Deep Part Induction from Articulated Object Pairs

LI YI, Stanford University
HAIBIN HUANG, Megvii (Face++) Research
DIFAN LIU, University of Massachusetts Amherst
EVANGELOS KALOGERAKIS, University of Massachusetts Amherst
HAO SU, University of California, San Diego
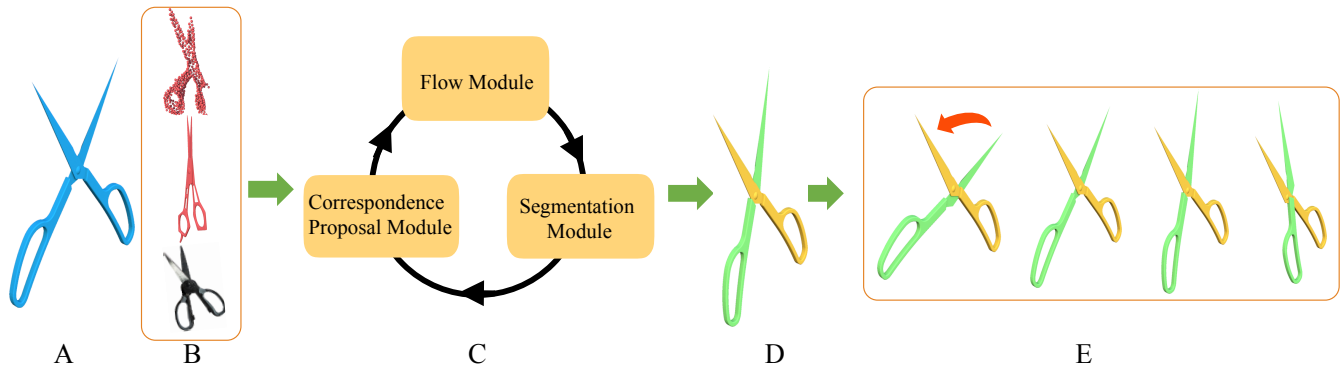LEONIDAS GUIBAS, Stanford University

Fig. 1. We present a neural network for discovering articulated parts of objects. Given a point set representing a 3D CAD model (A) and other functionally similar objects in the form of a scan, 3D mesh or RGB image (B), our network (C) can co-segment the input objects into their articulated parts and parse their underlying motion (D). The learned articulation can further be used for automatically animating the input CAD model (E).

Object functionality is often expressed through part articulation – as when the two rigid parts of a scissor pivot against each other to perform the cutting function. Such articulations are often similar across objects within the same functional category. In this paper we explore how the observation of different articulation states provides evidence for part structure and motion of 3D objects. Our method takes as input a pair of unsegmented shapes representing two different articulation states of two functionally related objects, and induces their common parts along with their underlying rigid motion. This is a challenging setting, as we assume no prior shape structure, no prior shape category information, no consistent shape orientation, the articulation states may belong to objects of different geometry, plus we allow inputs to be noisy and partial scans, or point clouds lifted from RGB images. Our method learns a neural network architecture with three modules that respectively propose correspondences, estimate 3D deformation flows, and perform segmentation. To achieve optimal performance, our architecture alternates between correspondence, deformation flow, and segmentation prediction iteratively in an ICP-like fashion. Our results demonstrate that

Authors' addresses: Li Yi, Stanford University, ericyi@stanford.edu; Haibin Huang, Megvii (Face++) Research; Difan Liu, University of Massachusetts Amherst; Evangelos Kalogerakis, University of Massachusetts Amherst; Hao Su, University of California, San Diego; Leonidas Guibas, Stanford University.

our method significantly outperforms state-of-the-art techniques in the task of discovering articulated parts of objects. In addition, our part induction is object-class agnostic and successfully generalizes to new and unseen objects.

CCS Concepts: • **Computing methodologies → Neural networks**; **Shape analysis**;

Additional Key Words and Phrases: shape correspondences, motion based part segmentation, deep learning, differentiable sequential RANSAC

1 INTRODUCTION

Our everyday living environments are largely populated with dynamic and articulated objects, which we can interact with through their moving parts e.g., swivel chairs, laptops, bikes, tools, to name a few. In order for autonomous agents to correctly interact with such objects, the agents need to be equipped with algorithms that are able to parse these objects into their functional parts and motion. Decomposing 3D shape representations into their moving parts is also important for several graphics, vision, and robotic applications, such as predicting object functionality, human-object interactions, guiding shape edits, animation, and reconstruction.

Recently, with the availability of large 3D datasets and the use of deep learning techniques, significant progress has been made in the task of supervised part segmentation. Given a large volume of

3D shapes with part segmentation annotations, we can train deep neural networks to reliably segment new shapes from the same object category. Although progress in supervised part segmentation is impressive, contemporary algorithms are still far inferior to humans when it comes to parsing 3D shapes from *novel* object categories, and also to discovering new functional parts whose types are not covered in the training sets. Being able to parse 3D objects into functional parts not seen before is fundamentally crucial towards building intelligent agents that must understand the object functionality, so as to have physical interactions with them, simulate such interactions, assist humans in augmented reality scenarios or autonomous robotics settings. Ideally, as new objects continuously emerge, an agent should possess the ability to induce their structures from the few observations and limited interaction experience.

In this work, we are interested in discovering 3D object structure according to the mobility of their underlying parts. Articulation can be a crucial clue in part structure determination, as it invariably involves motion of one part against others. We also aim to induce part structure from observations of different articulation states of an object in noisy settings e.g., 3D partial scans, or even when the articulation state observations originate from geometrically different, yet functionally related, objects. Take Figure 1 as an example: by observing different scissors under various opening angles, we expect to induce that a scissor is made of two thin blades that can rotate around a pivot.

Mobility-based shape parsing brings a novel angle to the part determination problem. Among different principles for finding parts of objects (e.g., Gestalt theory [Palmer 1977]), the mobility-based principle provides an unambiguous decomposition. Besides, mobility based parsing of an object can facilitate its functional understanding – man-made objects are designed to function or interact with other objects (including humans) in ways realized by particular moving structures. As in the previous example, a scissor is designed with two pivoting blades to enable cutting. Recently, there have been increasing efforts devoted to obtaining a functional understanding of 3D objects from a motion perspective [Hu et al. 2017; Pirk et al. 2017]. The recent work of [Hu et al. 2017] to infer part mobility types such as rotation and translation is particularly relevant to our effort, with the difference that they use a pre-segmentation of the object – something we do not assume.

Though seemingly trivial for humans, automatic part induction from observations of articulation states of objects is challenging for a number of reasons. First, the input objects can differ considerably in both geometry and pose. Second, the articulation differences must be aggregated into coherently moving parts with clean boundaries. Third, in the case of scans, part induction must be robust to both noise and missing data.

We designed a deep neural network-based system to address the problem, encouraged by its robustness to data variation in various 2D and 3D data understanding tasks. However, unlike previous supervised methods, we do not assume any prior knowledge of the input shape class or object structure (e.g., part labels, or pre-defined components). The design of our neural network is motivated by the following observations: (a) Establishing local correspondences between the input shapes requires much less global and high-level information compared with semantic understanding tasks such

as classification and segmentation, thus a learned correspondence module is potentially much more transferrable to novel categories. Based on the predicted correspondences, strong cues can be obtained to infer a deformation flow field, capturing the differences in the articulation states of the two input shapes, (b) However, local correspondences are often ambiguous and fuzzy due to shape symmetries, noise in scans, and geometric differences between the input shapes, thus one needs to incorporate global shape information to robustly translate correspondence cues into deformation flow. (c) Articulated parts can then be discovered by aggregating the deformation flow into rigid part motions. Following these observations, we designed a neural network that operates in three stages: (a) It learns to extract discriminative local features and propose possible correspondences between the input shape pair, (b) learns to disambiguate correspondence confusion caused by symmetries, noise, and geometric differences by integrating global shape information to predict deformation flow, and finally, (c) learns to group points into parts from the predicted flow, leveraging a part rigidity assumption. These three stages are executed by neural network modules trained from a massive synthetic training dataset of articulated shapes within a multi-task learning framework.

The three modules can also be executed iteratively to reinforce each other. This iterative procedure is akin to ICP approaches or RANSAC-based algorithms [Fischler and Bolles 1981] that iterate between model fitting and geometric verification to discover primitives out of clean and same instance pairs. However, our learning based algorithm enjoys much stronger robustness to input data variation and corruption, which allows us to do induction from different instances that may have limited overlap, and avoids the tedious and error-prone tuning of sensitive hyper-parameters.

We performed extensive qualitative and quantitative evaluation on both synthetic and real datasets. We also conducted ablation studies to confirm the utility of all the above mentioned network stages. As we demonstrate in the results, previous methods largely fail to obtain satisfactory results even for objects with a single Degree of Freedom (DoF) in their joints, while our network successfully parses objects with either one or several DoFs. Overall, results demonstrate that our network dramatically outperforms existing state-of-the-art methods.

In summary, this paper introduces a new deep learning method to parse 3D objects into moving parts based only on input static shape snapshots without any prior knowledge of the input object class or structure (part labels, or pre-existing components). Specifically, our method makes the following key contributions:

(1) Introduces a learning framework for mobility-based part segmentation from articulated object pairs that generalizes to novel object categories.

(2) Provides three new neural network modules: (a) a module for robust dense correspondence estimation between shapes with large geometric and articulation differences, with the additional capability of partial shape matching through a correspondence mask that handles structural shape differences and missing data, (b) a module, called PairNet, capable of inferring pairwise relationships, such as deformation flow, between two input shapes, and finally (c) a module for shape

segmentation by generating hypotheses of rigid motions from deformation flow and sequentially extracting parts whose motion is consistent with these hypotheses. This new module implements a differential, neural-based RANSAC procedure, which can be useful in other applications requiring structure discovery from noisy observed data.

(3) Demonstrates a neural net-based mutual reinforcement procedure iterating between correspondence, flow, and part estimation.

2 PRIOR WORK

Our work is primarily related to 3D shape segmentation approaches that aim to extract rigidly moving parts from 3D meshes, point clouds or RGBD sequences. Our architecture extracts matching probabilities between points on 3D shapes at an intermediate stage, thus it is also related to learning-based 3D shape correspondence approaches. We briefly overview these approaches here.

Rigid part extraction. Given an input sequence of meshes, point clouds, or RGBD data representing an underlying articulated 3D object under continuously changing poses, various approaches have been proposed to detect and extract its rigidly moving parts. In contrast to all these approaches, we do not assume that we are given a continuous, ordered sequence of 3D object poses. In addition, our method can infer the rigidly moving parts of the input 3D object by matching it to other geometrically different shapes available in online repositories, or partial scans. Thus, our setting is more general compared to previous rigid part extraction methods, yet we briefly overview them here for completeness.

In the case of RGBD sequences, early works attempt to estimate the 3D motion field (scene flow) between consecutive frames [Christoph et al. 2015; Hornacek et al. 2014; Quiroga et al. 2014; Vogel et al. 2014]. To recover parts, super segments can be extracted and grouped according to their estimated rigid transformations from the motion field [Golyani et al. 2017]. Alternatively, patches or points lifted from the RGBD frames can be clustered into segments based on their overall flow similarity across frames using Expectation-Maximization or coordinate descent formulations [Jaimez et al. 2015; Stückler and Behnke 2015]. Parts can also be extracted from 3D point flows through direct clustering on point trajectories [Pillai et al. 2014; Tzionas and Gall 2016a]. More similarly to our approach, the concurrent learning method by Shao et al. [2018] trains a joint flow estimation and segmentation network for motion-based part detection in objects. However, their approach mainly relies on RGB color to compute flow, and cannot handle complex structures or large articulation differences, as discussed in our results section.

In a similar spirit, in the case of raw 3D point cloud sequences, given established point-wise correspondences between consecutive point sets aligned through ICP, the point trajectories can be grouped through clustering and graph cut techniques [Kim et al. 2016; Yuan et al. 2016a]. Alternatively, joints with their associated transformations can be fitted according to these trajectories based on consensus voting techniques, such as RANSAC [Li et al. 2016]. Another set of methods attempts to fit pre-defined skeletons or templates to the input sequences using non-rigid registration techniques (see [Chang et al. 2012] for a survey), random forest regressors and classifiers [Shotton et al. 2013], or more recently through neural networks [Bogo et al. 2016; Mehta et al. 2017; Newell et al. 2016; Tomè et al. 2017; Toshev and Szegedy 2014]. However, these methods are specific to particular classes of objects, predominantly human bodies. Finally, in the case of deforming meshes with explicit vertices and triangle correspondences, mean shift clustering on rotation representations can be used to recover the rigid parts of the deforming shape [James and Twigg 2005].

The above approaches decouple shape correspondences and part extraction in separate steps, which often contain several hand-tuned parameters. Our method instead computes point correspondences and part segmentations in a single deep-learned architecture trained from a massive dataset of shapes.

Shape segmentation. A widely adopted approach in 3D shape segmentation is to train a classifier that labels points, faces, or patches based on an input training dataset of shapes with annotated parts (see [Xu et al. 2016] for a recent survey). More recent supervised learning approaches employ deep neural net architectures operating on multiple views [Kalogerakis et al. 2017], volumetric grids [Maturana and Scherer 2015], spatial data structures (kd-trees, octrees) [Klokov and Lempitsky 2017; Riegler et al. 2017; Wang et al. 2017], point sets [Qi et al. 2017a,b; Su et al. 2018], surface embeddings [Maron et al. 2017], or graph-based representations of shapes [Yi et al. 2017]. These methods can only extract parts whose labels have been observed in the training set, and cannot discover new parts.

Our work is more related to co-segmentation and joint segmentation approaches that aim to discover common parts in an input set of 3D shapes without any explicit tags. To discover common parts, geometric descriptors can be extracted per point or patch on the input 3D shapes, or geometric distances can be computed between candidate shape segmentation, then clustering can reveal common parts of shapes [Hu et al. 2012; Sidi et al. 2011; van Kaick et al. 2013]. However, the resulting clusters cannot be guaranteed to correspond to common functional parts. Our architecture is instead optimized to segment shapes under the assumption that functional parts in articulated objects predominantly undergo rigid motions, which is often the case for several man-made objects.

Alternatively, a family of approaches builds point-wise correspondences or functional maps between shapes [Golovinskiy and Funkhouser 2009; Huang et al. 2014, 2008; Kim et al. 2013], then employ an optimization approach that attempts to find parts that maximize geometric part similarity, or additionally satisfy cyclic consistency constraints. These methods largely depend on the quality of the initial correspondences and maps, while part similarity often relies on hand-engineered geometric descriptors. When input shape parts undergo large rigid transformations, their optimization approach can easily get stuck in unsatisfactory minima. Our approach instead learns to jointly extract shape correspondence and parts even in cases of large motions. Our experiments demonstrate significantly better results than co-segmentation approaches.

Shape correspondences. Since our architecture extracts point-wise correspondence probabilities as an intermediate stage, our work is also related to learning-based methods for 3D shape correspondences. In the context of deformable human bodies, deep learning architectures operating on intrinsic representations [Boscaini et al.

2015; Masci et al. 2015; Monti et al. 2017] have demonstrated excellent results. However, these methods cannot handle 3D shapes with largely different topology or structure, due to the instability of their spectral domain. Volumetric, view-based and point-based neural networks have been proposed to learn point-based descriptors for correspondences between structurally and geometrically different shapes [Huang et al. 2017; Qi et al. 2017b; Zeng et al. 2017]. Given shape correspondences extracted by these methods, one could attempt to recover rigid parts using RANSAC or Hough voting techniques [Li et al. 2016; Mitra et al. 2006] in a separate step. However, as we demonstrate in our results section, decoupling correspondences and part extraction yields significantly worse results compared to our architecture.

3 OVERVIEW

Our method co-segments input 3D shapes into rigidly moving parts through a deep architecture shown in Figure 1(C). Its modular design is motivated by the observation that estimating correspondences and deformation flows between shapes can provide cues for extracting rigidly moving parts, and in turn, the extracted piece-wise rigid motions can further improve the shape correspondences and deformation flows.

Neural network design. In contrast to prior rigid part extraction and traditional ICP approaches, our shape correspondences are not based on closest points and hand-engineered geometric descriptors but instead are extracted through a learned neural network module. This module, which we refer to as *correspondence proposal module*, is trained to map the input shape pair geometry into probabilistic point-wise correspondences (Figure 2). The module can handle large differences in both geometry and articulations in the input shapes as well as missing data and noise in the case of input 3D scans. Shape correspondences can provide strong cues for deformation flows and rigidly moving parts, but in general are not enough alone to reliably extract those. The reason is that correspondences are often ambiguous and fuzzy due to shape symmetries, missing parts and noise in scans, geometric and structural differences between the input shapes. Thus, our network incorporates another learned module, which we refer to as *flow module*, that learns to robustly translate the extracted fuzzy correspondences into a deformation flow field (Figure 3). The module is based on a new type of network, called PairNet, designed to extract pairwise relationships between point sets. To discover the underlying shape structure, the deformation flows are aggregated into piecewise rigid motions that reveal the underlying shape parts. Instead of using hand-engineered voting or clustering strategies, the deformation flows are aggregated through a third, learned neural network module, called the *segmentation module* (Figure 4). Since the number and motion of parts are not known *a priori*, the module first extracts rigid motion hypotheses from the deformation flows, discovers their support over the shape (i.e, groups points that tend to follow the same underlying motion), then sequentially extracts rigidly moving parts based on their support until no other parts can be discovered. The module is based on a new recurrent net-based architecture, called Recurrent Part Extraction Network, designed to handle sequential part discovery.

Iterative execution. Inspired by ICP approaches which alternate between estimating shape correspondences and alignment, our architecture iteratively executes the correspondence, deformation flow, and segmentation modules in a closed loop. Establishing correspondences provides cues for predicting deformation flow, the deformation flow helps extracting rigid parts, and in turn rigid parts helps improving shape correspondences and deformation flow. The loop is executed until the best possible alignment is achieved i.e., the total magnitude of the deformation flow field is minimized. In practice, we observed that this strategy converges and yields significantly better segmentations compared to executing the network pipeline only once.

4 NETWORK ARCHITECTURE

Our network takes as input a pair of shapes $\{P, Q\}$ in the form of 3D point sets. If either shape is in the form of a 3D mesh, we uniformly sample its surface using N points ($N = 512$ in our implementation). The only requirement for the input shape pair is that they should represent functionally related objects with rigidly moving parts in different articulation state. We do not make any assumptions on the shape orientation, order of points in the point set representations, number of underlying parts and DoFs. Next, we discuss the modules of our architecture in detail.

4.1 Correspondence Proposal Module

The processing of the input shape pair $\{P, Q\}$ starts with the correspondence proposal module visualized in Figure 2. Each shape in the pair is processed through a PointNet++ branch that outputs a $64-$dimensional feature representation for each point on the shape (we refer to the supplementary material for details regarding the PointNet++ structure). The two PointNet++ branches share their parameters i.e., have identical MLP layers such that the input geometry is processed in a consistent manner independently of the shape order in the pair. Then, for each pair of points across the two shapes, our architecture concatenates their extracted feature representation i.e., given the representation $\mathbf{u}_i^{(p)}$ for a point i on the shape P, and the representation $\mathbf{u}_j^{(q)}$ for a point j on the other shape Q, the resulting pair representation is $\mathbf{u}_{ij} = \{\mathbf{u}_i^{(p)}, \mathbf{u}_j^{(q)}\}$. Performing this concatenation for all pairs of points (or in other words, using the cartesian product of the point set representations of the two shapes) yields a tensor of size NxNx$2D$ ($N = 512$ is the number of input points per shape, $D = 64$ in our implementation). Each pair representation $\mathbf{u}_{ij}$ is transformed through a Multi-Layer Perceptron (MLP) (containing 128 nodes in each of its 3 hidden layers) into a confidence value $c_{i,j}$ that expresses how likely the point p_i matches, or corresponds to point q_j. The confidence is further transformed into a probability using the softmax function. Executing the same MLP for all pairs of points and passing the resulting confidences through softmax yields a pairwise matching probability matrix $\mathcal{M}$ with size NxN that describes the probability of matching any pair of points on the two shapes. The network is then trained to output high probabilities for corresponding points in the training data.

Since there might be points on the first shape P that have no correspondences to any other point on the second shape Q due
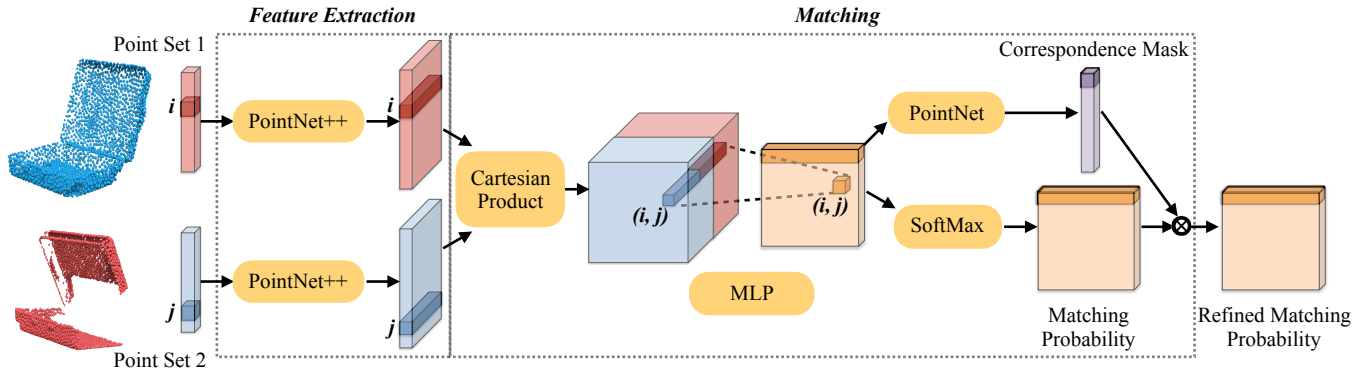
Fig. 2. Correspondence proposal module. We use a PointNet++ based sub-module to extract point-wise features for the input point clouds. The learned features are further fed into a matching sub-module for correspondence proposal. The sub-module also predicts a correspondence mask that determines which points should be matched or not.

to missing data or structural differences, the correspondence proposal module also outputs the probability for each point on the first shape being matched or not. We refer to this output as correspondence mask. Specifically, the confidences of each point on the shape P, stored in the vector $\mathbf{c}_i = c_{i,*}$ (where $*$ means all other points on the shape Q) are processed through a PointNet, that aggregates confidences throughout the whole vector, to determine the probability c_i for matching the point i of shape P with any other point on the shape Q. During training, the network also receives supervisory signal for this output i.e., whether each training point possesses a correspondence or not. We multiply each row of the above pairwise matching probability matrix $\mathcal{M}$ with the estimated probability of the corresponding point being matched, resulting in the refined pairwise matching probability matrix $\hat{\mathcal{M}}$ as the output of our correspondence proposal module.

4.2 Flow Module

The flow module, visualized in Figure 3, aims to produce a 3D deformation flow field $\mathbf{f}$ from the shape P to shape Q, which provides cues for determining common shape parts along with their rigid motions. One possibility would be to use the pairwise matching probability matrix $\hat{\mathcal{M}}$ alone to infer this field. However, we found that this is not sufficient, which is not surprising since the flow should also depend on point positions as well i.e., if we rotate one shape, the pairwise correspondence probabilities should remain the same, yet the flow would change. Thus, we pass both point position and correspondence information as input to the flow module.

Specifically, for each pair of points across the two shapes, we compute their relative displacement, or disparity i.e., $\mathbf{d}_{ij} = \mathbf{x}_j^{(q)} - \mathbf{x}_i^{(p)}$, where $\mathbf{x}_i^{(p)}$ is the position of point i from shape P, and $\mathbf{x}_j^{(q)}$ is the position of point j from shape Q. Computing the all-pairs displacement matrix yields a NxNx3 pairwise displacement matrix, where 3 corresponds to the xyz channels. The pairwise displacement matrix is concatenated together with the refined matching probability matrix $\hat{\mathcal{M}}$ along their 3rd dimension, forming a NxNx4 matrix passed as input to the flow module.

The flow module first processes each row of the stacked matrix through a PointNet (see supplementary material for architecture

details). The PointNet aggregates information from all possible displacements and correspondences for each point on shape P. Note here that the set, which PointNet aggregates on, is the set of points from Q. The output of the PointNet is a representation of dimensionality 256 that encodes this aggregated information per point. Processing all points of shape P through the same PointNet, yields a matrix of size Nx256 that stores all point representations of shape P. This point set representation is then processed through a PointNet++ (see supplementary material for architecture details). The PointNet++ hierarchically captures local dependencies in these point representations (e.g., neighboring points are expected to have similar flows), and outputs the predicted flow field $\mathbf{f}$ (Nx3) on shape P. As explained in the next section, the module is trained to extract flow using supervisory signal containing the ground-truth 3D flows for several possible shape orientations to ensure rotational invariance.

We refer to the combination of the PointNet and PointNet++ as PairNet. PairNet takes a pairwise matrix between two sets as input. It first globally aggregates information along the second set through the PointNet to extract a per-point representation for each point in the first set. The point representations are then hierarchically aggregated into a higher-level representation through a PointNet++, to encode local dependencies in the first set.

Even facilitated by the power of neural networks, the deformation flow estimation is still far from perfect (Figure 5). It is inherently not well-defined due to geometric or structural differences across shape pairs. As described in the next paragraph, learning plays a key role to reliably extract parts from the estimated flow.

4.3 Segmentation Module

Given the estimated deformation field $\mathbf{f}$ from shape P to Q, the segmentation module decodes it towards rigid motion modes as well as the corresponding part segments between the two shapes (Figure 4). The design of the module is inspired by RANSAC approaches. First, the module generates hypotheses of rigid motions, then for each rigid motion it finds support regions on the shape (i.e., groups of shape points that follow the same rigid motion hypothesis), and finally extracts the rigid parts from the support regions one-by-one starting from the most dominant ones i.e., the ones with largest
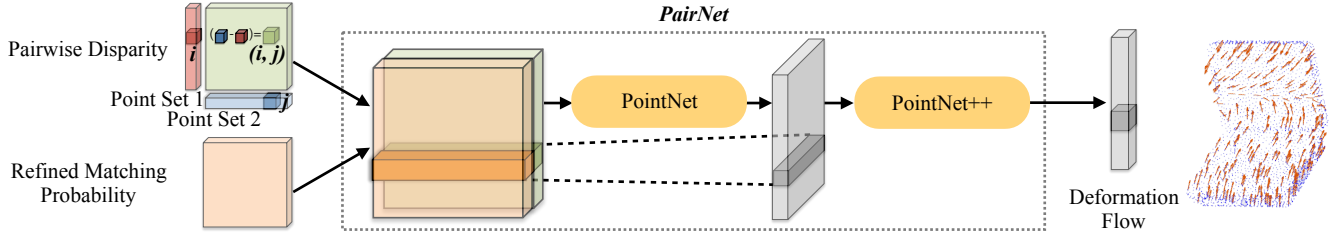
Fig. 3. Flow Module. The refined matching probabilities are concatenated with the pairwise disparity and fed into the flow module. The flow module learns a point-wise deformation flow from one point set to the other.

support. In contrast to traditional RANSAC approaches that employ hand-engineered techniques with user-defined thresholds in each of the stages (e.g., the target number of segments, inlier thresholds etc), our module implements them through learned neural network layers. In this sense, this module can be considered as a neural network based, differentiable sequential RANSAC procedure. Different from [Brachmann et al. 2017], we learn to generate hypothesis efficiently to avoid expensive sampling, plus we have a sequential hypothesis selection step which allows decoding multiple modes from input samples. In the next paragraphs, we explain this module in detail.

Hypothesis generation. The first stage of our segmentation module is to generate hypotheses of candidate part motions from the shape P towards corresponding parts of the shape Q. We here use a PointNet++ (see supplementary material for architecture details), to hierarchically aggregate the deformation flow $\mathbf{f}$ along with the point positions of the shape P to generate rigid motion hypotheses. The point positions are used as additional input to this stage along with the flow, since knowing the flow field alone without knowing the underlying geometry is not sufficient to determine a rigid motion. The output of the PointNet++ is a hypothesis for a rigid motion estimated per point i of the shape P. We experimented with various rigid motion output parameterizations, including predicting directly a 3x3 rotation matrix and a 3x1 translation vector, axis-angle and quaternion parameterizations of rotations, and affine matrice outputs followed by SVD to extract their rotational component. We found that the best performing rigid motion parameterization was through a 3x3 matrix $\hat{\mathbf{R}}_i$ and a 3x1 vector $\hat{\mathbf{t}}_i$, from which the rotational component is computed as $\mathbf{R}_i = \hat{\mathbf{R}}_i + \mathbf{I}$ followed by an SVD to project the matrix to the nearest orthogonal matrix, while the translational component is computed after applying the inferred rotation: $\mathbf{t}_i = -(\mathbf{R}_i - I) \cdot \mathbf{x}_i^{(p)} + \mathbf{f}_i + \hat{\mathbf{t}}_i$, where I is the identity transformation, $\mathbf{x}_i^{(p)}$ is the ith point position and $\mathbf{f}_i$ is the corresponding flow. We suspect that this parameterization resulted in better rigid motion and segmentation estimates due to the fact that the rotational and translational components of a rigid motion are not independent of each other and also because the elements of $\hat{\mathbf{R}}_i$ and $\hat{\mathbf{t}}_i$ have more compatible scales. Thus, for computing rotations, we predict the intermediate matrix $\hat{\mathbf{R}}_i$ which is equal to the zero matrix in case of the identity transformation (i.e., a "residual" rotation matrix), while the translation is predicted conditioned on the estimated rotation (again, a "residual" translation vector). We found that these residual representations are much easier to train and yield the best performance in terms of rigid motion and segmentation estimation.

Support prediction. Following the rigid motion hypothesis generation stage, our segmentation module predicts a probability for each point on the shape P to support, or in other words follow, each generated rigid motion hypothesis. To predict this probability, the segmentation module first examines how well each rigid motion hypothesis explains the predicted flow per point. This can be examined by applying the motion hypothesis to each point, computing the resulting displacement, and then comparing it with the predicted flow from our previous module. The displacement of a point i' after applying the hypothesis i is computed as $\mathbf{d}_{i,i'} = \mathbf{R}_i \mathbf{x}_{i'}^{(p)} + \mathbf{t}_i - \mathbf{x}_{i'}^{(p)}$, and the difference between this displacement and predicted flow is simply calculated as $\mathbf{d}_{i,i'} - \mathbf{f}_{i'}$. Computing the flow difference for each point i' and each motion hypothesis i yields a NxN pairwise flow difference matrix, where the rows correspond to rigid motion hypotheses (same number as the number of points of P) and columns corresponds to points. The module then aggregates information from all flow differences per rigid motion hypothesis to calculate its support. This is done through a PointNet operating on each row of the flow difference matrix. The PointNet is trained to output the probability of each point supporting a rigid motion hypothesis based on the estimated flow difference. Computing these probabilities for all available hypotheses yields a NxN matrix, which we refer to as support matrix $\mathcal{S}$. The rows of the matrix correspond to candidate rigid motion hypotheses and columns correspond to the per-point support probabilities.

Rigid part extraction. The last stage of the segmentation module is to decode the support matrix into a set of rigid segments in a sequential manner. Decoding is performed through a recurrent net-based architecture, that we refer to as Recurrent Part Extraction Network (RPEN). The RPEN outputs one segment at each step and also decides when to stop. It maintains a hidden state $\mathbf{h}_t = (\mathbf{e}_t, \mathbf{z}_t, \mathbf{s}_t)$, where $\mathbf{e}_t$ represents an internal memory encoding the already segmented regions so that subsequent steps decode the support matrix into different segments, $\mathbf{z}_t$ is a learned representation of the recurrent unit input designed to modulate the support matrix such that already segmented regions are downplayed, and $\mathbf{s}_t$ denotes a learned weight representing the importance of each hypothesis for segment prediction.

At each step t, the recurrent unit transforms its input support matrix $\mathcal{S}$, as well as the memory $\mathbf{e}_{t-1}$ from the previous time step, into a compact representation $\mathbf{z}_t$ through a PairNet (same combination of PointNet and PointNet++ used in the flow module). Each row of $\mathcal{S}$ is concatenated with $\mathbf{e}_{t-1}$, forming a NxNx2 matrix, and is then fed to the PairNet to generate $\mathbf{z}_t$ with dimensionality Nx32. Then
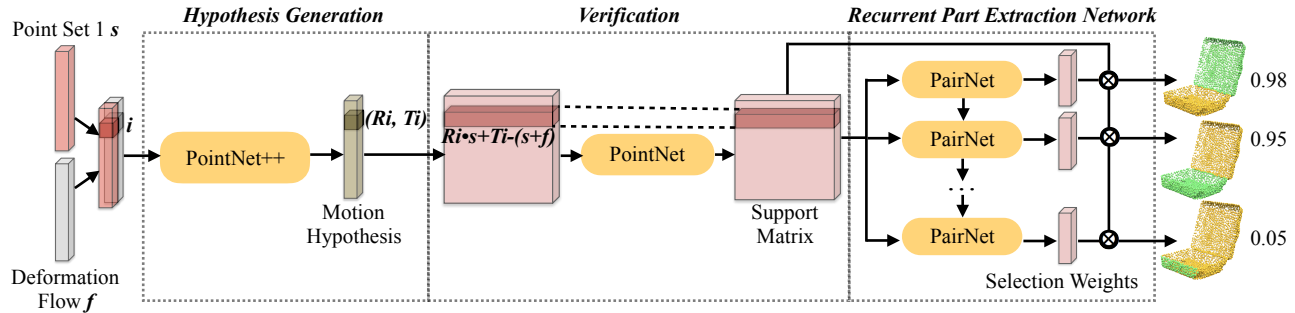
Fig. 4. Segmentation module. The predicted deformation flow on the first point set together with its point positions are fed into this module, which acts as a neural net-based, differentiable sequential RANSAC . Similar to sequential RANSAC, the inputs are processed through three sub-modules including hypothesis generation, verification and recurrent part extraction network, resulting in a set of soft segmentation indicator functions and part confidence scores.

the recurrent net decodes the representation z_t into the following two outputs through two different PointNets: (a) a continuation score r_t (a scalar value between 0 or 1) which indicates whether the network should continue predicting a new segment or stop, and (b) a $N \times 1$ vector s_t representing how much weight the support region of each hypothesis should be given to determine the output segment of shape P at the step t. The soft segmentation assignment variable y_t is generated by computing the weighted average of the corresponding hypothesis support probabilities, or mathematically $y_t = S^T \cdot s_t$. At inference time, the per-point soft segment assignments are converted into hard assignments through graph cuts [Boykov et al. 2001]. The associated rigid motion is estimated by fitting a rigid transformation to the deformation flow f on the segment. By applying the fitted rigid transformation on the points of the segmented part of shape P, we find the corresponding points on shape Q using a nearest neighbor search, then execute the same graph cuts procedure to further refine the corresponding segmented part on Q.

At each step, the Recurrent net also updates the hidden internal state h_t. The hidden state is updated in the following way: $z_t = f_{PairNet}(S, e_{t-1})$, $s_t = g(z_t)$, $e_t = (1 - e_{t-1}) \odot (S^T s_t) + e_{t-1}$, where $f_{PairNet}(\cdot)$ denotes the operation of PairNet, $g(\cdot)$ denotes a PointNet operation and $\odot$ denotes element-wise multiplication. Notice s_t in the hidden state is directly outputted to determine the segment of shape P. The segmentation module stops producing segments when the continuation score falls below 0.5 (50% probability).

4.4 Iterative Segmentation and Motion Estimation

When there are large articulation differences between the two input shapes or in the presence of noise and outliers in input scans, the execution of a single forward pass through our architecture often results in a noisy segmentation (Figure 5). In particular, an excessive number of small parts is often detected, which should be grouped instead into larger parts. We found that the main source of this problem is the estimation of the flow field f, which tends to be noisy in the above-mentioned conditions.

Inspired by ICP-like approaches, our method executes an iterative procedure to refine the prediction of the flow field. Specifically, given an initial estimated flow field f, along with rigid motions $\{H_t\}$ and segmented parts $\{P_t\}$ of shape P based on a first forward pass of the initial shape pair through our correspondence, flow, and
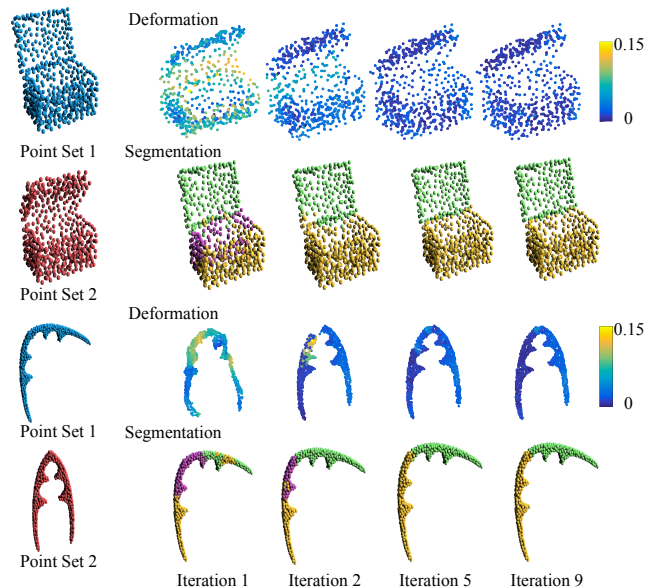

Fig. 5. Iterative refinement of deformation flow and segmentation. The outputs usually converge after 5 iterations.

segmentation modules, we produce a new deformed version P' of the original shape P. The deformed shape is produced by applying the detected rigid motions on the associated parts of the original shape i.e., $H_t \cdot P_t$. Then we compute a new "residual" flow field f' by passing the pair $\{P', Q\}$ through the same correspondence and flow module of our network. The "residual" flow field is added to the piecewise rigid deformation field $P' - P$ to compute a new refined flow field i.e., $P' - P + f'$. This refined flow field is subsequently processed through our segmentation module to further update the rigid motions $\{H_t\}$ and segmented parts $\{P_t\}$ of the original shape P. The procedure can repeat till it converges or reaches a maximum number of iterations (10 in our case).

Initialization. As in the case of ICP-like approaches, when the orientation of the two shapes differs significantly e.g., their upright or front-facing orientation is largely inconsistent, the algorithm might converge to a suboptimal configuration. In this case, at an initialization stage, we search over several different 3D global rotations

for one of the two shapes (in our implementation, 48 3D rotations, uniformly sampled from SO(3)), and initialize our iterative procedure from the one that yields the smallest flow magnitude according to the flow module. The flow evaluation is fast: it takes about 10 ms (measured on a TitanX GPU) to evaluate each candidate rotation, thus in the case of inconsistent shape orientation for the input pair, this initialization stage takes less than half of a second.

Results. The iterative version of our algorithm significantly improves both the deformation flow between P and Q as well as the segmentation of P, as demonstrated in the results section. We also refer to Figure 5 for qualitative results.

5 TRAINING

Our network is trained on a large synthetic dataset consisting of pairs of shapes with ground-truth annotations of corresponding parts along with their rigid motions. In this section, we describe the synthetic training dataset, multi-task objective function, then we describe the training procedure.

5.1 Training Dataset

Our training dataset is based on semantic part annotations [Yi et al. 2016] of ShapeNetCore [Chang et al. 2015], which contains $16,881$ segmented man-made shapes in 16 categories. For each shape P in this dataset, we generate 2 deformed versions of it by applying rotations on randomly picked segments about random axes passing through the contact regions between adjacent parts, including random translations along axes that are perpendicular to planar approximations of contact regions under the constraint that the resulting transformations keep the parts connected. We generate one pair of shapes for each P, resulting in a total number of $16,881$ pairs, with 2 to 6 moving parts per pair. We set a ratio of $1:3$ for the percentage of translations versus rotations in our deformation. For each pair of deformed shapes, we additionally generate 5 pairs of synthetic scans from random viewpoints and we conduct farthest point sampling on each synthetic scan, resulting 512 points per scan, which are then normalized to have bounding box centered at $(0,0,0)$ and with a diagonal 1.

We refer the readers to the supplementary material for a list of the categories and example visualizations of the training set. The resulting shape pairs contain (a) ground-truth part correspondences, (b) reference rigid motion per part, (c) ground-truth flow field from one shape to the other, (d) point-wise correspondences of corresponding parts since we know the underlying rigid motion that maps the points of one part onto the other, and finally (e) a binary flag for each point indicating whether it has correspondences with any other point on the other shape or not.

5.2 Multi-task objective function

Given a training set $\mathcal{T}$ of shape pairs, the network is trained according to a multi-task objective such that the predictions of each module in our architecture agrees as much as possible with the ground-truth annotations. Specifically, we minimize a loss L that includes terms related to the correct prediction of point correspondences along with existence of those (L_{corr}), flow field (L_{flow}),

rigid motions (L_{motion}), and part segmentations (L_{seg}):

$$L = \sum_{\{P,Q\} \in \mathcal{T}} \Big(L_{corr}(P,Q) + L_{flow}(P,Q) + \tag{1}$$

$$L_{motion}(P,Q) + L_{seg}(P,Q)\Big) \tag{2}$$

We discuss the above loss terms in the following paragraphs.

Correspondence loss. Given a set $\mathcal{M}$ of ground-truth pairs of corresponding points across a training shape pair $\{P,Q\}$, and a set $\mathcal{N}$ of points on shape P that do not match any point on shape Q, the correspondence loss penalizes outputs of the correspondence module that are incompatible with the above sets. Specifically, the correspondence loss is expressed as a weighted sum of two losses. The first loss L_a penalizes low probabilities for matching point pairs deemed as corresponding in the set $\mathcal{M}$. The second loss L_b penalizes low probabilities for matching a point that does not belong to the set $\mathcal{N}$ and similarly low probabilities for not matching a point that belongs to the set $\mathcal{N}$. Since the matching of point pairs in our correspondence module is posed as a multi-class classification problem, L_a is formulated as a softmax classification loss. Since the decision of whether a point on shape P has a correspondence is a binary classification problem, L_b is expressed through binary cross-entropy. Specifically, the correspondence loss is set as $L_{corr} = \lambda_a L_a + \lambda_b L_b$, where:

$$L_a = -\sum_{\{p,q\} \in \mathcal{M}} \log(softmax(c_{p,q}))$$

$$L_b = -\sum_{p \notin \mathcal{N}} \log(sigmoid(c_p)) - \sum_{p \in \mathcal{N}} \log(1 - sigmoid(c_p))$$

and λ_a, λ_b are weights both set to 1 through hold-out validation.

Flow loss. Given the ground truth flow field $\mathbf{f}^{(gt)}(P,Q)$ and the predicted field $\mathbf{f}(P,Q)$ from our flow module for a training shape pair $\{P,Q\}$, the flow loss directly penalizes their difference using the L^2 norm: $L_{flow}(P,Q) = \lambda_c ||\mathbf{f}^{(gt)} - \mathbf{f}||^2$, where λ_c is a weight for this loss term again set to 1 through hold-out validation.

Rigid motion loss. The rigid motion loss penalizes discrepancies between the ground-truth rotations and translations assigned to points of rigid parts of the training shapes and the hypothesized ones. We found that using a loss operating directly on the elements of the rotational and translational component of the rigid motion (e.g., Frobenius or L^2 norm difference) resulted in poor performance in terms of flow and segmentation prediction. The reason was that it was hard to balance the weights between the two components i.e., often either the rotational or translational component dominated at the expense of the other.

We instead found that significantly better performance was achieved when using a loss that compared the positions of points belonging to the same underlying rigid part of shape P, after applying the hypothesized rigid motion, with the positions of corresponding points on shape Q. Specifically, for each pair of ground-truth corresponding points in the set $\mathcal{M}$ for the training shape pair $\{P,Q\}$, we find all other pairs of corresponding points belonging to the same underlying rigid part, and measure the L^2 norm of the difference in

the positions of points on shape Q and position of the corresponding points on shape P after applying each hypothesized rigid motion:

$$L_{motion} = \lambda_d \sum_{\substack{\{p,q\} \in \mathcal{M}}} \sum_{\substack{\{p',q'\} \in \mathcal{M} \\ part(p')=part(p)}} ||q' - (\mathbf{R}_p p' + \mathbf{t}_p)||^2$$

where λ_d is a weight for this loss term set to 1 through hold-out validation, and $part(p), part(p')$ return the part that the points p, p' belong to respectively i.e., in the above summation, we consider pairs of $\{p, p'\}$ belonging to the same rigidly moving part.

Segmentation loss. We designed the segmentation loss such that supervisory signal is received on (a) the support prediction stage of the segmentation module, which outputs the support matrix $\mathcal{S}$ storing the probability of each point on a shape to follow the rigid motion hypotheses generated from each other point on the shape, (b) the recurrent net stage of the segmentation module, which outputs the soft segmentation assignment variables $\mathbf{y}_t$ and the continuation score r_t at each step t. We found that incorporating supervision for both these stages of the segmentation module offered the best performance. Specifically, the segmentation loss is a weighted sum of three terms: $L_{seg} = \lambda_e L_e + \lambda_f L_f + \lambda_g L_g$, where L_e is a loss term that evaluates the predicted support matrix, L_f evaluates the assignments of the recurrent net segmentation variables, L_g evaluates the recurrent net continuation scores, and $\lambda_e, \lambda_f, \lambda_g$ are loss term weights set to 0.5, 1, 1 respectively through hold-out validation.

The prediction of the support matrix $\mathcal{S}$ can be treated as a binary classification problem: a point p on shape P either follows or not the rigid motion hypothesis generated from another point p' on the same shape. Since we know whether $\{p', p\}$ fall onto the same rigid part or not based on the ground-truth annotations, we can evaluate the predicted support matrix through binary cross-entropy:

$$L_s = - \sum_{\substack{p,p': \\ part(p)=part(p')}} \log(S_{p,p'}) - \sum_{\substack{p,p': \\ part(p)\neq part(p')}} \log(1 - S_{p,p'})$$

where $part(p), part(p')$ return the rigid part the points p, p' belong to respectively.

Evaluating the output segmentation assignment variables of the recurrent net is more challenging because the order of the output segments does not necessarily match with the order of ground-truth segments specified for the training shapes. To handle the uncertainty in the order of the output segments, we use a loss function inspired by Romera and Torr [2016]. Assuming that the shape P in an input training pair has K annotated rigid segments, the segments can be represented through binary indicator vectors $\{\hat{\mathbf{y}}_k\}_{k=1...K}$, where each vector $\hat{\mathbf{y}}_k$ stores a binary value per point indicating whether it belongs to the segment with index k or not. The output of our recurrent net is a soft indicator vector $\{\mathbf{y}_t\}_{t=1...T}$, which contains the probability of a point belonging to the output segment at step t (where T is the number of the executed RNN steps). During training, we set the maximum number of RNN execution steps as $T = K + 2$ i.e., we predict two extra segments compared to ground-truth (we experimented with more steps, but did not have any noticeable effect in performance). We use the Hungarian algorithm [Kuhn 1955] to find a bipartite matching between

the ground-truth segment indicator vectors $\{\hat{\mathbf{y}}_k\}$ and predictions $\{\mathbf{y}_t\}$, then employ a relaxed version of the Intersection over Union (IoU) score [Krähenbühl and Koltun 2013] to evaluate the matched pairs of segments. The relaxed IoU between a predicted segment output $\mathbf{y}_t$ and a matched ground-truth segment $\hat{\mathbf{y}}_{k(t)}$ is defined as: $IoU(\mathbf{y}_t, \hat{\mathbf{y}}_{k(t)}) = \frac{<\mathbf{y}_t, \hat{\mathbf{y}}_{k(t)}>}{||\mathbf{y}_t||_1 + ||\hat{\mathbf{y}}_{k(t)}||_1 - <\mathbf{y}_t, \hat{\mathbf{y}}_{k(t)}>}$. Then the loss term L_f is expressed as the negative of a sum of IoUs over K matched segment pairs: $L_f = - \sum_{t=1...K} IoU(\mathbf{y}_t, \hat{\mathbf{y}}_{k(t)})$

Finally, the loss term L_g evaluates the recurrent net continuation scores, penalizing low continuation probability for the first $K - 1$ RNN execution steps, and high continuation probability after performing K steps. The decision to continue producing segments can be considered as a binary classification problem, thus the loss term on continuation can be expressed through binary cross-entropy: $L_g = \sum_{t<K} \log(r_t) + \sum_{t \geq K} \log(1 - r_t)$

5.3 Training Procedure and Implementation Details

We minimize our loss function using the Adam variant of batch gradient descent with a learning rate of 0.0001. To better balance variant losses, we adopt a stage-wise training strategy. We first optimize the correspondence proposal and flow module by minimizing the sum of L_{corrs} and L_{flow} for 100 epochs. Then we feed the ground truth flow to the segmentation module and optimize the hypothesis generation and verification submodule by minimizing L_{motion} and L_e for 100 epochs. Finally we include all the loss terms in the optimization and further train the whole pipeline in an end-to-end manner for another 100 epochs with a learning rate decay set to a factor of 0.001.

Hyper-parameter selection. Our architecture makes extensive use of PointNet and PointNet++ networks. Their number and type of layers were selected using the default architecture blocks provided in [Qi et al. 2017a,b]. Regarding the layer hyper-parameters of our architecture (grouping radius in PointNet++, dimensionality of intermediate feature representations, memory size in our RPEN), we performed a grid search over different values in a hold-out validation set with ground-truth shape segmentations, and selected the ones that offered the best performance in terms of IoU.

Implementation. Our method is implemented in Tensorflow. Our source code, datasets and trained models are available in our project page: https://github.com/ericyi/articulated-part-induction

6 EXPERIMENTS

In this section, we evaluate the quality of our approach and compare it to state-of-the-art methods. We conducted experiments on both synthetic and real datasets and demonstrate the performance of the whole framework as well as each module component.

6.1 Test Dataset

Synthetic Dataset. We leveraged the annotated dataset in [Hu et al. 2017], which contains articulated 3D CAD models with ground truth part segmentations and motion annotations, and generated three synthetic datasets: 1) Point cloud pairs originating from two different articulations of the same 3D CAD model (**SF2F**); 2) Point

cloud pairs consisting of one full shape and one partial scan from the same 3D model but with different articulations(**SF2P**); 3) RGBD pairs consisting of two partial views of the same object with different articulations(**SP2P**). For each CAD model, we randomly transformed its moving parts 10 times following the part segmentation and motion annotations in [Hu et al. 2017]. Then we randomly create 5 shape pairs by randomly selecting two different articulations of the same model out of its 10 generated configurations. We also added random perturbations to the global poses of the shapes. Then we uniformly sample points from the full shapes or conduct virtual scans from random viewpoints to generate partial point clouds using rendering tools developed from [Hassner et al. 2015]. We normalize the point sets so that their bounding boxes are centered at $(0, 0, 0)$ and have diagonals with length 1. We removed categories whose part motions cannot be distinguished from sampled point clouds. This could be due to either tiny parts (e.g., the button on a remote control, which might not be sampled in the point cloud, or sampled by less than 10 points), or due to rotational symmetries of a part (a rotating bottle cap cannot be distinguished in the sampled point clouds, because its rotations yield almost the same points). In total, for each synthetic dataset, we constructed 875 pairs covering 175 shapes from 23 categories. We refer readers to the supplementary material for a visualization of this test dataset, included a list of object categories in it. Note that our synthetic dataset has only one category overlapping with our training dataset (laptop). The rest of the categories are particularly useful for testing the cross-category generalization ability of all the different techniques included in our evaluation.

Real Dataset. We also collected real data for evaluation under two different settings. 1) Real scan pairs of the same object but with different articulations(**RP2P**) 2) 3D pairs consisting of a full CAD model downloaded online, and a partial scan captured from the real world. In this case, the pair represents geometrically different objects under different articulations (**RF2P**). For **RP2P**, we collected 231 scan pairs from 10 categories. We manually segmented the scans into their moving parts. For **RF2P** we collected 150 pairs from 10 categories and also manually segmented the CAD models into moving parts. The testing categories in all our test datasets are different from the ones used for training. We also note that in this setting, there is a potential domain shift in our testing since our training data do not necessarily include realistic motion, as in the case of real data.

6.2 Deformation Flow

We first evaluate our predicted deformation flow on two synthetic datasets **SF2F** and **SF2P** (where ground-truth flow is available).

Methods. We test our method against various alternatives, including both learning and non-learning approaches. Specifically, we compare with three learning approaches including a 3D scene flow estimation approach (3DFlow) [Liu et al. 2018], feature-based matching with learned descriptors from a volumetric CNN (3DMatch) [Zeng et al. 2017] and from a multi-view CNN (LMVCNN) [Huang et al. 2017]. For all three approaches, we train the corresponding networks from scratch using the same training data as ours. For

Table 1. EPE evaluation on dataset SF2F and SF2P for all competing methods. EPE measures the Euclidean distance between predicted flow and the ground-truth flow. Smaller EPE means more accurate flow prediction.

	Ours	3DFlow	ED	NRR	3DMatch	LMVCNN
SF2F	**0.0210**	0.0536	0.0481	0.0394	0.0715	0.0582
SF2P	**0.0422**	0.0892	0.0805	0.0556	0.138	0.093

3DMatch and LMVCNN, we extracted point-wise descriptors with the learned networks and estimated the flow by finding nearest neighboring points in descriptor space. In addition, we also compare with two non-learning approaches [Sumner et al. 2007] and [Huang et al. 2008], which leveraged non-rigid deformation for deformation flow estimation. We will refer to [Sumner et al. 2007] as ED and refer to [Huang et al. 2008] as NRR in our baseline comparison.

Metrics. We use two popular measures to evaluate the flow predicted by all methods. First, we use *End-Point-Error (EPE)* as defined in [Yan and Xiang 2016], which has been widely used for optical flow and scene flow evaluation. To be specific, given a ground truth flow field $\mathbf{f}^{(gt)}$ and a predicted flow field $\mathbf{f}$, the EPE is $E_{EPE} = \frac{1}{n} \sum_{i=1}^{n} \sqrt{(\mathbf{f}_i - \mathbf{f}_i^{gt})^2}$. We also use the *Percentage of Correct Correspondences (PCC)* curve as described in [Kim et al. 2011], where the percentage of correspondences that are consistent with ground truth under different prescribed distances is shown.

Results. We show the comparison of different approaches in Table 1 in the case of the EPE metric. Our approach outperforms all the baseline methods by a large margin. **SF2F** contains pairs of full shapes from the same objects with different articulations while pairs in **SF2P** include one full shape and one partial scan. Due to the large missing data, the flow predictions of all approaches are less accurate on **SF2P** compared with those on **SF2F**, but still, our approach demonstrates more robustness and achieves the best performance. We visualize the deformation flow predicted by various approaches
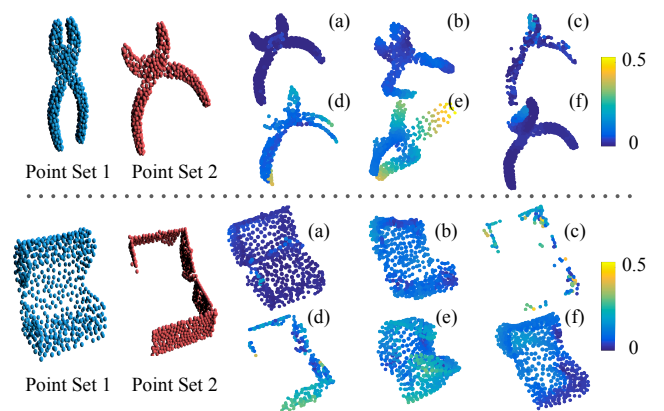

Fig. 6. Deformation flow visualization. We estimate a dense flow from the point set 1 to point set 2 and apply the flow to deform the point set 1. Deformation results are shown from (a) to (f): (a) Ours, (b) 3DFlow, (c) 3D match, (d) LMVCNN, (e) ED, (f) NRR. Colors on the deformed point set denote the flow error of each point, whose range is shown on the color bar at the right side of each row.
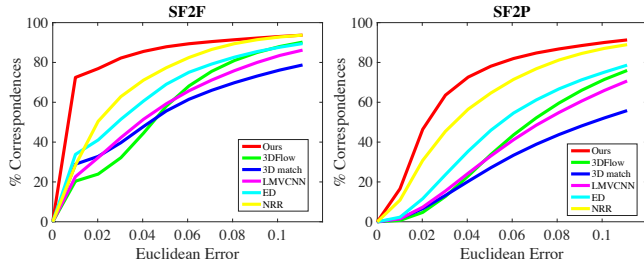
Fig. 7. Percent of correspondences which have Euclidean error smaller than a threshold. The x-axis corresponds to different thresholds.

in Figure 6. Similar to us, 3DFlow leverages PointNet in their scene flow estimation scheme. Their approach tends to have degraded performance while dealing with large motion, especially large rotations, which is a common scenario in part motions for man-made objects. They do not leverage motion structure between two frames, such as piecewise rigidity. In addition, they do not model missing data from one point cloud to another, resulting in a much worse performance on **SF2P** compared to our approach. 3DMatch and LMVCNN are not very suitable for dense flow estimation as they suffer from ambiguities due to symmetry, cannot handle missing data well and lack smoothness in their flow prediction. Deformation-based approaches (ED, NRR) ignore the piece-wise rigidity property of objects and also result in artifacts when the articulation difference in the input pair is large, where a good set of initial correspondences becomes hard to estimate. We also note that given large portions of missing data, deformation-based approaches tend to generate largely unrealistic deformations around missing shape regions. Compared to all the above methods, our approach is able to parse the piecewise rigid motion of the objects much more reliably. Due to the explicit modeling of missing data, our approach avoids deforming the portion of point set 1 which has no correspondences in point set 2, and instead tends to generate the flow field via considering the other points on the same rigid part.

Figure 7 shows the PCC curves of different approaches. We again observe that our method outputs more accurate flow estimation both in the local and global sense compared to other approaches.

6.3 Segmentation

We then evaluate our segmentation performance both on synthetic datasets **SF2F** and **SF2P** and real datasets **RF2F** and **RF2P**.

Methods. To ensure a fair comparison, we compare our approach with several other co-segmentation/motion segmentation baselines using our deformation flow prediction. Since our segmentation module can be regarded as a neural network-based version of sequential RANSAC with learnable hypothesis generation, verification and selection steps, we implemented a sequential RANSAC baseline, where we repeat the following steps until a stopping criterion is met: (a) figure out the largest rigid motion mode as well as its support in the current point set from the deformation flow; (b) remove all the supporting points from the discovered mode. We set the stopping criterion to be either a maximum number of iterations has been met (10 in our case), or the remaining points are less than 5% of the initial point set. Once we discovered the dominant motion modes as well

Table 2. RI and IoU evalution on both synthetic and real datasets for all competing methods. Numbers in each cell represent RI/IoU. Both RI and IoU measure the segmentation consistency. Higher RI and higher IoU mean better segmentation prediction.

	SeqRANSAC	SC	JLC	NRR	Ours
SF2F	60.2/55.8	80.6/69.4	74.7/67.3	74.1/57.3	**83.8/77.3**
SF2P	48.2/37.6	67.0/55.6	66.2/58.2	72.7/53.9	**75.6/66.6**
RS2S	56.7/43.0	79.1/67.1	80.4/73.0	78.4/65.5	**88.3/83.5**
RF2S	58.7/44.2	71.9/53.6	72.7/58.4	72.8/54.7	**87.6/81.8**

as their associated supporting points, we assign labels to the rest of the points according to their closest motion modes. In addition, we also implemented several other baselines for comparison, including a spectral clustering approach (SC) [Tzionas and Gall 2016b] and a JLinkage clustering approach (JLC)[Yuan et al. 2016b]. The spectral clustering approach leverages the fact that two points belonging to the same rigid part should maintain their Euclidean distance as well as the angular distance between their normals before and after deformation. The JLinkage clustering approach samples a large number of motion hypotheses first and associates each data point with a hypotheses set. The closeness among data points can be defined based on the hypotheses sets and an iterative merging step is adopted to generate the final segmentation. We also compare with the simultaneous flow estimation and segmentation approach (NRR) by Huang et al. [2008]. We found their segmentation results performs better using their own flow prediction (yet, the segmentation results are much worse than ours in any case). Therefore we report their segmentation results based on their own flow prediction.

Metrics. We use two evaluation metrics: Rand Index (RI) used in [Chen et al. 2009] and average per-part intersection over union (IoU) used in [Yi et al. 2016]. Rand index is a similarity measurement between two data clusterings. We use the implementation provided by [Chen et al. 2009]. Average per-part IoU is a more sensitive metric to small parts. To compute per-part IoU between a set of ground truth segments $\{\hat{y}_k\}_{k=1\ldots K}$ and the predicted segments $\{y_t\}_{t=1\ldots T}$, we first use the Hungarian algorithm to find a bipartite matching between the ground-truth segment indicator vectors $\{\hat{y}_k\}$ and the predicted segment indicator vectors $\{y_t\}$ so that $y_{t(k)}$ denotes the match of $\hat{y}_k$. We then compute the per-part IoU as: $IoU(y_t(k), \hat{y}_k) = \frac{<y_t(k),\hat{y}_k>}{||y_t(k)||_1+||\hat{y}_k||_1-<y_t(k),\hat{y}_k>}$. If a part $\hat{y}_k$ in the ground truth set has no match in the prediction set (the number of predicted parts is less than the ground truth), we count its $IoU(y_t(k), \hat{y}_k)$ as 0. The final average per-part IoU is simply an average of the above IoU for all parts and all shapes.

Results. We compare our approach with various baseline methods on four different datasets including two synthetic ones (**SF2F**,**SF2P**) and two real ones (**RP2P**,**RF2P**). The results are reported in Table 2 using the rand index and average per-part IoU as the evaluation metrics. Our approach outperforms all the baseline methods by a large margin, both on synthetic and real datasets, especially when using the per-part IoU metric, which indicates our approach has a better ability to capture the correct number of parts and is more capable to detect small parts.
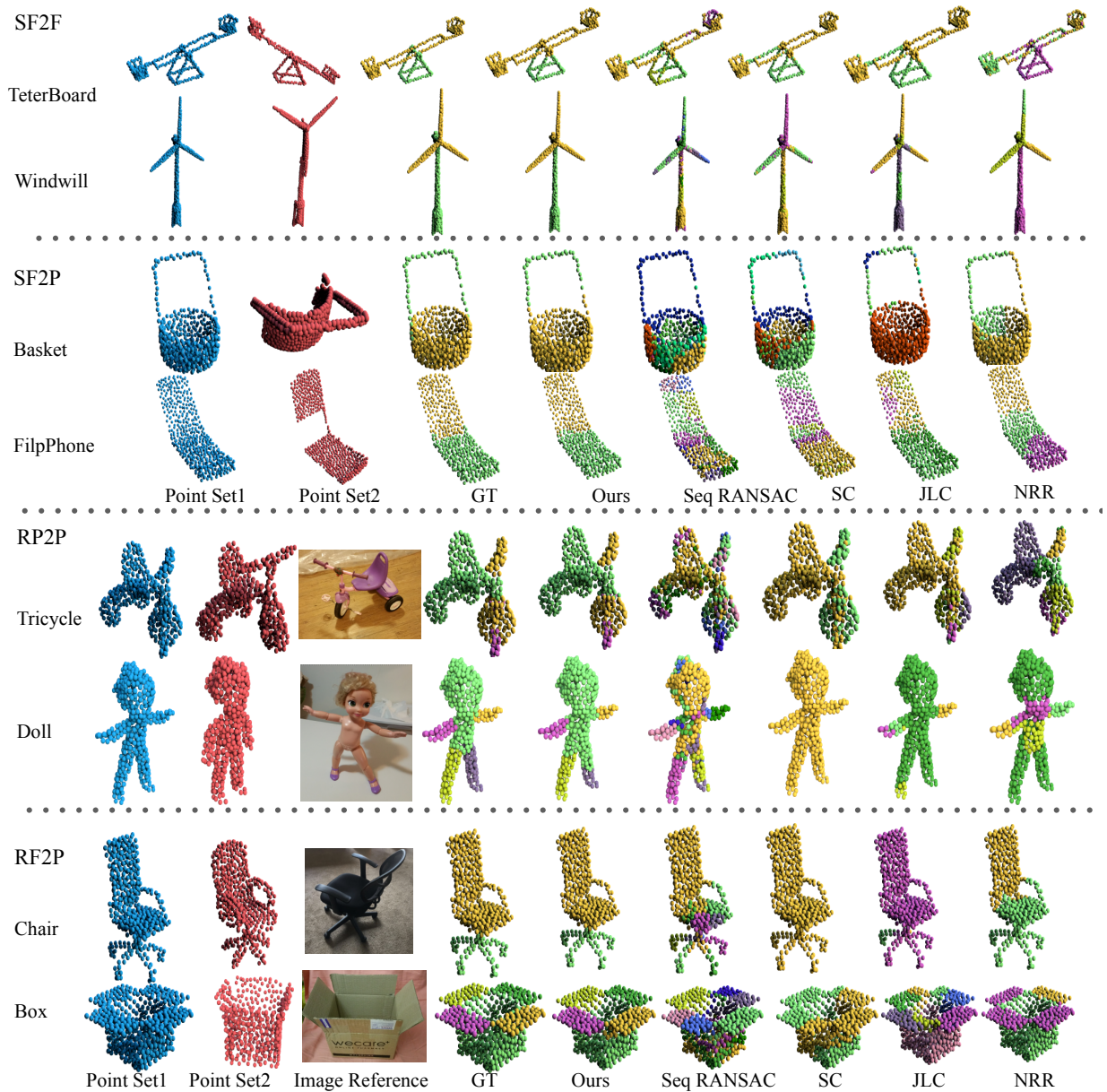
Fig. 8. Segmentation visualization for all competing methods on synthetic and real data.

To better understand the performance gain of our approach, we visualize the prediction results of different methods on the two synthetic datasets in Figure 8. Since it is hard to acquire perfect flow field prediction, the segmentation approach needs to be robust to input noise and imperfect flow while generating the predictions. Sequential RANSAC can handle input noise to some degree by properly setting a noise threshold while generating inlier supports. However different input shape pairs seem to require different thresholds – a single threshold fails to provide satisfactory results in all cases. Our implementation of sequential RANSAC uses a cross-validated threshold and it tends to generate small discontinuous pieces in the shown examples. The SC and JLC approaches predict segmentations

mainly based on the motion cues with little consideration of the underlying geometry, therefore they can over-group parts, such as the platform with the fulcrum. Our segmentation module instead considers both the motion cue and the underlying geometry, thus it is able to generate the correct segmentation even if the input flow is noisy and imperfect. Moreover, the SC and JLC approaches both require hand-tuned thresholds, which are quite sensitive to different types of shapes. Again we set the thresholds via cross-validation and find that they usually fail to discover rigid parts. NRR often leads to under-segmentations such as the USB example in Figure 8 while in other cases, it results in over-segmentations such as the flip phone example.

The experiments on the two real datasets demonstrate that our approach, trained on synthetic data, is able to generalize to real scans. We also visualize prediction results on **RP2P** and **RF2P** in Figure 8. The **RP2P** dataset contains two scans of the same underlying object with different articulations. In the challenging tricycle example, our approach successfully segments out the two small pedals and groups them together through their motion pattern. None of the baseline methods are capable of achieving this. When the number of parts and DoFs increase, such as the articulated doll example, the baseline approaches cannot generate a proper number of parts in contrast to ours. The **RF2P** dataset contains pairs of shapes from different objects, which is very challenging since the flow field from the point set 1 to point set 2 contains both motion flow and geometric flow caused by their geometric difference. To downweigh the influence of geometric flows, we also tried to optimize our predicted deformation flow with an as-rigid-as-possible (ARAP) objective [Sorkine and Alexa 2007] to preserve the local geometry before passing it to various segmentation approaches. Baseline methods either under-segment or over-segment the point sets and the segmentation boundaries are also very noisy. Our segmentation module again demonstrates robustness and is more capable of predicting the number of segments properly and generating cleaner motion boundaries.

We also refer readers to our supplementary material for an ablation study evaluating different design choices of our architecture, and a discussion on the computational cost of our method.

Comparison with Shao et al. We compare here our method with the concurrent learning method by Shao et al. [2018]. Their method trains a joint flow estimation and segmentation network for motion-based object segmentation. They take an RGB-D pair as input, and they convert the depth image into a partial point cloud with known camera parameters. Then their network consumes the RGB information as well as the point clouds and generates a motion-based segmentation for the input frames. To compare with that approach, we use our synthetic dataset **SP2P**, where we can render CAD models with different articulations into RGBD pairs and apply both their
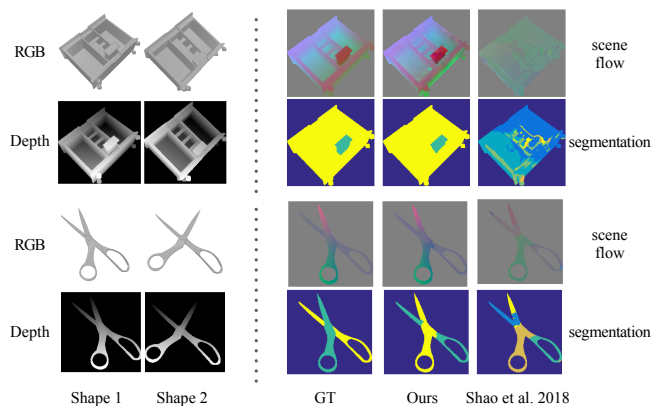

Fig. 9. Comparison with [Shao et al. 2018]. The input includes a pair of RGB and a pair of depth images representing the underlying shape pair. Both [Shao et al. 2018] and our method predict a scene flow plus a motion based segmentation, visualized here together with the ground truth (GT).

approach and ours for motion segmentation. Note that our approach only uses the point cloud information, while their approach exploits both the rendered images as well as the partial point clouds. We also rendered our training data into RGB-D pairs, and asked the authors of [Shao et al. 2018] to re-train their network on our training set so that they can handle the CAD rendered images. The quantitative comparison is shown in Table 3. Our flow field and segmentation estimation outperforms [Shao et al. 2018] by a large margin. We visualize the prediction results in Figure 9. [Shao et al. 2018] cannot reliably estimate flows for complex structures, in particular in texture-less settings, such as the drawer in a cabinet. This leads to largely inaccurate segmentation results. Our approach instead fully operates on 3D and is more effective to capture the object structure. In addition, [Shao et al. 2018] cannot group object parts whose centers are very close to each other and do not move much, e.g. the scissors example.

6.4 Applications
Our framework co-analyzes a pair of shapes, generating a dense flow field from one point set to another and also the motion-based part segmentation of the two shapes. These outputs essentially reveal the functional structure of the underlying dynamic objects and can benefit various applications we discuss below.

Shape Animation. The output of our framework can be directly used to animate shapes. Given a shape pair (P, Q), we co-segment them into rigid parts $\{P_t\}$ and also estimate the corresponding rigid motions $\{H_t\}$ We can then interpolate between the motion states of P and Q, generating in-between motion frames for P. To interpolate between two rigid motions H_i and H_j, we sample the geodesic paths connecting H_i and H_j following [Žefran and Kumar 1998]. Assuming 4x4 homogeneous matrices representation for H_i and H_j, an interpolation between H_i and H_j can be computed as $F(t; H_i, H_j) = \exp(t\log(H_jH_i^{-1}))H_i$, $t \in [0, 1]$. We fix the motion for a selected part and generate different motion states for the rest of the parts, and visualize our results in Figure 10. We are able to animate both revolute joints and prismatic joints. Such animation reveals the underlying functionality of the object and is useful for adding interactivity to that object in a virtual environment.

Part Induction from a Shape-Image Pair. Recently we have witnessed a lot of progress in single image-based 3D reconstruction, which opens up a new application of our framework, namely joint motion segmentation for a shape and image pair. Given an articulated 3D shape and a related 2D product image as input, we lift the 2D image onto 3D and then apply our framework to co-segment the lifted 3D shape and the input 3D shape. The segmentation information can be later propagated back from the lifted 3D shape back onto the 2D image, resulting in a motion segmentation for the image as well. This results in a co-segmentation for both the 3D shape

Table 3. Numerical evaluation compared to [Shao et al. 2018].

SP2P	EPE (Corrs)	RI (Seg)	IoU (Seg)
[Shao et al. 2018]	0.0862	0.686	0.563
Ours	**0.0369**	**0.833**	**0.756**

and the 2D image based on discovered motion cues. This setting is attractive since given an articulated 3D shape of a man-made object, we can easily find lots of images online describing related objects with different articulation states through product search. Being able to extract motion information for these shape image pairs benefit a detailed functional understanding of both domains. For the purpose of lifting the 2D product image onto 3D, we design another neural network and we refer to the *supplementary* for more details on this lifting network.

We visualize our shape-image pair co-analysis in Figure 11. In each row, we are given an input shape plus an input product image describing a related but different object. We first convert the 2D image into a 3D point set using the lifting network. The lifted 3D shapes are visualized in the 3rd column. We then apply our framework for the input pair, resulting in a deformed version of the input shape visualized in the 4th column. We note that articulation of the deformed input shape becomes more similar to the one of the lifted point set. We visualize the segmentation for the input shape in the 5th column and we propagate the segmentation of the lifted point set back to the 2D image (our lifting procedure maintains the correspondence between the image pixels and the lifted points), which is shown in the last column.

7 LIMITATIONS AND FUTURE WORK

We presented a neural network architecture that is able to discover parts of objects by analyzing the underlying articulation states and geometry of different shapes. Our network is able to generalize to novel objects and classes not observed during training.

There are several avenues for future research directions. First, our method uses 3D point cloud representations, which might undersample small parts, or cannot deal with rotationally symmetric parts, such as bottle caps. We also found that parts that slide towards the interior of a shape, such as sliding knives, are more challenging since point samples on folding shape layers are hard to differentiate . Increasing the point cloud resolution or introducing an attention mechanism that results in a dynamic adaptation of resolution could

Fig. 11. Our method can be used for joint shape-image analysis. Given a pair including a 3D shape and a 2D image, we can jointly align them and output the segmentation for both the shape and the image.

help to deal with these cases. Second, part induction between different shape instances with large topology variation remains a challenging problem. The point-wise deformation flow itself becomes harder to define in these cases. When it comes to large geometric differences, the segmentation module would need to be redesigned to deal with both local rigid and non-rigid deformation flow. Finally, our method currently infers parts and motions from pairs of shapes. In the future, it would be interesting to infer those from a single input, or analyze larger sets of objects to discover common articulation patterns within a shape family.

ACKNOWLEDGMENTS

REFERENCES

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *Proc. ECCV*.

D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. 2015. Learning Class-specific Descriptors for Deformable Shapes Using Localized Spectral Convolutional Networks. In *Proc. SGP*.

Yuri Boykov, Olga Veksler, and Ramin Zabih. 2001. Efficient Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 12 (2001), 1222–1239.

Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. 2017. DSAC-Differentiable RANSAC for camera localization. In *Proc. CVPR*.

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

Will Chang, Hao Li, Niloy Mitra, Mark Pauly, and Michael Wand. 2012. Dynamic Geometry Processing. In *Eurographics 2012 - Tutorials*.

Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. 2009. A benchmark for 3D mesh segmentation. 28, 3 (2009), 73.
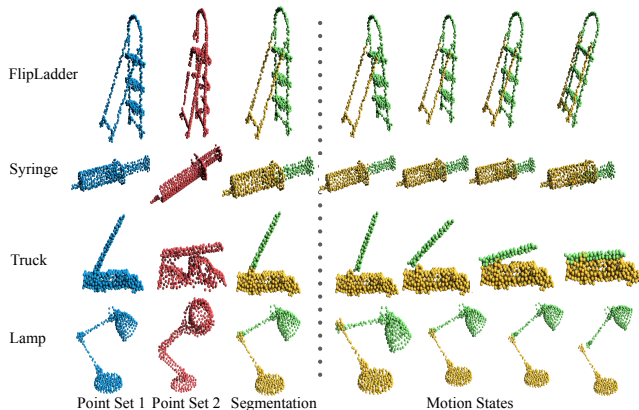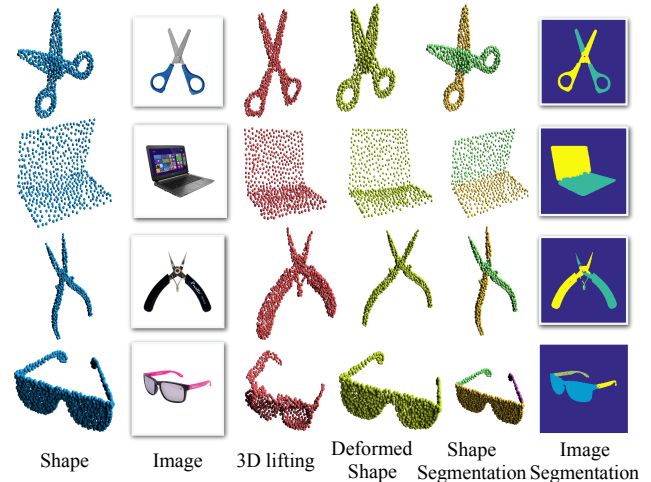

Fig. 10. Our method can be used for shape animation. Given the predicted segmentation of articulated parts (left), one can generate a sequence of animated shapes by interpolating the motion (right).

Vogel Christoph, Konrad Schindler, and Roth Stefan. 2015. 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. *International Journal of Computer Vision* 115, 1 (2015).

Martin A. Fischler and Robert C. Bolles. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* 24, 6 (1981).

Aleksey Golovinskiy and Thomas Funkhouser. 2009. Consistent Segmentation of 3D Models. *Computers & Graphics* 33, 3 (2009).

Vladislav Golyanik, Kihwan Kim, Robert Maier, Matthias Nießner, Didier Stricker, and Jan Kautz. 2017. Multiframe Scene Flow with Piecewise Rigid Motion. In *Proc. 3DV*.

Tal Hassner, Shai Harel, Eran Paz, and Roee Enbar. 2015. Effective face frontalization in unconstrained images. In *Proc. CVPR*.

Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. 2014. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3526–3533.

Ruizhen Hu, Lubin Fan, and Ligang Liu. 2012. Co-Segmentation of 3D Shapes via Subspace Clustering. *Computer Graphics Forum* 31, 5 (2012).

Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. 2017. Learning to predict part mobility from a single static snapshot. *ACM Transactions on Graphics* 36, 6 (2017), 227.

Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G. Kim, and Ersin Yumer. 2017. Learning Local Shape Descriptors from Part Correspondences with Multiview Convolutional Networks. *ACM Transactions on Graphics* 37, 1 (2017).

Qixing Huang, Fan Wang, and Leonidas Guibas. 2014. Functional Map Networks for Analyzing and Exploring Large Shape Collections. *ACM Transactions on Graphics* 33, 4 (2014).

Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J Guibas. 2008. Non-rigid registration under isometric deformations. 27, 5 (2008), 1449–1457.

M. Jaimez, M. Souiai, J. Stueckler, J. Gonzalez-Jimenez, and D. Cremers. 2015. Motion Cooperation: Smooth Piece-Wise Rigid Scene Flow from RGB-D Images. In *Proc. 3DV*.

Doug L. James and Christopher D. Twigg. 2005. Skinning Mesh Animations. *ACM Transactions on Graphics* 24, 3 (2005).

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 2017. 3D Shape Segmentation with Projective Convolutional Networks. In *Proc. CVPR*.

Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. 2013. Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics* 32, 4 (2013), 70:1–70:12.

Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. 2011. Blended intrinsic maps. In *ACM Transactions on Graphics*, Vol. 30. 79.

Youngji Kim, Hwasup Lim, Sang Chul Ahn, and Ayoung Kim. 2016. Simultaneous segmentation, estimation and analysis of articulated motion from dense point cloud sequence. In *Proc. IROS*.

Roman Klokov and Victor Lempitsky. 2017. Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models. In *Proc. ICCV*.

Philipp Krähenbühl and Vladlen Koltun. 2013. Parameter learning and convergent inference for dense random fields. In *Proc. ICML*.

Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)* 2, 1-2 (1955), 83–97.

Hao Li, Guowei Wan, Honghua Li, Andrei Sharf, Kai Xu, and Baoquan Chen. 2016. Mobility Fitting Using 4D RANSAC. *Computer Graphics Forum* 35, 5 (2016).

Xingyu Liu, Charles R Qi, and Leonidas J Guibas. 2018. Learning Scene Flow in 3D Point Clouds. *arXiv preprint* (2018).

Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Transactions on Graphics* 36, 4 (2017).

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on Riemannian manifolds. In *Proc. ICCV Workshops*.

Daniel Maturana and Sebastian Scherer. 2015. 3D Convolutional Neural Networks for Landing Zone Detection from LiDAR. In *Proc. ICRA*.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics* 36, 4 (2017), 14.

Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. 2006. Partial and Approximate Symmetry Detection for 3D Geometry. *ACM Transactions on Graphics* 25, 3 (2006).

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proc. CVPR*.

Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked Hourglass Networks for Human Pose Estimation. In *Proc. ECCV*.

Stephen Palmer. 1977. Hierarchical structure in perceptual representation. *Cognitive Psychology* 9, 4 (1977), 441–474.

Sudeep Pillai, Matthew R. Walter, and Seth J. Teller. 2014. Learning Articulated Motions From Visual Demonstration. In *Robotics: Science and Systems*.

Sören Pirk, Vojtech Krs, Kaimo Hu, Suren Deepak Rajasekaran, Hao Kang, Yusuke Yoshiyasu, Bedrich Benes, and Leonidas J. Guibas. 2017. Understanding and Exploiting Object Interaction Landscapes. *ACM Transactions on Graphics* 36, 3 (2017).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. CVPR*.

Charles R. Qi, Li Yi, Hao Su, and Leonidas Guibas. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proc. NIPS*.

Julian Quiroga, Thomas Brox, Frédéric Devernay, and James L. Crowley. 2014. Dense Semi-Rigid Scene Flow Estimation from RGBD images. In *Proc. ECCV*.

Gernot Riegler, Ali Osman Ulusoys, and Andreas Geiger. 2017. Octnet: Learning deep 3D representations at high resolutions. In *Proc. CVPR*.

Bernardino Romera-Paredes and Philip Hilaire Sean Torr. 2016. Recurrent instance segmentation. In *Proc. ECCV*.

Lin Shao, Parth Shah, Vikranth Dwaracherla, and Jeannette Bohg. 2018. Motion-based Object Segmentation based on Dense RGB-D Scene Flow. *arXiv preprint arXiv:1804.05195* (2018).

Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. 2013. Efficient Human Pose Estimation from Single Depth Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013).

Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. 2011. Unsupervised Co-Segmentation of a Set of Shapes via Descriptor-Space Spectral Clustering. *ACM Transactions on Graphics* 30, 6 (2011).

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible Surface Modeling. In *Proc. SGP*.

Jörg Stückler and Sven Behnke. 2015. Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video. *International Journal of Computer Vision* 113, 3 (2015).

Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. 2018. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. CVPR*.

Robert W. Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded Deformation for Shape Manipulation. *ACM Transactions on Graphics* 26, 3 (July 2007).

Denis Tomè, Chris Russell, and Lourdes Agapito. 2017. Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image. In *Proc. CVPR*.

Alexander Toshev and Christian Szegedy. 2014. DeepPose: Human Pose Estimation via Deep Neural Networks. In *Proc. CVPR*.

Dimitrios Tzionas and Juergen Gall. 2016a. Reconstructing Articulated Rigged Models from RGB-D Videos. In *Proc. ECCV*.

Dimitrios Tzionas and Juergen Gall. 2016b. Reconstructing Articulated Rigged Models from RGB-D Videos. In *Proc. ECCV*.

Oliver van Kaick, Kai Xu, Hao Zhang, Yanzhen Wang, Shuyang Sun, Ariel Shamir, and Daniel Cohen-Or. 2013. Co-hierarchical Analysis of Shape Structures. *ACM Transactions on Graphics*. 32, 4 (2013).

Christoph Vogel, Stefan Roth, and Konrad Schindler. 2014. View-consistent 3D scene flow estimation over multiple frames. In *European Conference on Computer Vision*. Springer, 263–278.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Transactions on Graphics* 36, 4 (2017).

Kai Xu, Vladimir G. Kim, Qixing Huang, Niloy Mitra, and Evangelos Kalogerakis. 2016. Data-driven Shape Analysis and Processing. In *SIGGRAPH ASIA 2016 Courses*.

Zike Yan and Xuezhi Xiang. 2016. Scene Flow Estimation: A Survey. *arXiv preprint arXiv:1612.02590* (2016).

Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, ARCewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. 2016. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics* 35, 6 (2016), 210.

Li Yi, Hao Su, Xingwen Guo, and Leonidas Guibas. 2017. SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proc. CVPR*.

Qing Yuan, Guiqing Li, Kai Xu, Xudong Chen, and Hui Huang. 2016a. Space-Time Co-Segmentation of Articulated Point Cloud Sequences. *Computer Graphics Forum* 35, 2 (2016).

Qing Yuan, Guiqing Li, Kai Xu, Xudong Chen, and Hui Huang. 2016b. Space-Time Co-Segmentation of Articulated Point Cloud Sequences. 35, 2 (2016), 419–429.

Miloš Žefran and Vijay Kumar. 1998. Interpolation schemes for rigid body motions. *Computer-Aided Design* 30, 3 (1998), 179–189.

Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 2017. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *Proc. CVPR*.