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

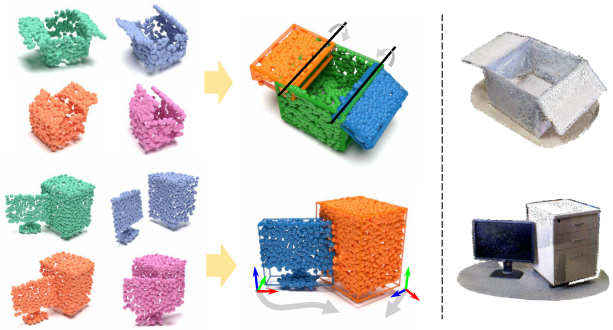
Jiahui Huang[1] He Wang[2] Tolga Birdal[2] Minhyuk Sung[3] Federica Arrigoni[4]
Shi-Min Hu[1] Leonidas Guibas[2]
[1]Tsinghua University [2]Stanford University [3]KAIST [4] University of Trento

Abstract

We present MultiBodySync, a novel, end-to-end trainable multi-body motion segmentation and rigid registration framework for multiple input 3D point clouds. The two non-trivial challenges posed by this multi-scan multi-body setting that we investigate are: (i) guaranteeing correspondence and segmentation consistency across multiple input point clouds capturing different spatial arrangements of bodies or body parts; and (ii) obtaining robust motion-based rigid body segmentation applicable to novel object categories. We propose an approach to address these issues that incorporates spectral synchronization into an iterative deep declarative network, so as to simultaneously recover consistent correspondences as well as motion segmentation. At the same time, by explicitly disentangling the correspondence and motion segmentation estimation modules, we achieve strong generalizability across different object categories. Our extensive evaluations demonstrate that our method is effective on various datasets ranging from rigid parts in articulated objects to individually moving objects in a 3D scene, be it single-view or full point clouds. Code at https://github.com/huangjh-pub/multibody-sync.

1. Introduction

Motion analysis in dynamic point clouds is an emerging area, required by various applications such as surveillance, autonomous driving, and robotic manipulation. Our human-made environments are dominated by *rigid body* movements, ranging from articulated objects to solids like furniture or vehicles. These settings require us to address rigid motions of objects or object parts – which is often referred to as the *multi-body* motion estimation problem. Despite its importance, previous work has mainly focused on specific scenarios with known category semantics, like category-level articulated object segmentation [44], indoor scene instance relocalization [70], or car movement detection [77], leaving the literature of generic motion segmentation relatively unexplored.



Input Point Clouds (Multi-Scan) | Multi-Body Segmentation & Motion Estimation | Colored Scan Visualization

Figure 1. MultiBodySync. Given an unordered set of point clouds, we simultaneously segment individual moving rigid parts/objects and register them. The transformed point clouds according to the first scan are aggregated and shown in the middle column. Note that the algorithm *does not* use color information and the right column is shown just for visualization.

Different from traditional single scan analysis algorithms like semantic segmentation [42], the most challenging part in multi-body motion analysis is to disambiguate and distinguish rigid bodies. There, we are naturally required to jointly process and relate *multiple* inputs, to effectively find *consistent* motion-based part/object segmentations as well as point correspondences to enable a multi-way registration. It is even more challenging when the capture is not temporally dense, *i.e.*, an intermittent acquisition that does not follow a stream such as a video, and might contain large pose variations, hampering naive temporal tracking.

In this paper, we introduce a multi-scan multi-body segmentation and motion estimation problem, where the goal is to simultaneously discover and register rigid bodies from multiple scans, represented either as full or partial point clouds, where objects come from unseen categories. As an effective solution, we present MultiBodySync, a fully end-to-end trainable deep declarative architecture [24] able to process an arbitrary number of unordered point sets. As shown in Fig. 1, given a set of scans, MultiBodySync begins relating pairs of scans via *3D scene flow* [83, 69] and con-

fidence estimation. Then, the following two differentiable (permutation and segmentation) synchronization modules, which are central to our approach, respectively enforce the consistency of pairwise point correspondences and motion segmentation labelings across different scans. Our design explicitly decouples geometry and motion, making MultiBodySync generalizable to *unseen categories* without sacrificing robustness.

We evaluate MultiBodySync on various datasets composed of full synthetic point clouds and partial real scans with articulated and solid objects. We also contribute a new dataset DynLab with 8 scenes and 64 scan fragments of distinctly moving objects. Our extensive evaluations demonstrate that our algorithm outperforms the state-of-the-art by a large margin on both multi-body motion segmentation and motion estimation. In brief, our contributions are:

1. We introduce a novel end-to-end trainable architecture for solving the multi-scan multi-body motion estimation and segmentation problem.
2. We theoretically analyze the spectral characteristics of the proposed weighted permutation synchronization.
3. To the best of our knowledge, we showcase the first cross-category generalization for the task at hand on both synthetic and real datasets, for both articulated part-level and object-level regimes.

2. Related Works

Dynamic scene understanding. The modeling of 3D dynamic scenes in deep learning literature is often formulated as a 4D data analysis, as done in seminal works like [46, 20]. Ability to infer spatiotemporal geometric properties has recently motivated research in *3D scene flow* as a form of low-level dynamic scene representation [45, 65, 74, 55, 51, 58, 47]. Domain-specific knowledge can be employed to give better predictions as done in autonomous driving [31, 5, 77] or articulated object analysis [82, 73]. The most recent dynamic SLAM works [33, 7, 86, 80] also rely heavily on semantic cues. While some works [51, 58] advocates continuous temporal-dynamics modeling, we instead assume discrete non-sequential input and enforce consistency using synchronization. Similarly, [27, 70] propose to perform instance-level re-localization in a changed scene. Nevertheless, we do not assume a pre-segmentation of the scene, but instead perform joint motion segmentation.

Multi-body motion. Provided point correspondences between two point clouds/images, rigid-body motion segmentation becomes a multi-model fitting problem, amenable for factorization techniques [21, 43, 81], clustering [34], graph optimization [48, 37, 12] or deep learning [41]. Among others, [83] handles raw scans and segments the rigidly moving parts using a Recurrent Neural Network (RNN). [29] fits non-parametric part models to sequential 3D data without needing explicit correspondences. However, to our best knowledge, no prior work can handle multiple scans while enforcing multi-way consistency like we do.

Synchronization. The art of consistently recovering absolute quantities from a collection of ratios is now a basic component of the classical multi-view/shape analysis pipelines [60, 15, 16]. Various aspects of the problem have been vastly studied: different group structures [26, 25, 13, 2, 1, 35, 28, 1, 71, 19, 64, 67, 4, 6], closed-form solutions [4, 2, 1], robustness [18], certifiability [59], global optimality [14], learning-to-synchronize [36, 54, 23] and uncertainty quantification [66, 11, 10, 13]. In this work, we are concerned with synchronizing correspondence sets, otherwise known as *permutation synchronization* (PS) [52] and motion segmentations [3]. PS is rich in the variety of algorithms: low-rank formulations [85, 72], convex programming [32], distributed optimization [32], multi-graph matching[61] or Riemannian optimization [13]. Out of all those, we are interested in the spectral methods of [2, 49] as they provide efficient, closed-form solutions deployable within a *deep declarative network* [24] like ours.

To the best of our knowledge, synchronization of correspondences [49] or motion segmentation [3] have not been explored in the context of deep learning. This is what we do in this paper to tackle the consistent multi-body motion estimation and segmentation.

3. Method

Problem setting and notation. Suppose we observe a set of K point clouds $\mathcal{X} = \{\mathbf{X}^k \in \mathbb{R}^{3\times N}, k \in [1, K]\}$ where each point cloud $\mathbf{X}^k = \left[\boldsymbol{x}_1^k, ..., \boldsymbol{x}_i^k, ..., \boldsymbol{x}_N^k\right]$ contains N points in $\mathbb{R}^3$ and sampled from the same object with S *independently moving* rigid parts indexed by s. Each point is assumed to belong to one of the S rigid parts and we denote the binary *point-part association matrices* as $\mathcal{G} = \{\mathbf{G}^k \in [0, 1]^{N\times S}\}$ where $G_{is}^k = 1$ if $\boldsymbol{x}_i^k$ belongs to the s^{th} rigid part and $G_{is}^k = 0$ otherwise[1]. The rigid motions for each part s in each point cloud k is defined as $\mathcal{T} = \{\mathbf{T}_s^k \in \text{SE}(3), k \in [1, K], s \in [1, S]\}$, with the rotational part being $\mathbf{R}_s^k \in \text{SO}(3)$ and the translational part being $\mathbf{t}_s^k \in \mathbb{R}^3$. Our final goal is to infer $\mathcal{G}$ and $\mathcal{T}$ given $\mathcal{X}$.

Summary. The core of our approach is a fully differentiable deep network fusing rigid dynamic information from multiple 3D scans as outlined in Fig. 2. We begin by explicitly predicting pairwise soft correspondences across all pairs of point clouds while enforcing consistency via a *weighted permutation synchronization* (§ 3.1). Next, the point clouds are segmented using a novel motion-based segmentation network and also further synchronized by a subsequent *motion segmentation synchronization* module (§ 3.2). Finally, the correspondences and segmentations are

[1]Throughout our paper we use superscript k, l to index point-clouds, subscript i, j to index points and subscript s to index rigid parts.

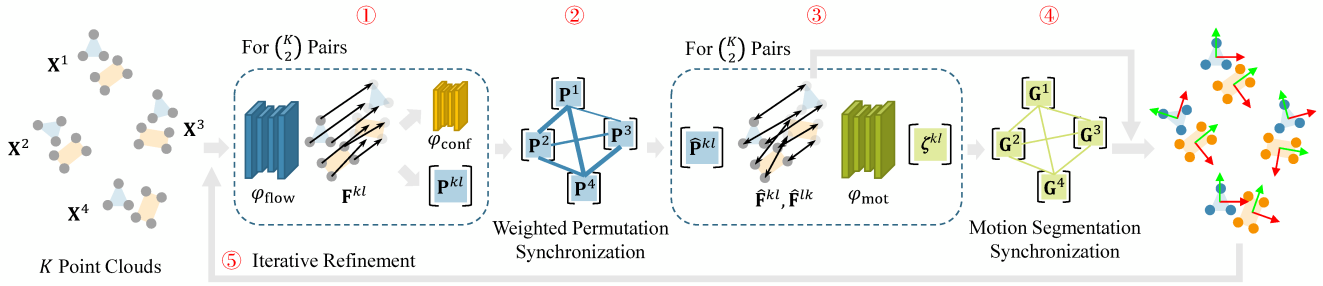


Figure 2. Our pipeline. ① Given multiple input point clouds, we first estimate pairwise scene flows. ② The point correspondences (permutations) computed from the flows are then synchronized in a weighted fashion to enforce consistencies. ③ Pairwise relative segmentations are subsequently estimated from the flows, and ④ further synchronized to get absolute motion segmentation. The pose of each part can be recovered by a weighted Kabsch algorithm. Our pipeline is fully differentiable and can be iterated (⑤) to get improved results.

used to recover the 6-DoF transformation for each of the individual rigid parts. The whole procedure can be iterated to refine the results. The pipeline can be readily trained end-to-end and we describe our training procedure in § 3.3.

3.1. Flow Estimation and Synchronization

Our approach starts with point correspondence estimation between all $\binom{K}{2}$ pairs of point clouds. We tackle this problem by predicting a 3D scene flow $\mathbf{F}^{kl} = \left[\boldsymbol{f}_1^{kl}, ..., \boldsymbol{f}_i^{kl}, ..., \boldsymbol{f}_N^{kl}\right] \in \mathbb{R}^{3\times N}$ for each point cloud pair indexed by (k, l) using a deep neural network $\varphi_{\text{flow}}(\cdot)$, *i.e.* $\mathbf{F}^{kl} = \varphi_{\text{flow}}(\mathbf{X}^k, \mathbf{X}^l)$, so that $\mathbf{X}^k + \mathbf{F}^{kl} \overset{P}{=} \mathbf{X}^l$ holds up to a permutation. The architecture of φ_{flow} inspired by *Point PWC-Net* [78] is detailed in the supplementary material.

Flow signals, estimated in a pairwise fashion, are not informed about the multiview configuration at our disposal. To ensure multi-way consistent flows, we employ the weighted variant of permutation synchronization [49] inspired by [23, 36] where a closed-form solution is given under spectral relaxation. We begin by the observation that any flow $\mathbf{F}^{kl}$ would induce a soft assignment matrix $\mathbf{P}^{kl} \in \mathcal{M}^{N\times N}$ based on the nearest-neighbor distances:

$$P_{ij}^{kl} = \frac{\exp\left(\delta_{ij}^{kl}/\tau\right)}{\sum_{\text{j}=1}^{N} \exp\left(\delta_{\text{ij}}^{kl}/\tau\right)}, \quad \delta_{ij}^{kl} = \|\boldsymbol{x}_i^k + \boldsymbol{f}_i^{kl} - \boldsymbol{x}_j^l\|^2 \quad (1)$$

where τ is the temperature of the *softmax*. The *multinomial manifold* $\mathcal{M}$ of row-stochastic matrices is a *continuous relaxation* of the (partial) permutation group $\mathcal{P}$.

Outlier filtering. To take into account the noise, missing points, or errors in the network, we further associate a confidence value $c_i^{kl} \in \mathbb{R}$ to each point $\boldsymbol{x}_i^k$ and its corresponding flow vector $\boldsymbol{f}_i^{kl}$ through another network $\varphi_{\text{conf}}(\cdot) : \mathbb{R}^{7\times N} \mapsto \mathbb{R}^N$ inspired from OANet [87]. The input to this network are the tuples $\{(\boldsymbol{x}_i^k, \boldsymbol{x}_i^k + \boldsymbol{f}_i^{kl}, \text{argmin}_j\, \delta_{ij}^{kl}) \in \mathbb{R}^7\}_{i=1}^N$ and we provide the architectural details in the supplementary. The last dimension of this tuple measures the quality of the flow vector via the distance between the transformed points and their nearest neighbors, thereby detecting spurious flow predictions. The final w^{kl} in Eq (3) reflects the overall quality of the corresponding $\mathbf{P}^{kl}$. Here we choose w^{kl} as the average confidence of all points, *i.e.*, $w^{kl} = \sum_{i=1}^N c_i^{kl}/N$.

Consistent correspondences. We now use the predictions $\{\mathbf{P}^{kl}, w^{kl}\}_{(k,l)}$ to achieve multiview consistent assignments. To this end, we deploy a *differentiable synchronization* algorithm inspired by [49]. We first introduce *absolute* permutation matrices $\mathbf{P}^k$ which map each point in $\mathbf{X}^k$ to a *universe* space and stack them as $\boldsymbol{p} = [\ldots, (\mathbf{P}^k)^\top, \ldots]^\top$. We solve for the best $\boldsymbol{p}$ minimizing:

$$E(\boldsymbol{p}) = \sum_{k=1}^{K}\sum_{l=1}^{K} w^{kl}\|\mathbf{P}^k - \mathbf{P}^{kl}\mathbf{P}^l\|_F^2. \quad (2)$$

Theorem 1 (Weighted synchronization)**.** *The spectral solution to the* weighted synchronization problem *in Eq* (2) $\boldsymbol{p}$ *is given by the* N *eigenvectors of* $\mathbf{L}$ *corresponding to the smallest* N *eigenvalues, where* $\mathbf{L} \in \mathbb{R}^{KN\times KN}$ *is the* ***weighted*** *Graph Connection Laplacian (GCL) constructed by tiling all* $\mathbf{P}^{kl}$ *matrices weighted by the related* w^{kl}*:*

$$\mathbf{L} = \begin{bmatrix} w^1\mathbf{I}_N & -w^{12}\mathbf{P}^{12} & \ldots & -w^{1K}\mathbf{P}^{1K} \\ -w^{21}\mathbf{P}^{21} & w^2\mathbf{I}_N & \ldots & -w^{2K}\mathbf{P}^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ -w^{K1}\mathbf{P}^{K1} & -w^{K2}\mathbf{P}^{K2} & \ldots & w^K\mathbf{I}_N, \end{bmatrix}, \quad (3)$$

with $w^k = \sum_{l=1,l\neq k}^K w^{kl}$ *and* $\mathbf{I}_N \in \mathbb{R}^{N\times N}$ *is the identity.*

Proof. Please refer to the supplementary material. □

This *spectral solution* requires only an eigen-decomposition lending itself to easy differentiation [36, 23]. The synchronized soft correspondence $\hat{\mathbf{P}}^{kl}$ is then extracted as the (k, l)-th $N \times N$ block of $\boldsymbol{p}\boldsymbol{p}^\top$. As a consequence of the relaxation, we cannot ensure that each sub-matrix of $\boldsymbol{p}\boldsymbol{p}^\top$ would be a valid permutation. To preserve differentiability we avoid Hungarian-like projection operators [49] and propose to directly compute the induced flow

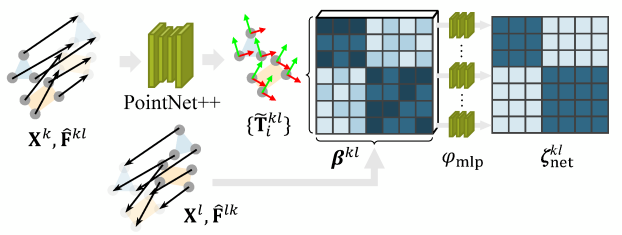


Figure 3. Illustration of our motion segmentation network φ_{mot}.

$\hat{\mathbf{F}}^{kl} = \left[..., \hat{\boldsymbol{f}}_i^{kl}, ...\right]$ using a softmax normalization on the synchronized soft correspondences:

$$\hat{\boldsymbol{f}}_i^{kl} = \frac{\sum_{j=1}^{N} \bar{P}_{ij}^{kl}(\boldsymbol{x}_j^l - \boldsymbol{x}_i^k)}{\sum_{j=1}^{N} \bar{P}_{ij}^{kl}}, \quad \bar{P}_{ij}^{kl} = \exp\left(\hat{P}_{ij}^{kl}/t\right). \quad (4)$$

Intuitively, this amounts to using the normalized synchronized result as a soft-assignment matrix, diminishing the effect of non-corresponding matches (false positives).

3.2. Motion Segmentation

Based upon the multiview consistent flow output $\hat{\mathbf{F}}^{kl}$, we now predict the point-part associations $\mathcal{G}$. Since we are not provided with consistent labeling of the parts, instead of predicting $\mathbf{G}^k$ directly, we estimate for all $\binom{K}{2}$ point cloud pairs a relative *motion segmentation matrix* $\boldsymbol{\zeta}^{kl} \in [0,1]^{N\times N}$, where ζ_{ij}^{kl} is 1 when $\boldsymbol{x}_i^k$ and $\boldsymbol{x}_j^l$ belong to the same rigid body, and 0 otherwise.

Our motion segmentation network $\varphi_{\text{mot}}(\cdot) : \mathbb{R}^{12\times N} \mapsto \mathbb{R}^{N\times N}$ illustrated in Fig. 3 takes the point cloud pair $\mathbf{X}^k, \mathbf{X}^l$ as well as flow $\hat{\mathbf{F}}^{kl}, \hat{\mathbf{F}}^{lk}$ estimated from the last step as input and outputs the matrix $\boldsymbol{\zeta}^{kl}$. It begins with a PointNet++ [57] predicting a transformation $\tilde{\mathbf{T}}_i^{kl}$ for each point in $\boldsymbol{x}_i^k \in \mathbf{X}^k$[2]. The predictions map the part in $\mathbf{X}^k$ containing $\boldsymbol{x}_i^k$ to $\mathbf{X}^l$. We then compute a residual matrix $\boldsymbol{\beta}^{kl} \in \mathbb{R}^{3\times N\times N}$ based on $\tilde{\mathbf{T}}_i^{kl}$, whose element is:

$$\boldsymbol{\beta}_{ij}^{kl} = (\tilde{\mathbf{T}}_i^{kl})^{-1} \circ \boldsymbol{x}_j^l - (\boldsymbol{x}_j^l + \hat{\boldsymbol{f}}_j^{lk}), \quad (5)$$

where $\circ$ denotes the action of $\tilde{\mathbf{T}}$. One can easily verify that the smaller the norm of the (i,j)-th entry of $\boldsymbol{\beta}^{kl}$ is, the more likely that $\boldsymbol{x}_i^k$ and $\boldsymbol{x}_j^l$ are in the same rigid part. Therefore, it contains valuable information for deducing the motion segmentation $\boldsymbol{\zeta}^{kl}$. We apply N denoising mini-PointNet [56] $\varphi_{\text{mlp}}(\cdot)$ to each horizontal $3\times N$ slice of $\boldsymbol{\beta}^{kl}$, concatenated with $\mathbf{X}^l$ to get a likelihood score for each pair of points $(\boldsymbol{x}_i^k \in \mathbf{X}^k, \boldsymbol{x}_j^l \in \mathbf{X}^l)$. The network output $\boldsymbol{\zeta}_{\text{net}}^{kl}$ is subsequently computed by applying a sigmoid on the output:

$$(\boldsymbol{\zeta}_{\text{net}}^{kl})_{i,:} = \text{sigmoid}\left(\varphi_{\text{mlp}}([\boldsymbol{\beta}_{i,:}^{kl}, \mathbf{X}^l])\right). \quad (6)$$

[2]In practice, instead of predicting $\tilde{\mathbf{T}}_i^{kl}$ directly, we estimate a residual motion w.r.t. the already obtained flow vectors similar to the method in [83]. This procedure is detailed in our supplementary material.

Motion segmentation consistency. Given all pairwise motion information $\boldsymbol{\zeta}^{kl}$, we adopt the method of Arrigoni and Pajdla [3] to compute an absolute motion segmentation $\boldsymbol{g} \in \mathbb{R}^{KN\times S}$ as a stack of matrices in $\mathcal{G}$. Once again, this is an instance of a synchronization problem, with the stacked relative and absolute motion segmentation matrices being:

$$\mathbf{Z} = \begin{bmatrix} \mathbf{0} & \boldsymbol{\zeta}^{12} & \dots & \boldsymbol{\zeta}^{1K} \\ \boldsymbol{\zeta}^{21} & \mathbf{0} & \dots & \boldsymbol{\zeta}^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\zeta}^{K1} & \boldsymbol{\zeta}^{K2} & \dots & \mathbf{0} \end{bmatrix}, \quad \boldsymbol{g} = \begin{bmatrix} \mathbf{G}^1 \\ \mathbf{G}^2 \\ \vdots \\ \mathbf{G}^K \end{bmatrix}, \quad (7)$$

A spectral approach similar to the one in § 3.1 optimizes for $\boldsymbol{g}$ so that $\mathbf{Z} = \boldsymbol{g}\boldsymbol{g}^\top$ is best satisfied. Then, $\boldsymbol{g}$ is just the S leading eigenvectors of $\mathbf{Z}$, scaled by the square root of its S largest eigenvalues. Here, the point-part association matrices $\mathbf{G}^k$ are relaxed to *fuzzy* segmentations by allowing its entries to take real values. As a subsequent step similar to § 3.1, we replace the projection step with a row-wise softmax on $\boldsymbol{g}$ to maintain differentiability.

Note that the output $\boldsymbol{\zeta}_{\text{net}}^{kl}$ of φ_{mot} is *unnormalized*, meaning that any submatrix in $\mathbf{Z}$ can be written as $\boldsymbol{\zeta}^{kl} = \sigma^{kl}\boldsymbol{\zeta}_{\text{net}}^{kl}$, where σ acts as a normalizer. This is akin to encoding a *confidence* in the norm of the matrix $\boldsymbol{\zeta}_{\text{net}}^{kl}$ and requires us to solve a weighted synchronization. However, as we prove in the following theorem, such a solution would involve an anisotropic scaling in the eigenvectors as a function of the number of points belonging to each part. As this piece of information is not available in runtime, we take an alternative approach and approximate the scaling factor as $q^{kl} = \text{mean}(\boldsymbol{\zeta}_{\text{net}}^{kl})$ and *pre-factor* it out of $\boldsymbol{\zeta}_{\text{net}}^{kl}$, by letting $\boldsymbol{\zeta}^{kl} = \boldsymbol{\zeta}_{\text{net}}^{kl}/q^{kl}$. In this way, we ensure that the eigenvectors yield the synchronized motion segmentation.

Theorem 2. *Under mild assumptions, the solution to the segmentation synchronization problem using a non-uniformly weighted matrix will result in a proportionally scaled version of the solution obtained by the eigenvectors of the unweighted matrix* $\mathbf{Z}$.

Proof. Please refer to the supplementary material. □

As we show in our supplement, entry k in the decomposed eigenvalues is related to the number of points belonging to motion k. To compute the number of rigid bodies S, *i.e.*, determine how many eigenvectors to use in $\boldsymbol{g}$, the spectrum of $\mathbf{Z}$ is analyzed during test time: We estimate S as the number of eigenvalues that are larger than α-percent of the sum of the first 10 eigenvalues. For training, we just fix $S = 6$ as an over-parametrization.

Pose Computation and Iterative Refinement. We finally estimate the motion for each part using a weighted Kabsch algorithm [39, 23] followed by a joint pose estimation. During test time we also iterate our pipeline several times to

gradually refine the correspondence and segmentation estimation by transforming input point clouds according to the estimated $\mathcal{T}$ and adding back the residual flow onto the flow predicted at the previous iteration. The details are provided in our supplementary.

3.3. Network Training

We propose to train each learnable component of our pipeline separately in a pairwise manner and then fine-tune their parameters using the full pipeline. Specifically, we first train the flow estimation network φ_{flow} supervised with ground-truth flow: $\mathcal{L}^{kl}_{\text{flow}} = \left\|\mathbf{F}^{kl} - \mathbf{F}^{kl,\text{gt}}\right\|^2_F$. Given the trained φ_{flow}, the confidence estimation network φ_{conf} is trained based on its output using a binary cross-entropy (BCE) loss supervised by comparing whether the error of the predicted flow is under a certain threshold:

$$\mathcal{L}^{kl}_{\text{conf}} = \sum_{i=1}^{N} \text{BCE}(c_i^{kl,\text{gt}}, c_i^{kl}), \tag{8}$$

with $c_i^{kl,\text{gt}} = 1$ if we have $\|\boldsymbol{f}_i^{kl} - \boldsymbol{f}_i^{kl,\text{gt}}\|_2^2 < \epsilon_{\boldsymbol{f}}$ and 0 otherwise. The motion segmentation network φ_{mot} is trained using joint supervision over the estimated transformation residual and the final motion segmentation matrix: $\mathcal{L}^{kl}_{\text{seg}} = \mathcal{L}^{kl}_{\text{trans}} + \mathcal{L}^{kl}_{\text{group}}$ where each term is defined as:

$$\mathcal{L}^{kl}_{\text{trans}} = \sum_{i=1}^{N} \frac{\sum_{j=1}^{N} \mathbb{I}(\zeta_{ij}^{kl} = 1) \left\|\boldsymbol{\beta}_{ij}^{kl}\right\|_2^2}{\sum_{j=1}^{N} \mathbb{I}(\zeta_{ij}^{kl} = 1)}, \tag{9}$$

$$\mathcal{L}^{kl}_{\text{group}} = \sum_{i=1}^{N}\sum_{j=1}^{N} \text{BCE}(\zeta_{ij}^{kl,\text{gt}}, \zeta_{ij}^{kl}). \tag{10}$$

After we train all the networks (*i.e.*, φ_{flow}, φ_{conf} and φ_{mot}), the entire pipeline is trained end-to-end with the supervision on both the pariwise flow $\sum_{k=1}^{K}\sum_{l=1}^{K}\mathcal{L}^{kl}_{\text{flow}}$ and the IoU (Intersection-over-union) loss, defined as:

$$\mathcal{L}_{\text{iou}} = \underset{\mathbf{A}}{\text{argmin}} \sum_{s,s'=1}^{S\times S} \frac{\mathbf{A}(s,s') \cdot (\boldsymbol{g}^{\text{gt}}_{:,s})^\top \boldsymbol{g}_{:,s'}}{\|\boldsymbol{g}^{\text{gt}}_{:,s}\|_2^2 + \|\boldsymbol{g}_{:,s'}\|_2^2 - (\boldsymbol{g}^{\text{gt}}_{:,s})^\top \boldsymbol{g}_{:,s'}},$$

where $\mathbf{A}$ is an $S \times S$ binary assignment matrix which we found using the Hungarian algorithm. The flow supervision is added to both the output of flow network, and the final pairwise *rigid flow* computed as $\boldsymbol{f}_i^{kl} = \mathbf{T}_s^l(\mathbf{T}_s^k)^{-1} \circ \boldsymbol{x}_i - \boldsymbol{x}_i$.

4. Experiments

Datasets. Our algorithm is tested on two main datasets: SAPIEN [79] and *DynLab* dataset contributed by this work: **SAPIEN** consists of realistic simulated articulated models with part mobility annotated. We ensure that the categories used for training and validation do not overlap with the test set, finally leading to 720 articulated objects with 20 different categories. We then perform K virtual 3D scans of the models, with each scan capturing the same object with a different camera (and hence object) pose and object articulating state. Later, furthest point sampling is applied to down-sample the number of points to N. **DynLab** (Dynamic Laboratory) contains 8 different scenes in a laboratory, each with 2-3 rigidly moving *solid* objects from various categories. Each of the scenes is captured 8 times, reconstructed using ElasticFusion [76] and between each capture, the object positions are randomly changed. The dataset also contains manual annotations of the object segmentation mask and rigid absolute transformations. For benchmarking, in each scene we choose different combinations of the 8 captures, leading to a total of $8 \cdot \binom{8}{4} = 560$ dataset items. We believe the two different scenarios (articulated single object and moving rigid bodies) reflected in the test sets are sufficient to verify the robustness and the general applicability of our algorithm.

Table 1. Rigid flow estimation on SAPIEN. Table reports mean and std. dev. (+/-) of the EPE3D over all pairwise flows, with (S) or w/o (NS) **S**ynchronization and with (W) or w/o (NW) **W**eighting.

	Deep Part [83]	NPP [29]	Ours		
			NS, NW	S, NW	S, W
Mean	5.95	21.22	6.20	6.08	**5.03**
+/-	3.57	6.29	4.06	3.47	**2.00**

The training data for articulated objects are generated using the dataset from [84], containing manually annotated semantic segmentation of 16 categories. Similar to [83], we generate K random motions for each connected semantic part of the shapes. For the training data of solid objects, we randomly sample independent motions for multiple objects taken from ShapeNet [17] as if they are floating and rotating in the air. Please refer to supplementary material for detailed data specifications and visualizations.

Metrics. Two main metrics are used: (1) **EPE3D** (End-Point Error in 3D) of all $\binom{K}{2}$ pairs of point clouds. The mean and standard deviation (+/-) measures the rigid 3D flow estimation quality: While the mean reflects an overall error in the transformation, the standard deviation shows how consistent the estimate is among all pairs - a desirable property in the multi-scan setting. (2) Segmentation accuracy assesses the motion segmentation quality. We use **mIoU** (mean Intersection-over-Union) and **RI** (Rand Index) to score the output based on 'Multi-Scan' and 'Per-Scan' segmentations. For '**Multi-Scan**', we evaluate the points from all K clouds altogether, revealing the consistency of the labeling across multiple scans. For '**Per-Scan**', we compute the score for each of the clouds separately and evaluate the mean and standard deviation across all scans.

Training. φ_{flow}, φ_{mot} and φ_{conf} are trained using Adam optimizer with initial learning rate of 10^{-3} and a 0.5/0.7/0.7 decay every 400K iterations for the three networks. The

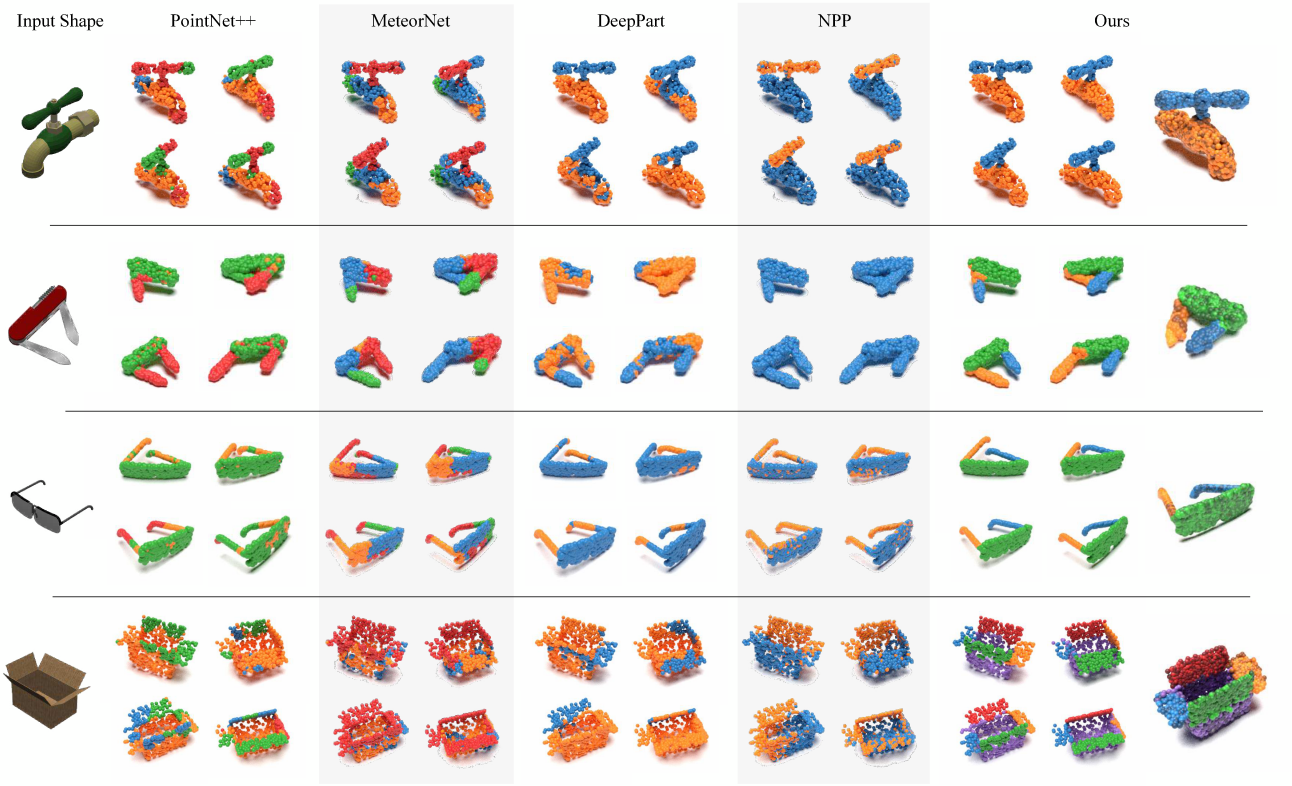


Figure 4. Qualitative results on SAPIEN [79] dataset. On the left-most column, we show reference rendering of the objects we tackle. Each method span two columns and the point colors show the motion segmentation. For our method, we additionally show the registration result as the last column, where the darkness of the points shows the point cloud index it comes from.

Table 2. Segmentation Accuracy on SAPIEN Dataset.

	Multi-Scan		Per-Scan	
	mIoU	RI	mIoU	RI
PointNet++ [57]	47.5	0.62	51.2±12.1	0.65±0.09
MeteorNet [46]	43.7	0.59	45.7±5.4	0.60±0.03
DeepPart [83]	49.2	0.64	53.0±8.9	0.67±0.06
NPP [29]	48.2	0.63	51.5±6.6	0.66±0.05
Ours (4 iters)	**66.7**	**0.76**	**67.3±4.3**	**0.77±0.03**

batch sizes are set to 32/8/32, respectively. The entire pipeline is trained end-to-end using $K = 4$ point clouds, with a learning rate of 10^{-6}. The gradient computation for eigen-decomposition will sometimes lead to numerical instabilities [22], so we roll back that iteration when the gradient norm is large. Our algorithm is implemented using PyTorch [53] with $N = 512$, $\tau = 0.01$, $\epsilon_f = 0.1$. We set $\alpha = 0.05$ for articulated objects and $\alpha = 0.15$ for solid objects.

4.1. Results on Articulated Objects

Baselines. Given our new multi-scan multi-body setting, we made adaptations to previous methods and compared to the following 4 baselines: (1) **PointNet++** [57]: We use the segmentation backbone to directly predict $\mathbf{G}^k$ matrices. We aggregate the bottleneck features by taking the max before feeding it to the individual K feature propagation modules. (2) **MeteorNet** [46]: We use the *MeteorNet-seg* model proposed to directly predict the segmentations. Both PointNet++ and MeteorNet are supervised with the IoU loss (Eq (11)) which counts in the ambiguity of rigid body labeling. (3) **DeepPart** [83]: As this method only allows pairwise input, we associate multiple point clouds using sequential label propagation. (4) **NPP** (Non-Parametric Part) [29]: This algorithm does not need training and a grid search is conveyed for its many tunable parameters.

Flow Accuracy. Tab. 1 shows that despite being based on [83], our method gives the lowest flow error and variance across different view pairs. This is thanks to the correspondence consistency among the provided K scans enforced by our synchronization module. The NPP method suffers from a surprisingly high flow error mainly because the point-level correspondence is not explicitly modeled. Note that PointNet++ and MeteorNet are excluded because they only output point-wise segmentations.

Segmentation Accuracy. For the segmentation benchmark, we achieve a significantly better result than all the baselines as shown in Tab. 2. Among the baselines, Me-

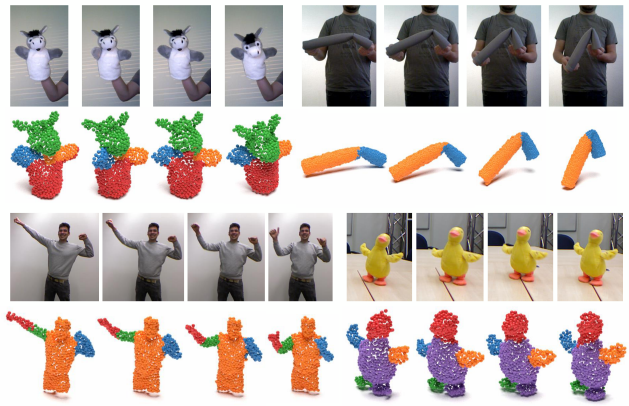


Figure 5. Qualitative segmentation results on two articulated sequences from [68] ('Donkey' and 'Pipe 3/4', first two rows) and [62] ('Alex' and 'Duck', last two rows).

teorNet fails because it assumes proximity of relevant data in the given point clouds, which is not robust to SAPIEN dataset because of change in both object pose and articulated parts. Even though PointNet++ reaches a relatively high mean score, the standard deviation and Multi-Scan score show the segmentation is not consistent across different input scans. DeepPart is specially designed for part-based motion segmentation, but only operates on two-views, which can cause drastic performance degradation if the input two-views have a large difference. Also the RNN they proposed for part segmentation tends to generate short sequences and most of the shapes are only divided into two parts. Despite the large error in flow estimation, NPP behaves reasonably in terms of segmentation. Qualitative comparisons are visualized in Fig. 4.

One important aspect of our network is that it can generalize to different objects and motions *without re-training*. To qualitatively showcase this, we use two additional dynamic RGB-D sequences from [68] and [62]. For each sequence, we use four views and back-project the depth map into point clouds for inference. As shown in Fig. 5, our model trained on full objects of synthetic SAPIEN dataset, can generalize to real dynamic depth sequences producing consistent motion-based segmentation. This is possible thanks to the property that our network anchors on the motion and not on the specific geometry.

4.2. Results on Full Objects

In DynLab, each rigid body (*i.e.* object) is now semantically meaningful, so apart from the 4 baseline methods from § 4.1, we additionally compare to the following two alternatives: (5) **InstSeg** (Instance Segmentation): We take the state-of-the-art indoor semantic instance module PointGroup [38] trained on ScanNet dataset to segment for each input cloud. (6) **Geometric**: We use the Ward-linkage [75] to agglomeratively cluster the points in each scan. In order to obtain consistent segmentation across multiple inputs, we

Table 3. Segmentation Accuracy on DynLab dataset.

	Multi-Scan		Per-Scan	
	mIoU	RI	mIoU	RI
PointNet++ [57]	37.2	0.53	39.4±7.1	0.54±0.03
MeteorNet [46]	58.5	0.69	71.8±9.7	0.76±0.06
DeepPart [83]	60.7	0.70	66.3±17.2	0.75±0.13
NPP [29]	65.7	0.74	71.6±7.7	0.78±0.05
Geometric	83.1	0.87	88.6±5.8	0.91±0.04
InstSeg [38]	56.5	0.66	72.4±12.5	0.78±0.09
Ours	**90.7**	**0.95**	**94.0±3.1**	**0.96±0.02**

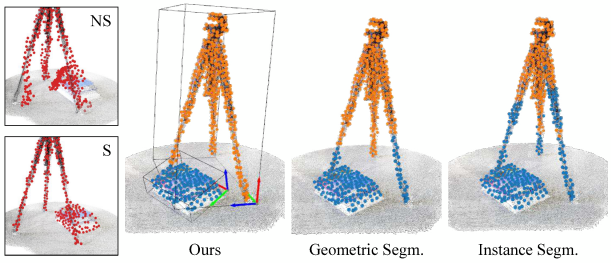


Figure 6. Example comparisons to baselines on DynLab. In the leftmost column we compare the warped point clouds without ('NS') and with ('S') synchronization. The right three sub-figures show the segmentation in different colors. For clarity we exclude the computed pose for the *geometric* & *InstSeg* approaches because the inaccuracy in segmentation leads to very noisy poses.

associate the segmentations between two different scans using a Hungarian search over the object assignment matrix, whose element is the *root mean squared error* measuring the fitting quality between any combinations of the object associations.

Interestingly, as listed in Tab. 3, all the previous deep methods lead to unsatisfactory results on this dataset. PointNet++ and MeteorNet are found to be inaccurate because by design they associate labels in the level of semantics (not motion) and no explicit consistencies across scans are considered. Even though the InstSeg method is trained on large-scale scene dataset, it is impossible for it to cover all real-world categories so wrong detections are observed in some scenes. The geometric approach is less robust in cluttered scenes where no obvious geometric cues can be used. Our method is motion-induced and is hence robust to geometric variations and out-of-distribution semantics, outperforming all baselines. A typical failure scenario for these approaches is visualized in Fig. 6. We show additional qualitative results in Fig. 7, demonstrating our ability to accurately segment, associate, and compute correct object transformations even if there are large pose changes.

Tab. 4 shows the rigid flow estimation result against the baselines. Apart from the influence of wrong per-scan segmentation and cross-scan associations, the iterative closest point (ICP) [8] method used to register object scans can also suffer from poor initializations. Our approach not only

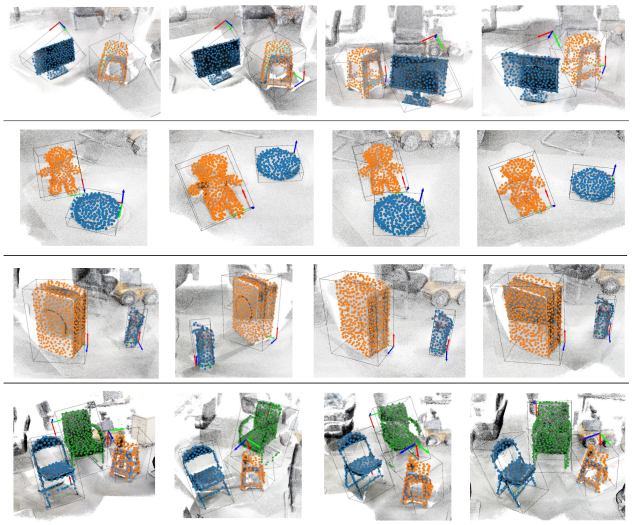


Figure 7. Results on DynLab dataset. Note that we detect and remove the ground for all baselines except InstSeg so only the points on the moving furniture are input. Point colors indicate segmentation and the bounding boxes show relative transformations.

Table 4. Rigid flow estimation result on DynLab dataset. The numbers represent mean and standard deviation (+/-) of the EPE3D from all pairwise flows. Note the value here does not reflect real-world metrics because the scales are uniformly normalized.

	DeepPart [83]	NPP [29]	Geometric	InstSeg [38]	Ours
Mean	16.89	51.14	21.61	46.40	**11.01**
+/-	11.39	18.38	9.76	20.73	**6.65**

reaches the lowest mean error, but also respects the motion consistency across multiple scans.

4.3. Ablation Study and System Analysis

Effect of synchronization. For permutation synchronization (§ 3.1), we can directly feed the network-predicted flow vector $\mathbf{F}^{kl}$ to subsequent steps instead of using synchronized $\hat{\mathbf{F}}^{kl}$ (Ours: NS, NW), or use an unweighted version of the synchronization by setting all $w^{kl} = 1$ (Ours: S, NW). However, as shown quantitatively in Tab. 1, both variants result in higher flow error due to the failure to find consistent correspondences. Similar results can be observed on DynLab dataset as demonstrated in the two sub-figures of Fig. 6, where direct flow prediction failed because the geometric variation is too large between two scans.

Effect of K. Our method can be naturally applied to an arbitrary number of views K even if we train using 4 views, because by design the learnable parameters are unaware of the input counts. As shown in Fig. 9, the segmentation accuracy improves given more views. This is because the introduction of additional scans helps build the connection between existing scans and benefits the 'co-segmentation' process.

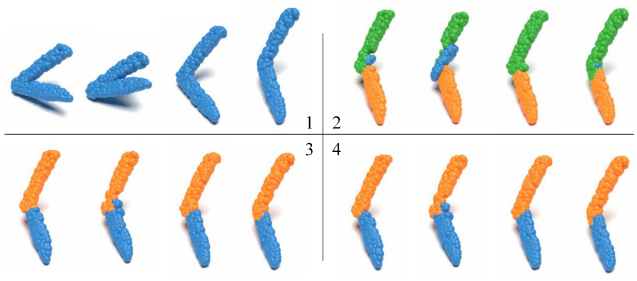


Figure 8. Iterative refinement on SAPIEN dataset. We show transformed and segmented point clouds according to recovered motions over iterations (shown in the middle).

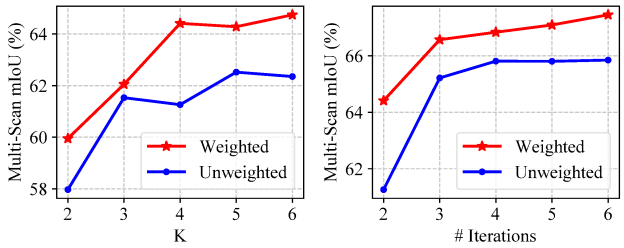


Figure 9. Influence of number of iterations (**left**) and number of views (**right**) on the final segmentation accuracy.

Number of iterations. As pointed out in § 3.2, our pipeline can be run multiple iterations to refine the results and an example is given in Fig. 8. Shown in Fig. 9, our method works better with more iterations because we estimate more accurate flows. Moreover, more iterations are demonstrated to be unnecessary because previous iterations already lead to converged results.

Timing. Our experiments are conducted using an Nvidia GeForce GTX 1080 card. For the input of 4 scans, the running time of our full model is ∼870ms *per iteration*. The entirety of a 4-iteration scheme hence takes ∼3.5s, while [83] and [29] take 11.5s and 60s resp. in comparison.

5. Conclusion

We presented *MultiBodySync*, a pipeline for simultaneously segmenting and registering multiple dynamic scans with multiple rigid bodies. We, for the first time, incorporated weighted permutation synchronization and motion segmentation synchronization into a fully-differentiable pipeline for generating consistent results across all input point clouds. However, currently MultiBodySync is not scalable to a large number (like hundreds) of scans or rigid bodies. Future directions include improvement of the pipeline's scalability and robustness in more complicated and dynamic settings.

Acknowledgements. We ack support from the China Scholarship Council, the Natural Science Foundation of China (No. 61521002), the Joint NSFC-DFG Research Program (No. 61761136018), a grant from Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, NSF grant CHS-1528025, a Vannevar Bush Faculty fellowship, a TUM/IAS Hans Fischer senior fellowship, and gifts from the Adobe, Amazon AWS, and Snap corporations. Arrigoni was supported by the EU Horizon 2020 Research and Innovation Programme under project SPRING (No. 871245).

References

[1] Federica Arrigoni and Andrea Fusiello. Synchronization problems in computer vision with closed-form solutions. *International Journal of Computer Vision*, Sep 2019. 2

[2] Federica Arrigoni, Eleonora Maset, and Andrea Fusiello. Synchronization in the symmetric inverse semigroup. In *International Conference on Image Analysis and Processing*, pages 70–81. Springer, 2017. 2

[3] Federica Arrigoni and Tomas Pajdla. Motion segmentation via synchronization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 2, 4, 13, 14

[4] Federica Arrigoni, Beatrice Rossi, and Andrea Fusiello. Spectral synchronization of multiple views in se (3). *SIAM Journal on Imaging Sciences*, 9(4):1963–1990, 2016. 2

[5] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7962–7971, 2019. 2

[6] Florian Bernard, Johan Thunberg, Peter Gemmar, Frank Hertel, Andreas Husch, and Jorge Goncalves. A solution for multi-alignment by transformation synchronisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2

[7] Berta Bescos, Carlos Campos, Juan D Tardós, and José Neira. Dynaslam ii: Tightly-coupled multi-object tracking and slam. *arXiv preprint arXiv:2010.07820*, 2020. 2

[8] 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. International Society for Optics and Photonics, 1992. 7

[9] 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. 16

[10] Tolga Birdal, Michael Arbel, Umut Simsekli, and Leonidas J Guibas. Synchronizing probability measures on rotations via optimal transport. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[11] Tolga Birdal, Umut Şimşekli, M. Onur Eken, and Slobodan Ilic. Bayesian Pose Graph Optimization via Bingham Distributions and Tempered Geodesic MCMC. In *Advances in Neural Information Processing Systems*, 2018. 2

[12] Tolga Birdal and Slobodan Ilic. Cad priors for accurate and flexible instance reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 2

[13] Tolga Birdal and Umut Simsekli. Probabilistic permutation synchronization using the riemannian structure of the birkhoff polytope. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[14] Jesus Briales and Javier Gonzalez-Jimenez. Cartan-sync: Fast and global se (d)-synchronization. *IEEE Robotics and Automation Letters*, 2(4):2127–2134, 2017. 2

[15] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 2

[16] Luca Carlone, Roberto Tron, Kostas Daniilidis, and Frank Dellaert. Initialization techniques for 3d slam: a survey on rotation estimation and its use in pose graph optimization. In *International Conference on Robotics and Automation*, pages 4597–4604. IEEE, 2015. 2

[17] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[18] Avishek Chatterjee and Venu Madhav Govindu. Robust relative rotation averaging. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):958–972, 2017. 2

[19] Kunal N Chaudhury, Yuehaw Khoo, and Amit Singer. Global registration of multiple point clouds using semidefinite programming. *SIAM Journal on Optimization*, 25(1), 2015. 2

[20] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[21] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3), 1998. 2

[22] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In *European Conference on Computer Vision*, pages 768–783, 2018. 6

[23] 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. 2, 3, 4, 16, 17

[24] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope. Technical report, Australian National University (arXiv:1909.04866), Sep 2019. 1, 2

[25] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE, 2004. 2

[26] Venu Madhav Govindu and A Pooja. On averaging multiview relations for 3d scan registration. *IEEE Transactions on Image Processing*, 23(3):1289–1302, 2014. 2

[27] Maciej Halber, Yifei Shi, Kai Xu, and Thomas Funkhouser. Rescan: Inductive instance segmentation for indoor rgbd scans. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2541–2550, 2019. 2

[28] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3), 2013. 2

[29] 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. 2, 5, 6, 7, 8, 19

[30] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012. 13

[31] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 5390–5399, 2019. 2

[32] Nan Hu, Qixing Huang, Boris Thibert, UG Alpes, and Leonidas Guibas. Distributable consistent multi-object matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[33] Jiahui Huang, Sheng Yang, Tai-Jiang Mu, and Shi-Min Hu. Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2168–2177, 2020. 2

[34] Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shi-Min Hu. Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5875–5884, 2019. 2

[35] Qixing Huang, Zhenxiao Liang, Haoyun Wang, Simiao Zuo, and Chandrajit Bajaj. Tensor maps for synchronizing heterogeneous shape collections. *ACM Trans. Graph.*, 38(4):106, 2019. 2

[36] 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. 2, 3, 16

[37] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *International journal of computer vision*, 97(2):123–147, 2012. 2

[38] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 7, 8

[39] 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, 16

[40] 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. 15

[41] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. Consac: Robust multi-model fitting by conditional sample consensus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[42] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 1

[43] Ting Li, Vinutha Kallem, Dheeraj Singaraju, and René Vidal. Projective factorization of multiple rigid-body motions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, 2007. 2

[44] Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1

[45] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 2

[46] 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. 2, 6, 7, 19

[47] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019. 2

[48] Luca Magri and Andrea Fusiello. Fitting multiple heterogeneous models by multi-class cascaded t-linkage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7460–7468, 2019. 2

[49] Eleonora Maset, Federica Arrigoni, and Andrea Fusiello. Practical and efficient multi-view matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4568–4576, 2017. 2, 3

[50] 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. 16

[51] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[52] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in neural information processing systems*, pages 1860–1868, 2013. 2

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. 6

[54] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. *arXiv preprint arXiv:1912.04485*, 2019. 2

[55] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *European Conference on Computer Vision*, 2020. 2

[56] 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, 16

[57] 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. 4, 6, 7, 19

[58] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. Caspr: Learning canonical spatiotemporal point cloud representations. In *Advances in Neural Information Processing Systems*, 2020. 2

[59] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38, 2019. 2

[60] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013. 2

[61] Michele Schiavinato and Andrea Torsello. Synchronization over the birkhoff polytope for multi-graph matching. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 266–275. Springer, 2017. 2

[62] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. KillingFusion: Non-rigid 3D Reconstruction without Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 7

[63] 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. 19

[64] Johan Thunberg, Florian Bernard, and Jorge Goncalves. Distributed methods for synchronization of orthogonal matrices over graphs. *Automatica*, 80:243–252, 2017. 2

[65] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. *arXiv preprint arXiv:2009.10467*, 2020. 2

[66] Roberto Tron and Kostas Daniilidis. Statistical pose averaging with non-isotropic and incomplete relative measurements. In *European Conference on Computer Vision*. Springer, 2014. 2

[67] Roberto Tron and Rene Vidal. Distributed 3-d localization of camera sensor networks from 2-d image measurements. *IEEE Transactions on Automatic Control*, 59(12), 2014. 2

[68] Dimitrios Tzionas and Juergen Gall. Reconstructing articulated rigged models from rgb-d videos. In *European Conference on Computer Vision Workshops*, 2016. 7

[69] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729. IEEE, 1999. 1

[70] Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance relocalization in changing indoor environments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7658–7667, 2019. 1, 2

[71] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: A Journal of the IMA*, 2(2):145–193, 2013. 2

[72] Qianqian Wang, Xiaowei Zhou, and Kostas Daniilidis. Multi-image semantic matching by mining consistent features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[73] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 2

[74] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 91–98, 2020. 2

[75] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963. 7

[76] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015. 5

[77] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020. 1, 2

[78] 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

[79] 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. 5, 6, 18, 19

[80] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Midfusion: Octree-based object-level multi-instance dynamic slam. In *International Conference on Robotics and Automation*, pages 5231–5237. IEEE, 2019. 2

[81] Xun Xu, Loong Fah Cheong, and Zhuwen Li. 3d rigid motion segmentation with mixed and unknown number of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[82] 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. *ACM Trans. Graph.*, 38(6):240:1–240:15, 2019. 2

[83] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Trans. Graph.*, 37(6), 2018. 1, 2, 4, 5, 6, 7, 8, 16, 17, 19

[84] 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. 5, 18

[85] Jin-Gang Yu, Gui-Song Xia, Ashok Samal, and Jinwen Tian. Globally consistent correspondence of multiple feature sets using proximal gauss–seidel relaxation. *Pattern Recognition*, 51:255–267, 2016. 2

[86] Jun Zhang, Mina Henein, Robert Mahony, and Viorela Ila. Vdo-slam: A visual dynamic object-aware slam system. *arXiv preprint arXiv:2005.11052*, 2020. 2

[87] 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. 3, 16

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) can be written as:

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

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

$$
\min_{\boldsymbol{p}} \mathrm{tr}(\boldsymbol{p}^\top\mathbf{L}\boldsymbol{p}) \quad \text{s.t.} \quad \boldsymbol{p}^\top\boldsymbol{p} = \mathbf{I}_N. \tag{S.11}
$$

This QCQP (Quadratically Constrained Quadratic Program) is known to have the closed form solution revealed by generalized Rayleigh problem [30] (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:

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

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

$$
\max_{\mathbf{U}} \mathrm{tr}(\mathbf{U}^\top\mathbf{Z}\mathbf{U}) \quad \text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}_S, \tag{S.13}
$$

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* $\mathbf{Z}$ *[3]:*

$$
\mathbf{U}\cdot\mathrm{diag}(\sqrt{\lambda_1},\dots,\sqrt{\lambda_S}) \approx \boldsymbol{g} = \begin{bmatrix} \mathbf{G}^1 \\ \mathbf{G}^2 \\ \vdots \\ \mathbf{G}^K \end{bmatrix}, \tag{S.14}
$$

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 [3]. The weighted synchronization matrix $\tilde{\mathbf{Z}}$ is composed of a set of anisotropically-scaled $\boldsymbol{\zeta}^{kl}$ matrices:

$$
\tilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{0} & \frac{1}{\sigma^{12}}\boldsymbol{\zeta}^{12} & \dots & \frac{1}{\sigma^{1K}}\boldsymbol{\zeta}^{1K} \\ \frac{1}{\sigma^{21}}\boldsymbol{\zeta}^{21} & \mathbf{0} & \dots & \frac{1}{\sigma^{2K}}\boldsymbol{\zeta}^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sigma^{K1}}\boldsymbol{\zeta}^{K1} & \frac{1}{\sigma^{K2}}\boldsymbol{\zeta}^{K2} & \dots & \mathbf{0} \end{bmatrix}. \tag{S.15}
$$

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*

$$
\max_{\tilde{\mathbf{U}}} \mathrm{tr}(\tilde{\mathbf{U}}^\top\tilde{\mathbf{Z}}\tilde{\mathbf{U}}) \quad \text{s.t.} \quad \tilde{\mathbf{U}}^\top\tilde{\mathbf{U}} = \mathbf{I}_S \tag{S.16}
$$

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

$$\tilde{\mathbf{U}} \cdot \text{diag}(\sqrt{\tilde{\lambda}_1}, \ldots, \sqrt{\tilde{\lambda}_S}) \approx \tilde{\boldsymbol{g}} = \begin{bmatrix} \mathbf{G}^1\mathbf{D}^1 \\ \mathbf{G}^2\mathbf{D}^2 \\ \vdots \\ \mathbf{G}^K\mathbf{D}^K \end{bmatrix}, \quad \text{(S.17)}$$

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 $\mathbf{Z}\boldsymbol{g} = \boldsymbol{g}\boldsymbol{\Lambda}$ [3]:

$$\mathbf{Z}\boldsymbol{g} = \boldsymbol{g}\boldsymbol{g}^\top\boldsymbol{g} = \boldsymbol{g}\sum_{k=1}^K \mathbf{G}^{k\top}\mathbf{G}^k = \boldsymbol{g}\boldsymbol{\Lambda}. \quad \text{(S.18)}$$

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:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}} = \begin{bmatrix} \sum_{l=1}^K w^{1l}\boldsymbol{\zeta}^{1l}\mathbf{G}^l\mathbf{D}^l \\ \sum_{l=1}^K w^{2l}\boldsymbol{\zeta}^{2l}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \sum_{l=1}^K w^{Kl}\boldsymbol{\zeta}^{Kl}\mathbf{G}^l\mathbf{D}^l \end{bmatrix}. \quad \text{(S.19)}$$

Then, using $\boldsymbol{\zeta}^{kl} = \mathbf{G}^k\mathbf{G}^{l\top}$ we can express Eq (S.19) as:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}} = \begin{bmatrix} \sum_{l=1}^K w^{1l}\mathbf{G}^1\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \sum_{l=1}^K w^{2l}\mathbf{G}^2\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \sum_{l=1}^K w^{Kl}\mathbf{G}^K\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \end{bmatrix} \quad \text{(S.20)}$$

$$= \begin{bmatrix} \mathbf{G}^1\sum_{l=1}^K w^{1l}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \mathbf{G}^2\sum_{l=1}^K w^{2l}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \\ \vdots \\ \mathbf{G}^K\sum_{l=1}^K w^{Kl}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l \end{bmatrix} = \begin{bmatrix} \mathbf{G}^1\mathbf{H}^1 \\ \mathbf{G}^2\mathbf{H}^2 \\ \vdots \\ \mathbf{G}^K\mathbf{H}^K \end{bmatrix} \quad \text{(S.21)}$$

where:

$$\mathbf{H}^k = \sum_{k=1}^K w^{kl}\mathbf{G}^{l\top}\mathbf{G}^l\mathbf{D}^l. \quad \text{(S.22)}$$

$\mathbf{H}$ 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):

$$\mathbf{E}^{kl} = w^{kl}\mathbf{G}^{l\top}\mathbf{G}^l, \quad \text{(S.23)}$$

This form is very similar to Eq (S.17) 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 $\tilde{\mathbf{Z}}$:

$$\tilde{\mathbf{Z}}\tilde{\boldsymbol{g}}^s = \tilde{\lambda}_s\tilde{\boldsymbol{g}}^s. \quad \text{(S.24)}$$

In other words, we seek the existence of $\tilde{\lambda}_s$ such that Eq (S.24) 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.17) and Eq (S.21) into Eq (S.24) to see that:

$$\begin{bmatrix} (\mathbf{G}^1\mathbf{H}^1)^s \\ (\mathbf{G}^2\mathbf{H}^2)^s \\ \vdots \\ (\mathbf{G}^K\mathbf{H}^K)^s \end{bmatrix} = \tilde{\lambda}_s \begin{bmatrix} (\mathbf{G}^1\mathbf{D}^1)^s \\ (\mathbf{G}^2\mathbf{D}^2)^s \\ \vdots \\ (\mathbf{G}^K\mathbf{D}^K)^s \end{bmatrix}. \quad \text{(S.25)}$$

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:

$$\begin{cases} H^1_{ss} = \sum_{l=1}^{K} E^{1j}_{ss} D^l_{ss} = \tilde{\lambda}_s D^1_{ss} \\ H^2_{ss} = \sum_{l=1}^{K} E^{2j}_{ss} D^l_{ss} = \tilde{\lambda}_s D^2_{ss} \\ \vdots \\ H^K_{ss} = \sum_{l=1}^{K} E^{Kj}_{ss} D^l_{rr} = \tilde{\lambda}_s D^K_{ss} \end{cases} \tag{S.26}$$

where E is as defined in Eq (S.23). Note that both D and $\tilde{\lambda}_s$ are unknowns in this seemingly non-linear problem. Yet, we can re-arrange Eq (S.26) into another eigen-problem:

$$\mathbf{J}^s \mathbf{d}^s = \tilde{\lambda}'_s \mathbf{d}^s, \tag{S.27}$$

where:

$$\mathbf{J}^s = \begin{bmatrix} E^{11}_{ss} & E^{12}_{ss} & \cdots & E^{1K}_{ss} \\ E^{21}_{ss} & E^{22}_{ss} & \cdots & E^{2K}_{ss} \\ \vdots & \ddots & \vdots & \\ E^{K1}_{ss} & E^{K2}_{ss} & \cdots & E^{KK}_{ss} \end{bmatrix} \mathbf{d}^s = \begin{bmatrix} D^1_{ss} \\ D^2_{ss} \\ \vdots \\ D^K_{ss} \end{bmatrix}. \tag{S.28}$$

Hence, we conclude that the eigenvectors of the weighted synchronization have the form of Eq (S.27) if and only if we can solve Eq (S.17). This is possible as soon as $\mathbf{E}^{kl}$ are known and $\mathbf{J}^s$ has real eigenvectors. Besides an *existence* condition, Eq (S.17) 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.28) 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.27) 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 $\tilde{\boldsymbol{g}}$ itself as the estimate of motion segmentation, as our closed form expression for $\mathbf{D}^k$ (Eq (S.28)) cannot be evaluated in test time. Then, assuming $\mathbf{D}^k$ to be the identity, for each k it holds:*

$$\begin{aligned} \sum_{l=1}^{K} E^{kl}_{ss} &= \sum_{l=1}^{K} w^{kl} \mathbf{G}^{k\top} (\mathbf{G}^l)^s && \text{(S.29)} \\ &= w^{k1}\mathbf{G}^{1\top}(\mathbf{G}^1)^s + \cdots + w^{kK}\mathbf{G}^{K\top}(\mathbf{G}^K)^s \\ &= \mathbf{w}_k \cdot \begin{bmatrix} \mathbf{G}^{1\top}(\mathbf{G}^1)^s & \cdots & \mathbf{G}^{K\top}(\mathbf{G}^K)^s \end{bmatrix} \\ &= \mathbf{w}_k \boldsymbol{\varphi}^s = \tilde{\lambda}'_s. && \text{(S.30)} \end{aligned}$$

where $(\mathbf{G}^l)^s$ is the s-th column of $\mathbf{G}^l$. The final equality follows directly from Eq (S.26) when $D_{ss} = 1$. Note that we can find multiple weights $\mathbf{w}_k$ satisfying Eq (S.30). For instance, if $\boldsymbol{\varphi}$ and λ were known, one solution for any s would be:

$$w^{kl} = \frac{\tilde{\lambda}_s}{K\varphi^s_k}. \tag{S.31}$$

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 [78] by changing layer sizes and the number of pyramids. As illustrated in Fig. S10, 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' [40] 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 [78] for more details.

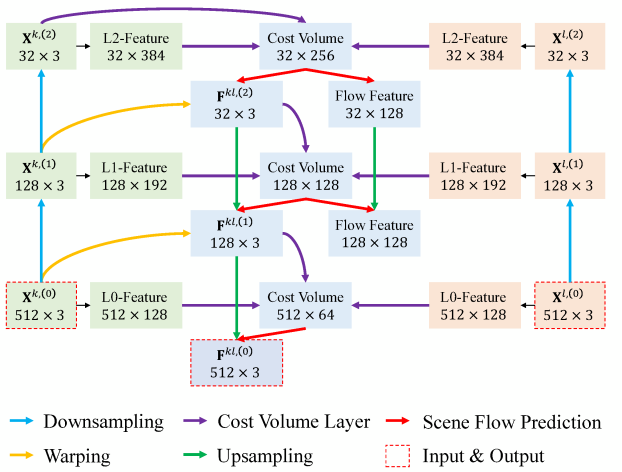


Figure S10. Our adapted version of PointPWC-Net φ_{flow}. Each rectangular block denotes a tensor, whose size is written as $N \times C$ (batch dimension is ignored) below its name, with N being the number of points and C being the feature dimension. The network is composed of 3 hierarchical levels. 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) [87], 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 [56], 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 [9].

The network starts and ends with 6 PointCN [50] 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 $M = 64$.

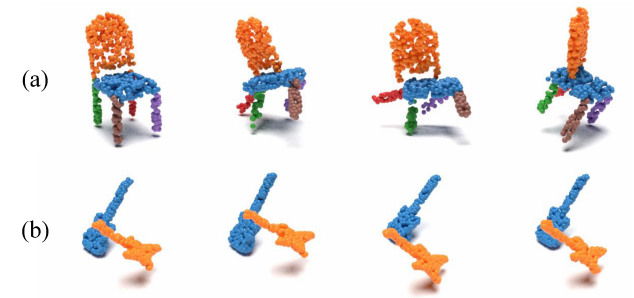


Figure S11. Examples from our training set for (a) articulated objects and (b) multiple solid objects. Different colors indicate rigidly moving parts.

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 [83]. 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:

$$\tilde{\mathbf{R}}_i^{kl} = \mathbf{R}_{\text{net}} + \mathbf{I}_3, \quad \tilde{\mathbf{t}}_i^{kl} = \mathbf{t}_{\text{net}} - \mathbf{R}_{\text{net}}\boldsymbol{x}_i^k + \boldsymbol{f}_i^{kl}. \quad \text{(S.32)}$$

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 [39]. 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 [23, 36] 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.

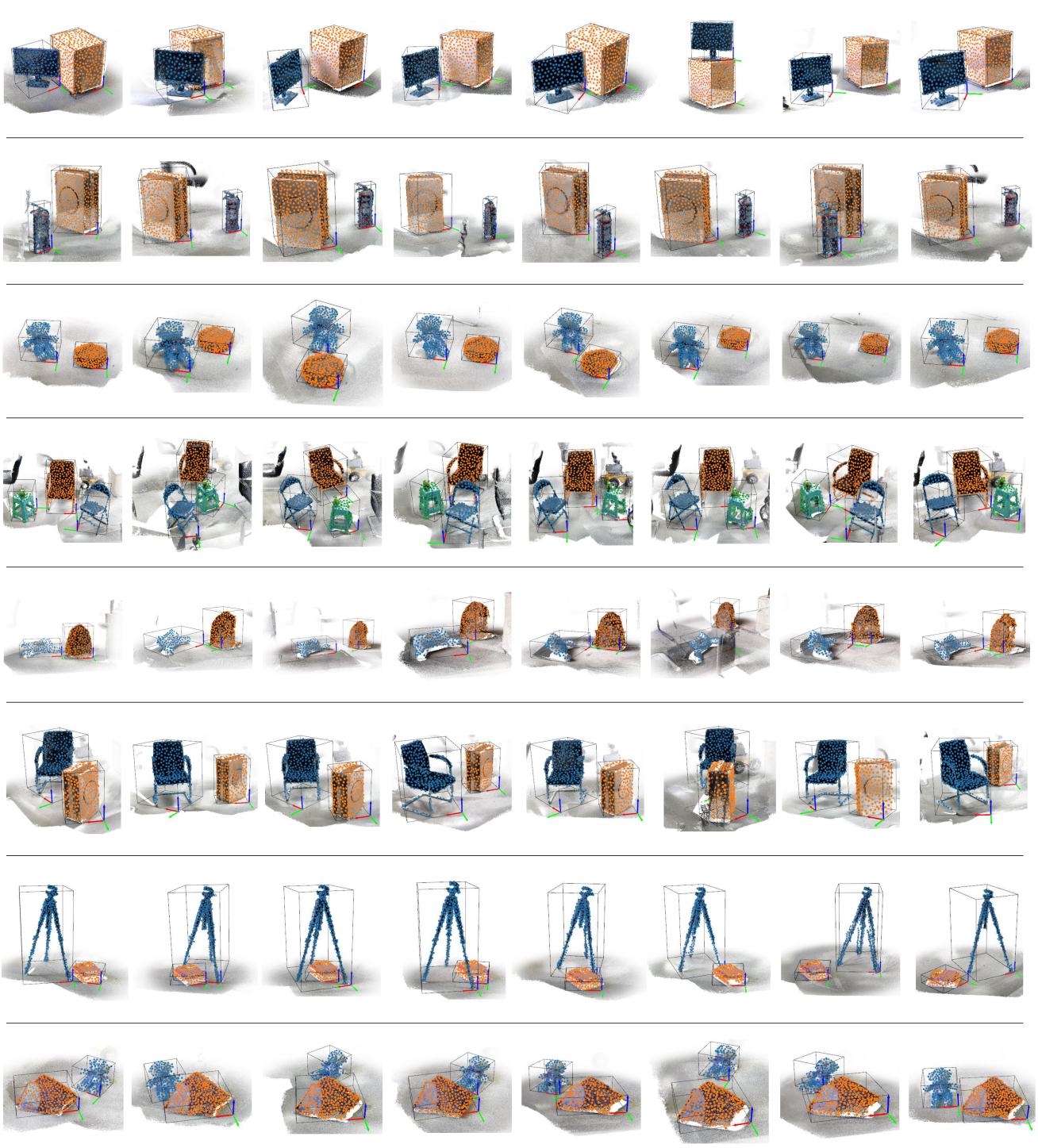


Figure S12. Visualization of the DynLab dataset. Each row shows 8 different dynamic configurations of the same set of rigid objects. Annotated bounding boxes are parallel to the ground plane and reflect the objects' absolute poses.

This might lead to wrong results in the subsequent steps. Inspired by point cloud registration works [83, 23], 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

Table S5. Training and validation categories from [84] used for articulated objects.

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

Table S6. Training and validation categories from [84] used for multiple solid objects.

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

belonging to part s to the *canonical* pose of the k^*-th point 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. S5. For multiple solid objects, the categories are listed in Tab. S6. Examples from our training set are visualized in Fig. S11.

DynLab dataset. A full visualization of the DynLab dataset with manual annotations is shown in Fig. S12. 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. 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}.

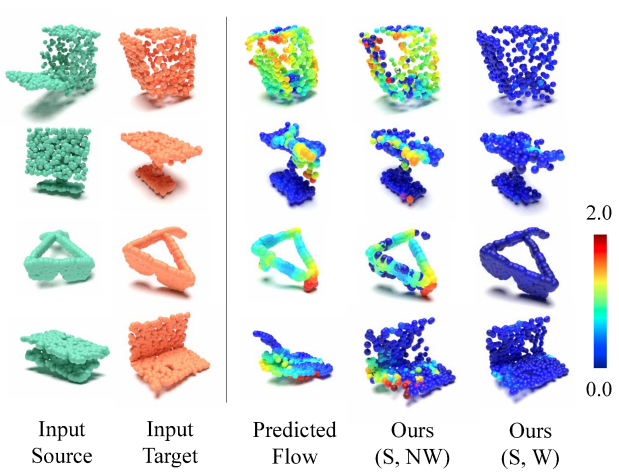


Figure S13. Visual comparisons of the pairwise flow. To visualize the flow we warp the source point cloud and compare the its similarity with the target point cloud. The color bar on the right shows the end-point error (EPE3D). 'Ours (S, W)' represents the output of our method with the **W**eighted permutation **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.15)) 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. S14 and S15 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) 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. S13, 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 [79] dataset in Tab. S7. 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 S7. Per-category mIoU comparisons on SAPIEN [79] dataset.

	Box	Dishwasher	Display	Storage Furniture	Eyeglasses	Faucet	Kettle	Knife	Laptop	Lighter
PointNet++ [57]	43.5	46.8	54.8	51.3	34.6	42.4	65.7	43.0	58.5	52.3
MeteorNet [46]	47.0	42.2	41.7	36.9	36.1	47.1	67.2	36.2	57.9	61.3
DeepPart [83]	53.3	55.1	47.4	48.7	31.8	43.4	64.7	38.5	67.3	39.0
NPP [29]	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++ [57]	51.5	42.6	46.2	63.6	55.7	43.0	42.7	40.0	51.2	49.8	47.5
MeteorNet [46]	37.4	37.1	41.7	43.4	33.7	54.7	33.3	38.3	61.5	41.9	43.7
DeepPart [83]	65.9	49.8	41.9	32.9	57.5	47.0	38.6	39.1	65.1	59.5	49.2
NPP [29]	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

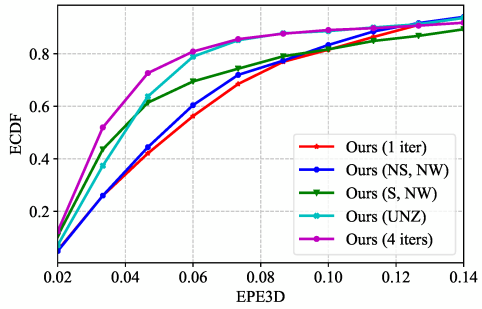


Figure S14. Empirical cumulative distribution function (ECDF) of rigid flow error (EPE3D) on SAPIEN [79] dataset. The higher the curve, the better the results.

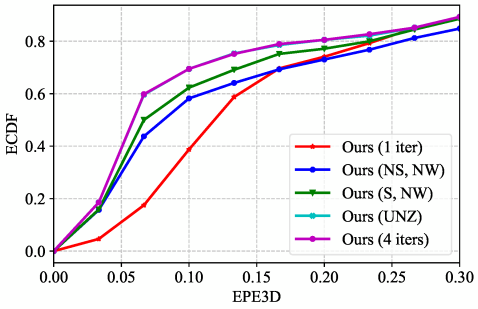


Figure S15. Empirical cumulative distribution function (ECDF) of rigid flow error (EPE3D) on DynLab dataset. The higher the curve, the better the results.

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. S16 the scenarios with co-existing articulated/solid objects and multiple cars in a scene of Waymo Open dataset [63] (though the car category is within our training set). Moreover, we show in Fig. S17 to S19 our segmentation and registration results for each category in SAPIEN [79] 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,

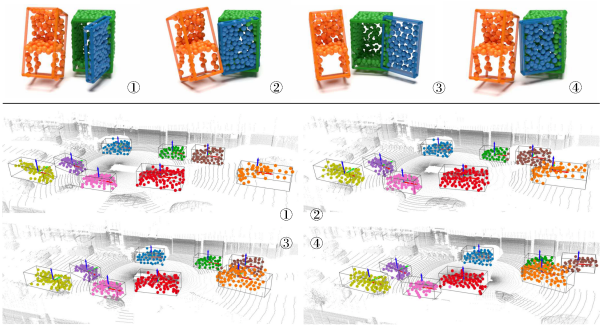


Figure S16. Quantitative demonstrations on complex scans. The first row is estimated using our trained articulated objects model while the last row is obtained by hierarchically apply our method to each segmented part until convergence. ①-④ indicates scan index. Best viewed with 200% zoom in.

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. S20. Our method can generate robust object associations under challenging settings.

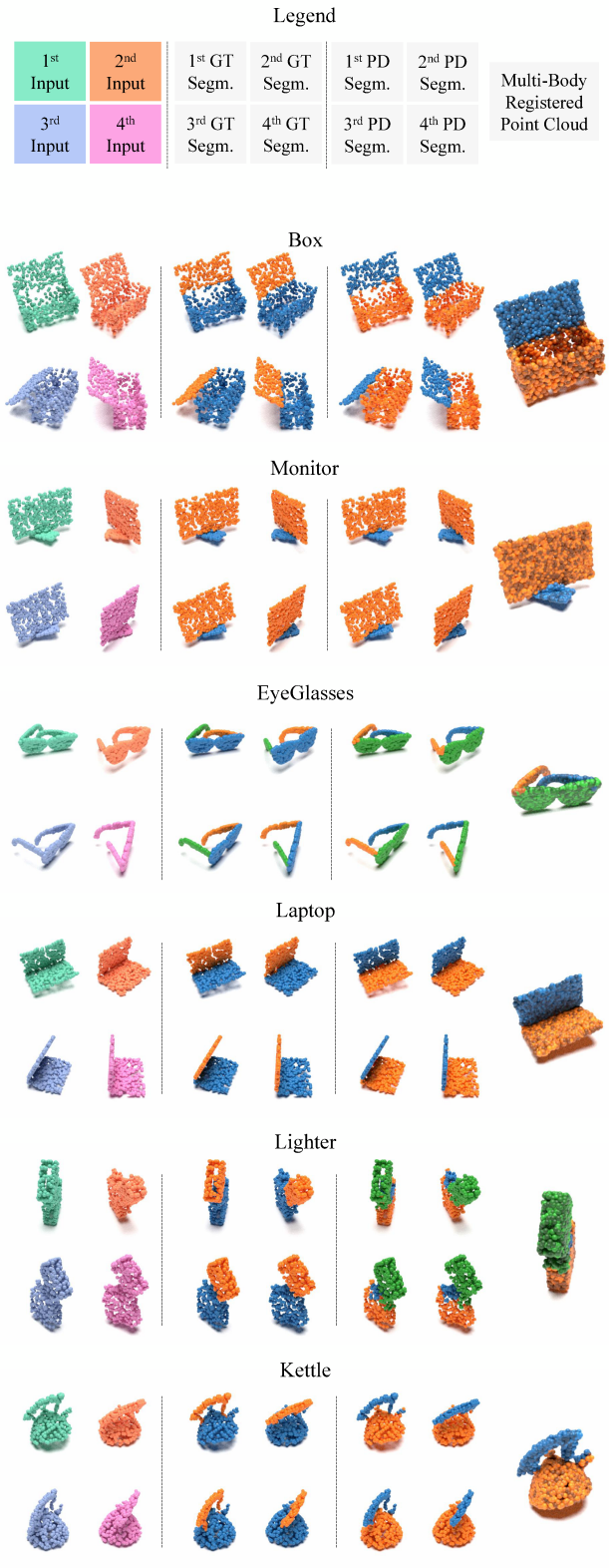


Figure S17. Qualitative results on SAPIEN dataset (1/3).

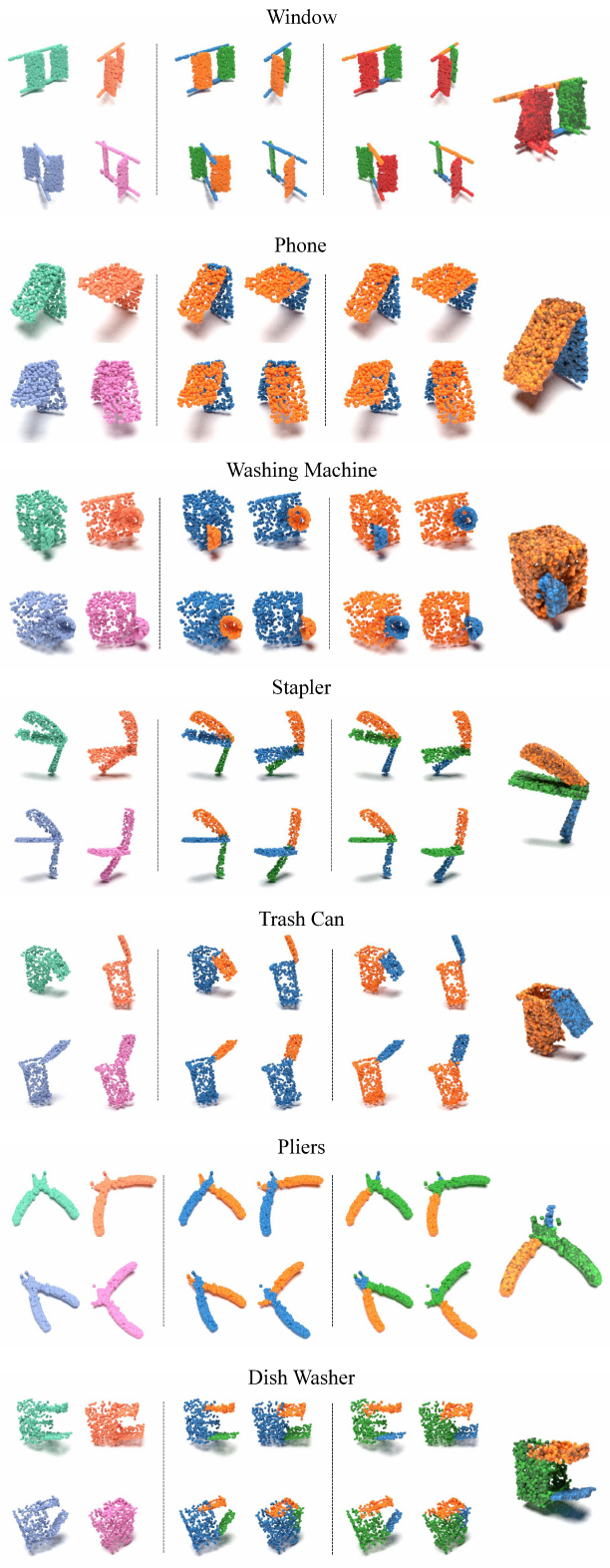


Figure S18. Qualitative results on SAPIEN dataset (2/3).

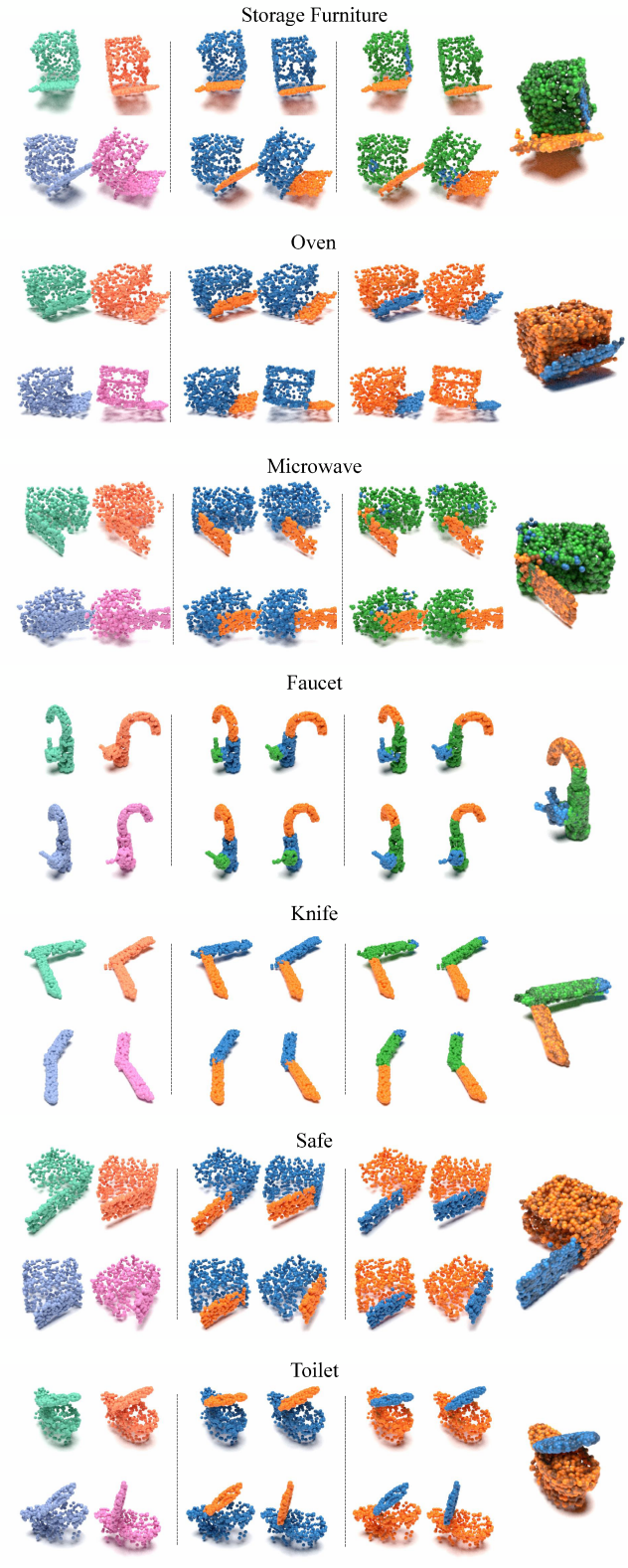


Figure S19. Qualitative results on SAPIEN dataset (3/3).

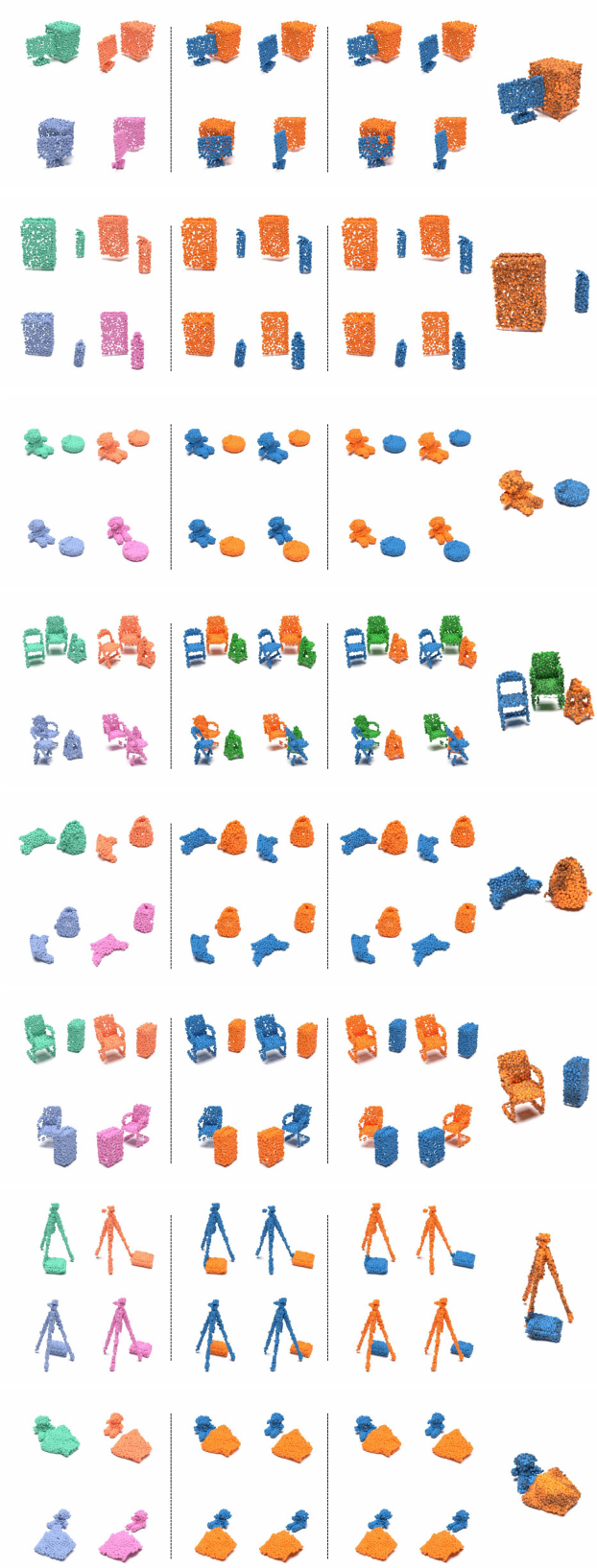


Figure S20. Qualitative results on DynLab dataset.