Shape2Motion: Joint Analysis of Motion Parts and Attributes from 3D Shapes

Xiaogang Wang[1] Bin Zhou[1] Yahao Shi[1] Xiaowu Chen[1] Qinping Zhao[1] Kai Xu[2*]
[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
[2]National University of Defense Technology

Abstract

For the task of mobility analysis of 3D shapes, we propose joint analysis for simultaneous motion part segmentation and motion attribute estimation, taking a single 3D model as input. The problem is significantly different from those tackled in the existing works which assume the availability of either a pre-existing shape segmentation or multiple 3D models in different motion states. To that end, we develop Shape2Motion which takes a single 3D point cloud as input, and jointly computes a mobility-oriented segmentation and the associated motion attributes. Shape2Motion is comprised of two deep neural networks designed for mobility proposal generation and mobility optimization, respectively. The key contribution of these networks is the novel motion-driven features and losses used in both motion part segmentation and motion attribute estimation. This is based on the observation that the movement of a functional part preserves the shape structure. We evaluate Shape2Motion with a newly proposed benchmark for mobility analysis of 3D shapes. Results demonstrate that our method achieves the state-of-the-art performance both in terms of motion part segmentation and motion attribute estimation.

1. Introduction

The analysis of part mobilities is a key step towards function analysis of 3D shapes [7], finding numerous potential applications in robot-environment interaction [17, 18]. In this work, we approach this problem from a data-driven perspective: Given a 3D shape as static observation, learn to simultaneously segment the shape into motion parts and estimate associated motion attributes (type and parameters).

In most existing approaches, mobility analysis is conducted on the pre-segmented parts of a 3D model [6]. However, the pre-existing segmentation of a 3D model does not necessarily conform to its mobilities. For example, in many car models, the door is not a separate component (Figure 1(a)). This greatly limits the utility of those methods on existing shape repositories where mobility-oriented segmentation is often unavailable. Another type of method extracts mobility through comparing multiple different motion states of a single object, e.g., a scissor with different opening angles [32]. Such requirement is, unfortunately, hard to meet for most testing shapes. Moreover, not all mobility can be easily spot by motion state comparison. For instance, it would be very hard to capture a rotating sphere without very accurate slippage analysis [4]. Last but not least, existing methods can only output a single mobility for each motion part, while in reality one part may possess multiple motions (e.g., the seat of a swivel chair in Figure 1(c)).

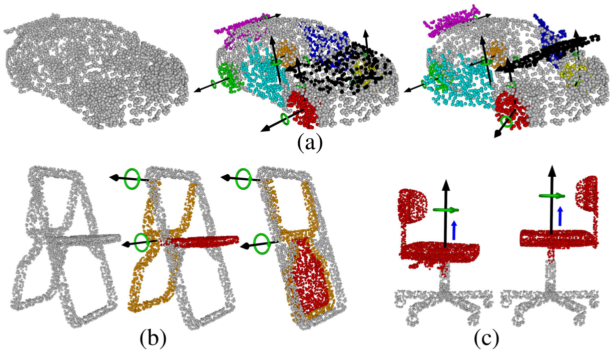

Figure 1: Three examples of simultaneous motion part segmentation and motion attribute inference. In each example, the left is the input point cloud; the middle demonstrates the analyzed result (segmented motion parts are shaded in distinct colors and their associated motions depicted with arrows); the right shows the motion parts are moved according to the analyzed motions.

We propose Shape2Motion, a method that consumes a single 3D shape in point cloud as input, and jointly computes a mobility-oriented segmentation and estimates the corresponding motion attributes. Shape2Motion adopts a propose-and-optimize strategy, in a similar spirit to the proposal-based object detection from images [21]. It consists of two carefully designed deep neural networks, i.e., a mobility proposal network (MPN) followed by a mobility optimization network (MON). MPN generates a collection of mobility proposals and selects a few high-quality ones.

[*]Corresponding author: kevin.kai.xu@gmail.com

Each mobility proposal is comprised of a motion part and its associated motion attributes; the latter refers to motion type (*translation*, *rotation* and *translation+rotation*) and motion parameters (*translation direction* and *rotation axis*). MON optimizes the proposed mobilities through jointly optimizing the motion part segmentation and motion attributes. The optimized mobilities are then merged, yielding the final set of mobilities. Figure 2 shows an overview of our method.

Our key insight in designing these networks is to fully exploit the coupling between a motion part and its mobility function: The movement of a functional part does not break the shape structure. See the examples in Figure 1: the opening or closing of a car door keeps it hinged on the car frame; the folding or unfolding of a folding chair preserves the attachment relations between its constituent parts. Therefore, we propose a motion-driven approach to encode mobilities and to measure the correctness (or loss) of motion part segmentation and motion attribute inference. Given a mobility proposal, we first move the part according to the corresponding motion and then inspect how much the movement preserves the shape structure. Given the ground-truth part mobility, we can measure the pose deviation of the moved part from that of its ground-truth counterpart under ground-truth motion. Motion-driven pose loss amplifies the direct loss of segmentation and motion parameters, thus greatly improves the training convergence.

We perform extensive evaluations of Shape2Motion over a newly proposed benchmark for mobility analysis of 3D shapes. Results demonstrate that our method achieves the state-of-the-art performance both in terms of motion part segmentation and motion attribute estimation, and show the advantages of our design choices over several baselines. Our work makes the following contributions:

- We propose the problem of joint analysis for motion part segmentation and motion attribute prediction from a single 3D shape.
- We design the first deep learning architecture to approach the above problem with two carefully designed networks, responsible for mobility proposing and mobility optimization, respectively.
- We contribute the first benchmark of 3D shape mobility analysis, encompassing both motion part segmentation and motion attribute estimation.

2. Related Work

Mobility analysis of 3D shapes. Dynamic functionalities of articulated 3D objects can be characterized by the motion of one or more of their constituent parts, which is commonly referred to as part mobility [7]. There have been a few works on discovering part mobility from an input object [30, 16, 23], or from a sequence of RGBD scans of the dynamic motion of an articulated model [15]. Hu et al. [6] propose a method for inferring part mobilities of a static 3D object by learning from motion sequences of parts from different classes of objects. Common to these existing approaches is the assumption of a pre-existing part segmentation, which, however, does not necessarily conform to part mobilities. Our method learns to simultaneously segment a 3D shape into motion parts and infer their motion attributes. Moreover, our method does not need a motion sequence of either object or parts, during either training or testing.

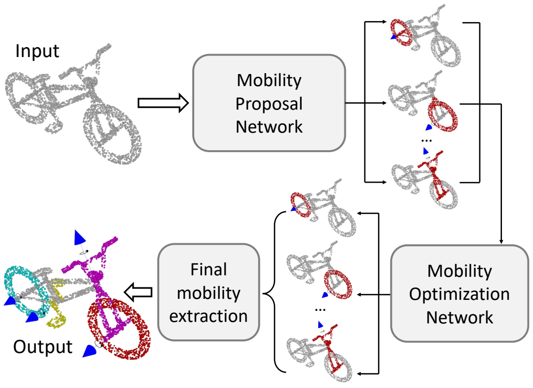
Input
Mobility Proposal Network
Mobility Optimization Network
Final mobility extraction
Output
...

Figure 2: An overview of the three stages of our method.

Semantics- / function- / mobility-induced segmentation of 3D shapes. Semantic segmentation of 3D shapes has gained significant research progress in recent year, benefiting from the advanced machine learning techniques [10, 24, 27, 28] (see also the comprehensive survey in [29]), and especially from the powerful feature learning of deep-learning models [20, 33, 25, 9]. The goal of semantic segmentation is to decompose a 3D shape into parts which are meaningful from the assembling or functional point of view. A semantically meaningful part does not necessarily imply a mobility.

Inferring functional parts is another promising way of 3D shape segmentation. Pechuk et al. [19] introduce a supervised method to recognize the functional regions of a shape according to a model of functionality of the shape class. Kim and Sukhatme [11] propose to learn a classifier of regions of shapes with functional labels such as “graspable”, “liftable”, and “pushable” in a supervised manner. Similarly, Laga et al. [13] introduce a supervised method for labeling shape parts that with functional tags, based on both geometric features and the context of parts. The method of Hu et al. [8] defines weight fields over a point-sampled surface, based on a learned classifier predicting the probability of a point belonging to a specific functional region, which could be used to infer a function-induced segmentation.

Mobility-induce segmentation is relatively less studied. The slippage analysis approach of Gelfand and Guibas [4] segments a triangle mesh into kinematic surfaces, which indicate regions of the shape that undergo a similar type of

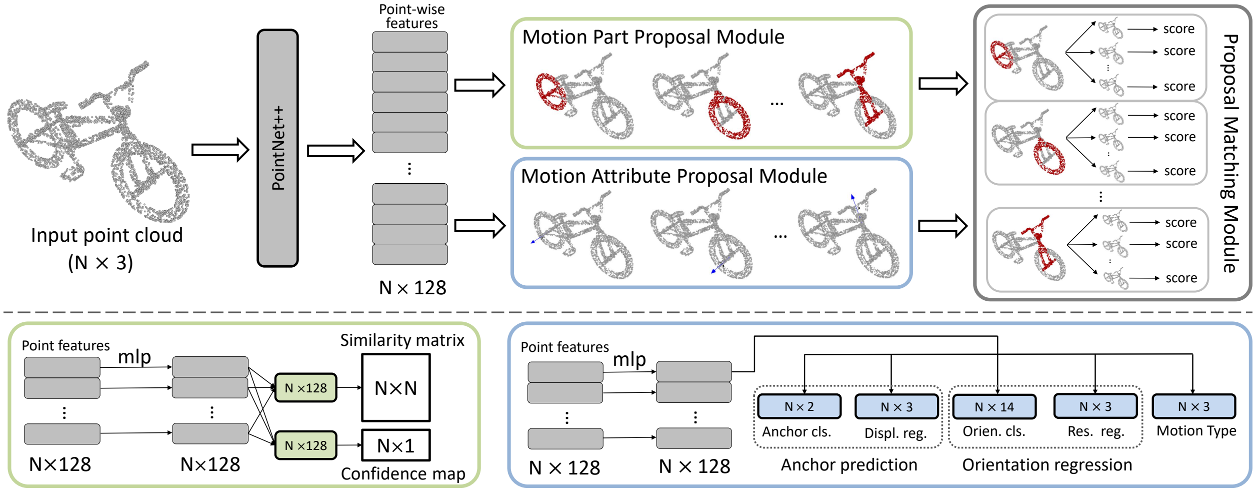

Figure 3: Architecture of Motion Proposal Network (MPN). MPN contains three modules: a Motion Part Proposal Module (green boxes), a Motion Attribute Proposal Module (blue boxes) and a Proposal Matching Module (grey box to the top-right).

motion. Each resulting segment is classified into one of a few types of kinematic surfaces, such as planes, spheres, and cylinders, which are not yet parts with functional mobilities. There is a large body of literature on co-segmentation of a sequence of animated meshes (e.g. [14, 22, 5]). These works typically handle organic objects (e.g., human and animal) undergoing smooth surface deformation. In contrast, Yi et al. [32] studies mobility analysis of man-made objects through comparing multiple different motion states of an object, e.g., a scissor with different opening angles. Our work infers part mobilities from a single, static 3D model, through joinly learning motion part segmentation and motion attribute regression from a large annotated dataset.

3. Mobility Proposal Network (MPN)

Terminology and notations. Given a 3D shape represented as point cloud $P = \{p_i\}_{i=1}^{N}$ ($p_i \in \mathbb{R}^3$), our goal is to extract a set of *part mobilities*: $\mathcal{M} = \{M_k\}_{k=1}^{K}$. Each part mobility is a tuple: $M_k = \langle P_k, A_k \rangle$, which consists of a *motion part segmentation*, $P_k \subset P$, and the corresponding *motion attributes*, $A_k = \langle t_k, m_k \rangle$, where t_k is the *motion type* and m_k the *motion parameter* of the motion. We consider three motion types: translation (T), rotation (R), and rotation+translation (RT). Motion parameter is a line in $\mathbb{R}^3$ along the motion axis referring to either a translation direction or a rotation axis, or both. Other motion parameters, such as motion range or joint limit, are left for future works.

Our Mobility Proposal Network (MPN) is designed to generate a collection of quality proposals of part mobilities, $\mathcal{M}_k^{\mathrm{P}} = \{M_k^{\mathrm{P}}\}$, so that the final part mobilities could be selected from them via an optimization stage in Section 4. MPN itself is composed of three modules: a *motion part proposal module*, a *motion attribute proposal module* and a *proposal matching module*. The first two modules are devised to propose motion part segmentations and the corresponding motion attributes, respectively. The third module is used to select a set of good-quality mobility proposals, through matching the two kinds of proposals based on a motion-driven filtering scheme.

3.1. Motion Part Proposal Module

Unlike the existing approaches, we do not assume the availability of mobility-induced shape segmentation, but instead try to solve it through coupling it with the inference of motion attributes. First of all, off-the-shelf shape segmentation methods cannot be used since a semantically meaningful part may not correspond to a motion part. Meanwhile, it is quite challenging to cast motion part segmentation into a plain labeling problem, as what is done in semantic segmentation, since the possible categories of motion parts could be extremely large, considering the combination of different functions and various motions.

We therefore opt for a proposal-based solution and devise a motion part proposal network based on the recently proposed SGPN (Similarity Group Proposal Network) [26]. SGPN is developed for object proposal in scene segmentation. It regresses a point similarity matrix S to accentuate point group proposals, from which object instances could be extracted. Each entry S_{ij} in the similarity matrix indicates whether two points, p_i and p_j, belong to the same object instance. Each row can then be viewed as an object proposal. They also regress a confidence score for each object proposal. In what follows, we explain the main adaptions we make for our problem setting.

Similarity matrix. To achieve motion part proposing, we define a motion-based point similarity matrix $S^{\mathrm{M}} \in \mathbb{R}^{N \times N}$

to encode whether two points belong to the same motion part. We train a fully-connected network to estimate S^{M}, based on the motion similarity matrix loss:

$$L_{\text{sim}} = \sum_{p_i, p_j \in P, i \neq j} l_{i,j}, \tag{1}$$

where

$$l_{i,j} = \begin{cases} \|F(p_i) - F(p_j)\|_2, & mp(i,j) = 1 \\ \max\{0, K - \|F(p_i) - F(p_j)\|_2\}, & mp(i,j) = 0 \end{cases}$$

where $mp(i,j)$ indicates whether p_i and p_j belong to the same motion part in the ground-truth (1 means 'yes'). F is point-wise feature computed with PointNet++ [20]. K is a constant that controls the degree of motion-based dissimilarity in a proposal; we use $K = 100$ by default. The rows of S^{M} then represent N different motion part proposals. During testing, we use a threshold τ_{sim} to binarize the regressed similarity matrix so that each row of the matrix represents a binary segmentation of motion part. We set $\tau_{\text{sim}} = 100$ throughout our experiments.

Confidence map. To rate the quality of the motion part proposals in S^M, we regress a confidence map $C \in \mathbb{R}^{N\times 1}$ where i-th entry corresponds to i-th motion part proposal in S^{M}. The ground-truth confidence values in C^{gt} are computed as the IoU between the point set of a proposed motion part P_i and its ground-truth counterpart P_i^{gt}. The loss for training the fully-connected regression network, L_{conf}, is the mean-square-error (MSE) loss between C and C^{gt}. During testing, only those proposals with a confidence value higher than $\tau_{\text{conf}} = 0.5$ are considered as valid ones.

3.2. Motion Attribute Proposal Module

Existing works usually estimate a single motion for each motion part. In reality, however, a motion part can possess multiple different motions. Taking the models in Figure 1 for example, the front wheel of a car can rotate along two different axes (steering and rolling); the seat of a swivel chair goes up and down (translating) while rotating.

Instead of estimating several motions for each motion part proposal, which can cause combinatorial explosion, we opt for an independent motion attribute proposal, based on a motion attribute proposal module. Given a 3D point cloud P, this module proposes a set of motion attributes $\{\langle t, m\rangle\}$. Each attribute contains a *motion type* t and a motion axis line m. The latter is parameterized by an *anchor point* $p^{\mathrm{A}} \in P$, a *displacement vector* $\mathbf{d}^{\mathrm{A}}$, and an *orientation vector* $\mathbf{v}^{\mathrm{O}}$. By selecting a point in the point cloud as anchor, the motion axis line m can be uniquely determined by a displacement to the anchor $\mathbf{d}^{\mathrm{A}}$ and an orientation $\mathbf{v}^{\mathrm{O}}$ (Figure 4(a)).

Figure 3 summarizes the network architecture of motion attribute proposal module. It contains five branches: two for anchor prediction, two for orientation regression and one for motion type prediction. Below we explain them in detail.

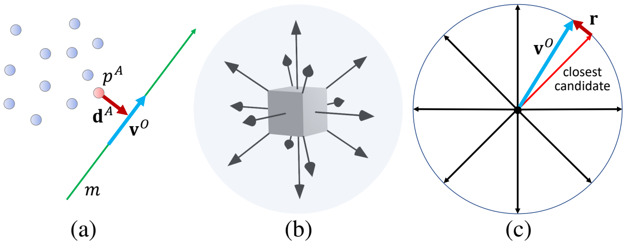

Figure 4: Illustrations of anchor-based motion axis parameterization (a), orientation discretization (b), and orientation prediction via classification and residual regression (c).

Anchor prediction. To spatially pin down a motion axis, we train a network to select a point in the input point cloud that is closest to the line along a ground-truth motion axis. We use a binary indicator vector to encode point selection. In addition, we regress the displacement vector between the anchor point and the ground-truth line; see Figure 4(a). By doing this, motion axes prediction is invariant to the shape pose. The anchoring loss is computed as:

$$L_{\text{anchor}} = L_{\text{ap}} + L_{\text{dis}}, \tag{2}$$

where L_{ap} is softmax loss for binary classification of whether a point is anchor or not, and L_{dis} an L2 loss between the predicted displacement vector and ground-truth.

Orientation regression. Direct regression of orientation is quite difficult to train. We instead turn the problem into a classification problem through discretizing the space of orientation into 14 candidates. For each candidate orientation, the network estimates both a classification probability and a residual vector used to correct the error caused by the discretization; see Figure 4(b) for illustration. The loss of orientation regression is:

$$L_{\text{orien}} = L_{\text{class}} + L_{\text{res}}, \tag{3}$$

where L_{class} is the softmax classification loss and L_{res} an L2 loss between the estimated residual vector and ground-truth.

Motion type prediction. Another network is trained to classify the mobility into one of the three motion types, with the loss L_{type} being a softmax classification loss.

3.3. Proposal Matching Module

Having obtained a set of motion part proposals and a set of motion attribute proposals, this module selects the best combinations of the two, leading to a set of high-quality mobility proposals. To do so, for each motion part proposal, we find a set of motion attribute proposals that

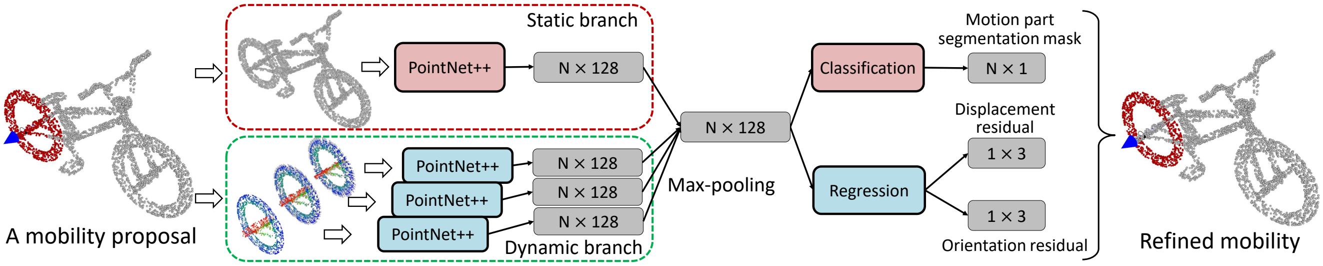

Figure 5: Architecture of Motion Optimization Network (MON). Given a part mobility, MON encodes both static shape geometry (red box) and dynamic part motion (green box). It produces a refined segmentation of the motion part (removing the points of "back fork" which are mis-labeled as "wheel" in the input) and residual vectors for correcting the motion axis.

best matches it. This is achieved by training another network, which takes the feature maps of a motion part and a motion attribute learned from the respective modules, and regresses a matching score for them. To train this network, we compute a matching score loss between a proposed mobility M^{prop} and its ground-truth counterpart M^{gt}: $L_{\text{ms}} = |S^{\text{pred}} - S^{\text{gt}}|$. S^{pred} is the matching score predicted by the network. $S^{\text{gt}} = score(M^{\text{prop}}, M^{\text{gt}})$, where $score$ is a scoring function measuring the similarity between a proposed mobility and the ground-truth.

To measure the similarity between two mobilities, we propose a motion-driven metric which accounts for both the similarity of motion part segmentation and motion attributes. The basic idea is to let the motion parts move a prescribed amount according to their respective motions, for both mobilities, and measure the degree of spatial alignment between the moved parts:

$$score(M_j, M_k) = \frac{1}{N}\sum_{p_i \in P} \|move(p_i, M_j) - move(p_i, M_k)\|_2, \tag{4}$$

where $move(p_i, M_k)$ is a moving function:

$$move(p_i, M_k) = \begin{cases} T(p_i, m_k, \delta)/dist(p_i, m_k), & p_i \in P_k \\ 0, & p_i \notin P_k \end{cases}$$

which moves point $p_i \in P$ according to the motion defined in M_k, if p_i belongs to the point cloud of the corresponding motion part. $T(p_i, m_k, \delta)$ transforms a point according to the motion axis line by an amount of δ. δ takes 15% of the diagonal length of P's bounding box for translation, and 90° for rotation. $dist(p_i, m_k)$ is the distance from p_i to motion axis m_k, which is used to normalize the movement.

4. Mobility Optimization Network

Since the proposals for motion parts and attributes are generated separately, they could be inaccurate and insufficiently conforming with each other. Mobility Optimization Network (MON) is trained to jointly optimize both in an end-to-end fashion (Figure 5). Given a proposed mobility, MON refines it through predicting a binary segmentation of motion part out of the input shape point cloud, and regressing two residual vectors for correcting the displacement $\mathbf{d}^{\text{A}}$ (w.r.t. the anchor point) and orientation $\mathbf{v}^{\text{O}}$ of the motion axis respectively.

To account for both shape geometry and part motion, MON takes both static and dynamic information as input. The static branch encodes the shape point cloud with PointNet++ features. The dynamic branch takes as input a bunch of moved point clouds corresponding to the motion part being refined. Specifically, we move the part about the motion axis by three specific amounts (5%, 10% and 15% of the diagonal length of shape bounding box for translation and 30°, 60° and 90° for rotation). The moved point clouds are again encoded with PointNet++ except that the point-wise moving vectors are also encoded (using the point normal channel). Note that the two branches do not share weights. By integrating both static geometric features and *motion-driven* dynamic features, MON achieves highly accurate segmentation and regression.

The loss for MON is composed of two parts, a point-wise labeling loss and the residual losses for motion axis:

$$L_{\text{MON}}(P, M_k) = \sum_{p_i \in P} L_{\text{label}}(p_i) + L_{\text{res}}(r^{\text{pred}}_{\text{displ}}, r^{\text{gt}}_{\text{displ}}) + L_{\text{res}}(r^{\text{pred}}_{\text{orien}}, r^{\text{gt}}_{\text{orien}}), \tag{5}$$

where L_{label} is a negative log-likelihood loss for labeling and L_{res} takes L2 loss. The ground-truth residuals are computed by comparing the predicted displacement/orienation vectors and their ground-truth counterparts.

5. Final Mobility Extraction

During the testing phase, having obtained a set of high-quality mobility proposals with associated matching scores, we also need a final extraction process to merge the proposals as output. This process is similar to the Non-Maximum Suppression (NMS) step employed in many proposal-based object detection works. Different from NMS, however, we need to select both the motion part and its associated motion attributes. Furthermore, one part may have multiple possible motions.

We first select motion parts: When multiple motion part proposals overlap significantly, we keep only the one with the highest matching score. Then for each selected motion part, we select distinct motion attributes from those mobility proposals whose motion part has a significantly large IoU against that part. For translation, only one direction with the highest score is selected, assuming a part can slide along only one direction. For rotation, we perform a greedy selection of high score axes while ensuring the angle between every two selected axes is larger than a threshold (45°).

6. Details, Results and Evaluations

6.1. Network training

Training scheduling. Our network is implemented with Tensorflow. We used Adam [12] for training and set the initial learning rate to 0.001. Our network is trained in three stages. In the *first stage*, we train the Motion Part Proposal Module and Motion Attribute Proposal Module, each for 100 epochs. The mini-batch size is 8 shapes. In the *second stage*, we train the Proposal Matching Module using the motion part proposals and motion attribute proposals generated from the first stage, again, for 100 epochs. Here, we use mini-batches of size 64 (4 shapes ×16 motion part proposals). The training motion part proposals are selected from those whose IoU against ground-truth is greater than 0.5 and within top 25%. The *third stage* trains the Motion Optimization Network for 100 epoch. The training mobility proposals are selected from those whose matching error at most 0.05 and within top 10%. The mini-batch is 8 mobility proposals. Our networks make extensive use of PointNet++ [20], for which we use the default architecture and parameters provided in the original paper. The training and testing time is reported in the supplemental material.

Data augmentation. Given a training dataset with ground-truth mobilities, we perform data augmentation via generating two kinds of shape variations. We first generate a number of *geometric variations* for each training shape based on the method described in [3]. Furthermore, based on the ground-truth mobility in the training shapes, we move the motion parts according to their corresponding motion attributes, resulting in a large number of *motion variations*. See the supplemental material for a few samples of our training data augmentation.

6.2. Shape2Motion benchmark

We contribute the first benchmark for mobility analysis, called Shape2Motion, encompassing both motion part segmentation and motion attribute estimation. It contains 2440 models in 45 shape categories. These shapes are selected from ShapeNet [2] and 3D Warehouse [1]. A overview and detailed information of the benchmark dataset can be found int the supplemental material. For each shape, we manually label its motion parts and the corresponding motion attributes, using an easy-to-use annotation tool developed by ourselves. The benchmark along with the annotation tool are released at: http://www.kevinkaixu.net/projects/shape2motion.html and http://z.buaa.edu.cn/shape2motion.html.

Annotation tool. Mobility annotation of 3D shapes is tedious. To ease the process, we developed an annotation tool with an easy-to-use user interface. The tool consists of two parts, one for motion part annotation and one for motion attribute annotation. A notable feature of our tool is that it allows the user to visually verify the correctness of a annotated mobility, by animating the annotated motion part with the corresponding motion attributes prescribed by the user. Using our tool, it takes 80 seconds in average to annotate a 3D shape. In the supplemental material, we provide more details about the annotation tool.

6.3. Evaluation on Shape2Motion

We train and test our model on the Shape2Motion dataset, with a training / testing split of 8 : 2. After data enhancement, we obtain 11766 shapes with 32298 mobility parts in total as training data. Figure 6 shows a couple of visual results of mobility analysis. More results can be found in the supplemental material.

To compare with the state-of-the-art method in [32], we train their model on the Shape2Motion dataset. This model requires shapes in two different motion states as input both for training and testing. To this end, we compute for each shape another motion state according to the ground-truth mobilities. Their method is trained to predict motion parts and associated 3D motion flows, while our method predicts motion parts and motion attributes. To make the two methods comparable, we convert the mobilities predicted by our method into 3D motion flows. We measure IoU to evaluate motion part segmentation and End-Point-Error (EPE) [31] to evaluate 3D motion flow. The comparison is shown in Table 1. Our method outperforms theirs because our method can detect those mobilities which are difficult to capture by motion state comparison. For instance, it would be very hard to capture a rotating wheel with two motion states.

In addition, we develop a baseline method (Table 1, row 2) for directly segmenting motion parts and regressing motion attributes. Specifically, we use the instance segmentation network SGPN [26] to perform motion part segmentation. For motion attribute prediction, we design a baseline network that takes the point cloud of a motion part as input and regress the motion type and motion axis (more details in the supplemental material). To facilitate comparison, we

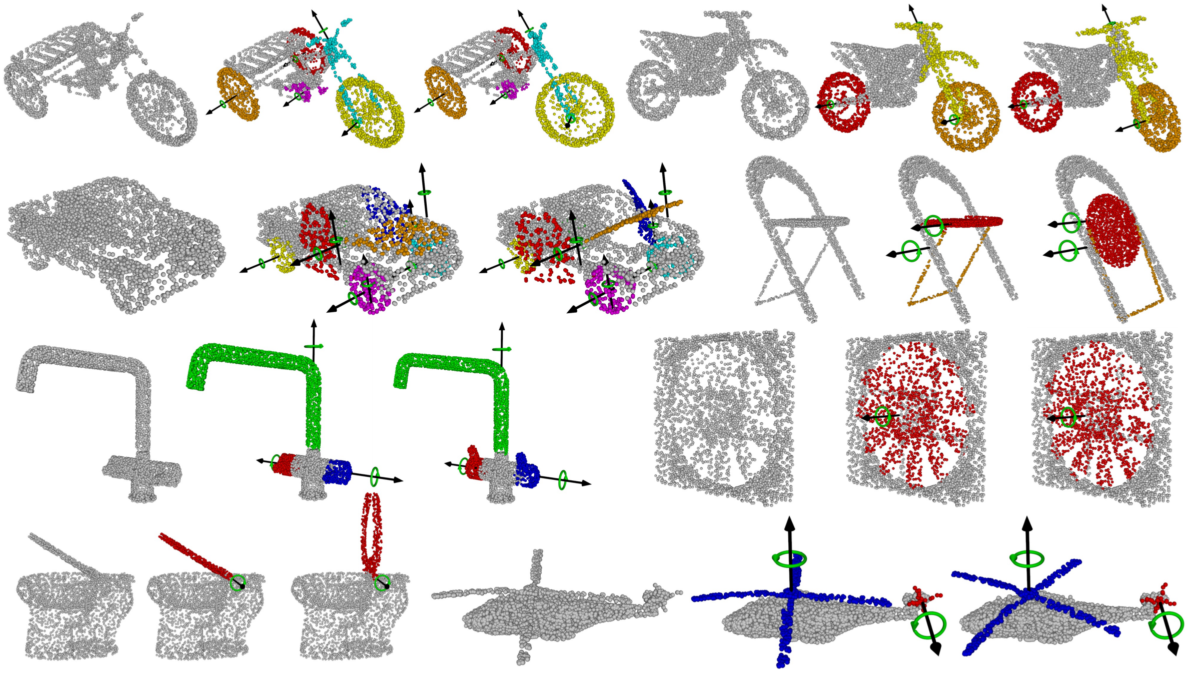

Figure 6: Visual results of mobility analysis. For each example, we show from left to right the input point cloud, the extracted mobilities (motion parts and motion axes) and the point cloud with motion parts moved according to detected mobility.

	IoU	EPE	MD	OE	TA
Yi et al. [32]	61.0	0.051	-	-	-
SGPN [26]+BL	79.4	-	0.11	3.13	0.96
Ours (w/o MON)	82.3	0.028	**0.01**	0.12	**0.98**
Ours	**84.7**	**0.025**	**0.01**	**0.11**	**0.98**

Table 1: Comparison on the Shape2Motion benchmark.

	IoU	EPE	MD	OE	TA
Hu et al. [6]	-	-	0.030	8.14	-
SGPN [26]+BL	54.3	-	0.145	11.8	0.74
Ours (w/o MON)	58.6	0.074	0.028	0.78	**0.93**
Ours	**64.7**	**0.061**	**0.024**	**0.12**	**0.93**

Table 2: Comparison on the dataset of Hu et al. [6].

define three metrics for evaluating motion attributes. Minimum Distance (MD) and Orientation Error (OE) measure the distance and angle between the predicted motion axis line and the ground-truth. Type Accuracy (TA) is accuracy of motion type prediction. Our method achieves better performance for both motion part segmentation and motion attribute estimation, through leveraging the synergy between the two tasks in a joint optimization framework.

6.4. Evaluation on the dataset in [6]

We also evaluate our method on the dataset in the work of Hu et al. [6], which contains 315 shapes in 37 categories, with 368 mobility parts in total. We again performed data enhancement on this dataset. Table 2 reports the comparison of the afore-mentioned metrics between the method of Hu et al. [6], the SGPN+baseline method, and our method.

In the method of Hu et al., the availability of segmented motion parts is assumed. To predict the motion attributes (motion type and motion parameters) of a part, it takes the part and a static part adjacent to that part as input, and perform prediction via metric learning. We report their performance on two metrics, i.e., MD and OE. The results shows our method is advantageous thanks to our powerful deep-learning model trained effectively with the motion-driven strategy. Since their motion types are different from ours, we do not report TA comparison.

6.5. Analysis of parameters and networks

Effect of the parameter τ_{sim} and τ_{conf}. As mentioned in Section 3.1, τ_{sim} is used to binarize the similarity matrix to form a binary segmentation of motion part in each row of the matrix. We use $\tau_{\text{sim}} = 100$ by default. In Figure 7 (left), we study the quality of motion part proposals over the different values of τ_{sim}, over four relatively difficult shape categories. For a fixed IoU threshold (0.5), we find that the recall rate grows fast with the increasing τ_{sim}, and then drops as τ_{sim} continues to increase. The peak is reached around $\tau_{\text{sim}} = 100$. This parameter τ_{sim} is strongly correlated with the parameter K in Equation (1), which is the margin used in defining the hinge similarity loss.

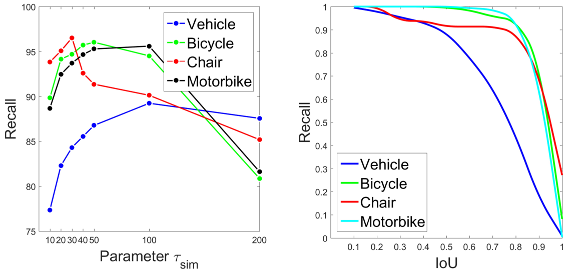

Figure 7: Left: Recall rate of motion parts over increasing τ_{sim}, when IoU against ground-truth is fixed to 0.5. Right: Recall rate over IoU.

	Vehicle	Bicycle	Chair	Motorbike
IoU ($\tau_{\text{conf}} \geqslant 0.3$)	57.8	84.3	93.0	83.3
IoU ($\tau_{\text{conf}} \geqslant 0.5$)	**70.7**	**90.6**	**98.0**	**86.9**
IoU ($\tau_{\text{conf}} \geqslant 0.7$)	64.3	84.0	**98.0**	85.6

Table 3: Average IoU of motion part proposals over different values of τ_{conf} on the Shape2Motion benchmark.

In Section 3.1, τ_{conf} is used to filter motion part proposals based on the confidence map. In Table 3, we also study the motion part quality (average IoU) over different confidence threshold τ_{conf}. We find that the proposal quality is the best when $\tau_{\text{conf}} = 0.5$.

Effect of Motion Optimization Network (MON). To evaluate the effectiveness of MON, we experiment an ablated version of our network without MON, over the two datasets. The results are reported in row 3 of Table 1 and 2. By incorporating MON, our method achieves higher performance for both motion part segmentation and motion attributes estimation, verifying its optimization effect.

Effect of Motion Part Proposal Module. In Figure 7 (right), we evaluate the quality of motion part proposals via plotting the recall rate over average IoU. It can be observed that our method can generate many high-quality motion part proposals. It can be seen that our method achieves very high recall rate (> 0.9) for an IoU of 0.5.

Effect of Motion Attribute Proposal Module. To verify the effectiveness of this module, we compared to an ablated version of our method which replaces this module by direct motion attributes regression as in the SGPN+BL in Table 1. The quality of a predicted mobility is measured as the similarity between the mobility and its ground-truth, using similarity score in Equation (4). In Figure 8, we plot the distribution of motion part numbers over varying similarity scores (quality). The plots show that our full method works the best: The similarity of approximately 80% mobility proposals is less than 0.02.

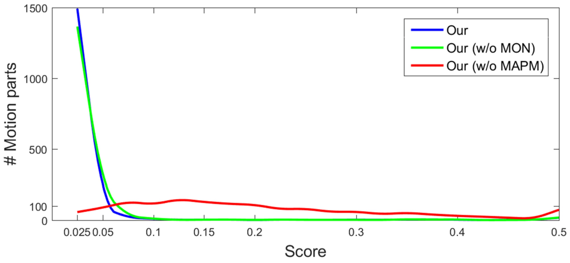

Figure 8: The distribution of motion part numbers over varying similarity scores (quality) of proposals.

Timing The training of MPN and MON takes 31 and 35 hours for 50 epochs on a NVIDIA TITIAN X GPU, respectively. The testing time per 3D shape is 0.4 seconds for MPN and 1 seconds for MON. The total computational cost is approximately 10 seconds for each shape.

7. Conclusion

We have presented, Shape2Motion, an approach to simultaneous motion part segmentation and motion attribute estimation, using a single 3D shape as input. The method adopts a proposal-and-optimize strategy, and consists of two deep neural networks, i.e., a mobility proposal network (MPN) followed by a mobility optimization network (MON). A key insight in training these networks is to fully exploit the coupling between a motion part and its mobility function, leading to a novel concept of motion-driven training, which may be valuable also for other scenarios.

Limitations and future works. Our approach has a few limitations, which point out the directions of future study. Representative failure cases can be found in the supplemental material. First, our method works with point cloud representation, which may not be able to represent shapes with highly detailed structure, such as a knife with scabbard. However, we believe the general framework of Shape2Motion can adapt to other shape representations. Second, as a proposal-and-optimize framework, our method as a whole is not end-to-end trainable. Third, our method does not support hierarchical mobility extraction, which is ubiquitous in real-world objects. This involves higher-order analysis which is an interesting direction for future work. We would also like to study the mobility analysis of scanned real-world objects.

Acknowledgement

We thank the anonymous reviewers for their valuable comments. This work was supported in part by Natural Science Foundation of China (61572507, 61532003, 61622212, U1736217), and Natural Science Foundation of Hunan Province (2017JJ1002).

References

[1] 3d warehouse. https://3dwarehouse.sketchup.com/, 2017. Accessed: 2017-05-18. 6

[2] A. X. Chang, T. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. (arXiv:1512.03012 [cs.GR]). 6

[3] Q. Fu, X. Chen, X. Su, J. Li, and H. Fu. Structure-adaptive Shape Editing for Man-made Objects. *Computer Graphics Forum*, 2016. 6

[4] N. Gelfand and L. J. Guibas. Shape segmentation using local slippage analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 214–223. ACM, 2004. 1, 2

[5] S. Ghosh, M. Loper, E. B. Sudderth, and M. J. Black. From deformations to parts: Motion-based segmentation of 3d objects. In *Advances in Neural Information Processing Systems*, pages 1997–2005, 2012. 3

[6] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang. Learning to predict part mobility from a single static snapshot. *ACM Trans. on Graph. (SIGGRAPH Asia)*, 36(6):227, 2017. 1, 2, 7

[7] R. Hu, M. Savva, and O. van Kaick. Functionality representations and applications for shape analysis. *Computer Graphics Forum (Eurographics State of The Art Report)*, 37(2), 2018. 1, 2

[8] R. Hu, O. van Kaick, B. Wu, H. Huang, A. Shamir, and H. Zhang. Learning how objects function via co-analysis of interactions. *ACM Transactions on Graphics (TOG)*, 35(4):47, 2016. 2

[9] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. In *Proc. CVPR*, volume 1, page 8, 2017. 2

[10] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 29(4):102:1–102:12, 2010. 2

[11] D. I. Kim and G. S. Sukhatme. Semantic labeling of 3d point clouds with object affordance for robot manipulation. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5578–5584. Citeseer, 2014. 2

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014. 6

[13] H. Laga, M. Mortara, and M. Spagnuolo. Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(5):150, 2013. 2

[14] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen. Segmenting a deforming mesh into near-rigid components. *The Visual Computer*, 22(9-11):729, 2006. 3

[15] H. Li, G. Wan, H. Li, A. Sharf, K. Xu, and B. Chen. Mobility fitting using 4d ransac. *Computer Graphics Forum*, 35(5):79–88, 2016. 2

[16] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala. Illustrating how mechanical assemblies work. *ACM Transactions on Graphics-TOG*, 29(4):58, 2010. 2

[17] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA*, pages 1374–1381, 2015. 1

[18] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis. Detecting object affordances with convolutional neural networks. In *Proc. IROS*, pages 2765–2770. IEEE, 2016. 1

[19] M. Pechuk, O. Soldea, and E. Rivlin. Learning function-based object classification from 3d imagery. *Computer Vision and Image Understanding*, 110(2):173–191, 2008. 2

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2, 4, 6

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[22] G. Rosman, A. M. Bronstein, M. M. Bronstein, X.-C. Tai, and R. Kimmel. Group-valued regularization for analysis of articulated motion. In *European Conference on Computer Vision*, pages 52–62. Springer, 2012. 3

[23] A. Sharf, H. Huang, C. Liang, J. Zhang, B. Chen, and M. Gong. Mobility-trees for indoor scenes manipulation. *Computer Graphics Forum*, 33(1):2–14, 2014. 2

[24] O. Sidi, O. van Kaick, Y. Kleiman, H. Zhang, and D. Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graph. (SIGGRAPH Asia)*, 30(6), 2011. 2

[25] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017. 2

[26] W. Wang, R. Yu, Q. Huang, and U. Neumann. SGPN: similarity group proposal network for 3d point cloud instance segmentation. *CoRR*, abs/1711.08588, 2017. 3, 6, 7

[27] Y. Wang, M. Gong, T. Wang, D. Cohen-Or, H. Zhang, and B. Chen. Projective analysis for 3d shape segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 32(6):192:1–192:12, 2013. 2

[28] Z. Xie, K. Xu, and L. L. abd Yueshan Xiong. 3d shape segmentation and labeling via extreme learning machine. *Computer Graphics Forum (Proc. SGP)*, 33(5):85–95, 2014. 2

[29] K. Xu, V. G. Kim, Q. Huang, N. Mitra, and E. Kalogerakis. Data-driven shape analysis and processing. In *SIGGRAPH ASIA 2016 Courses*, page 4. ACM, 2016. 2

[30] W. Xu, J. Wang, K. Yin, K. Zhou, M. Van De Panne, F. Chen, and B. Guo. Joint-aware manipulation of deformable models. *ACM Transactions on Graphics (TOG)*, 28(3):35, 2009. 2

[31] Z. Yan and X. Xiang. Scene flow estimation: A survey. *CoRR*, abs/1612.02590, 2016. 6

[32] L. Yi, H. Huang, D. Liu, E. Kalogerakis, H. Su, and L. Guibas. Deep part induction from articulated object pairs. *ACM Trans. on Graph. (SIGGRAPH Asia)*, 2018. 1, 3, 6, 7

[33] L. Yi, H. Su, X. Guo, and L. J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017. 2