MultiBodySync: Multi-Body Segmentation and Motion Estimation via 3D Scan Synchronization — Supplementary Material

In this supplementary material, we first give the proofs of the theorems in Sec. A, then provide more details of our implementation and our dataset in Sec. B. Additional ablations and results are shown in Sec. C.

A. Proofs of Theorems

A.1. Theorem 1

Proof. The energy function in Eq (2) of the main paper can be written as:

$$\begin{aligned}
E(\boldsymbol{p}) &= \sum_{k=1}^{K}\sum_{l=1}^{K} w^{kl}\|\mathbf{P}^k - \mathbf{P}^{kl}\mathbf{P}^l\|_F^2 \\
&= \sum_{k=1}^{K}\sum_{l=1}^{K}\sum_{i=1}^{N} w^{kl}\|\mathbf{P}^k_{:i} - \mathbf{P}^{kl}\mathbf{P}^l_{:i}\|^2 \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{l=1}^{K} w^{kl}\|\mathbf{P}^k_{:i}\|^2 + w^{lk}\|\mathbf{P}^l_{:i}\|^2 \\
&\quad - w^{kl}(\mathbf{P}^k_{:i})^\top(\mathbf{P}^{kl}\mathbf{P}^l_{:i}) - w^{lk}(\mathbf{P}^l_{:i})^\top(\mathbf{P}^{lk}\mathbf{P}^k_{:i}) \\
&= \sum_{i=1}^{N} 2\sum_{k=1}^{K}(\mathbf{P}^k_{:i})^\top\left(\sum_{l=1}^{K} w^{kl}(\mathbf{P}^k_{:i} - \mathbf{P}^{kl}\mathbf{P}^l_{:i})\right) \\
&= \sum_{i=1}^{N} 2\sum_{k=1}^{K}(\mathbf{P}^k_{:i})^\top\left(\left(w^k\mathbf{I}_N\right)\mathbf{P}^k_{:i} - \sum_{l\neq k} w^{kl}\mathbf{P}^{kl}\mathbf{P}^l_{:i}\right) \\
&= 2\sum_{i=1}^{N}\boldsymbol{p}_{:i}^\top\mathbf{L}\boldsymbol{p}_{:i} = 2\mathrm{tr}(\boldsymbol{p}^\top\mathbf{L}\boldsymbol{p}).
\end{aligned}$$

The spectral solution additionally requires each column of $\boldsymbol{p}$ to be of unit norm and orthogonal to others relaxing $\{\mathbf{P}^{kl} \in \mathcal{M}\}_{k,l}$:

$$\min_{\boldsymbol{p}} \mathrm{tr}(\boldsymbol{p}^\top\mathbf{L}\boldsymbol{p}) \quad \text{s.t.} \quad \boldsymbol{p}^\top\boldsymbol{p} = \mathbf{I}_N. \tag{S.1}$$

This QCQP (Quadratically Constrained Quadratic Program) is known to have the closed form solution revealed by generalized Rayleigh problem [5] (or similarly, the Courant-Fischer-Weyl min-max principle). The solution is given by the N eigenvectors of $\mathbf{L}$ corresponding to the smallest N eigenvalues. □

A.2. Theorem 2

We first recall the spectral solution of the synchronization problem and then extend the result to the weighted variant we propose. For completeness, here we include $\mathbf{Z} = \boldsymbol{g}\boldsymbol{g}^\top$, the **unweighted** motion segmentation matrix:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{0} & \boldsymbol{\zeta}^{12} & \dots & \boldsymbol{\zeta}^{1K} \\ \boldsymbol{\zeta}^{21} & \mathbf{0} & \dots & \boldsymbol{\zeta}^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\zeta}^{K1} & \boldsymbol{\zeta}^{K2} & \dots & \mathbf{0} \end{bmatrix}. \tag{S.2}$$

Lemma 1 (Spectral theorem of synchronization)**.** *In the noiseless regime and under spectral relaxation, the synchronization problem can be cast as*

$$\max_{\mathbf{U}} \mathrm{tr}(\mathbf{U}^\top\mathbf{Z}\mathbf{U}) \quad \text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}_S, \tag{S.3}$$

where $\mathbf{U} \in \mathbb{R}^{KN\times S}$ *denotes the sought solution,* i.e. *absolute permutations. Then each column in* $\mathbf{U}$ *will be one of the* S *leading eigenvectors of matrix* $\mathbf{Z}$ *[1]:*

$$\mathbf{U}\cdot\mathrm{diag}(\sqrt{\lambda_1},\dots,\sqrt{\lambda_S}) \approx \boldsymbol{g} = \begin{bmatrix} \mathbf{G}^1 \\ \mathbf{G}^2 \\ \vdots \\ \mathbf{G}^K \end{bmatrix}, \tag{S.4}$$

where $\lambda_1,\dots,\lambda_S$ *are the leading eigenvalues of* $\mathbf{Z}$*.*

We now recall the weighted synchronization problem. Here we assume the $\boldsymbol{\zeta}^{kl}$ matrices are binary and satisfy the properties listed in [1]. The weighted synchronization matrix $\tilde{\mathbf{Z}}$ is composed of a set of anisotropically-scaled $\boldsymbol{\zeta}^{kl}$ matrices:

$$\tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{0} & \frac{1}{\sigma^{12}}\boldsymbol{\zeta}^{12} & \dots & \frac{1}{\sigma^{1K}}\boldsymbol{\zeta}^{1K} \\ \frac{1}{\sigma^{21}}\boldsymbol{\zeta}^{21} & \mathbf{0} & \dots & \frac{1}{\sigma^{2K}}\boldsymbol{\zeta}^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sigma^{K1}}\boldsymbol{\zeta}^{K1} & \frac{1}{\sigma^{K2}}\boldsymbol{\zeta}^{K2} & \dots & \mathbf{0} \end{bmatrix}. \tag{S.5}$$

Remind that in the main paper we use the *unweighted* synchronization (*i.e.* without $\frac{1}{\sigma}$) by cancelling the effect of the weights via a normalization. Thm. 2, which we now state more formally, is then concerned about the linear scaling of the solution proportional to the weights in the motion segmentation matrix:

Theorem 2 (Weighted synchronization for segmentation)**.** *The spectral solution to the weighted version of the synchronization problem*

$$\max_{\tilde{\mathbf{U}}} \mathrm{tr}(\tilde{\mathbf{U}}^\top\tilde{\mathbf{Z}}\tilde{\mathbf{U}}) \quad \text{s.t.} \quad \tilde{\mathbf{U}}^\top\tilde{\mathbf{U}} = \mathbf{I}_S \tag{S.6}$$

is given by the columns of $\tilde{\boldsymbol{g}}$:

$$\tilde{\mathbf{U}} \cdot \text{diag}(\sqrt{\tilde{\lambda}_1}, \ldots, \sqrt{\tilde{\lambda}_S}) \approx \tilde{\boldsymbol{g}} = \begin{bmatrix} \mathbf{G}^1\mathbf{D}^1 \\ \mathbf{G}^2\mathbf{D}^2 \\ \vdots \\ \mathbf{G}^K\mathbf{D}^K \end{bmatrix}, \quad \text{(S.7)}$$

Here $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_S$ are the leading eigenvalues of $\tilde{\mathbf{Z}}$, and $(\mathbf{D}^1, \ldots, \mathbf{D}^K \in \mathbb{R}^{S\times S})$ are diagonal matrices. In other words, the columns of $\tilde{\boldsymbol{g}}$ being the eigenvectors of $\tilde{\mathbf{Z}}$ are related to the non-weighted synchronization by a piecewise linear anisotropic scaling.

Proof. We begin by the observation that $\mathbf{K}^k = \mathbf{G}^{k\top}\mathbf{G}^k$ is a diagonal matrix where K^k_{ss} counts[1] the number of points in point cloud k belonging to part s. Hence, each element along $\boldsymbol{g}^\top\boldsymbol{g} = \sum_{k=1}^K(\mathbf{K}^k)$ counts the number of points over all point clouds that belong to part s. Because $\mathbf{Z} = \boldsymbol{g}\boldsymbol{g}^\top$, we have the following spectral decomposition $\mathbf{Z}\boldsymbol{g} = \boldsymbol{g}\boldsymbol{\Lambda}$ [1]:

$$\mathbf{Z}\boldsymbol{g} = \boldsymbol{g}\boldsymbol{g}^\top\boldsymbol{g} = \boldsymbol{g}\sum_{k=1}^K \mathbf{G}^{k\top}\mathbf{G}^k = \boldsymbol{g}\boldsymbol{\Lambda}. \quad \text{(S.8)}$$

To simplify notation we overload w^{kl} by setting $w^{kl} = \frac{1}{\sigma^{kl}}$ for the rest of this subsection. Let us now write $\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}}$ in a similar fashion and seek the similar emergent property of eigen-decomposition:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}} = \begin{bmatrix} \sum_{l=1}^K w^{1l}\boldsymbol{\zeta}^{1l}\mathbf{G}^l\mathbf{D}^l \\ \sum_{l=1}^K w^{2l}\boldsymbol{\zeta}^{2l}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \sum_{l=1}^K w^{Kl}\boldsymbol{\zeta}^{Kl}\mathbf{G}^l\mathbf{D}^l \end{bmatrix}. \quad \text{(S.9)}$$

Then, using $\boldsymbol{\zeta}^{kl} = \mathbf{G}^k\mathbf{G}^{l\top}$ we can express Eq (S.9) as:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}} = \begin{bmatrix} \sum_{l=1}^K w^{1l}\mathbf{G}^1\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \sum_{l=1}^K w^{2l}\mathbf{G}^2\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \sum_{l=1}^K w^{Kl}\mathbf{G}^K\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \end{bmatrix} \quad \text{(S.10)}$$

$$= \begin{bmatrix} \mathbf{G}^1\sum_{l=1}^K w^{1l}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \mathbf{G}^2\sum_{l=1}^K w^{2l}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \mathbf{G}^K\sum_{l=1}^K w^{Kl}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \end{bmatrix} = \begin{bmatrix} \mathbf{G}^1\mathbf{H}^1 \\ \mathbf{G}^2\mathbf{H}^2 \\ \vdots \\ \mathbf{G}^K\mathbf{H}^K \end{bmatrix} \quad \text{(S.11)}$$

where:

$$\mathbf{H}^k = \sum_{k=1}^K w^{kl}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l. \quad \text{(S.12)}$$

$\mathbf{H}$ is a diagonal matrix because $\mathbf{D}^l$ is diagonal by assumption. Note that, the first part in the summation is assumed to be a *known*[2] diagonal matrix (see the beginning of proof):

$$\mathbf{E}^{kl} = w^{kl}\mathbf{G}^{l\top}\mathbf{G}^l, \quad \text{(S.13)}$$

This form is very similar to Eq (S.7) scaled by the corresponding diagonal matrices. Let us know consider the s^{th} column of $\tilde{\boldsymbol{g}}$ responsible for part s. We are interested in showing that such column is an eigenvector of $\tilde{\mathbf{Z}}$:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}}^s = \tilde{\lambda}_s\tilde{\boldsymbol{g}}^s. \quad \text{(S.14)}$$

In other words, we seek the existence of $\tilde{\lambda}_s$ such that Eq (S.14) is satisfied. Moreover, a closed form expression of $\tilde{\lambda}_s$ would allow for the understanding of the effect of the weights on the problem. Let us now plug Eq (S.7) and Eq (S.11) into Eq (S.14) to see that:

$$\begin{bmatrix} (\mathbf{G}^1\mathbf{H}^1)^s \\ (\mathbf{G}^2\mathbf{H}^2)^s \\ \vdots \\ (\mathbf{G}^K\mathbf{H}^K)^s \end{bmatrix} = \tilde{\lambda}_s \begin{bmatrix} (\mathbf{G}^1\mathbf{D}^1)^s \\ (\mathbf{G}^2\mathbf{D}^2)^s \\ \vdots \\ (\mathbf{G}^K\mathbf{D}^K)^s \end{bmatrix}. \quad \text{(S.15)}$$

As $\mathbf{G}^k$ is a binary matrix, it only actas as a column selector, where for a single part s, a column of the motion segmentation $\tilde{\boldsymbol{g}}$ should contain only ones. We can use this idea and the diagonal nature of $\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}}^s$ to cancel $\mathbf{G}^k$ on each side. Re-

[1] According to our assumption, this 'count' hereafter is only valid when $\boldsymbol{\zeta}^{kl}$s are binary and can be viewed as *soft counting* when such an assumption is relaxed.

[2] We will see later in remark 1 why this is only an assumption.

arranging the problem in terms of scalars on the diagonal yields:

$$\begin{cases} H^1_{ss} = \sum_{l=1}^{K} E^{1j}_{ss} D^l_{ss} = \tilde{\lambda}_s D^1_{ss} \\ H^2_{ss} = \sum_{l=1}^{K} E^{2j}_{ss} D^l_{ss} = \tilde{\lambda}_s D^2_{ss} \\ \vdots \\ H^K_{ss} = \sum_{l=1}^{K} E^{Kj}_{ss} D^l_{rr} = \tilde{\lambda}_s D^K_{ss} \end{cases} \tag{S.16}$$

where E is as defined in Eq (S.13). Note that both D and $\tilde{\lambda}_s$ are unknowns in this seemingly non-linear problem. Yet, we can re-arrange Eq (S.16) into another eigen-problem:

$$\mathbf{J}^s \mathbf{d}^s = \tilde{\lambda}'_s \mathbf{d}^s, \tag{S.17}$$

where:

$$\mathbf{J}^s = \begin{bmatrix} E^{11}_{ss} & E^{12}_{ss} & \cdots & E^{1K}_{ss} \\ E^{21}_{ss} & E^{22}_{ss} & \cdots & E^{2K}_{ss} \\ \vdots & \ddots & \vdots & \\ E^{K1}_{ss} & E^{K2}_{ss} & \cdots & E^{KK}_{ss} \end{bmatrix} \mathbf{d}^s = \begin{bmatrix} D^1_{ss} \\ D^2_{ss} \\ \vdots \\ D^K_{ss} \end{bmatrix}. \tag{S.18}$$

Hence, we conclude that the eigenvectors of the weighted synchronization have the form of Eq (S.17) if and only if we can solve Eq (S.7). This is possible as soon as $\mathbf{E}^{kl}$ are known and $\mathbf{J}^s$ has real eigenvectors. Besides an *existence* condition, Eq (S.7) also provides an explicit closed form relationship between the weights and the eigenvectors once $\mathbf{E}^{kl}$ are available.

□

Remark 1. *Note that the symmetric eigen-problem given in Eq (S.18) only requires the matrix $\mathbf{E}^{kl}$ for all k, l. By definition, each element along the diagonal of $\mathbf{E}^{kl} = w^{kl}\mathbf{G}^{k\top}\mathbf{G}^l$ denotes the number of points in each point cloud belonging to each part weighted by w. Hence, it does not require the complete knowledge on the part segmentation but only the amount of points per part. While this is unknown in practice, for the sake of our theoretical analysis, we might assume the availability of this information. Hence, we could speak of solving Eq (S.17) for each part s.*

Remark 2. *It is also interesting to analyze the scenario where one assumes $\mathbf{d}^s = \mathbf{1}$ for each s. In fact, this is what would happen if one were to naively use the unweighted solution for a weighted problem,* i.e. *use $\tilde{\mathbf{g}}$ itself as the estimate of motion segmentation, as our closed form expression for $\mathbf{D}^k$ (Eq (S.18)) cannot be evaluated in test time. Then, assuming $\mathbf{D}^k$ to be the identity, for each k it holds:*

$$\sum_{l=1}^{K} E^{kl}_{ss} = \sum_{l=1}^{K} w^{kl}\mathbf{G}^{k\top}(\mathbf{G}^l)^s \tag{S.19}$$

$$= w^{k1}\mathbf{G}^{1\top}(\mathbf{G}^1)^s + \cdots + w^{kK}\mathbf{G}^{K\top}(\mathbf{G}^K)^s$$
$$= \mathbf{w}_k \cdot \begin{bmatrix} \mathbf{G}^{1\top}(\mathbf{G}^1)^s & \cdots & \mathbf{G}^{K\top}(\mathbf{G}^K)^s \end{bmatrix}$$
$$= \mathbf{w}_k \boldsymbol{\varphi}^s = \tilde{\lambda}'_s. \tag{S.20}$$

where $(\mathbf{G}^l)^s$ is the s-th column of $\mathbf{G}^l$. The final equality follows directly from Eq (S.16) when $D_{ss} = 1$. Note that we can find multiple weights $\mathbf{w}_k$ satisfying Eq (S.20). For instance, if $\boldsymbol{\varphi}$ and λ were known, one solution for any s would be:

$$w^{kl} = \frac{\tilde{\lambda}_s}{K\varphi^s_k}. \tag{S.21}$$

Because (i) we cannot assume a uniform prior on the number of points associated to each part and (ii) it would be costly to perform yet another eigendecomposition, we choose to cancel the effect of the predicted weights w_{ij} as we do in the paper by a simple normalization procedure. However, such unweighted solution would only be possible because our design encoded the weights in the norm of each entry in the predicted $\boldsymbol{\zeta}^{kl}_{\text{net}}$.

B. Implementation Details

B.1. Network Structures

B.1.1 Flow Prediction Network

We adapt our own version of flow prediction network φ_{flow} from PointPWC-Net [14] by changing layer sizes and the number of pyramids. As illustrated in Fig. S1, the network predicts 3D scene flow in a coarse-to-fine fashion. Given input $\mathbf{X}^k$ as source point cloud and $\mathbf{X}^l$ as target point cloud, a three-level pyramid is built for them using furthest point sampling as $\{\mathbf{X}^{k,(0)} = \mathbf{X}^k, \mathbf{X}^{k,(1)}, \mathbf{X}^{k,(2)}\}$ and $\{\mathbf{X}^{l,(0)} = \mathbf{X}^l, \mathbf{X}^{l,(1)}, \mathbf{X}^{l,(2)}\}$, with point counts being 512, 128, 32, respectively. Similarly, we denote the flow predicted at each level as $\{\mathbf{F}^{kl,(0)}, \mathbf{F}^{kl,(1)}, \mathbf{F}^{kl,(2)}\}$. Per-point features for all points are then extracted with dimension 128, 192 and 384 for each hierarchy. A 3D 'Cost Volume' [8] is then computed for the source point cloud by aggregating the features from $\mathbf{X}^k$ and $\mathbf{X}^l$ for the point pyramid, with feature dimension 64, 128 and 256. This aggregation uses the neighborhood information relating the target point cloud and the warped source point cloud in a patch-to-patch manner. The cost volume, containing valuable information about the correlations between the point clouds, is fed into a scene flow prediction module for final flow prediction. The predicted flow at the coarser level can be upsampled via interpolation and help the prediction of the finer level. Readers are referred to [14] for more details.

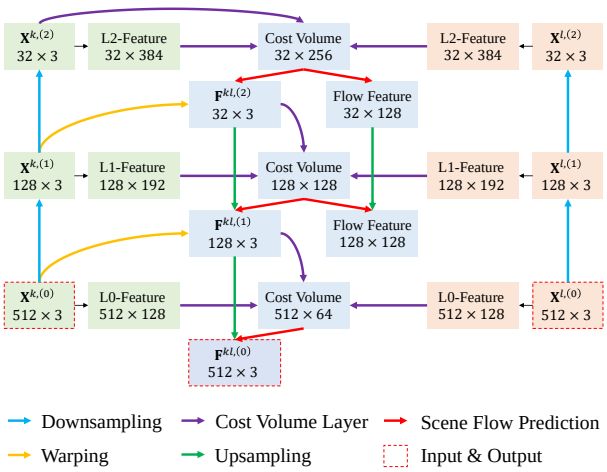

Figure S1. Our adapted version of PointPWC-Net φ_{flow}. Each rectangular block denotes a tensor, whose size is written as $N \times C$ (batch dimension is ignored) below its name, with N being the number of points and C being the feature dimension. The network is composed of 3 hierarchical levels. At each level, features from the two input point clouds are fused via a Cost Volume Layer, which digests warped point cloud and features from the upsampled coarse flow estimated from the last level and provides a cost volume for flow prediction.

B.1.2 Confidence Estimation Network

The confidence estimation network φ_{conf} we use, adapted from OANet (Order-Aware Network) [18], learns inlier probability of point correspondences. In our case, each correspondence is represented as a $\mathbb{R}^7$ vector as described in the main paper. Different from other network architectures like PointNet [11], OANet features in the novel differentiable pooling (DiffPool) and unpooling (DiffUnpool) operations as well as the order-aware filtering block, which are demonstrated to effectively gather local context and are hence useful in geometric learning settings, especially for outlier rejection [2].

The network starts and ends with 6 PointCN [10] layers, which globally exchanges the point feature information by context normalization (*i.e*. whitening along the channel dimension to build cross-point relationship). In between the PointCNs lies the combination of DiffPool layer, order-aware filtering block and DiffUnpool layer. The DiffPool layer learns an $N \times M$ soft assignment matrix, where each row represents the classification score of each point being assigned to one of the M 'local clusters'. These local clusters represent local structures in the correspondence space and are implicitly learned. As the M clusters are in canonical order, the feature after the DiffPool layer is permutation-invariant, enabling the order-aware filtering block afterward to apply normalization along the spatial dimension (*i.e*., 'Spatial Correlation Layer') for capturing a more complex global context. In our φ_{conf}, we choose $M = 64$.

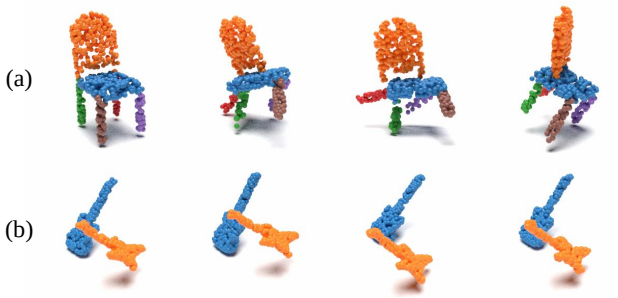

Figure S2. Examples from our training set for (a) articulated objects and (b) multiple solid objects. Different colors indicate rigidly moving parts.

B.1.3 Motion Segmentation Network

The architecture of φ_{mot} has been already introduced in the main paper. Here we elaborate how the transformations $\tilde{\mathbf{T}}_i^{kl}$ are estimated by PointNet++. The input to the network is the stacked $[(\mathbf{X}^k)^\top \ (\hat{\mathbf{F}}^{kl})^\top]^\top \in \mathbb{R}^{6\times N}$ and the output is in $\mathbb{R}^{12\times N}$, where for each point we take the first 9 dimensions as the elements in the rotation matrix and the last 3 dimensions as the translation vector.

In practice, direct transformation estimation from the PointNet++ is not accurate. Given that we have already obtained the flow vectors, instead of estimating $\tilde{\mathbf{T}}_i^{kl}$, we compute a residual motion w.r.t. the given flow similar to the method in [16]. Specifically, when the actual outputs from the network are $\mathbf{R}_{\text{net}} \in \mathbb{R}^{3\times3}$ and $\mathbf{t}_{\text{net}} \in \mathbb{R}^3$, the transformations used in subsequent steps of the pipeline $\tilde{\mathbf{T}}_i^{kl} = [\tilde{\mathbf{R}}_i^{kl}|\tilde{\mathbf{t}}_i^{kl}]$ are computed as follows:

$$\tilde{\mathbf{R}}_i^{kl} = \mathbf{R}_{\text{net}} + \mathbf{I}_3, \quad \tilde{\mathbf{t}}_i^{kl} = \mathbf{t}_{\text{net}} - \mathbf{R}_{\text{net}}\boldsymbol{x}_i^k + \boldsymbol{f}_i^{kl}. \quad \text{(S.22)}$$

Note that we do not ensure $\tilde{\mathbf{T}}_i^{kl}$ is in SE(3) with SVD-like techniques. In fact the transformation is not directly supervised (neither in this module nor in the entire pipeline) and the nearest supervision comes from $\boldsymbol{\beta}^{kl}$ matrix through Eq (9). This avoids the efforts to find a delicate weight for balancing the rotational and translational part of the transformation.

B.2. Pose Computation and Iterative Refinement

Given the synchronized pairwise flow $\hat{\boldsymbol{f}}^{kl}$ and motion segmentation $\mathbf{G}^k$, we estimate the motion separately for each rigid part using a weighted Kabsch algorithm [7]. The weight for point $\boldsymbol{x}_i^k$ and the rigid motion s between $\mathbf{X}^k$ and $\mathbf{X}^l$ is taken as $c_i^{kl}G_{is}^k$. We then use similar techniques as in [3, 6] to estimate the motions separately for each part.

The point clouds to register can have a large difference in poses making it hard for the flow network to recover. This might lead to wrong results in the subsequent steps. In-

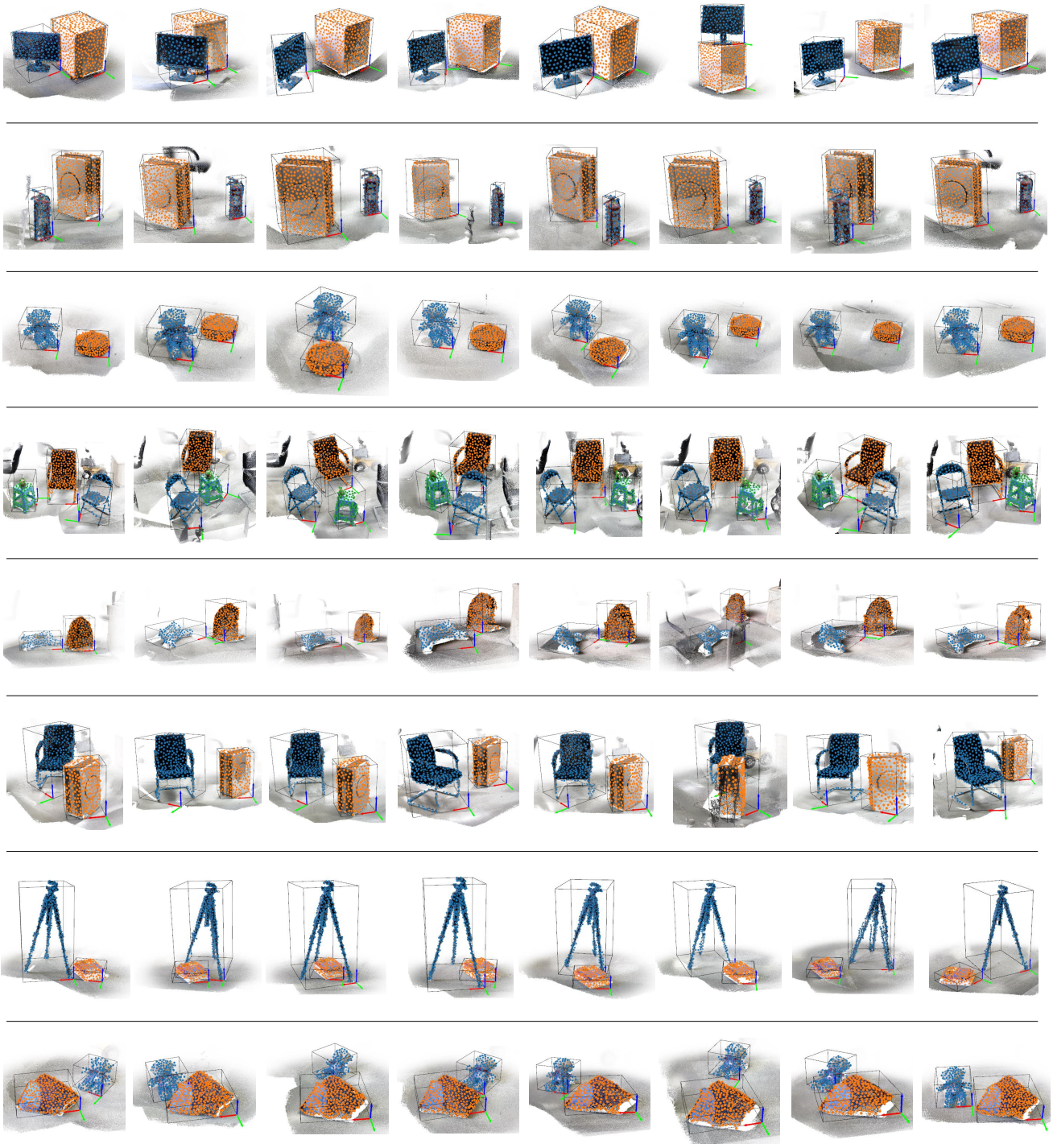

Figure S3. Visualization of the DynLab dataset. Each row shows 8 different dynamic configurations of the same set of rigid objects. Annotated bounding boxes are parallel to the ground plane and reflect the objects' absolute poses.

spired by point cloud registration works [16, 3], during test time we iterate our pipeline several times to gradually refine the correspondence and segmentation estimation. In particular, we use the transformation $\mathbf{T}_s^{k^*}(\mathbf{T}_s^k)^{-1}$ estimated at iteration $t-1$ to transform all the points in all point sets belonging to part s to the *canonical* pose of the k^*-th point

Table S1. Training and validation categories from [17] used for articulated objects.

Training Categories	Table	Chair	Plane	Car
	Guitar	Bike	Suitcase	
Validation Categories	Lamp	Pistol	Mug	Skateboard
	Earphone	Rocket	Cap	

Table S2. Training and validation categories from [17] used for multiple solid objects.

Training Categories	Table	Knife	Plane	Car
	Guitar	Bike	Suitcase	Laptop
Validation Categories	Lamp	Pistol	Mug	Skateboard
	Earphone	Rocket	Cap	

cloud. Note that the choice of $k^\star$ is arbitrary, and we choose $k^\star = 1$. Then at iteration t, we feed the transformed point clouds to the flow network again to compute the residual flow, which is added back onto the flow predicted at iteration $t-1$ to form the input of the segmentation network. The progress works reciprocally, as differences in poses of the point clouds are gradually minimized and the flow estimation will hence become more accurate, leading to better segmentation and transformations. Specially, during the first iteration where pose differences are usually large, we treat the point clouds as if they are composed of only one rigid part to globally align the shapes. This will provide a good pose initialization for subsequent iterations.

B.3. Dataset

Training Data. To demonstrate the generalizability of our method across different semantic categories, we ensure the categories used for training, validation and test have no overlap. For articulated objects, the categories we use are shown in Tab. S1. For multiple solid objects, the categories are listed in Tab. S2. Examples from our training set are visualized in Fig. S2.

DynLab dataset. A full visualization of the DynLab dataset with manual annotations is shown in Fig. S3. We will make the scans publicly available.

C. Additional Results

C.1. Extended Ablations

In this subsection we provide more complete ablations extending § 4.3 of the main paper. A full listing of the baselines we compare is as follows:

- **Ours (1 iter)**: The pipeline is iterated only once, without the global alignment step as described in § B.2.
- **Ours (NS, NW)**: Same as the main paper, we directly feed $\mathbf{F}^{kl}$ instead of $\hat{\mathbf{F}}^{kl}$ to the motion network φ_{mot}.

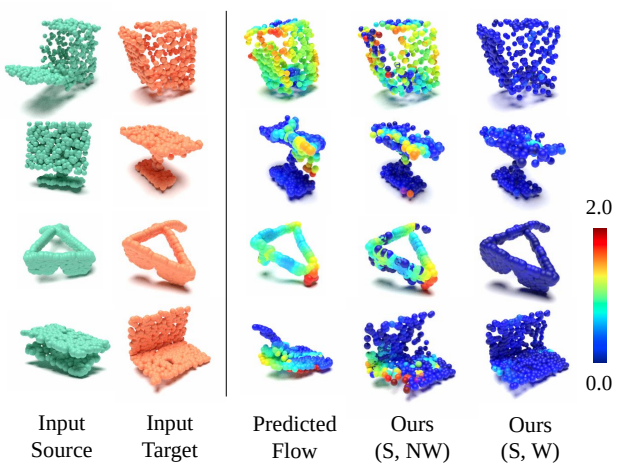

Figure S4. Visual comparisons of the pairwise flow. To visualize the flow we warp the source point cloud and compare the its similarity with the target point cloud. The color bar on the right shows the end-point error (EPE3D). 'Ours (S, W)' represents the output of our method with the **W**eighted permutation **S**ynchronization scheme.

- **Ours (S, NW)**: Same as the main paper, we set all weights of the permutation synchronization $w^{kl} = 1$.
- **Ours (UNZ)**: The unnormalized matrix $\tilde{\mathbf{Z}}$ (Eq (S.5)) is used as input to motion segmentation synchronization, *i.e.*, the normalizing factors are set to $\sigma^{kl} = 1$.
- **Ours (4 iters)**: Full pipeline of our method, with 4 steps of iterative refinement.

We show comparisons of the final rigid flow error using EPE3D metric on both SAPIEN and DynLab dataset in Fig. S5 and S6 respectively. Results indicate that all the components introduced in our algorithm, including iterative refinement, weighted synchronization, and the pre-factoring of the motion segmentation matrix, contribute to the improvement of accuracy under different scenarios. Note that in DynLab dataset, the performance of 'Ours (UNZ)' is very similar to 'Ours (4 iters)' because the motion segmentation accuracy is already high (Tab. 3 of the main paper) due to the good quality of each individual $\boldsymbol{\zeta}_{\text{net}}$ output, rendering normalization optional in practice.

We provide additional visual examples demonstrating the effectiveness of our weighted permutation synchronization in Fig. S4, where direct flow output fails due to large pose changes between the input clouds, and a naive unweighted synchronization still suffers from such failure because the influence of wrong correspondences is not eliminated.

For completeness we include per-category segmentation accuracy of articulated objects on SAPIEN [15] dataset in Tab. S3. The variants of our method perform consistently better than other methods for nearly all categories, showing the robustness of our model for accurate multi-scan motion-

Table S3. Per-category mIoU comparisons on SAPIEN [15] dataset.

	Box	Dishwasher	Display	Storage Furniture	Eyeglasses	Faucet	Kettle	Knife	Laptop	Lighter
PointNet++ [12]	43.5	46.8	54.8	51.3	34.6	42.4	65.7	43.0	58.5	52.3
MeteorNet [9]	47.0	42.2	41.7	36.9	36.1	47.1	67.2	36.2	57.9	61.3
DeepPart [16]	53.3	55.1	47.4	48.7	31.8	43.4	64.7	38.5	67.3	39.0
NPP [4]	41.4	63.7	57.3	48.0	35.3	45.4	50.7	44.5	61.1	50.7
Ours (1 iter)	67.6	57.3	66.3	68.1	57.8	54.7	83.3	55.5	78.6	52.0
Ours (NS, NW)	67.1	61.6	62.6	67.5	60.6	50.3	78.3	53.6	77.5	51.5
Ours (S, NW)	71.4	58.9	68.8	71.3	61.7	57.2	81.4	57.8	82.7	64.6
Ours (UNZ)	71.5	59.6	69.1	71.6	62.1	58.0	78.9	57.8	82.7	65.0
Ours (4 iters)	72.0	62.0	67.4	73.1	66.2	56.2	80.7	56.4	83.3	62.6

	Microwave	Oven	Phone	Pliers	Safe	Stapler	Door	Toilet	TrashCan	Washing Machine	Overall
PointNet++ [12]	51.5	42.6	46.2	63.6	55.7	43.0	42.7	40.0	51.2	49.8	47.5
MeteorNet [9]	37.4	37.1	41.7	43.4	33.7	54.7	33.3	38.3	61.5	41.9	43.7
DeepPart [16]	65.9	49.8	41.9	32.9	57.5	47.0	38.6	39.1	65.1	59.5	49.2
NPP [4]	56.4	39.7	48.4	61.3	55.9	45.5	40.4	31.2	51.0	48.4	48.2
Ours (1 iter)	61.6	54.7	52.5	50.6	59.4	67.0	47.1	55.7	79.3	64.2	62.9
Ours (NS, NW)	74.6	59.0	49.4	57.0	62.2	65.6	45.1	52.0	76.1	72.9	63.3
Ours (S, NW)	62.8	52.3	54.1	51.4	62.5	72.0	49.0	57.2	81.1	71.2	65.6
Ours (UNZ)	62.7	52.2	55.1	49.9	61.3	72.3	48.8	57.5	81.4	71.4	65.8
Ours (4 iters)	69.3	56.1	54.6	63.9	63.9	70.2	48.3	56.5	80.4	72.1	66.7

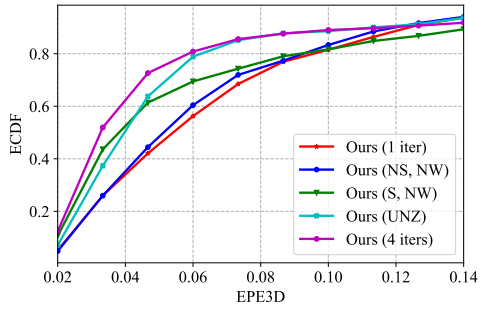

Figure S5. Empirical cumulative distribution function (ECDF) of rigid flow error (EPE3D) on SAPIEN [15] dataset. The higher the curve, the better the results.

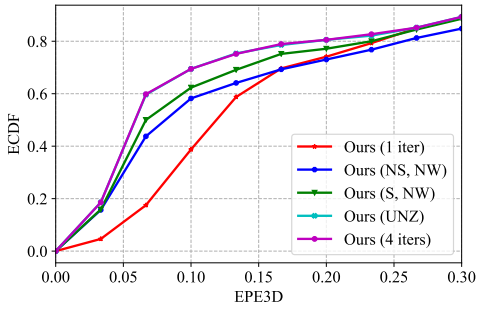

Figure S6. Empirical cumulative distribution function (ECDF) of rigid flow error (EPE3D) on DynLab dataset. The higher the curve, the better the results.

based segmentation.

C.2. Qualitative Results

To provide the readers with a more intuitive understanding of our performance under different cases, we illustrate in Fig. S7 the scenarios with co-existing articulated/solid objects and multiple cars in a scene of Waymo Open dataset [13] (though the car category is within our training set). Moreover, we show in Fig. S8 to S10 our segmentation and registration results for each category in SAPIEN [15] dataset, covering most of the articulated objects in real world. Due to the irregular random point sampling pattern and the natural motion ambiguity, in some examples, our method may generate excessive rigid parts,

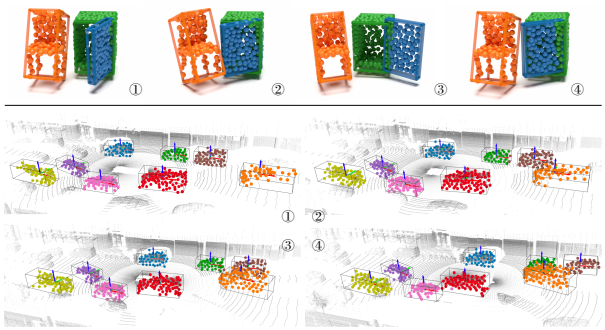

Figure S7. Quantitative demonstrations on complex scans. The first row is estimated using our trained articulated objects model while the last row is obtained by hierarchically apply our method to each segmented part until convergence. ①-④ indicates scan index. Best viewed with 200% zoom in.

which can be possibly eliminated by a carefully-designed post-processing step and is out of the scope of this work. We also show results from the DynLab dataset in Fig. S11. Our method can generate robust object associations under challenging settings.

References

[1] Federica Arrigoni and Tomas Pajdla. Motion segmentation via synchronization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 1, 2

[2] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 4

[3] Zan Gojcic, Caifa Zhou, Jan D Wegner, Leonidas J Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 4, 5

[4] David S Hayden, Jason Pacheco, and John W Fisher. Nonparametric object and parts modeling with lie group dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 7

[5] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012. 1

[6] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J Guibas, and Qixing Huang. Learning transformation synchronization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8082–8091, 2019. 4

[7] 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*, 1976. 4

[8] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–75, 2017. 3

[9] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9246–9255, 2019. 7

[10] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 4

[11] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 4

[12] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 7

[13] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 7

[14] 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 *European Conference on Computer Vision*, pages 88–107. Springer, 2020. 3

[15] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2020. 6, 7

[16] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Trans. Graph.*, 37(6), 2018. 4, 5, 7

[17] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6):1–12, 2016. 6

[18] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 4

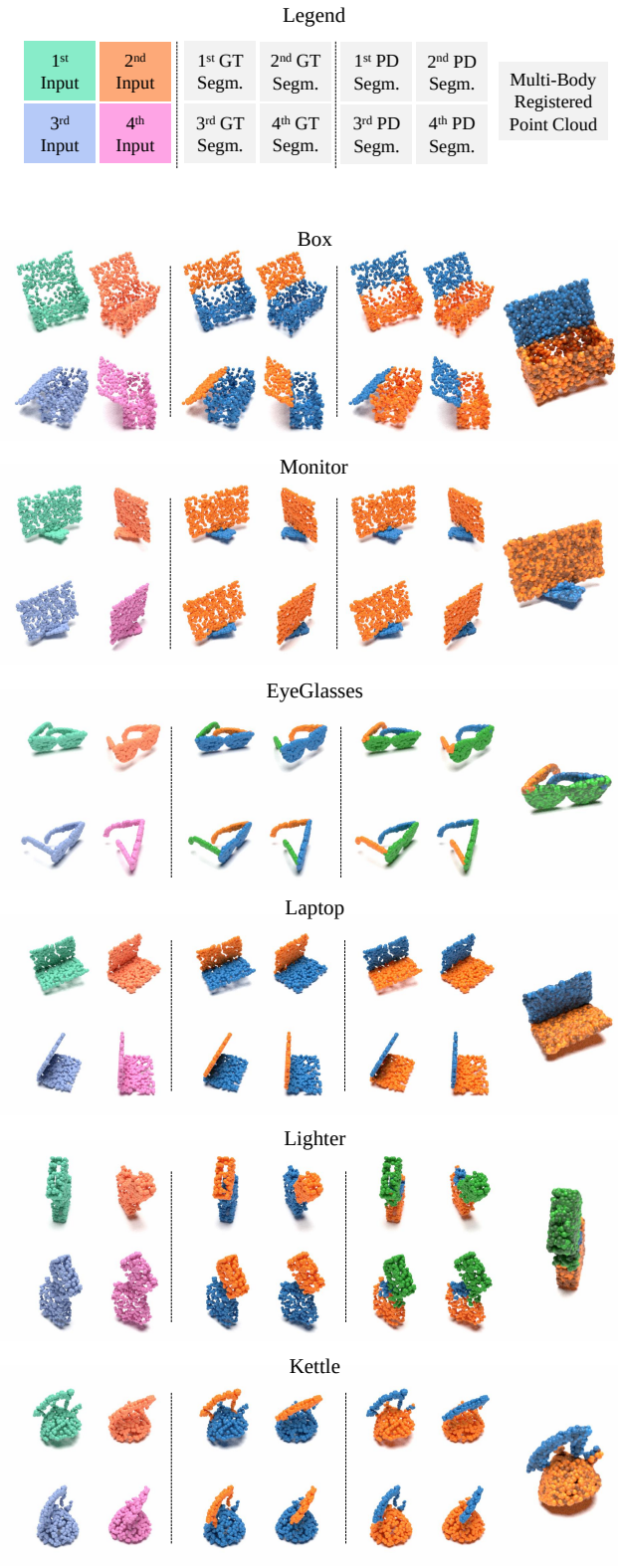

Figure S8. Qualitative results on SAPIEN dataset (1/3).

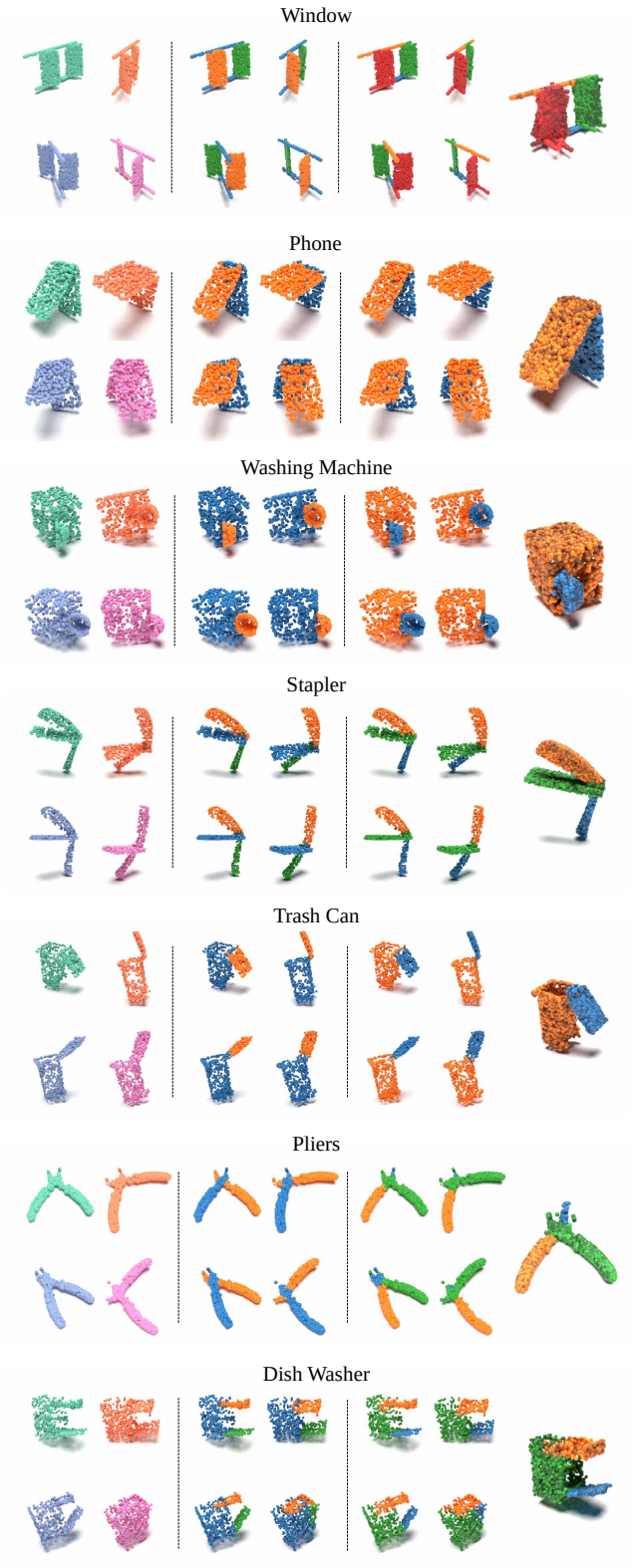

Figure S9. Qualitative results on SAPIEN dataset (2/3).

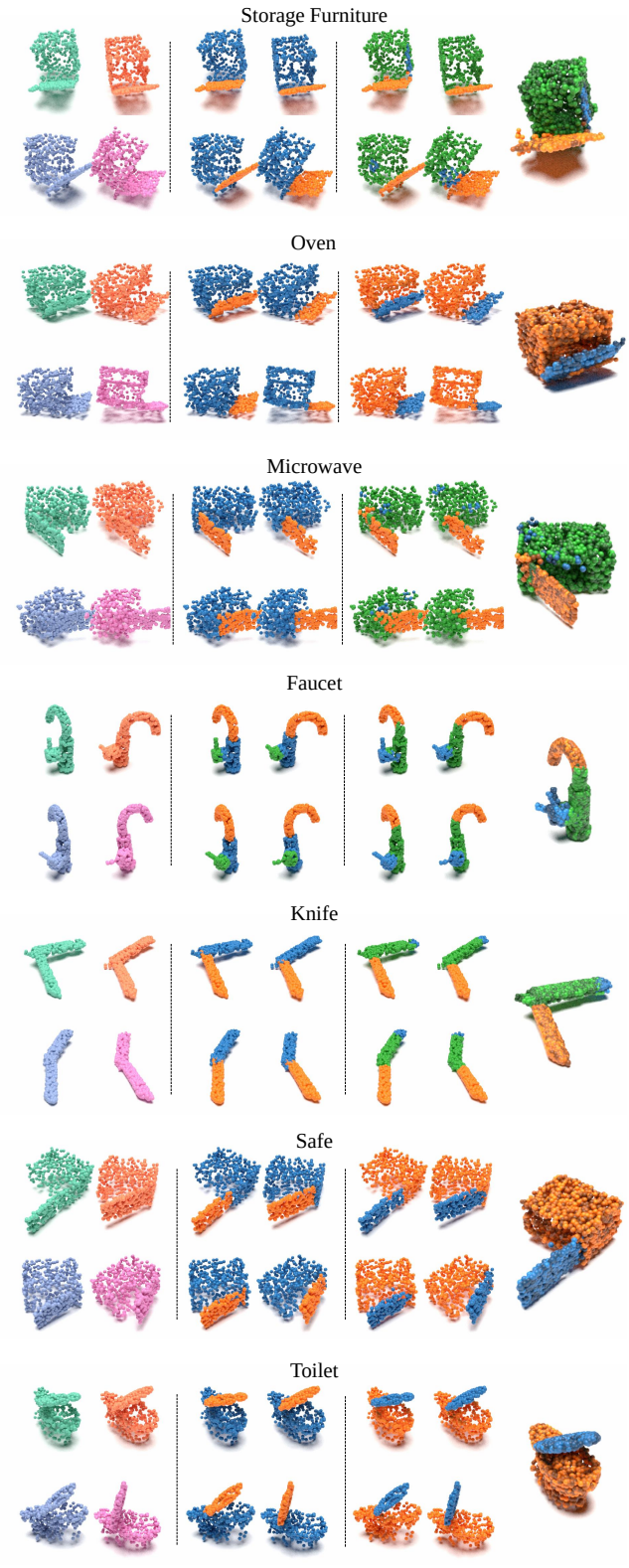

Figure S10. Qualitative results on SAPIEN dataset (3/3).

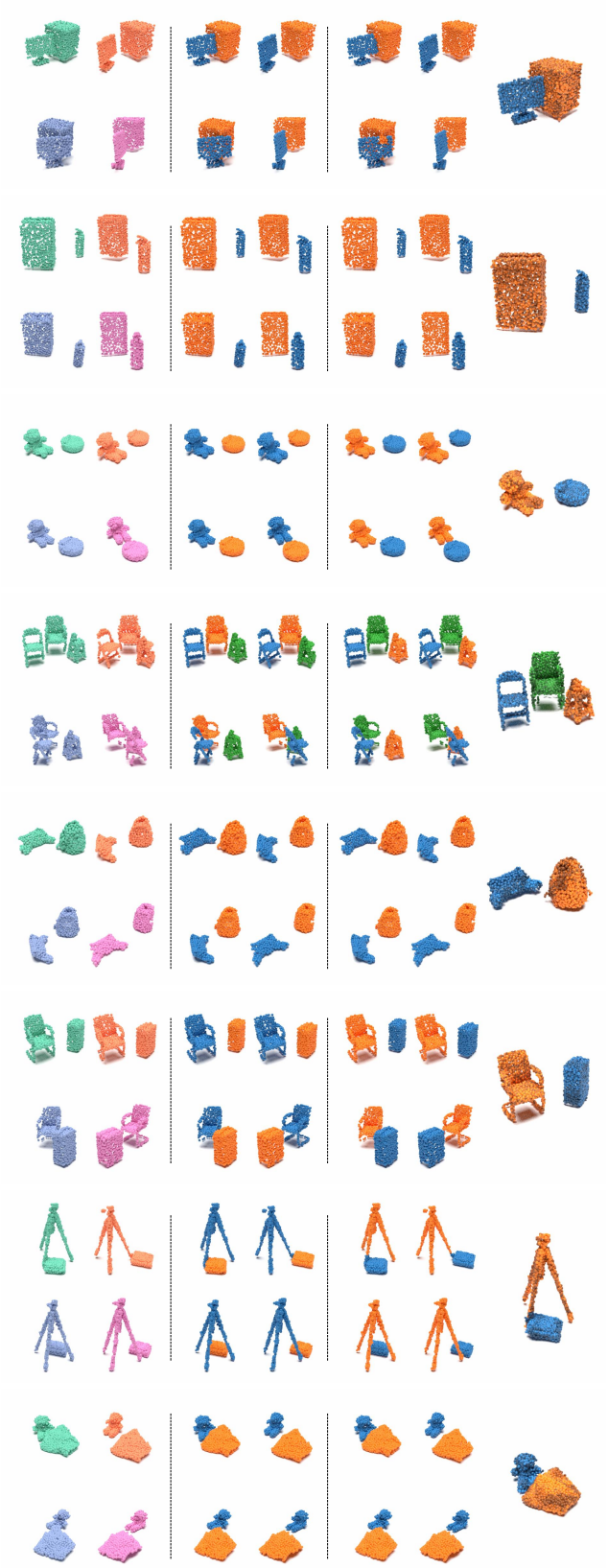

Figure S11. Qualitative results on DynLab dataset.