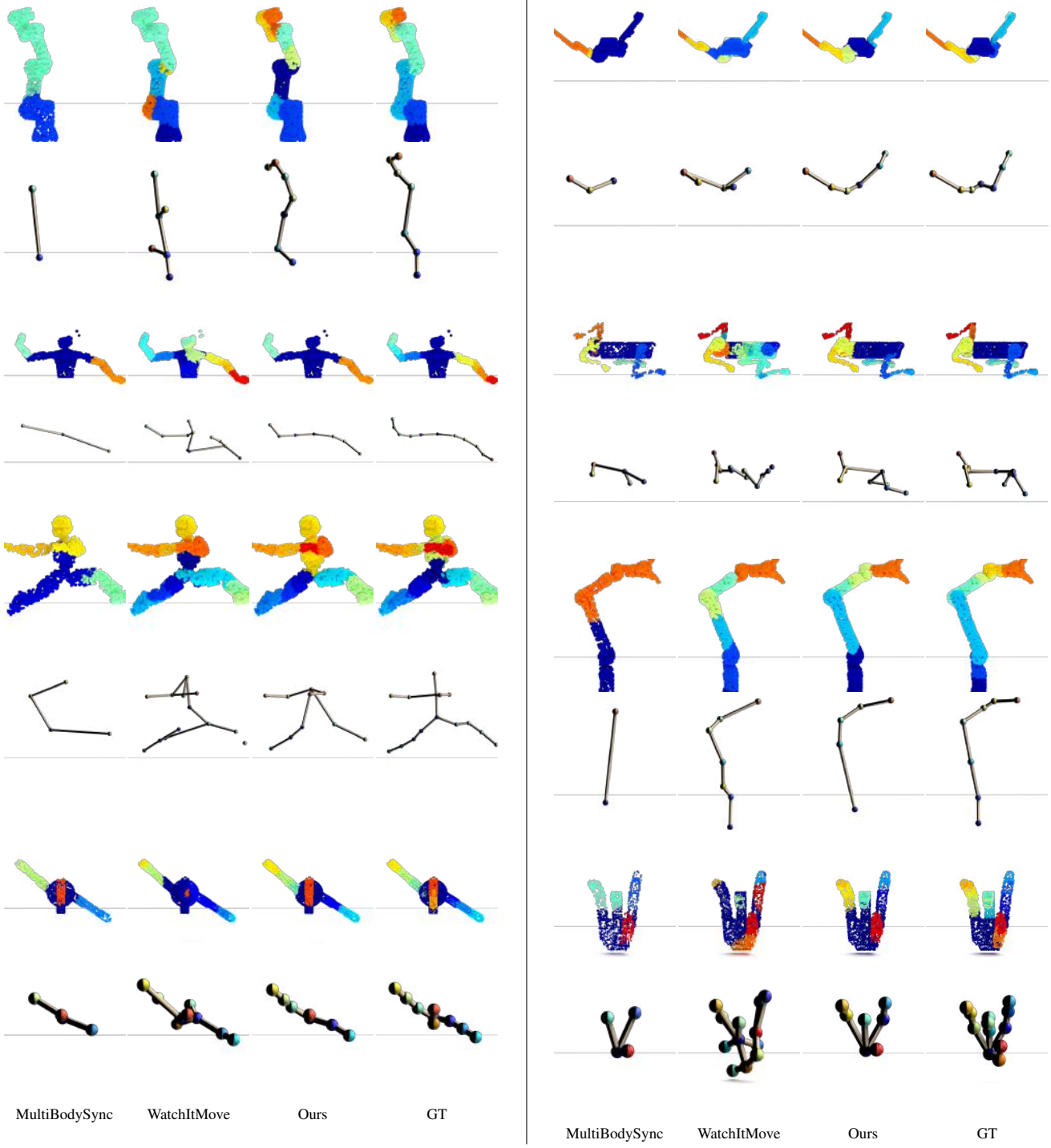


Figure 5. Qualitative comparison against MultiBodySync [3] and WatchItMove [6] on the RoboArt dataset test set. Note, a) MultiBodySync by itself doesn't produce a kinematic tree, we use our method on top of their output to generate one, and b) we provide WatchItMove [6] with the ground truth SDFs and 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.

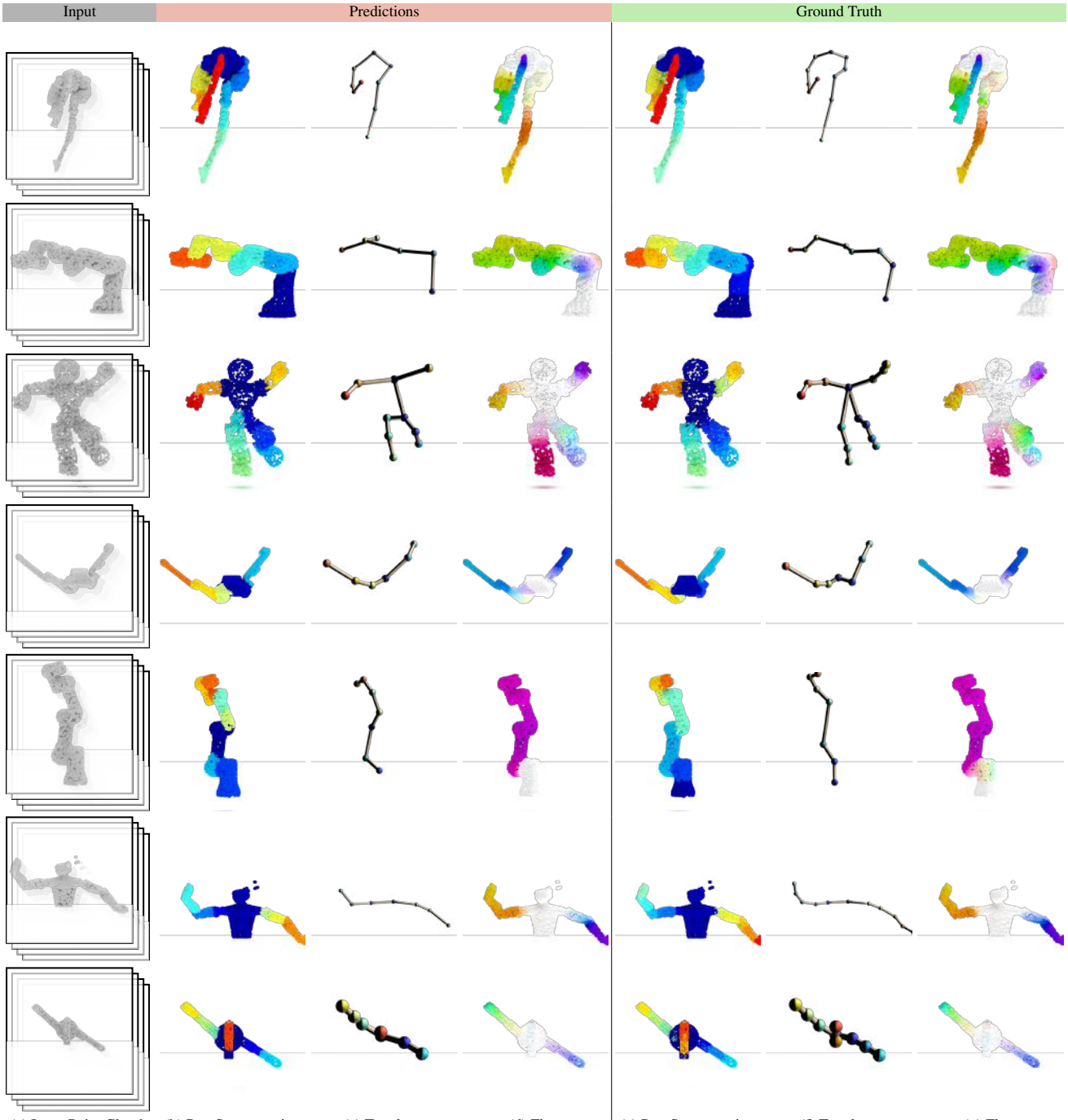


Figure 6. Qualitative Results on RoboArt Dataset (1/2). Given the input point cloud sequence (shown in (a)), we show the part segmentation, part connectivity, and implied flow using our inferred articulated model (in (b, c, d)) and the ground truth articulated model (in (e, f, g))

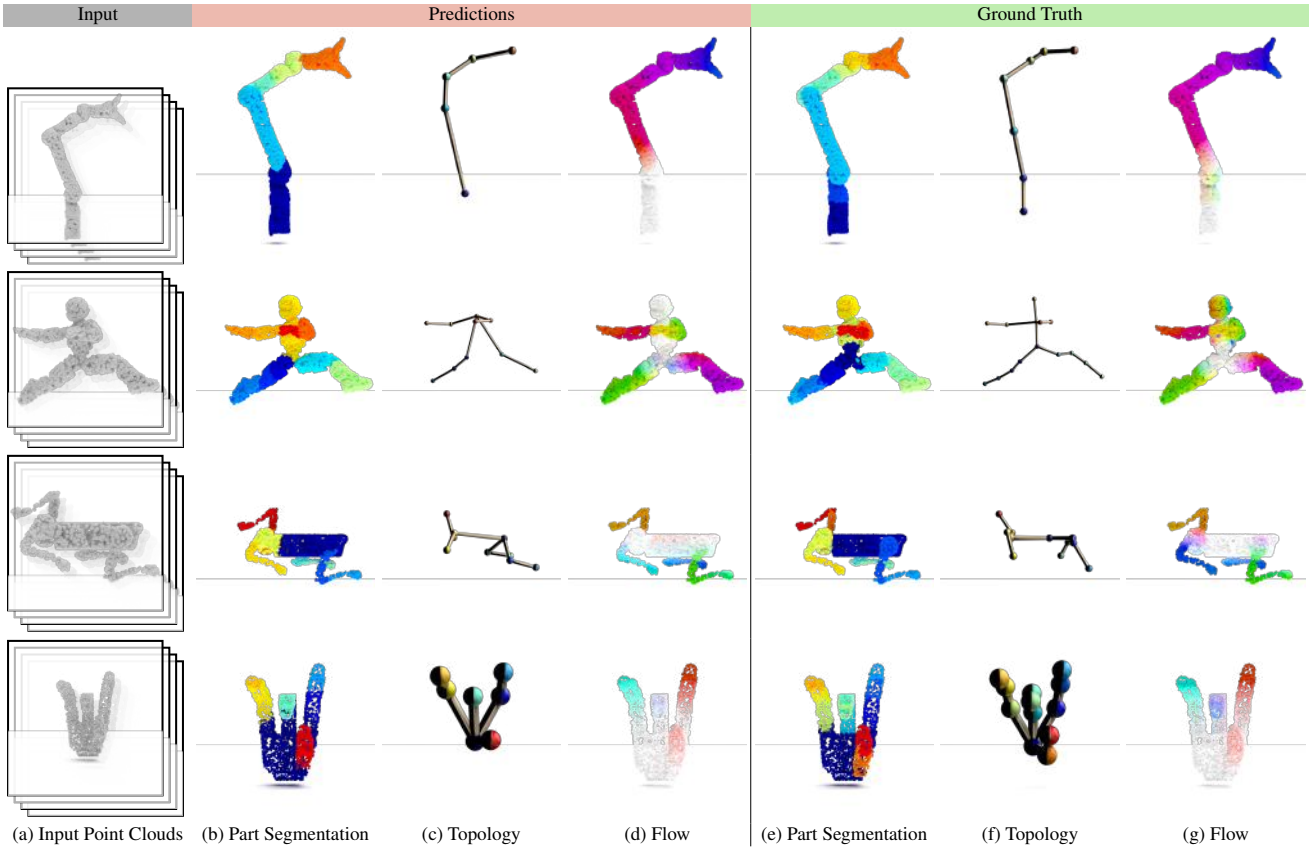


Figure 7. Qualitative Results on RoboArt Dataset (2/2).

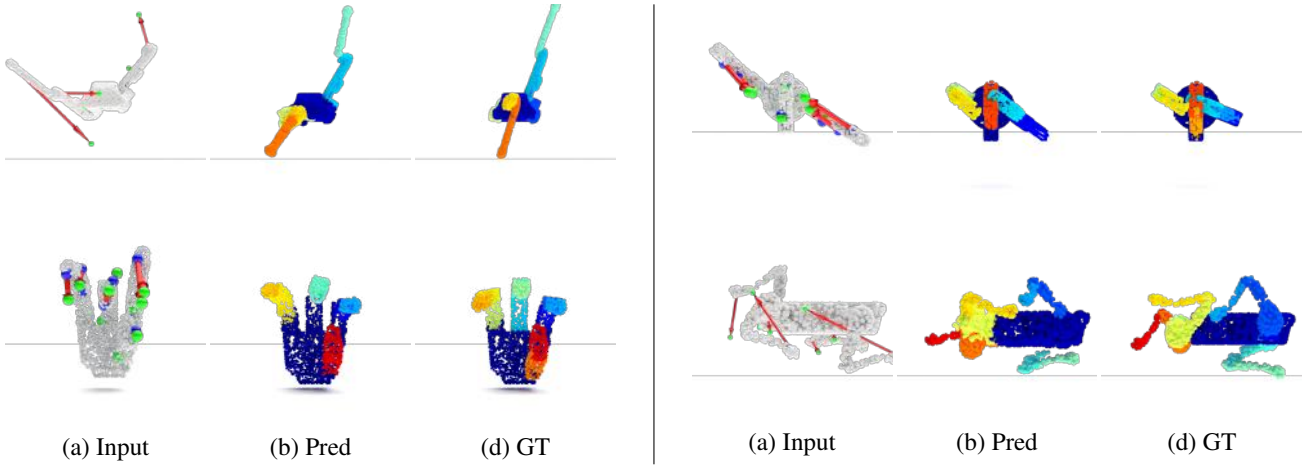


Figure 8. Reanimation Results on the RoboArt Dataset (1/2). Given new locations for a sparse set of points on the object(shown in (a)), our method (shown in (b)) is able to generate a reasonable reanimation to match the specified points.

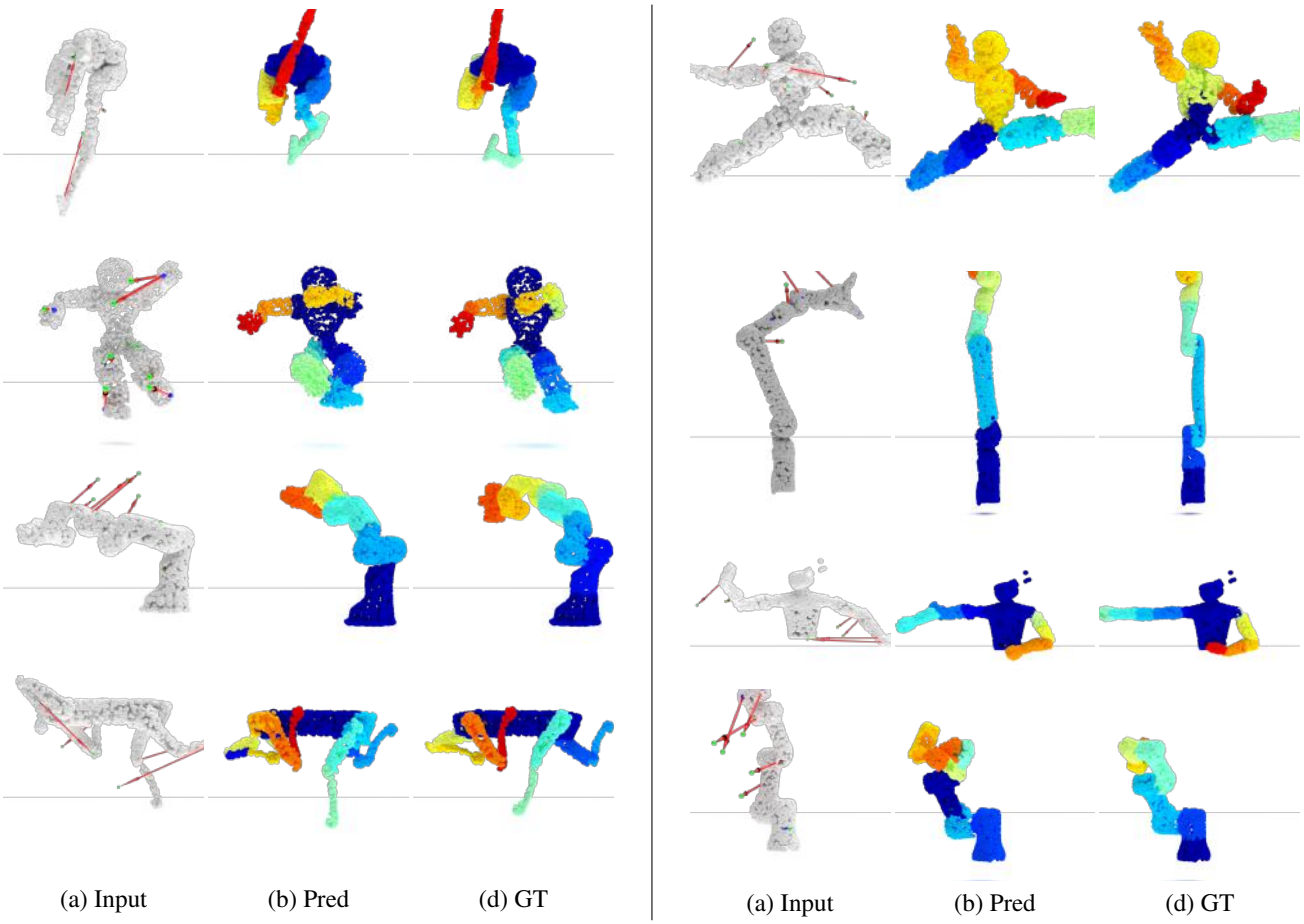


Figure 9. Robot reanimation Results on the RoboArt Dataset (2/2).

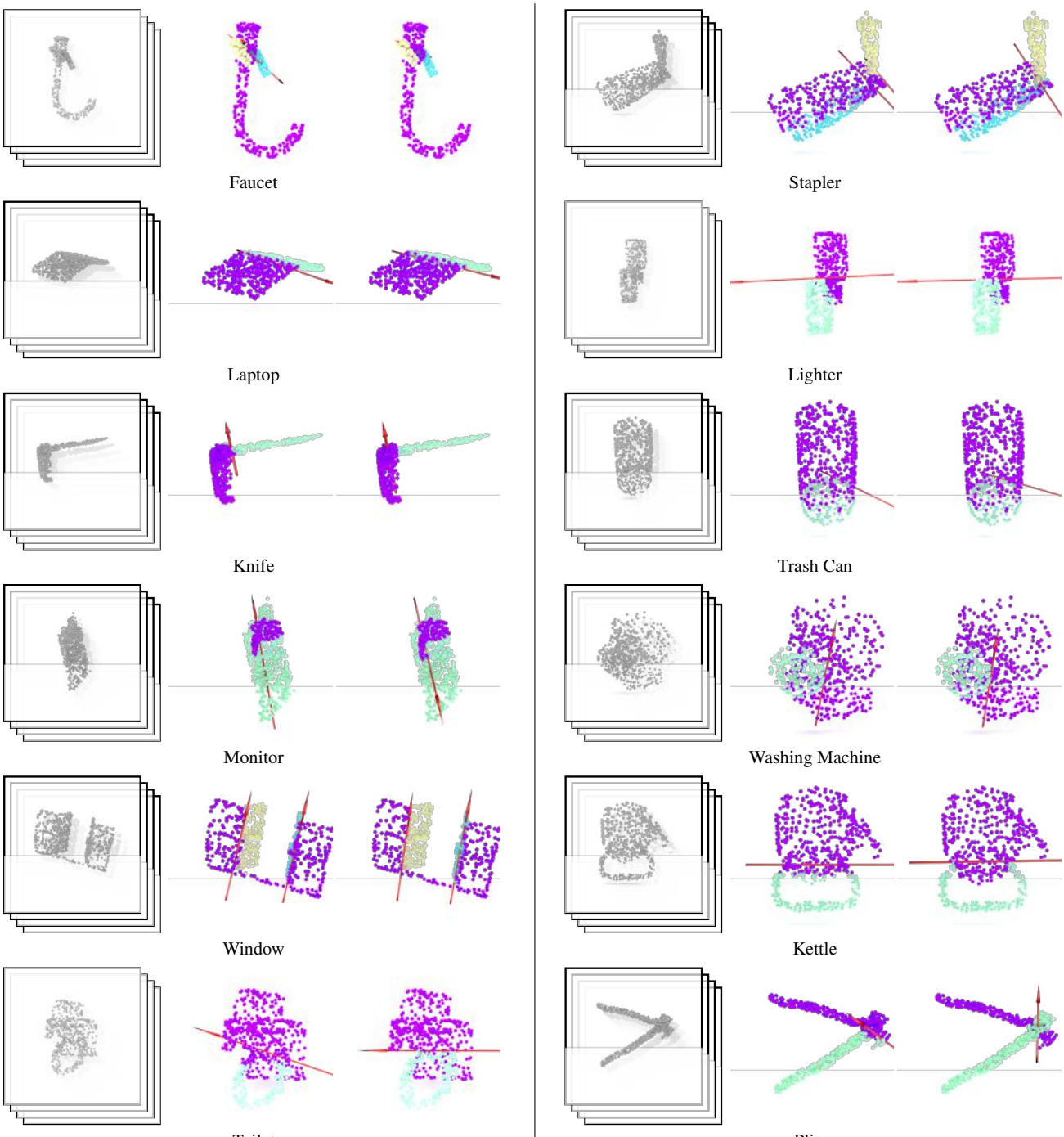


Figure 10. Qualitative results of common categories on Sapiens dataset from [3]. We visualize the predicted and ground truth articulated models. Different parts are in different colors, and we also show the screw parameters (in red) for the inferred joints. We use the provided flow estimation model [3]. Last row show some inaccurate results mostly caused by the noise in flow estimation.

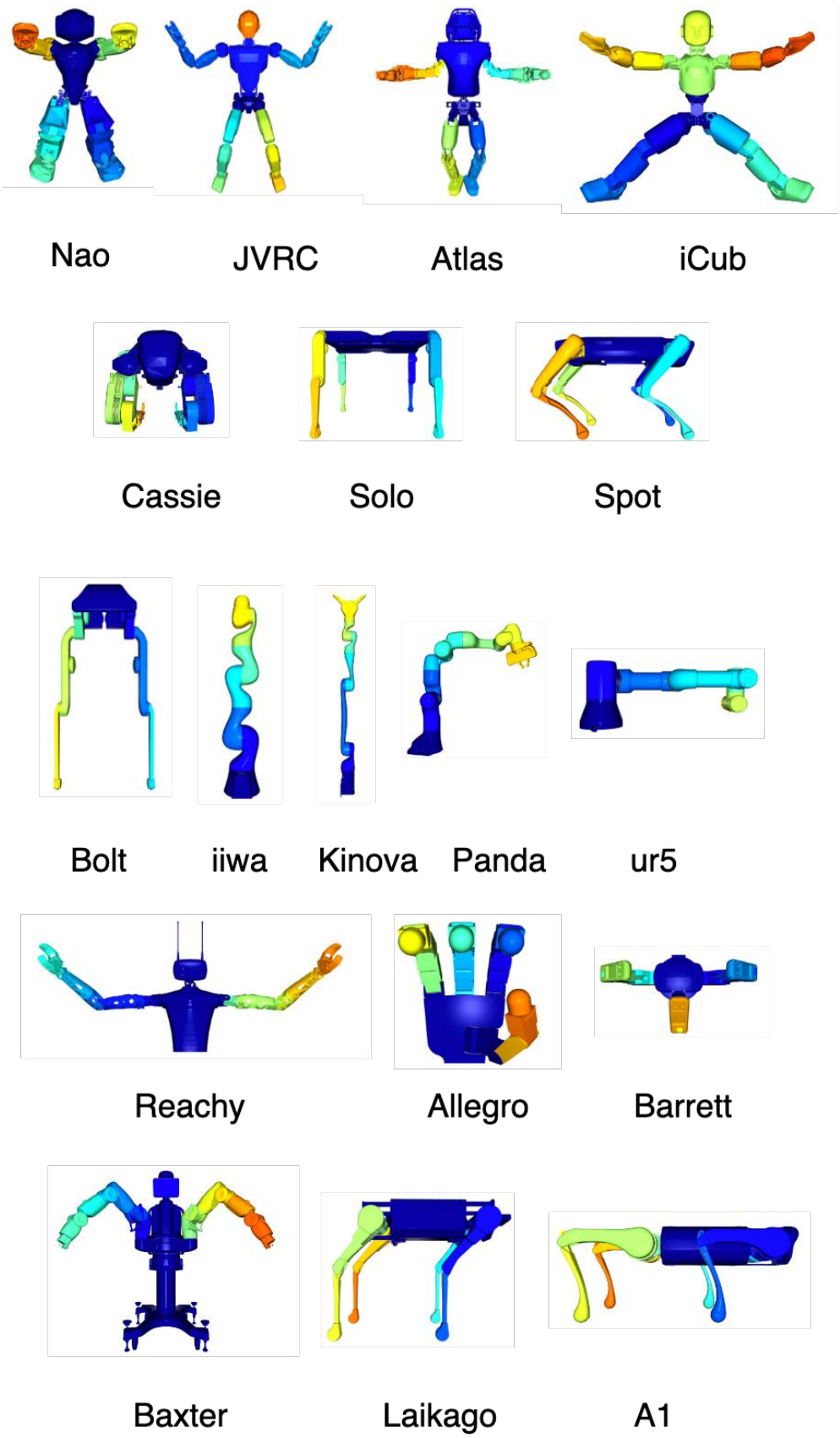


Figure 11. Robot categories visualization on RoboArt dataset. Different parts are in different colors.