Unsupervised Kinematic Motion Detection for Part-segmented 3D Shape Collections

Xianghao Xu
xianghao_xu@brown.edu
Brown University
USA

Yifan Ruan
yifan_ruan@brown.edu
Brown University
USA

Srinath Sridhar
srinath_sridhar@brown.edu
Brown University
USA

Daniel Ritchie
daniel_ritchie@brown.edu
Brown University
USA

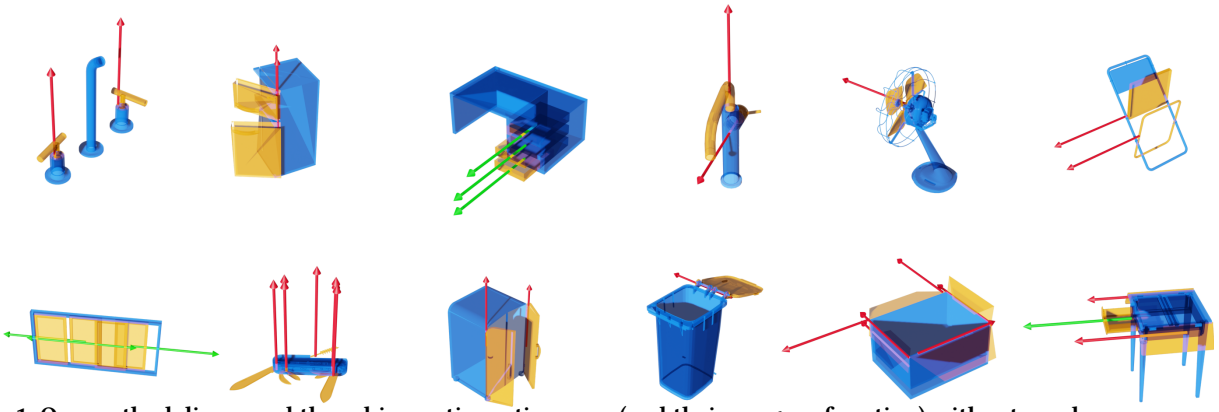

Figure 1: Our method discovered these kinematic motion axes (and their ranges of motion) without any human supervision. It works by finding motion parameters such that one shape can transform into another from the same category. Moving parts are orange; static parts are blue; translation axes are green; rotation axes are red.

ABSTRACT

3D models of manufactured objects are important for populating virtual worlds and for synthetic data generation for vision and robotics. To be most useful, such objects should be *articulated*: their parts should move when interacted with. While articulated object datasets exist, creating them is labor-intensive. Learning-based prediction of part motions can help, but all existing methods require annotated training data. In this paper, we present an unsupervised approach for discovering articulated motions in a part-segmented 3D shape collection. Our approach is based on a concept we call *category closure*: any valid articulation of an object's parts should keep the object in the same semantic category (e.g. a chair stays a chair). We operationalize this concept with an algorithm that optimizes a shape's part motion parameters such that it can transform into other shapes in the collection. We evaluate our approach by using it to re-discover part motions from the PartNet-Mobility dataset. For almost all shape categories, our method's predicted motion parameters have low error with respect to ground truth annotations, outperforming two supervised motion prediction methods.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530742

CCS CONCEPTS

• **Computing methodologies** → **Shape analysis**; **Unsupervised learning**.

KEYWORDS

kinematic motion prediction, articulation, 3D shape databases

ACM Reference Format:
Xianghao Xu, Yifan Ruan, Srinath Sridhar, and Daniel Ritchie. 2022. Unsupervised Kinematic Motion Detection for Part-segmented 3D Shape Collections. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3528233.3530742

1 INTRODUCTION

3D models of manufactured objects are important for many applications: populating virtual worlds for games, AR/VR experiences, animation, interior design, and architectural visualization; creating synthetic training data for data-hungry computer vision models [Richter et al. 2016; Zhang et al. 2017]; simulated training for

robots to learn to navigate or to detect and manipulate objects before being deployed in the real world [Das et al. 2018; Kolve et al. 2017; Manolis Savva* et al. 2019; Savva et al. 2017; Xiang et al. 2020; Yan et al. 2018]. Ideally, such 3D objects should be *articulated*: each part should specify how it moves (e.g. cabinet drawers slide open). Recognizing the value of such data, researchers have created datasets of articulated 3D object models [Hu et al. 2017a; Wang et al. 2019; Xiang et al. 2020]. However, annotating 3D objects with kinematic motions requires human time and effort. One alternative approach is to use machine learning to predict kinematic part motions for a shape, automating some manual annotation effort. While such methods have been proposed, all rely on supervised learning with 3D shapes that already have motion annotations.

In this paper, we present an unsupervised method that discovers kinematic motions in a consistently part-segmented 3D shape collection. What makes our method possible is an insight we call *category closure*: given an object of category C (e.g. chairs), all valid articulations of its parts will produce a shape that is still in category C. While appealing in theory, this insight is challenging to apply in practice, as it is non-trivial to determine without supervision whether an articulated shape belongs to a category. We address this challenge via the following observation: given a collection of shapes of the same category, an articulation of one shape's parts definitely remains in the same category if that articulation can transform the shape into other shapes from the collection. Implicit in this observation is the assumption that the shape collection contains some degree of part pose variation.

Based on these insights, we design an alternating optimization scheme for discovering part articulations in a collection of shapes. In one optimization phase, the system learns an embedding space in which shapes which can transform to one another via articulation are close. The learning signal is based on feedback from another phase, in which groups of nearby shapes in the embedding space are selected and articulation parameters are optimized to try to transform one shape in the group into the others. As this optimization problem is underconstrained, the system uses commonsense and physically-inspired priors to avoid finding implausible part motions. Our approach predicts the type of motion (rotational, translation, or static) as well as the motion parameters (axes of motion, centers of rotation, ranges of motion).

We evaluate our approach by predicting articulations for shapes in PartNet-Mobility, a dataset of consistently part-segmented objects which have ground-truth kinematic motion annotations with which we can compare [Xiang et al. 2020]. Our approach discovers motion parameters which exhibit low error with respect to the ground truth, outperforming two supervised motion prediction approaches on almost all shape categories. In summary, our contributions are:

- The concept of category closure as self-supervision for discovering valid kinematic part motions.
- An alternating optimization scheme which implements this concept by finding motion parameters which transform objects into other objects of the same category.

Code and data for this paper are at https://github.com/xxh43/UKMD

2 RELATED WORK

Articulated object datasets. Researchers have built datasets of part-segmented shapes with kinematic motions. One includes an unreleased dataset of 368 moving joints [Hu et al. 2017a], manually annotated from ShapeNet [Chang et al. 2015]. The Shape2Motion dataset [Wang et al. 2019] (no longer available) contained 2,240 3D objects from 45 categories, sourced from ShapeNet and the 3D Warehouse [Inc. 2021] and manually annotated with kinematic motions. PartNet-Mobility [Xiang et al. 2020] consists of over 2,000 objects in 47 categories, also from ShapeNet. All these datasets were manually annotated, which is labor intensive. Other prior work proposes a machine-learning-assisted interface for rapidly writing simple programs to annotate shapes with kinematic motions [Xu et al. 2020]. This system reduces human labeling effort but does not eliminate it. We seek a method that require no human labeling.

Predicting part mobilities. Early work on automatic mobility prediction includes illustrating the motions of mechanical assemblies [Mitra et al. 2010], analyzing multiple instances of an object in a scene [Sharf et al. 2014], slippage analysis for deformable mesh models [Xu et al. 2009], and inferring kinematic chains based on motion trajectories [Yan and Pollefeys 2006]. These methods rely on having high-fidelity, physically accurate joint geometry or access to multiple observations of the same shape in different poses; in contrast, large shape collections have widely varying geometric quality and only contain a single observation of each shape. The problem has also been studied in robotics for manipulating unknown articulating objects [Hausman et al. 2015; Pillai et al. 2015; Sturm et al. 2011]. In computer vision, machine learning has been applied to unstructured point clouds to jointly segment them into parts and predict their motions [Hu et al. 2017b; Wang et al. 2019; Yan et al. 2019; Yi et al. 2019]. Closet to our work is the system of Hu et al. [2017a], which also assumes consistently-segmented manufactured object meshes. Given a new object, it retrieves the best-matching example and transfers its motion to the input. These methods require labeled examples; in contrast, our approach leverages the principle of category closure as form of self-supervision. In concurrent work, Kawana et al. [2021] jointly predict part segmentation and part articulations via a neural network trained with adversarial self-supervision. Training this network requires many pose variations of each training shape. In contrast, our method works with only *one* observation of each unique training object and requires fewer unique training objects.

Estimating articulation from images. Estimating 3D articulation from images and depth maps has been widely studied for humans [Alldieck et al. 2018; Ballan et al. 2012; Huang et al. 2020; Joo et al. 2018; Kanazawa et al. 2018; Mehta et al. 2017; Mueller et al. 2018; Shotton et al. 2011] and more recently for articulating objects [Abbatematteo et al. 2019; Li et al. 2020; Zhang et al. 2021], assuming a known kinematic structure. When the structure is unknown, a recent method [Mu et al. 2021] has proposed to disentangle shape and appearance using a neural network to estimate parts, joints, and joint angles. Unlike these methods, ours requires no kinematic structure or other supervision..

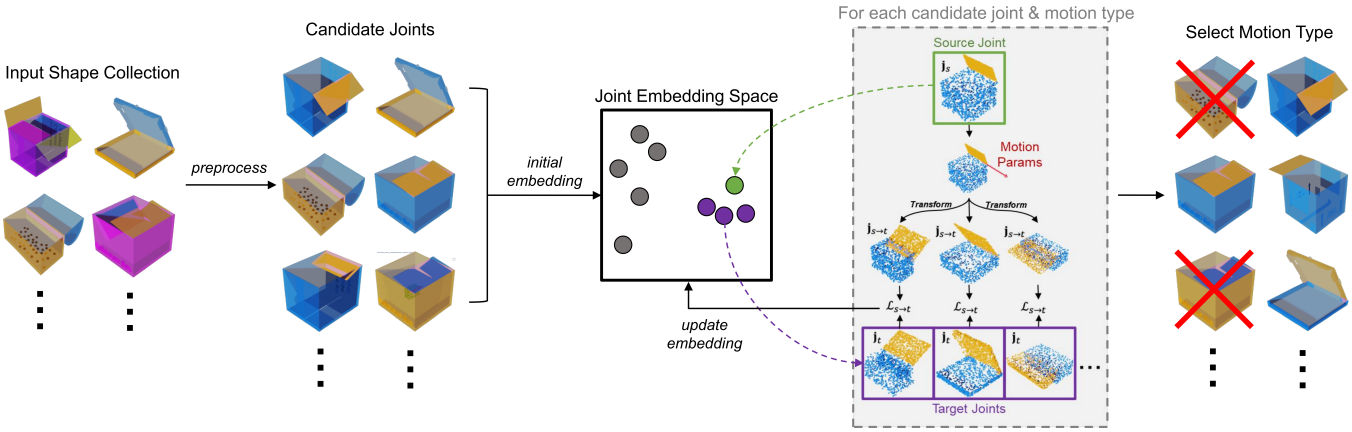

Figure 2: Given a collection of consistently part-segmented shapes from the same semantic category, our system first extracts a set of candidate *joints* consisting of connected parts, one of which may move about the other. It then initializes an embedding space in which two joints should be nearby if one can be transformed into the other through a valid hinge or prismatic motion. For each candidate joint j_s, it samples nearby joints j_t and optimize for motion parameters which approximately transform j_s to each j_t (producing $j_{s\to t}$). The reconstruction error is used as feedback to improve the embedding space, and the process repeats. This results in multiple possible motions for each joint, so we heuristically select which motion (if any) is best. Best viewed zoomed.

3 APPROACH

Figure 2 shows an overview of our system, which takes as input a set of part-segmented shapes. We assume the segmentations are consistent (i.e. shapes are segmented at the same granularity, though some shapes may have parts that others do not) but do not require part labels. Such data can be scalably produced from online 3D model repositories [CGTrader 2020; Inc. 2020; Turbosquid 2020] using e.g. machine-learning-assisted segmentation tools [Yi et al. 2016]. We also assume that input shapes exhibit some part pose variations (i.e. parts are not modeled in exactly the same pose across every shape in the dataset). We examine the impact of input pose variation on our method's performance in Section 7.

From the input shapes, we create a set of candidate *joints*: each joint $\mathbf{j}$ consists of a moving part $\mathbf{j}^m$ and a base part $\mathbf{j}^b$, i.e. the fundamental unit of kinematic motion. For each part in each shape, we create one joint in which that part is $\mathbf{j}^m$ and the largest other part to which it is connected is $\mathbf{j}^b$. Part connectivity is automatically inferred from geometry; see supplemental. For each joint, our goal is to determine what type of motion (if any) applies to it, and what its motion parameters are. We consider two types of joints: hinge (rotational) joints, parameterized by an axis, a center of rotation, and a range of angles; and prismatic (translational) joints, parameterized by an axis and a range of displacements [Murray et al. 2017].

To solve this problem, we observe that a valid joint motion is one that can transform the joint into other joints from the same collection (assuming parts occur in different poses throughout the collection); this is the concept of *category closure*. Our solution is a two phase, alternating optimization scheme. The first phase, for a given joint, identifies 'target joints' to which it should be transformed. Not all joints are good targets, e.g. we do not want to transform a cabinet door into a cabinet drawer. Our system identifies good targets by building an embedding space in which joints are nearby if they are good transformation targets for one another. The more other joints into which we can transform one joint through a motion, the more confident we can be that this motion is correct.

To learn this embedding space, the system relies on feedback from the second phase. Here, given a source joint, a set of target joints, and a candidate motion type, the system optimizes for motion parameters that transform the source into the target. This problem is underconstrained and can produce implausible motions; thus, we introduce commonsense and physically-inspired priors to steer the system toward good solutions.

These two phases are iterated: feedback about how well a source joint can be transformed to its targets is used to improve the embedding space; the improved embedding space leads to new target joints which help the system optimize for better motions. This iterative process produces multiple possible motions for each part. Thus, the system uses a heuristic final phase to determine which type of motion (or no motion) is most plausible.

4 IDENTIFYING TRANSFORMATION TARGET JOINTS

Given a candidate joint, the goal of this phase is to construct a set of 'target joints' to which that joint should be transformed via a kinematic motion. The more other joints into which we can transform one joint through a motion, the more confident we can be in that motion. To solve this problem, we construct an embedding space in which two joints are close by if one is a good target for the other.

Initial embedding. Initially, the system has no information about which joints can transform into other joints via valid motions. Thus, we construct an initial embedding based on which joints can transform into others through *any* affine transformation. For every pair of joints $(\mathbf{j}_1, \mathbf{j}_2)$, we optimize for a rotation, translation, and scale (where we penalize the anistropy of the scale) for both $\mathbf{j}_1^m$ and

$\mathbf{j}_1^b$ to bring them as close as possible to $\mathbf{j}_2^m$ and $\mathbf{j}_2^b$ by minimizing bidirectional chamfer distance (assuming all objects are consistently upright-oriented, $\mathbf{j}_1^b$'s rotation reduces to a single rotation about the up axis).

We then use the optimization residuals to produce a $N \times N$ similarity matrix for a collection of N joints ($N \in [100, 200]$, in our experiments). We set the similarities between joints that have a different number of connected components in either their moving part or base part to zero, to prevent these structurally-different joints from being grouped as source-target pairs. An embedding can be constructed from this matrix, but we do not need to do so—for our purposes, it suffices to select, for a source joint $\mathbf{j}_s$ the 16 most similar joints as its set of potential target joints.

Iterative improvement. On each iteration of the system, for each source joint and its target joints, we run the motion optimization procedure described in Section 5. This produces a transformation reconstruction loss $\mathcal{L}_{s\to t}^{\text{recon}}$ for each pair of source and target joints $(\mathbf{j}_s, \mathbf{j}_t)$. The system uses these losses to learn a new embedding space, where the distance between two joint embeddings should be proportional to their loss:

$$\mathcal{L}^{\text{embed}} = \frac{1}{N}\sum_{s=1}^{N}\frac{1}{k}\sum_{t\in\mathcal{T}_s}|\mathcal{L}_{s\to t}^{\text{recon}} - \alpha||E(\mathbf{j}_s) - E(\mathbf{j}_t)||_2| \quad (1)$$

where N is the total number of joints, $k = 5$ is the number of target joints per source joint, $\mathcal{T}_s$ is the set of k targets for source joint s, and E is a PointNet encoder [Qi et al. 2017] whose parameters (and α) are the variables of optimization. A joint is fed to the encoder as a point cloud with a per-point one-hot indicator of whether the point belongs to the moving part or base part. We minimize this loss using Adam [Kingma and Ba 2014]. We then select k new target joints for each source joint $\mathbf{j}_s$ by sampling $\mathbf{j}_t \sim \exp(-||E(\mathbf{j}_s) - E(\mathbf{j}_t)||_2)$. The system then moves to the next iteration.

5 OPTIMIZING FOR JOINT MOTION PARAMETERS

Given a source joint $\mathbf{j}_s$, a set of target joints $\{\mathbf{j}_t\}$, and a motion type (hinge or prismatic), the goal of this phase is to optimize for motion parameters that can transform the source joint to each of the targets. As a pre-process, we first optimize for rotations $\theta_{t\to s}$ about the up axis that bring each target joint $\mathbf{j}_t$ into closest alignment with $\mathbf{j}_s$ (via bidirectional chamfer distance).

5.1 Parameterized transformation model

We first define the parametric function by which one joint $\mathbf{j}_s$ is transformed into another joint $\mathbf{j}_t$:

Kinematic motion. We use T_s^J to denote a prismatic (translational) joint transformer for joint s (implicitly parameterized by a translation direction vector). We use $T_s^J(\mathbf{j}_s^m, d)$ to denote articulating the source joint $\mathbf{j}_s$'s moving part $\mathbf{j}_s^m$ with translational displacement d. Similarly, we use R_s^J to denote a hinge (rotational) joint transformer for joint s (implicitly parameterized by an axis and center of rotation). We use $R_s^J(\mathbf{j}_s^m, \theta)$ to denote articulating the source joint $\mathbf{j}_s$'s moving part $\mathbf{j}_s^m$ with rotation angle θ.

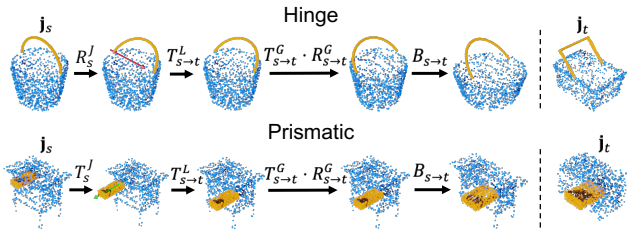

Figure 3: How our transformation models for hinge and prismatic motions transform one joint $\mathbf{j}_s$ to another $\mathbf{j}_t$.

Additional pose transformations. In addition to kinematic motion, we may need additional pose transformations to align the source and target joint. We use $T_{s\to t}^G$ and $R_{s\to t}^G$ to denote a translation and a rotation about world-up that are applied to the entire joint $\mathbf{j}_s$ to help globally align it with the target joint $\mathbf{j}_t$. The moving part sometimes also needs additional degrees of freedom relative to the base part. For example, to transform the bottom drawer in a cabinet into the top drawer, we need an additional upward translation. For this, we also define a local alignment translation $T_{s\to t}^L$. For translational joints, to ensure that this local alignment translation cannot become redundant with joint motion, it is projected into the plane perpendicular to the axis of translation.

Geometric deformers. To transform a joint to a geometrically different joint, we must also permit some deformation of joint geometries, in addition to pose variation. $B_{s\to t}$ denotes a *box deformer*, whose degrees of freedom are the scales of the 6 faces of a part's bounding box, which allows adjusting the bulk shape of a part. This box is aligned with the part's local coordinate frame, so its deformations are independent of the part's pose.

Final transformation model. Given the functions defined above, we can now define the complete, optimizable transformation function that takes the moving part of a joint s to that of another joint t via a prismatic motion:

$$\mathbf{j}_{s\to t}^m = B_{s\to t}^m(\{T^G \cdot R^G \cdot T^L\}_{s\to t} \cdot T_s^J(\mathbf{j}_s^m, d_{s\to t})) \quad (2)$$

Similarly, for a hinge motion:

$$\mathbf{j}_{s\to t}^m = B_{s\to t}^m(\{T^G \cdot R^G \cdot T^L\}_{s\to t} \cdot R_s^J(\mathbf{j}_s^m, \theta_{s\to t})) \quad (3)$$

For both types of motion, the base part transforms as:

$$\mathbf{j}_{s\to t}^b = B_{s\to t}^b(\{T^G \cdot R^G\}_{s\to t} \cdot \mathbf{j}_s^b) \quad (4)$$

Figure 3 illustrates these transformation sequences.

5.2 Loss functions

To produce plausible transformations from a source joint $\mathbf{j}_s$ to a target $\mathbf{j}_t$, we optimize the parameters of the above transformation model with respect to several loss functions:

Reconstruction loss. First and foremost, the transformed source joint must approximately reconstruct the target joint:

$$\mathcal{L}_{s\to t}^{\text{recon}} = \frac{D_{\text{chamfer}}(\mathbf{j}_{s\to t}^m, \mathbf{j}_t^m)}{\text{diag}(\mathbf{j}_{s\to t}^m)} + \frac{D_{\text{chamfer}}(\mathbf{j}_{s\to t}^b, \mathbf{j}_t^b)}{\text{diag}(\mathbf{j}_{s\to t}^b)} \quad (5)$$

where D_{chamfer} denotes bidirectional chamfer distance between two point-sampled parts, and diag$(\cdot)$ is the diagonal length of a part (to normalize these distances).

Many settings of the transformation model parameters will give low reconstruction loss, most of which do not correspond to valid kinematic motions. Thus, we design priors to encourage the optimization to find plausible motions:

Small joint motion penalty. The optimization can find spurious motions by setting the amount of motion (i.e. the displacement d or rotation angle θ) to a small value. Thus, we propose to penalizes motions smaller than a threshold τ. For prismatic joints,

$$\mathcal{L}^{\text{joint}}_{s\to t} = w_{\text{joint}} \cdot \max(\tau_d - |d_{s\to t}|, 0) \tag{6}$$

and for hinge joints,

$$\mathcal{L}^{\text{joint}}_{s\to t} = w_{\text{joint}} \cdot \max(\tau_\theta - |\theta_{s\to t}|, 0) \tag{7}$$

This term is particularly important for finding correct motions for parts whose geometry does not visibly change over the course of articulation (e.g. a spinning wheel).

Large alignment transform penalties. While the global alignment transforms $T^G_{s\to t}$, $R^G_{s\to t}$ and local alignment transform $T^L_{s\to t}$ are often necessary, the optimization should try to perform as much of the transformation as possible using the joint motion. To this end, we introduce loss terms to penalize the magnitude of the alignment transforms:

$$\mathcal{L}^{\text{align}}_{s\to t} = w^G_{\text{align}}|R^G_{s\to t}| + w^L_{\text{align}}|T^L_{s\to t}| \tag{8}$$

Large deformation penalty. Similarly, we must also restrict the box deformers $B_{s\to t}$ from being responsible for more of the transformation than is necessary:

$$\mathcal{L}^{\text{deform}}_{s\to t} = w_{\text{deform}}(|B^m_{s\to t}| + |B^b_{s\to t}|) \tag{9}$$

where $|B|$ is the sum of all box face absolute displacements.

Collision penalty. For a joint motion to be physically valid, the moving part must not collide with the base part. We define a collision penalty loss $\mathcal{L}^{\text{collide}}_{s\to t}$ which enforces this property. We sample n equally-spaced values along the interval $[0, d_{s\to t}]$ for prismatic joints ($[0, \theta_{s\to t}]$ for hinge joints) and transform the moving part to that pose. The collision penalty for each pose is the mean penetration distance of each point in the base part point cloud to the moving part. The overall collision penalty is then the mean of these per-timestep penalties.

For a hinge joint:

$$\mathcal{L}^{\text{collide}}_{s\to t} = \frac{w_{\text{collide}}}{n|\mathbf{j}^b_s|}\sum_{i=1}^{n}\sum_{\mathbf{x}\in\mathbf{j}^b_s}\max(0, d_0 - \text{sdf}(\mathbf{x}, R^J_s(\mathbf{j}^m_s, i\cdot\theta_{s\to t}))) \tag{10}$$

where $\text{sdf}(\mathbf{x}, \mathbf{j}^m)$ is the signed distance from the point $\mathbf{x}$ to the minimum volume bounding box of the moving part $\mathbf{j}^m$ (negative signed distance $\rightarrow$ the point is inside the box) and d_0 is the largest penetration distance of any point $\mathbf{x} \in \mathbf{j}^b_s$ at $i = 0$ (i.e. some joints may initially have a small degree of interpenetration, which we should not penalize). The loss for a prismatic joint is defined analogously (replace $R^J_s(\mathbf{j}^m_s, i\cdot\theta_{s\to t})$ with $T^J_s(\mathbf{j}^m_s, i\cdot d_{s\to t})$).

Detachment penalty. In addition to not colliding with the base part, a moving part should also not *detach* from its base part over the course of its motion. For hinge joints, we minimize the distance between the nearest 10 original contact points in the rotated moving part $R^J_s(\mathbf{j}^m_s, i\cdot\theta_{s\to t})$ (denoted as $\mathcal{S}_{\mathbf{j}^m_s}$) and the nearest 10 original contact points in $\mathbf{j}^b_s$ (denoted as $\mathcal{S}_{\mathbf{j}^b_s}$):

$$\mathcal{L}^{\text{detach}}_{s\to t} = \frac{w_{\text{detach}}}{n}\sum_{i=1}^{n}\max(0, \sum_{\mathbf{x},\mathbf{y}\in\mathcal{S}_{\mathbf{j}^m_s},\mathcal{S}_{\mathbf{j}^b_s}} ||\mathbf{x}-\mathbf{y}||_2 - r) \tag{11}$$

where $r = 0.01$ is a distance penalty threshold. In addition to the above term, we also add a term which penalizes the center of rotation from non-physically falling outside of the moving part's bounding box.

For prismatic joints, we penalize the distance between the moving part $\mathbf{j}^m_s$ and the 50 points on the base part $\mathbf{j}^b_s$ which are closest to it (denoted as $\mathcal{N}_{\mathbf{j}_s}$):

$$\mathcal{L}^{\text{detach}}_{s\to t} = \frac{w_{\text{detach}}}{n|\mathcal{N}_{\mathbf{j}_s}|}\sum_{i=1}^{n}\sum_{\mathbf{x}\in\mathcal{N}_{\mathbf{j}_s}}\max(0, \text{sdf}(\mathbf{x}, R^J_s(\mathbf{j}^m_s, i\cdot\theta_{s\to t}))) \tag{12}$$

Hyperparameters. We empirically define two sets of values for the various loss weights w: one set for optimizing hinge joints; one set for optimizing prismatic joints. These weights are kept constant across all shape categories in all of our experiments. Values for weights and other hyperparameters can be found in supplemental.

5.3 Optimization procedure

To optimize the parameters of the transformation model, we combine all the above losses together into one:

$$\mathcal{L}_{s\to t} = \mathcal{L}^{\text{recon}}_{s\to t} + \mathcal{L}^{\text{joint}}_{s\to t} + \mathcal{L}^{\text{align}}_{s\to t} + \mathcal{L}^{\text{deform}}_{s\to t} + \mathcal{L}^{\text{collide}}_{s\to t} + \mathcal{L}^{\text{detach}}_{s\to t}$$

$$\mathcal{L} = \frac{1}{kN}\sum_{s=1}^{N}\sum_{t\in\mathcal{T}_s}\mathcal{L}_{s\to t} \tag{13}$$

We minimize $\mathcal{L}$ using the Adam optimizer.

Multiple initializations. As this optimization problem is non-convex, we solve it multiple times with different initializations to avoid local minima. For hinge joints, we use the 3 axes of the minimum volume bounding box of the moving part as the initial rotation axes. We use the centroid of the moving part and the centers of 4/6 of its bounding box faces as the initial rotation centers (the four with the largest distance to the part centroid). For prismatic joints, we use the longest 2 axes of the minimum volume bounding box of the moving part as the initial axes. This results in 2 initializations for prismatic joints and 15 (3×5) initializations for hinge joints; we choose the one which gives the smallest $\mathcal{L}_{s\to t}$.

Axis post-processing. The optimization often gives motion axes that noisily oscillate around a good solution. Thus, we use a post-processing step to 'snap' the axes. We check if the axis is close to any of the three world axes or the three principal axes of the moving part or base part. If the dot product of the optimized axis and any of these axes is > 0.975, we snap the axis to it.

Determining range of motion. Finally, given optimized motion parameters for a joint $\mathbf{j}_s$, we estimate its range of motion. For this, we sample 16 nearby target joints from the embedding space (using the sampling procedure from Section 4) and optimize for a transformation from $\mathbf{j}_s$ to each of these targets, holding motion parameters fixed and only optimizing pose, deformation, and alignment transforms. We estimate the joint's motion range as the range of poses for all of these target joints whose post-optimization $\mathcal{L}^{\text{recon}}$ is less than a threshold (see supplemental). We call these joints the 'valid' targets for a motion. This results in a motion range relative to the initial pose of $\mathbf{j}_s^m$ (i.e. for a hinge joint, $\theta = 0$ is the initial pose).

6 DETERMINING JOINT MOTION TYPE

After multiple iterations of optimization, we are left with multiple potential hinge and prismatic motions $\tilde{\mathbf{m}}$ for each candidate joint $\mathbf{j}$. In this section, we describe our procedure for determining (a) which of these motions is the best for each joint and (b) whether a part moves at all or should instead be labeled as static.

Selecting the best candidate motion. We start by considering the set of 'valid' target joints for each potential motion $\tilde{\mathbf{m}}$, as described above. Intuitively, a motion is more likely to be correct if (a) it allows the source joint to reach more valid targets, and (b) the targets exhibit a wider range of poses. Let $N_{\text{valid}}^{\tilde{\mathbf{m}}}$ be the number of valid target joints for a motion, which addresses (a). For (b), we discretize the predicted range of motion into a set of equally-sized bins and let $N_{\text{bin}}^{\tilde{\mathbf{m}}}$ be the number of these bins which contain the pose of at least one of the valid target joints. We then define our confidence in this potential motion as:

$$C^{\tilde{\mathbf{m}}} = \lambda_1 N_{\text{valid}}^{\tilde{\mathbf{m}}} + \lambda_2 N_{\text{bin}}^{\tilde{\mathbf{m}}} \tag{14}$$

We select whichever potential motion $\tilde{\mathbf{m}}$ has the highest confidence as the best motion $\mathbf{m}^*$. See supplemental for the values of λ_1, λ_2.

Distinguishing moving vs. static parts. To identify whether a part $\mathbf{p}$ should be movable or static, we look at the number of candidate motions $\tilde{\mathbf{m}}$ in which it is is used as a base or moving part, as well as our confidence in those motions. Intuitively, a part used as a moving part in many high-confidence motions is more likely to be movable; a part used as a base part in many high-confidence motions is more likely to be static. Let $\mathcal{M}_{\text{mov}}^{\mathbf{p}}$ and $\mathcal{M}_{\text{base}}^{\mathbf{p}}$ be the set of candidate motions in which part $\mathbf{p}$ is used as a moving or base part, respectively. Our confidences that this part is movable or static are:

$$\begin{aligned} C_{\text{mov}}^{\mathbf{p}} &= \frac{\lambda_3}{|\mathcal{M}_{\text{mov}}^{\mathbf{p}}|} \sum_{\tilde{\mathbf{m}} \in \mathcal{M}_{\text{mov}}^{\mathbf{p}}} C^{\tilde{\mathbf{m}}} + \lambda_4 |\mathcal{M}_{\text{mov}}^{\mathbf{p}}| \\ C_{\text{static}}^{\mathbf{p}} &= \frac{\lambda_3}{|\mathcal{M}_{\text{base}}^{\mathbf{p}}|} \sum_{\tilde{\mathbf{m}} \in \mathcal{M}_{\text{base}}^{\mathbf{p}}} C^{\tilde{\mathbf{m}}} + \lambda_4 |\mathcal{M}_{\text{base}}^{\mathbf{p}}| \end{aligned} \tag{15}$$

If $C_{\text{static}}^{\mathbf{p}} / C_{\text{mov}}^{\mathbf{p}}$ is greater than a threshold, the part is labeled static. We also always label the largest part of every shape as static. See supplemental for λ_3, λ_4, and threshold values.

Table 1: Comparing the performance of BaseNet (with and without pre-alignment) vs. Our method on predicting the motion attributes of shapes in PartNet-Mobility.

Method	Type Acc↑	Axis Err (°)↓	Center Err (%)↓	Range IoU↑
BaseNet	0.87	30.71	24.28	0.44
BaseNet + align	0.93	13.24	22.18	0.45
Ours	0.84	**6.09**	**9.12**	0.46

Table 2: Comparing the performance of Shape2Motion (with and without pre-alignment) vs. Our method on predicting the motion attributes of shapes in PartNet-Mobility. S2M does not handle static motion type and does not predict motion range.

Method	Type(w/o static) Acc↑	Axis Err (°)↓	Center Err (%)↓
S2M	0.91	33.65	25.58
S2M + align	0.92	15.80	16.67
Ours	0.80	**6.09**	**9.12**

7 RESULTS

Here we evaluate our system's ability to discover accurate kinematic motions without supervision.

Dataset. We evaluate our method on PartNet-Mobility, a dataset of part-segmented 3D shapes annotated with ground-truth kinematic articulations [Xiang et al. 2020]. We run experiments on 18 categories of objects: Box, Bucket, Clock, Door, Fan, Faucet, Folding Chair, Knife, Laptop, Pliers, Refrigerator, Scissors, Stapler, StorageFurniture, Table, Trash Can, USB, and Window. These categories were chosen to give good coverage of the different types of motions which occur in PartNet-Mobility. We perform various filtering steps on this data: joints with invalid motion ranges, moving parts that are extremely small relative to the overall shape (which are not well-represented in point cloud from), etc. See supplemental for details. Our final evaluation dataset contains 753 shapes with 1939 parts.

Our method assumes that the input shapes exhibit pose variations. Some shape categories in PartNet-Mobility have this property; others have all shapes in a neutral pose. We normalize pose variation across categories by randomly sampling a pose from within each movable part's range of motion. In some of our experiments, we examine the impact that the amount of pose variation has on our method. Also, all shapes in a PartNet-Mobility category are aligned to a common coordinate frame, but not all in-the-wild shape collections exhibit this property. We randomly rotate each shape about its up vector.

Comparison to baselines. There is no existing prior work which performs unsupervised kinematic motion detection; thus, we compare to existing supervised approaches. We stress that these approaches require training data, whereas ours can be applied on categories of shapes that have never been seen before. We compare our method to two supervised baselines:

- *BaseNet:* This network takes in a point cloud of a shape, where each point carries an extra one-hot dimension indicating whether it is a member of the part for which motion should be predicted. This point cloud is passed through a PointNet encoder, the output

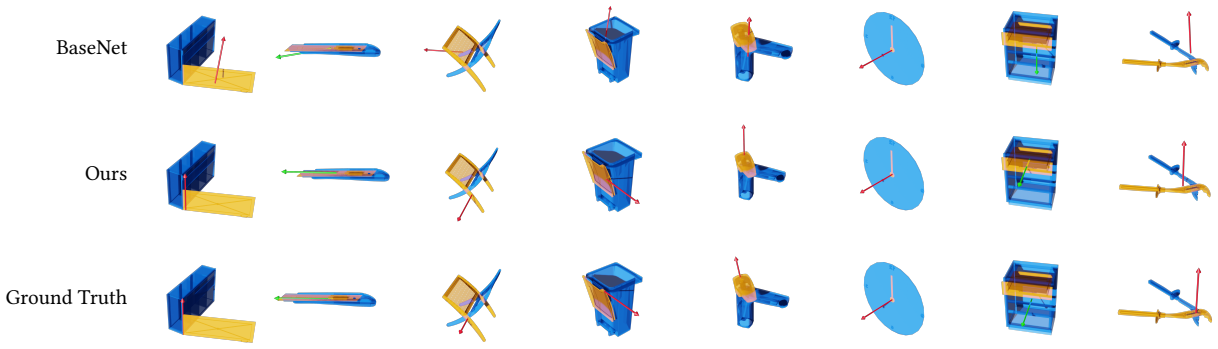

Figure 4: Qualitative examples of our method vs. BaseNet on predicting motion parameters for PartNet-Mobility.

of which is fed into four separate fully connected branches which predict motion type (hinge, prismatic, static), axis ($\mathbb{R}^3$), center of rotation ($\mathbb{R}^3$), and range ([min, max] $\in \mathbb{R}^2$). The entire network is trained jointly, with cross entropy loss for the motion type branch and MSE loss for the other branches.

- *Shape2Motion (S2M):* A network which jointly predicts part segmentations and part motions for point clouds [Wang et al. 2019]. For fair comparison with our method, we modify the network to take ground truth part segmentations and only predict motions.

For both supervised methods, we evaluate them with and without a pre-alignment step, in which objects are rotated above the up axis to roughly align them via chamfer distance.

See supplemental for more details on these baselines. We split our filtered collection of joints 60%/40% into train/test sets. The supervised baselines are trained on the train set, our method is run on all joints, and all methods are evaluated on the test set.

We evaluate test set motion predictions with these metrics:

- *Motion type accuracy (Type Acc)*: percentage of joints whose motion type (static, prismatic, hinge) is correctly predicted. Since Shape2Motion does not handle 'static' parts, we omit that label for its training and evaluation.
- *Axis angular error (Axis Err)*: mean difference (in degrees) between predicted axis directions and their ground-truth values.
- *Rotation center error (Center Err)*: mean distance (in percentage of the part's bounding box diagonal length) between predicted centers of rotation and ground truth rotation axes.
- *Range of motion accuracy (Range IoU)*: mean intersection over union between predicted and ground truth ranges of motion.

Table 1 and Table 2 show quantitative results of this comparison; a breakdown by category is in supplemental. Our method has complementary strengths to BaseNet and Shape2Motion: These two supervised methods are better at predicting motion type; ours is better at motion parameters. BaseNet's and Shape2Motion's higher type accuracy is not surprising: ternary motion type is the easiest quantity for a network to learn to predict (given the limited training data, it is harder to learn to regress continuous motion parameters). Motion type is also the easiest information for a human annotator to provide. A hybrid approach might be best in practice: combining weakly-supervised motion type labeling with our approach for predicting motion parameters. Shape2Motion doesn't outperform

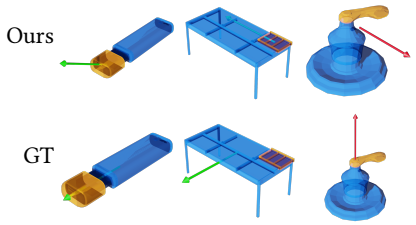

Figure 5: Typical failure cases (USB, Table, Faucet). The Table and Faucet examples disagree with ground-truth annotations but are still plausible.

BaseNet by a large margin, which we hypothesize is due to some combination of (1) the network being complicated and requiring more training data, (2) training on ground truth segmentations rather than jointly inferring them resulting in a weaker learned representation.

Figure 4 qualitatively compares our method to BaseNet. Our physically-inspired approach degrades more gracefully than the learning-based approach, which can produce nonsensical outputs due to insufficient training data. Figure 5 additionally shows some of the ways our method gracefully fails. For the table and faucet joints shown, the motions found are different from PartNet-Mobility's ground truth but are nonetheless plausible: the table drawer could slide side-to-side rather than front-to-back; the faucet handle could rotate vertically rather than / as well as laterally. The USB failure may be due to incorrect grouping of joints, not enough similar joints to group, or suboptimal loss weight hyperparameters.

Sensitivity to amount of pose variation. Table 3 shows how our method performs on Doors when the amount of pose variation is increased (see supplemental for details) Results are poor with no variation, but even a little increases performance dramatically.

Running time. Our method iterates complex optimizations on multiple joint groups, so it can be time-consuming: one iteration on a set of 100 joints with target joint number $k = 5$ takes our PyTorch implementation about 1 hour on an 8-core Intel i9 machine with 32 GB RAM and a NVIDIA RTX 2080Ti GPU. However, compute time can be small price to pay to avoid human annotation time.

Table 3: Analyzing how the performance of our method on Doors varies with amount of pose variation in the input shape collection. The number of target joints $k = 3$ for this experiment.

Pose variation level	Type Acc↑	Axis Err (°)↓	Center Err (%)↓	Range IoU↑
0	0.54	39.21	12.70	0.03
1	0.89	0.02	16.33	0.25
2	0.91	0.03	10.09	0.42
3	0.91	0.09	10.48	0.49
4	0.91	0.09	4.59	0.37
5	0.91	0.00	6.34	0.43

Additional results. The supplemental contains additional qualitative results, a study on sensitivity to number of target joints, an ablation study on model components, and a visualization of the learned joint embedding space.

8 CONCLUSION

In this paper, we presented an unsupervised approach for discovering part motions in part-segmented 3D shape collections. Our approach is based on *category closure*: a valid articulation of a shape's parts should not change the semantic category of that shape. We operationalize this insight via an algorithm that finds motion parameters for a joints that transforms into other joints from the same category. Our approach successfully rediscovers a large percentage of motions in the PartNet-Mobility dataset, often outperforming a supervised motion prediction network.

Our system has some limitations. It cannot handle moving parts whose size is a small fraction of their base parts, due to point cloud resolution limits. More fundamentally, our method assumes that the input shape collection contains similar joints in different poses. This is often true (e.g. lamp arms, swivel chair bases), but some shapes are typically modeled in a canonical pose (e.g. cabinet doors are usually modeled as closed). These shapes may pose a challenge for our method (or any unsupervised method).

Finally, our method assumes the input part segmentation is fairly consistent, e.g. it would not perform well on cabinets if each cabinet door was broken into a different number of segments. In the future, we would like to extend our method to handle such data by developing a system for proposing ways to group different part fragments. Combined with an automatic shape over-segmentation method, this would allow our method to discover shape parts as well as their motions without any supervision.

ACKNOWLEDGMENTS

This work was funded in part by NSF Award #1941808. Daniel Ritchie is an advisor to Geopipe and owns equity in the company. Geopipe is a start-up that is developing 3D technology to build immersive virtual copies of the real world with applications in various fields, including games and architecture. Srinath Sridhar was supported by a Google Research Scholar Award.

REFERENCES

Ben Abbatematteo, Stefanie Tellex, and George Konidaris. 2019. Learning to generalize kinematic models to novel objects. In *Proceedings of the 3rd Conference on Robot Learning*.

Thiemo Alldieck, Marcus Magnor, Weipeng Xu, Christian Theobalt, and Gerard Pons-Moll. 2018. Detailed human avatars from monocular video. In *2018 International Conference on 3D Vision (3DV)*. IEEE, 98–109.

Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*. Springer, 640–653.

CGTrader. 2020. CGTrader - 3D Models for VR / AR and CG Projects. https://www.cgtrader.com/. Accessed: 2020-05-22.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *1512.03012* (2015).

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied Question Answering. In *CVPR*.

Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S Sukhatme. 2015. Active articulation model estimation through interactive perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3305–3312.

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

Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Ariel Shamir, Hao Zhang, and Hui Huang. 2017b. Learning to predict part mobility from a single static snapshot. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

Zeng Huang, Yuanlu Xu, Christoph Lassner, Hao Li, and Tony Tung. 2020. Arch: Animatable reconstruction of clothed humans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3093–3102.

Adobe Systems Inc. 2020. Royalty-free 3D assets to enhance your projects. https://stock.adobe.com/3d-assets. Accessed: 2020-10-20.

Trimble Inc. 2021. 3D Warehouse. https://3dwarehouse.sketchup.com/.

Hanbyul Joo, Tomas Simon, and Yaser Sheikh. 2018. Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8320–8329.

Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. 2018. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7122–7131.

Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. 2021. Unsupervised Pose-Aware Part Decomposition for 3D Articulated Objects. (2021).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

Eric Kolve, Roozbeh Mottaghi, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An Interactive 3D Environment for Visual AI. *CoRR* arXiv:1712.05474 (2017).

Xiaolong Li, He Wang, Li Yi, Leonidas J Guibas, A Lynn Abbott, and Shuran Song. 2020. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3706–3715.

Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. 2019. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

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 (TOG)* 36, 4 (2017), 1–14.

Niloy J Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. 2010. Illustrating how mechanical assemblies work. *ACM Transactions on Graphics-TOG* 29, 4 (2010), 58.

Jiteng Mu, Weichao Qiu, Adam Kortylewski, Alan Yuille, Nuno Vasconcelos, and Xiaolong Wang. 2021. A-SDF: Learning Disentangled Signed Distance Functions for Articulated Shape Representation. *arXiv preprint arXiv:2104.07645* (2021).

Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 49–59.

Richard M Murray, Zexiang Li, and S Shankar Sastry. 2017. *A mathematical introduction to robotic manipulation*. CRC press.

Sudeep Pillai, Matthew R Walter, and Seth Teller. 2015. Learning articulated motions from visual demonstration. *arXiv preprint arXiv:1502.01659* (2015).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 652–660.

Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision (ECCV) (LNCS, Vol. 9906)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, 102–118.

Manolis Savva, Angel X. Chang, Alexey Dosovitskiy, Thomas Funkhouser, and Vladlen Koltun. 2017. MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments. *arXiv:1712.03931* (2017).

Andrei Sharf, Hui Huang, Cheng Liang, Jiapei Zhang, Baoquan Chen, and Minglun Gong. 2014. Mobility-trees for indoor scenes manipulation. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 2–14.

Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. 2011. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*. Ieee, 1297–1304.

Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. 2011. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research* 41 (2011), 477–526.

Turbosquid. 2020. 3D Models for Professionals. https://turbosquid.com. Accessed: 2020-10-20.

Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. 2019. Shape2Motion: Joint Analysis of Motion Parts and Attributes from 3D Shapes. In *CVPR*. 8876–8884.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. 2020. SAPIEN: A SimulAted Part-based Interactive ENvironment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Weiwei Xu, Jun Wang, KangKang Yin, Kun Zhou, Michiel Van De Panne, Falai Chen, and Baining Guo. 2009. Joint-aware manipulation of deformable models. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–9.

Xianghao Xu, David Charatan, Sonia Raychaudhuri, Hanxiao Jiang, Mae Heitmann, Vladimir Kim, Siddhartha Chaudhuri, Manolis Savva, Angel X. Chang, and Daniel Ritchie. 2020. Motion Annotation Programs: A Scalable Approach to Annotating Kinematic Articulations in Large 3D Shape Collections. In *3DV*.

Claudia Yan, Dipendra Kumar Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. CHALET: Cornell House Agent Learning Environment. *CoRR* arXiv:1801.07357 (2018).

Jingyu Yan and Marc Pollefeys. 2006. Automatic kinematic chain building from feature trajectories of articulated objects. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1. IEEE, 712–719.

Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. 2019. RPM-Net: recurrent prediction of motion and parts from point cloud. *Proceedings of SIGGRAPH Asia* 38, 6 (2019), 240.

Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, and Leonidas Guibas Hao S and. 2019. Deep Part Induction from Articulated Object Pairs. *Proceedings of SIGGRAPH Asia* 37, 6 (2019), 209.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A Scalable Active Framework for Region Annotation in 3D Shape Collections. *SIGGRAPH Asia* (2016).

Ge Zhang, Or Litany, Srinath Sridhar, and Leonidas Guibas. 2021. StrobeNet: Category-Level Multiview Reconstruction of Articulated Objects. *arXiv preprint arXiv:2105.08016* (2021).

Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. 2017. Physically-Based Rendering for Indoor Scene Understanding Using Convolutional Neural Networks. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).