P^3-Net: Part Mobility Parsing from Point Cloud Sequences via Learning Explicit Point Correspondence

Yahao Shi[1], **Xinyu Cao**[1], **Feixiang Lu**[2], **Bin Zhou**[1,3]

[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China,
[2]Robotics and Autonomous Driving Laboratory, Baidu Research, Beijing, China,
[3]Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen, China,
zhoubin@buaa.edu.cn

Abstract

Understanding an articulated 3D object with its movable parts is an essential skill for an intelligent agent. This paper presents a novel approach to parse 3D part mobility from point cloud sequences. The key innovation is learning explicit point correspondence from a raw unordered point cloud sequence. We propose a novel deep network called P^3-Net to parallelize the trajectory feature extraction and the point correspondence establishment, performing joint optimization between them. Specifically, we design a Match-LSTM module to reaggregate point features among different frames by a point correspondence matrix, *a.k.a.* the matching matrix. To obtain this matrix, an attention module is proposed to calculate the point correspondence. Moreover, we implement a Gumbel-Sinkhorn module to reduce the many-to-one relationship for better point correspondence. We conduct comprehensive evaluations on public benchmarks, including the motion dataset and the PartNet dataset. Results demonstrate that our approach outperforms SOTA methods on various 3D parsing tasks of part mobility, including motion flow prediction, motion part segmentation, and motion attribute (*i.e.*, axis & range) estimation. Moreover, we integrate our approach into a robot perception module to validate its robustness.

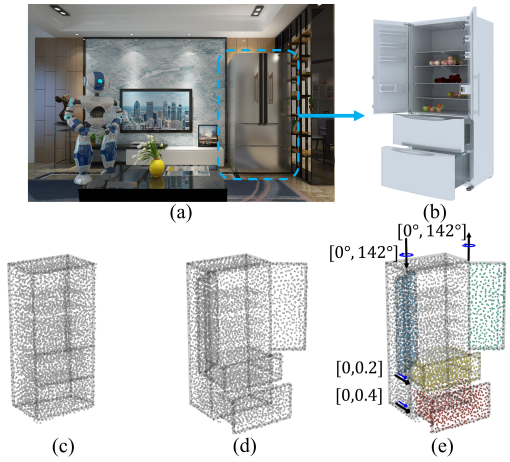

Figure 1: Part mobility parsing. A key aspect of the robot manipulation application (a) is part mobility parsing. Given a motion sequence of an articulated object (b), we obtain a point could sequence (c-d). Then, our algorithm parses part mobility, including motion flow prediction, motion part segmentation, and motion attribute estimation (e).

Introduction

Our environment packs with plenty of articulated objects, *e.g.*, a cabinet or a fridge. Automatically parsing an articulated object into its 3D part mobility is indispensable for driving an agent to complete a manipulation task (Mo et al. 2021). Fig. 1 demonstrates a typical scenario of robot-object interaction. A human user says "please open the fridge and carry an apple to me". Following this instruction, the household robot should find out *where* the door of the fridge is and *how* to open it. Therefore, the robot perception module requires various 3D part mobility parsing techniques, including motion flow prediction (Yi et al. 2018; Liu, Qi, and Guibas 2019), motion part segmentation (Yi et al. 2018; Yan et al. 2019), and motion attribute (*i.e.*, axis & range) estimation (Li et al. 2020; Hu et al. 2017; Wang et al. 2019).

Several attempts have been made to parse part mobility from a single snapshot such as a point cloud (Wang et al. 2019; Yan et al. 2019; Mo et al. 2021) and a depth image (Li et al. 2020). However, these snapshot-based approaches are limited for a robot to accomplish interaction tasks, due to lacking the consecutive part motion information, *e.g.*, the observation of a single snapshot confuses the robot whether a closed door should be opened from the right side or the left side. Thanks to the recent advances in real-time 3D acquisition techniques such as commercial RGB-D and LiDAR cameras, point cloud sequences are readily available for visual perception. Unlike a single *static* snapshot, point cloud sequences contain meaningful semantic, spatial, and temporal information, enabling a robot to perform *dynamic* perception of part mobility with fine granularity.

However, it is difficult to learn the spatial-temporal information from the unordered point cloud sequences. In general, the main challenges are threefold: (1) how to extract a proper trajectory feature among multiple frames for each 3D point; (2) how to establish the one-to-one, point-level, consecutive correspondences from the point cloud sequences; and (3) how to optimize above 3D point trajectories. The pioneer work (Shi, Cao, and Zhou 2021) obtains aligned point

cloud sequences via a non-rigid ICP-like method (Papazov and Burschka 2011), which takes much time for preprocessing. Meanwhile, its preprocessing cannot be learned, hurting its capacity on modeling point cloud sequences. Other existing works process point cloud sequences via correlating frames by the nearest points (Liu, Yan, and Bohg 2019; Min et al. 2020) or learning implicit point relationship by the attention mechanism (Cao et al. 2020). However, the nearest point usually cannot represent the actual direction of motion, particularly in part-level motions. In addition, the implicit point relationship is not controllable to reveal motion correlation between two frames.

This paper presents a novel approach to parse 3D part mobility, including motion flow prediction, motion part segmentation, and motion attribute (*i.e.*, axis & range) estimation, from point cloud sequences. Learning explicit point correspondence is our key innovation, which can improve the trajectory feature extraction and has better interpretability. To this end, we design a novel deep network called P^3-Net to jointly optimize trajectory feature extraction and point correspondence establishment, benefiting the processing of unordered point cloud sequences.

Specifically, we adopt a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) network to process point cloud sequences because of its generalizability in processing sequential and time-series data (Sutskever, Vinyals, and Le 2014). However, it is problematic to utilize LSTM to model point trajectories directly since the unordered point cloud sequence is not organized in the form of point trajectories. LSTM-cell has a tuple of features, which requires the correspondence in the input data. To this end, we propose Match-LSTM (Fig. 3), which reaggregates point features according to a matching matrix. This way, we obtain a set of motion trajectory features.

Therefore, we implement an attention-based method to obtain the matching matrix (Fig. 4). Given two consecutive frames, we devise a shared PointNet++ (Qi et al. 2017b) to extract the point geometric features. We utilize two kinds of attention mechanisms to obtain the matching matrix based on pairs of geometric features. The first is the intra-frame attention which searches for a region containing similar points for each point in one frame, encoding non-local semantic information. The second is inter-frame attention for finding point correspondences between two consecutive frames. These attention modules can accommodate continuously adjustable point-to-point relationships from two neighboring regions in the feature level. Finally, we apply a Gumbel-Sinkhorn algorithm (Mena et al. 2018) to transform the raw matching matrix into an approximate doubly stochastic matrix, with which we build the point correspondence.

Finally, we conduct extensive experiments with comparisons to many state-of-the-art (SOTA) methods (Yi et al. 2018; Liu, Qi, and Guibas 2019; Cao et al. 2020; Liu, Yan, and Bohg 2019; Shi, Cao, and Zhou 2021) on multiple tasks over several public benchmarks including the motion dataset (Wang et al. 2019) and the PartNet dataset (Mo et al. 2019). Results show that our approach outperforms the existing methods on part mobility parsing, including motion flow prediction, motion part segmentation, and motion attribute estimation.

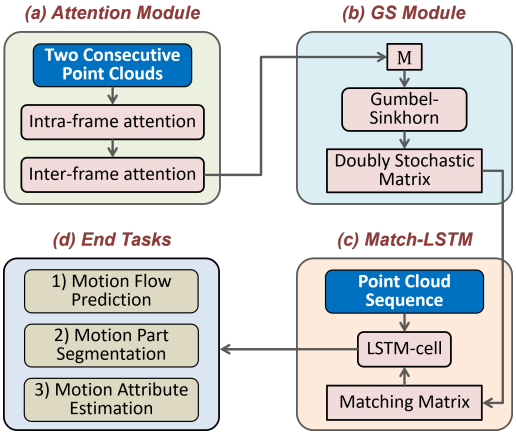

Figure 2: An overview of our P^3-Net, which contains three modules and three end tasks.

Our work makes the following contributions:

- We present a novel approach to parsing 3D part mobility from point cloud sequences. The parsing results include motion flow prediction, motion part segmentation, and motion attribute (*i.e.*, axis & range) estimation, which are essential for robot perception.
- We design a novel deep network (P^3-Net) with three efficient modules (*i.e.*, Match-LSTM, Attention, and Gumbel-Sinkhorn), which can jointly optimize the trajectory feature extraction and the point correspondence establishment from point cloud sequences.
- Comprehensive evaluations of our method are conducted on public benchmarks. Results show that our method outperforms existing works on various tasks. Moreover, we integrate our approach into a robot perception module to validate its robustness.

Related Work

Deep Learning on the Point Cloud. Deep learning on point cloud processing (Guo et al. 2021) has received significant achievements in several tasks, including classification, segmentation, detection, *etc.* Moreover, researchers tend to study new tasks that are associated with part mobility for robot from a single point cloud (Li et al. 2020; Mo et al. 2021). Compared to deep learning on the 2D image (regular domains), the classic convolution (Krizhevsky, Sutskever, and Hinton 2012) is difficult to be applied to a 3D point cloud because of the challenge of processing unordered sets. The pioneering work, PointNet (Qi et al. 2017a), considers each point independently and aggregates point features into the global feature using a max pooling operation to address the problem of order-invariance. Subsequent works explored several different neighborhood aggregation mechanisms to obtain hierarchical local point set features, such as ball-query strategy (Qi et al. 2017b), CNN-like point convolution (Li et al. 2018), voxel-based technique (Maturana

and Scherer 2015), graph-based approach (Simonovsky and Komodakis 2017), *etc.*

Deep Learning on Point Cloud Sequence. Most recently, researchers are not confined to study on a single point cloud. The newly emerged applications, including motion flow prediction, motion segmentation, and motion attribute estimation, contribute to autonomous driving and personal assistant robots. In the flow prediction, Yi *et al.* (Yi et al. 2018) infer part motion flow and part segmentation by comparing two different motion states from an articulated object. Liu *et al.* (Liu, Qi, and Guibas 2019) propose FlowNet3D that estimates scene flow from a pair of consecutive point clouds in an end-to-end fashion and evaluates it on the real LiDAR scans from the KITTI scene flow dataset (Menze and Geiger 2015). In the case of segmentation, Liu *et al.* (Liu, Yan, and Bohg 2019) propose MeteorNet, which is the first work on deep learning for dynamic point cloud sequences. They merge all frames into a single point cloud and then process the point cloud using two different ball-query strategies: direct grouping and chained-flow grouping. Yan *et al.* (Yan et al. 2019) introduces RPM-Net to infer movable parts of a single point cloud by forecasting a sequence motion with a Recurrent Neural Network (RNN). Cao *et al.* (Cao et al. 2020) propose an attention-based network called ASAP-Net used for semantic segmentation in dynamic point clouds. In the case of motion attribute estimation, Shi *et al.* (Shi, Cao, and Zhou 2021) propose a self-supervised deep learning method on regular trajectories that segments the motion part and estimates the motion axis and motion range.

Point Correspondence. Compared to the single point cloud, processing raw point cloud sequences has a challenge that two consecutive frames do not have explicit point correspondences. Before deep learning became prevalent, traditional iterative methods usually establish point correspondences and then compute these correspondences' optimal transformation. These methods employ either handcrafted (Rusu, Blodow, and Beetz 2009; Tombari, Salti, and di Stefano 2010) or learned (Khoury, Zhou, and Koltun 2017; Deng, Birdal, and Ilic 2018; Yew and Lee 2018) 3D local feature descriptors to estimate candidate correspondences in combination with a RANSAC-like estimator (Fischler and Bolles 1981). Recent works improve traditional registration methods with deep learning (Elbaz, Avraham, and Fischer 2017; Aoki et al. 2019; Yang et al. 2019; Choy, Dong, and Koltun 2020; Bai et al. 2020; Poiesi and Boscaini 2021; Ao et al. 2021) in an end-to-end fashion. Moreover, Attention mechanism (Vaswani et al. 2017; Wang et al. 2018; Wang and Solomon 2019) has been proven to efficiently model dependencies from sequences and images without regard to their element distance and order, which this characteristic makes it can be applied to point correspondence estimation potentially.

Overview of Our Approach

We propose a novel deep network called P^3-Net to parse 3D part mobility, including motion flow prediction, motion part segmentation, and motion attribute estimation, by learning explicit point correspondence. The inputs to our network are point cloud sequences $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_T\}$ captured from an articulated object, where T denotes the number of frames. The network aims to first segment l motion parts $\{S^{(i)} | i = 1, 2, ..., l\}$. Then, for each motion part, the network estimates the motion attributes, including the motion axis and the motion range. We denote the motion axis as the start point μ and axis orientation ω. The motion ranges have rotation angle θ and shift distance ϕ for two different motion joints: revolute joint and prismatic joint. Moreover, the network utilizes actual motion flow to establish point correspondence. To obtain accurate point correspondence, the network also predicts the motion flow.

P^3-Net Architecture

In this section, we introduce our P^3-Net architecture with three efficient modules for 3D part mobility parsing from point cloud sequences, which parallelizes the *trajectory feature extraction* and the *point correspondence establishment* with jointly optimizing them (Fig. 2). Specifically, The Match-LSTM module is designed to process the unordered point cloud sequence, which the sequence is converted to motion trajectories by a match matrix. We utilize attention mechanisms module to calculate the matrix using point correspondence. Moreover, the Gumbel-Sinkhorn module without trainable parameter convert the matrix to a approximate doubly stochastic matrix when using the matrix in the Match-LSTM. Finally, we describe how to apply our architecture on three end tasks.

Trajectory Feature Extraction

The wide applications of LSTM in natural language processing and video understanding have demonstrated its strong ability on processing regular sequential data. The LSTM-cell is responsible for keeping track of dependencies between elements in a sequence. Compared to the vanilla RNN only maintaining a hidden state ($\mathbf{h}_t$), the LSTM-cell also contains a cell state ($\mathbf{c}_t$).

Match-LSTM Module The original LSTM cannot be directly applied to unordered point cloud sequences due to lacking the relationship between the input ($\mathbf{x_t}$) and a pair of features ($\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$). To this end, we propose Match-LSTM to extract trajectory feature. Formally, our Match-LSTM accepts a point cloud sequence containing T frames $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_T\}$ as input (T=8 by default), where each frame is a point cloud consisting of n points $\{p_1, p_2, .., p_n\}$ (n=1024 by default). At each step i, our model firstly estimates a point correspondence matrix, *a.k.a.* the matching matrix $\mathbb{M} \in \mathbb{R}^{n \times n}$ for two consecutive frames $\mathbb{P}_i$ and $\mathbb{P}_{i+1}$ by the attention module. Then, Match-LSTM employs $\mathbb{M}_i$ to transform the states, $\mathbf{h}_i$ and $\mathbf{c}_i$, for future steps.

To obtain trajectory features, it is unreasonable to rearrange $\mathbb{P}_{i+1}$ into the same point order as $\mathbb{P}_i$, to construct a normalized point cloud for the next step, since there is no guarantee on the one-to-one mapping between points of $\mathbb{P}_i$ and $\mathbb{P}_{i+1}$ due to the uneven point density. Therefore, we implement the rearrangement in the feature level. Denoting $\mathcal{F}_{1 \to i}$ as trajectory features from the first frame to the i-th frame, each of them contains a hidden state and a cell

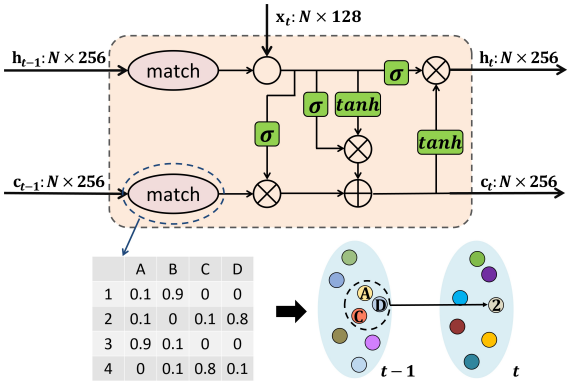

Figure 3: The schematic illustration of the Match-LSTM module. $\circ$ is the concatenation, $\otimes$ is the element-wise multiplication, and $\oplus$ is the element-wise summation. We reaggregate the input $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$ by a matching matrix.

state obtained by the Match-LSTM. When processing $\mathbb{P}_{i+1}$, Match-LSTM first reaggregates $\mathcal{F}_{1\to i}$ instead of $\mathcal{F}_{1\to i+1}$ by a matching matrix, and then obtains $\mathcal{F}_{1\to i+1}$ followed by a naive LSTM-cell (Fig. 3). Therefore, the direction of the matching matrix is $\mathbb{P}_i \leftarrow \mathbb{P}_{i+1}$, which indicates that feature, $\mathcal{F}_{1\to i+1}$, is based on the order of $\mathbb{P}_{i+1}$. The first advantage is that it avoids fewer and fewer available points since all points in $\mathbb{P}_{i+1}$ are used. The second advantage is that we only need consider the point correlation between two consecutive frames rather than from the first frame to the current frame. Finally, the end task results are based on the last frame, $\mathbb{P}_T$. Moreover, we apply the soft correspondence, which searches the top k similar points to improve the robustness ($k = 8$ by default). The forward pass of the Match-LSTM at step $i+1$ can be formulated as:

$$\mathbf{h}_{i+1}, \mathbf{c}_{i+1} = \text{LSTM}(\mathbf{x}_{i+1}, \mathbb{M}_i \cdot \mathbf{h}_i, \mathbb{M}_i \cdot \mathbf{c}_i). \quad (1)$$

Point Correspondence Establishment

To convert the raw point cloud sequence into motion trajectories, we calculate the matching matrix ($\mathbb{M}$). Specifically, we first calculate point similarities by computing points' geometric distances and then use an attention-based method to obtain point correspondence, which is called the attention module. Moreover, we implement a Gumbel-Sinkhorn module to reduce many-to-one relationships for better point correspondence.

Attention Module To measure the point similarity between $\mathbb{P}_i$ (denoted as $\mathcal{P}$ for simplicity) and $\mathbb{P}_{i+1}$ ($\mathcal{Q}$), we employ a standard attention mechanism to effectively weigh the relevance of two points by exploiting their local geometrical features. The attention function (Vaswani et al. 2017) can be described as a mapping between a query feature vector Q and a set of key-value feature vector pairs (K, V) as:

$$\text{Att}\big(Q, (K, V)\big) = \text{SoftMax}(\frac{K^T \cdot Q}{\sqrt{d_Q}})V, \quad (2)$$

where d_Q is the dimension of Q for scaling the numbers.

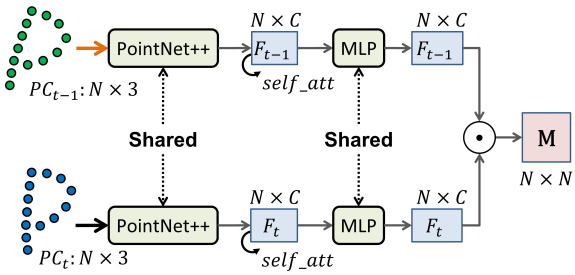

Figure 4: The schematic illustration of our point correspondence establishment via attention mechanisms. $\odot$ represents the dot product operation. C is 128 by default.

As for obtaining geometrical features of $\mathcal{P}$ and $\mathcal{Q}$ (*i.e.*, $\mathcal{F}_\mathcal{P}, \mathcal{F}_\mathcal{Q} \in \mathbb{R}^{n\times 128}$), we employ a shared PointNet++ to achieve it. To integrate more information inside each frame, we adopt a self-attention mechanism to establish an intra-frame relationship. A self-attention module computes the response at p_i by attending it to $\mathcal{P} \setminus p_i$ and takes their weighted summation. Then, we can obtain non-local point semantic features $\mathcal{F}_{\mathcal{P},\text{self}}$ and $\mathcal{F}_{\mathcal{Q},\text{self}} \in \mathbb{R}^{n\times 128}$ by:

$$\mathcal{F}_{\mathcal{P},\text{self}} = \mathcal{F}_\mathcal{P} + \text{Att}\big(\mathcal{F}_\mathcal{P}\boldsymbol{W}_Q, (\mathcal{F}_\mathcal{P}\boldsymbol{W}_K, \mathcal{F}_\mathcal{P}\boldsymbol{W}_V)\big), \quad (3)$$

where $\boldsymbol{W}_Q$, $\boldsymbol{W}_K$, and $\boldsymbol{W}_V$ are trainable parameters.

To obtain the final geometric point feature, we further feed $\mathcal{F}_{\mathcal{P},\text{self}}$ and $\mathcal{F}_{\mathcal{Q},\text{self}}$ into a shared Multi-Layer Perception (MLP) network. Applying the cross-frame attention module, we can obtain the point correspondence $\mathbf{M}$ (Fig. 4) by:

$$\mathbf{M} = \eta(\mathcal{F}_{\mathcal{P},\text{self}}) \times \eta(\mathcal{F}_{\mathcal{Q},\text{self}}), \quad (4)$$

where $\eta : \mathbb{R}^{n\times 128} \to \mathbb{R}^{n\times 128}$ is a differentiable function, and $\times$ is the Cartesian product which enumerates similarities for each pair (p_i, q_j).

Model Training of Point Correspondence To conduct the model training, supervision signals are needed. And here we choose the actual motion flow as the learning supervision. Concretely, we translate point correspondence cues into a motion flow for a motion sequence of an articulated object, which can help us discover part motions. For each point in $\mathcal{P}$ plus the motion flow ($\mathcal{P} \to \mathcal{Q}$), we can obtain the ground truth point correspondence $\tilde{\mathbf{M}}$ by searching its nearest point in $\mathcal{Q}$. Then we utilize the cross-entropy loss to optimize our model as:

$$\mathcal{L}_{match} = \text{H}(\tilde{\mathbf{M}}, \mathbf{M}), \quad (5)$$

where H is the cross-entropy function.

Gumbel-Sinkhorn Module Match-LSTM employs a forward motion flow as supervised information (direction based on row) and utilizes the backward direction of the matching matrix (direction based on column). Meanwhile, to reduce many-to-one relationships, the matching matrix should be close to a doubly stochastic matrix, which is a square matrix whose values are non-negative, with each row and column summing to one. Hence, the matching matrix generation can be regarded as a bipartite matching problem,

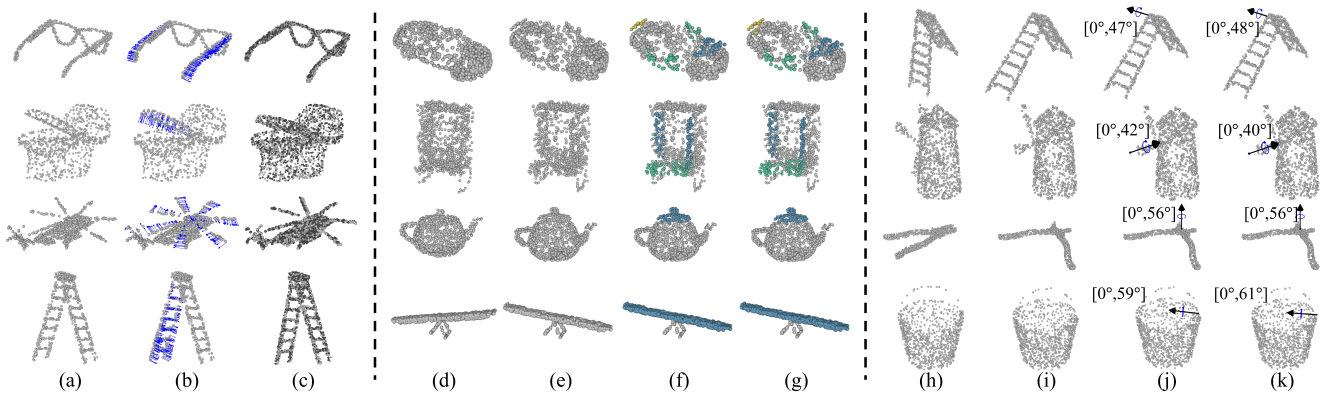

Figure 5: Visualization of our part mobility parsing results, including motion flow prediction (a-c), motion part segmentation (d-g), and motion attribute estimation (h-k). For motion flow prediction, (a): the first frame; (b): the first frame with predicted motion flow; (c): the predicted second frame and the ground-truth second frame. For motion part segmentation, (d): the first frame; (e): the last frame; (f): our segmentation results; (g): ground-truth segmentation results. For motion attribute estimation, (h): the first frame; (i): the last frame; (j): our predicted axis; (k): ground-truth axis.

which in theory can be solved by the Hungarian algorithm (Kuhn 2010). However, it is challenging to apply the Hungarian algorithm to deep learning since it is non-differentiable. Therefore, we adopt a differentiable approximate solution, the Gumbel-Sinkhorn algorithm proposed by Mena *et. al.* (Mena et al. 2018), which is similar to an iterative method of the row-based and column-based SoftMax. To ensure $\mathbf{M}$ is a positive matrix, we employ a ReLU activation function (Agarap 2018) before the dot product operation. Based on the Gumbel-Sinkhorn (GS) module, we can transform the continuous correspondence matrix $\mathbf{M}$ into a discrete correspondence matrix $\mathbb{M}$:

$$\mathbb{M} = \lim_{\tau \to 0^+} \mathrm{GS}(\mathbf{M}/\tau), \tag{6}$$

where τ is a temperature parameter, and the lower temperature can lead to an approximate sampling of a doubly stochastic matrix.

End Tasks

Three end tasks including motion flow prediction, motion part segmentation, and motion attribute estimation are considered. Formally, given a point cloud sequence $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_T\}$, we first obtain the geometric feature sequence $\{\mathcal{F}_1, \mathcal{F}_2, ..., \mathcal{F}_T\}$ resort to a Pointnet++ network due to its simplicity and effectiveness. Meanwhile, we obtain $T-1$ matching matrix $\{\mathbb{M}_1, \mathbb{M}_2, ..., \mathbb{M}_{T-1}\}$ computed by the attention module. Last, we feed these features and matrices into the Match-LSTM module to obtain the motion trajectory feature $\mathcal{F}_{1\to T}$. For end tasks, they accept the last frame $\mathbb{P}_T$ and the trajectory feature $\mathcal{F}_{1\to T}$ as input, since the order of trajectory feature is based on the order of the last frame.

Motion Flow Prediction. Motion flow prediction only accepts paired point clouds ($\mathcal{P}$, $\mathcal{Q}$) as the input to ensure a fair comparison with Yi *et. al.* (Yi et al. 2018) and Liu *et. al.* (Liu, Qi, and Guibas 2019). Moreover, we evaluate the backward motion flow $flow_{\mathcal{P}\leftarrow\mathcal{Q}}$, because the trajectory feature is based on the order of $\mathcal{Q}$. We utilize the L_2 loss function $\mathcal{L}_{flow}$, to learn the deformation flow as:

$$\mathcal{L}_{flow} = \left\| \tilde{flow}_{\mathcal{P}\leftarrow\mathcal{Q}}, flow_{\mathcal{P}\leftarrow\mathcal{Q}} \right\|^2. \tag{7}$$

Motion Part Segmentation. The input is the raw point cloud sequence $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_T\}$, and our segmentation results is on the last frame $\mathbb{P}_T$. Then, we adopt a cross-entropy loss function $\mathcal{L}_{seg}$, which is designed as:

$$\mathcal{L}_{seg} = \mathrm{H}(\xi(\mathbb{P}_T, \mathcal{F}_{1\to T}), l(\mathbb{P}_T)), \tag{8}$$

where $\xi : \mathbb{R}^{n\times 259} \to \mathbb{R}^{n\times c}$ is a PointNet++, and $l(\mathbb{P}_T)$ is the ground truth segmentation label.

Motion Attribute Estimation. Taking the point cloud sequence $\{\mathbb{P}_1, \mathbb{P}_2, ..., \mathbb{P}_T\}$ as input, we estimate its motion axis and motion range. Each point cloud sequence has one moving part. We denote the motion axis as the start point μ and axis orientation ω. The motion range is defined by the rotation angle θ and the shift distance ϕ. The part rigid transformation is noted as $[\boldsymbol{R}, \boldsymbol{t}]$, where $\boldsymbol{R} \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$.

It is not suitable for using the L_2 loss function because any point along the motion axis can be the start point. By observing that the start point is not essential for the translation and not changed around an axis, we ignore the loss $\mathcal{L}_\mu$ in translation, which is designed as:

$$\mathcal{L}_\mu = \|\boldsymbol{R}\tilde{\mu} - \tilde{\mu}\|^2. \tag{9}$$

Finally, we use the cosine distance for motion direction prediction and L_2 loss function for motion range estimation:

$$\mathcal{L}_{\omega,\theta,\phi} = \cos(\tilde{\omega}, \omega) + ||\tilde{\theta} - \theta||^2 + ||\tilde{\phi} - \phi||^2. \tag{10}$$

Experiments and Results

Dataset

The Motion Dataset (Wang et al. 2019). It is a 3D benchmark for part mobility analysis, which encompasses both

Method	Time (h)	EPE	ACC (0.05)	ACC (0.1)
Yi *et al.*	10.3	0.219	0.258	0.473
FlowNet3D	10	0.199	**0.424**	0.524
P^3-Net	**6.7**	**0.181**	0.415	**0.541**

Table 1: Flow estimation results on the motion dataset and the PartNet dataset. We adopt end-point-error (EPE) and flow estimation accuracy (ACC) to measure the performance of the predicted motion flow.

Method	Time (h)	IoU	ACC
MeteorNet	15.7	0.82	0.88
ASAP-Net	16.5	0.81	0.89
P^3-Net	**15.4**	**0.84**	**0.92**

Table 2: Motion semantic segmentation results on the motion dataset and the PartNet dataset.We show the IoU and overall segmentation accuracy (ACC)

motion part segmentation and motion attribute estimation, including 44 object categories and 2440 3D models.

The PartNet Dataset (Mo et al. 2019). It is a consistent dataset of 3D objects annotated with fine-grained, instance-level, and hierarchical 3D part information, and Xiang *et al.* (Xiang et al. 2020) enriched the dataset with motion attributes. It provides 26671 3D models in 24 categories.

Compared with the SOTA Methods

Motion Flow Prediction. We benchmark our method against two alternatives, Yi *et al.* (Yi et al. 2018) and FlowNet3D (Liu, Qi, and Guibas 2019). We use three metrics to test the flow prediction by all methods. We employ 3D end-point-error defined in (Yan and Xiang 2016), which is the average L_2 distance between the predicted flow and the ground truth flow. Moreover, we adopt flow estimation accuracy with two different thresholds defined in (Liu, Qi, and Guibas 2019), which describes point flows within a certain precision. We report the results of P^3-Net and two SOTA methods that take a pair of point clouds as input in Tab. 1. The results show that our method is superior to the SOTA methods. Moreover, our network requires less training time than theirs. Fig. 5 (left) shows that our predicted second frame and ground-truth second frame are overlapped.

Motion Part Segmentation. We compare our network with MeteorNet (Liu, Yan, and Bohg 2019) and ASAP-Net (Cao et al. 2020). We choose two metrics to analyze the performance of these methods. The mean per-part Intersection over Union (IoU) defined in (Yi et al. 2016), as well as overall segmentation accuracy (ACC), is the popular measure for 3D segmentation. We report the IoU and ACC in total shown in Tab. 2. The results demonstrate that our P^3-Net outperforms two SOTA methods. The reason is our method makes full use of motion-related temporal information from point cloud sequences by explicit learning point correspondences, which can transfer similar motion part features into the last frame. We show the visualization in Fig. 5 (middle).

Method	Time (h)	MD	OE	θ_e	ϕ_e
Shi *et al.*	**16**	0.045	**0.036**	0.045	0.017
P^3-Net	17.8	**0.039**	0.042	**0.038**	**0.013**

Table 3: The comparison of our method with Shi *et al.* in the task of motion attribute estimation. We choose MD, OE, θ_e and ϕ_e as the measures.

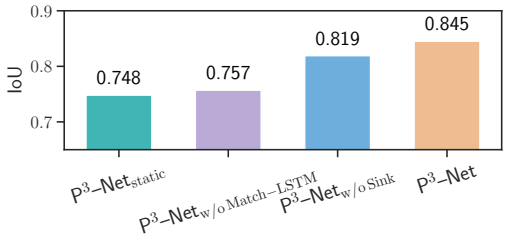

Figure 6: The histogram for the performance of four different variants P^3-Net$_{\text{static}}$, P^3-Net$_{\text{w/o Match−LSTM}}$, P^3-Net$_{\text{w/o Sink}}$ and P^3-Net.

Motion Attribute Estimation. Motion attribute estimation consists of motion axis prediction and motion range estimation. Unlike Shi *et al.* (Shi, Cao, and Zhou 2021), our method can directly accept the raw point cloud sequence as input and train in an end-to-end fashion. We utilize four metrics to evaluate the accuracy of the predicted motion axis and motion range. To evaluate the motion axis, we adopt minimum distance (MD) from the predicted start point to the ground truth axis and orientation error (OE) between the predicted axis and the ground truth axis, which are defined in (Wang et al. 2019). We also employ the θ_e and ϕ_e to measure the rotation range error and translation range error, which are defined in (Shi, Cao, and Zhou 2021). We report the results in Tab. 3, which demonstrates that our methods achieve better performance than theirs. Visualization results are shown in Fig. 5 (right).

Ablation Study

The Effect of Our Modules. In our approach, we adopt the attention module to establish point correspondences. Then, we feed the match matrix to the Match-LSTM module to extract features. The core of Match-LSTM module is the match matrix, otherwise it becomes a vanilla LSTM module without the match matrix. To validate the effect of individual modules, we compare four different variants incrementally. P^3-Net$_{\text{static}}$ inputs the last frame of each point cloud sequence. Conversely, P^3-Net$_{\text{w/o Match−LSTM}}$ inputs the point cloud sequence, which contains a vanilla LSTM to aggregates features. P^3-Net$_{\text{w/o Sink}}$ contains the Match-LSTM, which the match matrix is obtain by the at-

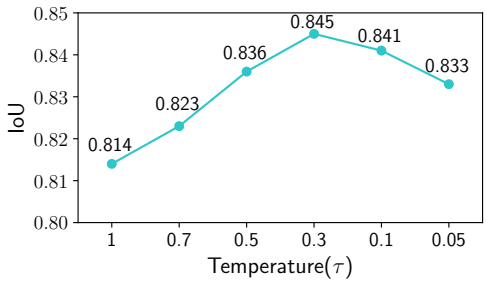

Figure 7: The impact of the temperature on the performance, where mean Interaction over Union (IoU) represents the segmentation accuracy.

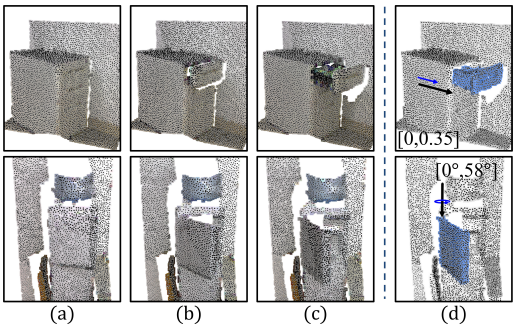

Figure 8: The visualization of the real data. We show three frames of the point cloud sequence (a-c). We demonstrate the motion part segmentation and motion attribute estimation on the last frame (d).

tention module without the GS module. For P^3-Net$_{\text{static}}$ and P^3-Net$_{\text{w/o Match-LSTM}}$, we verified using LSTM to obtain the dynamic information improves the performance. For P^3-Net$_{\text{w/o Match-LSTM}}$ and P^3-Net$_{\text{w/o Sink}}$, the comparison verifies the success of attention module. For P^3-Net$_{\text{w/o Sink}}$ and P^3-Net, we found that the GS module improves the performance.

The Effect of the Temperature τ. Match-LSTM requires a matching matrix to reaggregate point features, which is better to make the matching matrix to be a sparse matrix because the sparse matrix can help the network to find better point correspondence. Thus, we adopt a Gumbel-Sinkhorn module for Match-LSTM. In the Gumbel-Sinkhorn module, the hyperparameter, τ, controls the matching matrix's sparse degree, and the lower temperature can improve the sparse degree. Nevertheless, the lower temperature usually leads to higher variance in gradients, which makes training unstable. Thus, we conduct an ablation study to find a better setting of τ. We illustrate the result in Fig. 7 and find out that the network performance improves until it peaks at around 0.3.

Qualitative Experiment on Real Scan

It often appears real data with scanning artifacts because of the single view and noise. To process real data, we train our network with synthetic scan dataset by the simulation environment. We choose the τ that is not too low in the Gumbel-Sinkhorn module and adopt the soft correspondence to mitigate those effects. However, few large real scan datasets contain part-level motion. To verify the effectiveness, we scan the real data by ourselves, and Fig. 8 shows examples of part mobility prediction on real scan data. Despite these challenges, our network still outputs reasonable results and demonstrates its robustness.

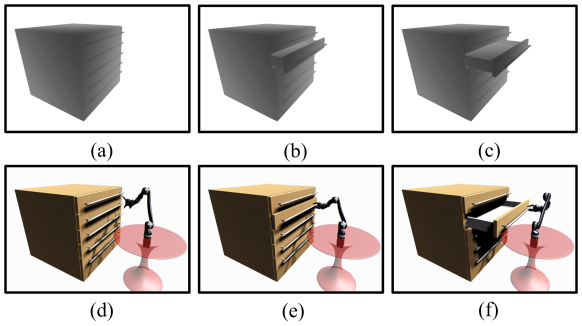

Figure 9: The visualization of opening a drawer by a robot arm (d-f), which motion attributes (*i.e.*, axis & range) are estimated from the motion sequence (a-c) by our method.

Robot Arm Manipulation Application

We deploy a simulation environment based on Unity and ROS (Quigley et al. 2009) for robot arm manipulation. Fig. 9 shows an example of our system. We first demonstrate the actual motion of the drawer, and then the robot arm repeats the interaction by estimating the motion attribute. We adopt GG-CNN (Morrison, Leitner, and Corke 2018) to detect the contact point and use our network to estimate motion attributes, including motion axis and range. The experiment indicates that our method would be conducive to the household robot and personal assistant robot.

Conclusion

Part mobility parsing is essential for a robot to perceive the surroundings and interact with the real world. In this work, we present a novel approach to parse part mobility from point cloud sequences via learning explicit point correspondences. To this end, we design a new deep network architecture (P^3-Net) with three efficient modules (*i.e.*, Match-LSTM, Attention, and Gumbel-Sinkhorn), which can jointly optimize the trajectory feature extraction and the point correspondence establishment. We conduct intensive experiments on public benchmarks to validate the parsing performance. Comparison results show that our approach outperforms other SOTA methods on various tasks. Moreover, we integrate our approach into a robot perception module to perform part mobility parsing. The parsing results can effectively support the robot planning and control modules to accomplish the manipulation tasks.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China (2019YFF0302902), and National Natural Science Foundation of China (U1736217 and 61932003).

References

Agarap, A. F. 2018. Deep Learning using Rectified Linear Units (ReLU). *CoRR*, abs/1803.08375.

Ao, S.; Hu, Q.; Yang, B.; Markham, A.; and Guo, Y. 2021. SpinNet: Learning a General Surface Descriptor for 3D Point Cloud Registration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, 11753–11762.

Aoki, Y.; Goforth, H.; Srivatsan, R. A.; and Lucey, S. 2019. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 7163–7172.

Bai, X.; Luo, Z.; Zhou, L.; Fu, H.; Quan, L.; and Tai, C. 2020. D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 6358–6366.

Cao, H.; Lu, Y.; Pang, B.; Lu, C.; Yuille, A. L.; and Liu, G. 2020. ASAP-Net: Attention and Structure Aware Point Cloud Sequence Segmentation. In *31st British Machine Vision Conference 2020, BMVC 2020*.

Choy, C.; Dong, W.; and Koltun, V. 2020. Deep Global Registration. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 2511–2520.

Deng, H.; Birdal, T.; and Ilic, S. 2018. PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 195–205.

Elbaz, G.; Avraham, T.; and Fischer, A. 2017. 3D Point Cloud Registration for Localization Using a Deep Neural Network Auto-Encoder. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2472–2481.

Fischler, M. A.; and Bolles, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6): 381–395.

Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; and Bennamoun, M. 2021. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(12): 4338–4364.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780.

Hu, R.; Li, W.; van Kaick, O.; Shamir, A.; Zhang, H.; and Huang, H. 2017. Learning to predict part mobility from a single static snapshot. *ACM Trans. Graph.*, 36(6): 227:1–227:13.

Khoury, M.; Zhou, Q.; and Koltun, V. 2017. Learning Compact Geometric Features. In *2017 IEEE International Conference on Computer Vision, ICCV 2017*, 153–161.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems 2012*, 1106–1114.

Kuhn, H. W. 2010. The Hungarian Method for the Assignment Problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, 29–47.

Li, X.; Wang, H.; Yi, L.; Guibas, L. J.; Abbott, A. L.; and Song, S. 2020. Category-Level Articulated Object Pose Estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 3703–3712.

Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; and Chen, B. 2018. PointCNN: Convolution On X-Transformed Points. In *Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 828–838.

Liu, X.; Qi, C. R.; and Guibas, L. J. 2019. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 529–537.

Liu, X.; Yan, M.; and Bohg, J. 2019. MeteorNet: Deep Learning on Dynamic 3D Point Cloud Sequences. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*, 9245–9254.

Maturana, D.; and Scherer, S. A. 2015. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015*, 922–928.

Mena, G. E.; Belanger, D.; Linderman, S. W.; and Snoek, J. 2018. Learning Latent Permutations with Gumbel-Sinkhorn Networks. In *6th International Conference on Learning Representations, ICLR 2018*.

Menze, M.; and Geiger, A. 2015. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, 3061–3070.

Min, Y.; Zhang, Y.; Chai, X.; and Chen, X. 2020. An Efficient PointLSTM for Point Clouds Based Gesture Recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 5760–5769.

Mo, K.; Guibas, L. J.; Mukadam, M.; Gupta, A.; and Tulsiani, S. 2021. Where2Act: From Pixels to Actions for Articulated 3D Objects. *CoRR*, abs/2101.02692.

Mo, K.; Zhu, S.; Chang, A. X.; Yi, L.; Tripathi, S.; Guibas, L. J.; and Su, H. 2019. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 909–918.

Morrison, D.; Leitner, J.; and Corke, P. 2018. Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach. In *Robotics: Science and Systems XIV*.

Papazov, C.; and Burschka, D. 2011. Deformable 3D Shape Registration Based on Local Similarity Transforms. *Comput. Graph. Forum*, 30(5): 1493–1502.

Poiesi, F.; and Boscaini, D. 2021. Generalisable and distinctive 3D local deep descriptors for point cloud registration. *CoRR*, abs/2105.10382.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 77–85.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Conference on Neural Information Processing Systems 2017*, 5099–5108.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, 5. Kobe, Japan.

Rusu, R. B.; Blodow, N.; and Beetz, M. 2009. Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation, ICRA 2009*, 3212–3217.

Shi, Y.; Cao, X.; and Zhou, B. 2021. Self-Supervised Learning of Part Mobility from Point Cloud Sequence. *Comput. Graph. Forum*, 40(6): 104–116.

Simonovsky, M.; and Komodakis, N. 2017. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 29–38.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks. In *Conference on Neural Information Processing Systems 2014*, 3104–3112.

Tombari, F.; Salti, S.; and di Stefano, L. 2010. Unique Signatures of Histograms for Local Surface Description. In *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision Proceedings, Part III*, volume 6313 of *Lecture Notes in Computer Science*, 356–369.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *Conference on Neural Information Processing Systems 2017*, 5998–6008.

Wang, X.; Girshick, R. B.; Gupta, A.; and He, K. 2018. Non-Local Neural Networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 7794–7803.

Wang, X.; Zhou, B.; Shi, Y.; Chen, X.; Zhao, Q.; and Xu, K. 2019. Shape2Motion: Joint Analysis of Motion Parts and Attributes From 3D Shapes. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 8876–8884.

Wang, Y.; and Solomon, J. 2019. Deep Closest Point: Learning Representations for Point Cloud Registration. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019*, 3522–3531.

Xiang, F.; Qin, Y.; Mo, K.; Xia, Y.; Zhu, H.; Liu, F.; Liu, M.; Jiang, H.; Yuan, Y.; Wang, H.; Yi, L.; Chang, A. X.; Guibas, L. J.; and Su, H. 2020. SAPIEN: A SimulAted Part-Based Interactive ENvironment. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 11094–11104.

Yan, Z.; Hu, R.; Yan, X.; Chen, L.; van Kaick, O.; Zhang, H.; and Huang, H. 2019. RPM-Net: recurrent prediction of motion and parts from point cloud. *ACM Trans. Graph.*, 38(6): 240:1–240:15.

Yan, Z.; and Xiang, X. 2016. Scene Flow Estimation: A Survey. *CoRR*, abs/1612.02590.

Yang, Z.; Pan, J. Z.; Luo, L.; Zhou, X.; Grauman, K.; and Huang, Q. 2019. Extreme Relative Pose Estimation for RGB-D Scans via Scene Completion. In *2019 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*, 4531–4540.

Yew, Z. J.; and Lee, G. H. 2018. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration. In *Computer Vision - ECCV 2018 - 15th European Conference Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, 630–646.

Yi, L.; Huang, H.; Liu, D.; Kalogerakis, E.; Su, H.; and Guibas, L. J. 2018. Deep part induction from articulated object pairs. *ACM Trans. Graph.*, 37(6): 209:1–209:15.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. J. 2016. A scalable active framework for region annotation in 3D shape collections. *ACM Trans. Graph.*, 35(6): 210:1–210:12.