Category-Level Articulated Object Pose Estimation

Xiaolong Li[1] He Wang[2] Li Yi[2] Leonidas Guibas [2] A. Lynn Abbott [1] Shuran Song[3]
[1]Virginia Tech [2]Stanford University [3]Columbia University

Abstract

This paper addresses the task of category-level pose estimation for articulated objects from a single depth image. We present a novel category-level approach that correctly accommodates object instances not previously seen during training. A key aspect of the work is the new Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH), which represents the different articulated objects for a given object category. This approach not only provides the canonical representation of each rigid part, but also normalizes the joint parameters and joint states. We developed a deep network based on PointNet++ that is capable of predicting an A-NCSH representation for unseen object instances from single depth input. The predicted A-NCSH representation is then used for global pose optimization using kinematic constraints. We demonstrate that constraints associated with joints in the kinematic chain lead to improved performance in estimating pose and relative scale for each part of the object. We also demonstrate that the approach can tolerate cases of severe occlusion in the observed data. Project webpage: articulated-pose.github.io

1. Introduction

Our environment is populated with articulated objects, ranging from furniture such as cabinets or ovens to small tabletop objects such as laptops or eyeglasses. Effectively interacting with these objects requires a detailed understanding of their articulation states and part-level poses. Such understanding is beyond the scope of typical 6D pose estimation algorithms, which have been designed for rigid objects [28, 23, 22, 26]. Algorithms that do consider object articulations [11, 12, 10, 14] often require the exact object CAD model and the associated joint parameters at test time, preventing them from generalizing to new object instances.

In this paper, we focus on the task of category-level pose estimation for articulated objects from a single depth image – a task that aims at producing the detailed per-part pose and scale, joint parameters, and joint states of a novel articulated object instance from a known category. An overview

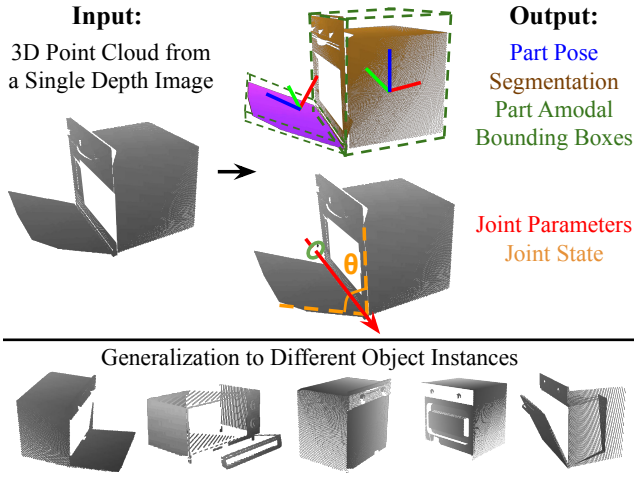


Figure 1. **Category-level articulated object pose estimation.** Given a single depth image of a novel articulated object from a known category, the goal of the algorithm is to estimate detailed per-part poses, segmentation, amodal bounding boxes, as well as joint parameters and joint states of the object.

is shown in Figure 1. To achieve this goal, several major challenges need to be addressed:

First, to handle a new object instance without a precise 3D CAD model, the algorithm needs to define a shared object representation that can accommodate different object instances within a given category. The representation needs to be able to generalize to different variations of part geometry, joint parameters, and self-occlusion patterns.

Second, in contrast to rigid objects, the pose of articulated objects naturally requires a much higher degree of freedom. Depending on the length of kinematic chain, the degrees of freedom in the target pose representation will vary significantly. It becomes a challenging problem to accurately estimate pose in such a high-dimensional space while being faithful to physical constraints.

Third, various joint types present different articulation characteristics along with different physical constraints and priors. We are particularly interested in the two most common joint types, revolute joints that cause rotational motion (*e.g.*, door hinges), and prismatic joints that allow translational movement (*e.g.*, drawers in a cabinet). Designing a framework that can consider and leverage both

joint types effectively is still an open research problem.

To address the first representation challenge, we propose a shared category-level representation for different articulated object instances, which we call Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH). Concretely, A-NCSH represents different articulated objects in a "canonical" space, which normalizes part shapes, joint parameters, and joint states. The pose of each rigid part and the state of each joint in the depth image can then be defined with respect to their normalized states in A-NCSH. This representation provides us a way to define category-level pose even for unseen object instances regardless of intra-class variance.

To address the second pose estimation challenge, which involves high dimensionality, we segment objects into rigid parts and estimate the pose on a per-part basis. However, separate per-part pose estimation could easily lead to physically impossible solutions since joint constraints are not considered and the estimation may not conform with the actual degrees of freedom. To cope with this issue, we treat joints as "first-class citizens." Our approach estimates joint parameters, and leverages the induced kinematic priors to constrain the pose of each part. We formulate articulated pose fitting from the A-NCSH space as a combined optimization problem, taking both rigid part pose fitting and joint constraints into consideration.

To handle both revolute and prismatic joints, each joint is associated with a unique index, and our system explicitly predicts joint index and the corresponding joint parameters. We model the constraints introduced by each type of joint mathematically with the predicted parameters, and selectively use these constraints during the pose fitting stage according to the associated joint type. Although we focus only on revolute and prismatic joints in our work, the approach is sufficiently general to support other types of joints.

In summary, the primary contribution of our paper is the formulation of a unified framework for the task of category-level articulated pose estimation that is able to handle previously unseen object instances and multiple articulation types. In support of this framework, we designed:

- A new category-level representation for articulated objects – Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH).
- A PointNet++ based neural network that is capable of predicting A-NCSH representations for unseen articulated object instances from single depth input.
- A strategy for global optimization that leverages kinematic constraints along with all the information in the A-NCSH representation to improve the overall pose estimation accuracy.

Our experiments demonstrate that the A-NCSH representation and the global optimization using A-NCSH prediction lead to improved performance in both part pose prediction and joint parameter estimation.

2. Related Work

This section summarizes related work on pose estimation for rigid and articulated objects.

Rigid object pose estimation. Classically, the goal of pose estimation is to infer an object's 6D pose (3D rotation and 3D location) relative to a given reference frame. Most previous work has focused on estimating instance-level pose by assuming that exact 3D CAD models are available. For example, traditional algorithms such as iterative closest point (ICP) [3] perform template matching by aligning the CAD model with an observed 3D point cloud. Another family of approaches aim to regress the object coordinates onto its CAD model for each observed object pixel, and then use voting to solve for object pose [4, 5]. These approaches are limited by the need to have exact CAD models for particular object instances.

Category-level pose estimation aims to infer an object's pose and scale relative to a category-specific canonical representation. Recently, Wang *et al*. [26] extended the object coordinate based approach to perform category-level pose estimation. The key idea behind the intra-category generalization is to regress the coordinates within a Normalized Object Coordinate Space (NOCS), where the sizes are normalized and the orientations are aligned for objects in a given category. Whereas the work by [26] focuses on pose and size estimation for rigid objects, the work presented here extends the NOCS concept to accommodate articulated objects at both part and object level. In addition to pose, our work also infers joint information and addresses particular problems related to occlusion.

Articulated object pose estimation. Most algorithms that attempt pose estimation for articulated objects assume that instance-level information is available. The approaches often use CAD models for particular instances along with known kinematic parameters to constrain the search space and to recover the pose separately for different parts [16, 7]. Michel *et al*. [16] use a random forest to vote for pose parameters on canonical body parts for each point in a depth image, followed by a variant of the Kabsch algorithm to estimate joint parameters using RANSAC-based energy minimization. Desingh *et al*. [7] adopted a generative approach using a Markov Random Field formulation, factoring the state as individual parts constrained by their articulation parameters. However, these approaches only consider known object instances and cannot handle different part and kinematic variations.

Another line of work relies on active manipulation of an object to infer its articulation pattern [11, 12, 10, 14, 29]. For example, Katz *et al*. [12], use a robot manipulator

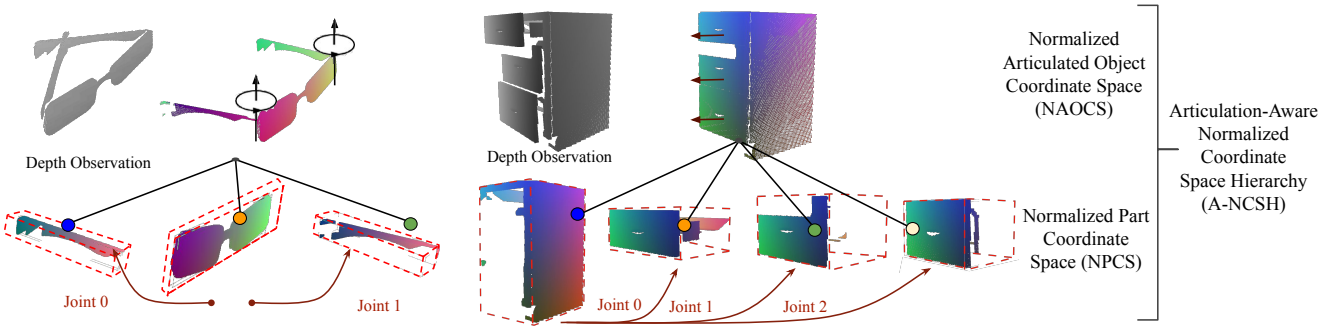


Figure 2. **Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH)** is a shared object representation for different object instances in a given category. It consists of a two-level hierarchy: at the leaf level, it uses Normalized Part Coordinate Space (NPCS) to represent each individual part; at the root level, it uses Normalized Articulated Object Coordinate Space (NAOCS), which is a single coordinate space that transforms all the NPCS-based parts to represent a complete articulated object in a pre-defined rest state. Here we show two examples of A-NCSH representation, where each point is colored according to its coordinate location in the corresponding representation (NAOCS or NPCS).

to interact with articulated objects as RGB-D videos are recorded. Then the 3D points are clustered into rigid parts according to their motion. Although these approaches could perform pose estimation for unknown objects, they require the input to be a sequence of images that observe an object's different articulation states, whereas our approach is able to perform the task using a single depth observation.

Human body and hand pose estimation. Two specific articulated classes have gained considerable attention recently: the human body and the human hand. For human pose estimation, approaches have been developed using end-to-end networks to predict 3D joint locations directly [15, 21, 17], using dense correspondence maps between 2D images and 3D surface models [2], or estimating full 3D shape through 2D supervision [13, 18]. Techniques for hand pose estimation (*e.g.*, [25, 9]) often start with per-pixel estimates, such as pixel-level segmentation, coordinate regression or joint voting. The pixel-level prediction is then aggregated to infer 3D joint coordinates. Approaches for both body and hand pose estimation are often specifically customized for those object types, relying on a fixed skeletal model with class-dependent variability (*e.g.*, expected joint lengths) and strong shape priors (*e.g.*, using parametric body shape model for low-dimensional parameterization). Also, such hand/body approaches accommodate only revolute joints and do not generalize well to other object types. In contrast, our algorithm is able to handle general articulated objects with any kinematic chain topology, allowing both revolute joints and prismatic joints.

3. Problem Statement

The input to the system is a 3D point cloud $P = \{\mathbf{p}_i \in \mathbb{R}^3 | \, i = 1, ..., N\}$ representing an unknown object instance from a known category, where N denotes the number of points. The goal is to segment the point cloud into rigid moving parts $\{S^{(j)}\}$, recover the 3D rotation, 3D translation, and size for each part $\{R^{(j)}, \mathbf{t}^{(j)}, s^{(j)}\}$, and predict the joints with their parameters $\{\phi_k\}$ and states $\{\theta_k\}$. We consider two types of joint in this work, 1D revolute joints (*e.g.*, door hinges) and 1D prismatic joints (*e.g.*, drawers for a cabinet). For a revolute joint, the joint parameters include the direction of the rotation axis $\mathbf{u}_k^{(r)}$ as well as a pivot point on the rotation axis $\mathbf{q}_k$. The state is defined as the relative rotation angle between the two connected parts, as compared with a pre-defined rest state. For a prismatic joint, the joint parameters are simply the direction of the translation axis $\mathbf{u}_k^{(t)}$, and the joint state is defined as the relative translation distance between the two connected parts compared with a pre-defined rest state. This representation scheme not only encodes many common articulated objects effectively and compactly, but also suggests possible articulations.

4. Method

It is challenging to define part poses and joint states for unseen object instances with potentially large geometric variations and articulation differences. To tackle this problem, we introduce Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH), a shared object representation for different object instances in a given category. An overview is given in Figure 2. We will provide details on A-NCSH and how it allows defining part poses and joints for unseen objects in Sec. 4.1. We then present a deep neural network capable of predicting the A-NCSH representation in Sec. 4.2. Last, Sec. 4.3 describes how the A-NCSH representation is used within a voting scheme to jointly optimize part poses and joint states with explicit kinematic chain constraints.

4.1. A-NCSH Representation

Our A-NCSH representation is inspired by and closely related to Normalized Object Coordinate Space (NOCS) [26], which we will briefly review first. NOCS is

defined as a 3D space contained within a unit cube, *i.e.*, $\{x, y, z\} \in [0, 1]$, and was introduced in [26] to estimate the category-level 6D pose and size of rigid objects. Specifically, known objects from a certain category are consistently aligned by their centers and orientations. At the same time, these objects are pre-scaled and pre-centered so that their tight bounding boxes all have a diagonal distance of 1 and are centered in the NOCS. The object pose and size can then be defined as the rigid transformation plus scaling from the NOCS coordinate to the camera space observations. NOCS provides a common reference frame for each category with a canonical global pose and size, enabling pose estimation even for unseen object instances. However, NOCS is not well-suited for articulated objects. Instead of the global pose and size, we care more about the states of rigid parts and joints, which are all ignored in NOCS.

Therefore, we need a representation which not only normalizes the pose and size of each rigid part, but also normalizes the joint parameters and states. For this purpose, we present A-NCSH, a two-level hierarchy of normalized coordinate spaces. At the leaf level, for each individual part we introduce one Normalized Part Coordinate Space (NPCS) to normalize the part pose and size. At the root level, we use a single coordinate space into which all the NPCSs are transformed so that the part coordinates represent the articulated object in a pre-defined rest state. We name the root space as Normalized Articulated Object Coordinate Space (NAOCS). NAOCS complements the set of NPCSs with normalized joint parameters as well as a rest joint state defined for each joint (Figure 2). We explain both NPCS and NAOCS in detail below.

NPCS. NPCS is defined similarly to NOCS [26] but for single parts instead of whole objects. We use a unit cube to normalize the 6D pose and size of each rigid part. Given a set of 3D shapes from a known category, we assume they all share a similar kinematic chain with M parts. We consistently segment the shapes into M sets of rigid parts, and each part is represented by a separate NPCS. We use the same protocol as is in [26] to pre-align and pre-scale the parts, which not only allows us to define the 6D pose and size of each part but also provides a natural way to obtain the amodal 3D bounding box for each part. We define the 6D pose and size of a part as the rigid transformation plus scaling from its NPCS coordinates to its camera coordinates. This definition naturally generalizes to unseen instances once the corresponding NPCS get predicted.

NAOCS. NPCS is defined for each component in the kinematic chain separately, and does not consider the relationship between different parts. Therefore normalizing of joint parameters is difficult. Moreover, NPCS is not able to fully normalize the states of different types of joints. For example, for a unseen cabinet instance with prismatic joints,

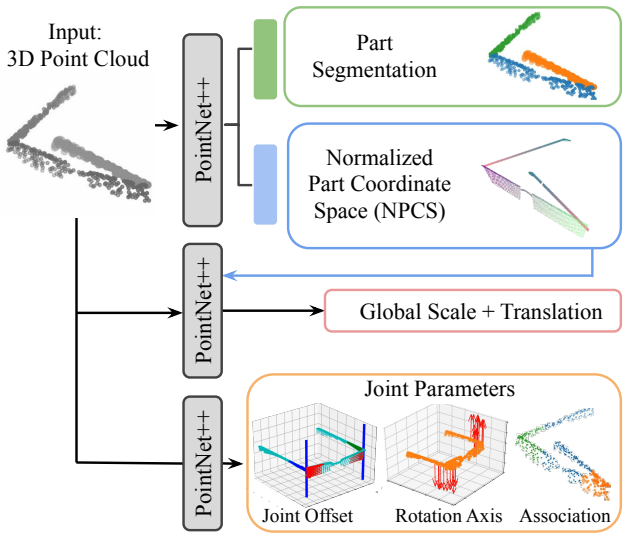


Figure 3. **A-NCSH network example.** The network uses three PointNet++ [19] modules to predict the A-NCSH representation that includes part segmentation, per-part NPCS coordinates, global transformation (scale and translation) from each NPCS to NAOCS, and joint parameters in NAOCS.

it is hard to determine how much a individual drawer is pulled out by using part coordinate alone. Therefore, to normalize joint parameters and joint states, we introduce NAOCS, a canonical frame that brings different NPCS coordinates together. Specifically, we transform the part coordinates in each NPCS to a global NAOCS space where different parts compose the original articulated object in a pre-defined rest state (*e.g.*, a cabinet with a closed drawer). The rest state of the object will define the rest state of each joint. In addition, the rotation axes of revolute joints or translation axes of prismatic joints from different object instances all follow same canonical orientations and get naturally aligned in NAOCS. With these joint parameters, we can explicitly incorporate the kinematic constraints when estimating the part poses and joint states. In this work, we specify the transformation from each NPCS to NAOCS by a global translation and scaling.

Compared with NOCS, NAOCS not only normalizes the global pose and size of the articulated objects, but also normalizes their joint parameters and states. It is worth mentioning that from NAOCS alone we can not obtain the per-part amodal 3D bounding box. Therefore, to generate a full state description for an articulated object, we need to use NAOCS together with multiple NPCS cases, namely our A-NCSH representation.

4.2. A-NCSH Network

We present a deep neural network capable of predicting the A-NCSH representation for unseen articulated object instances. As shown in Figure 3, the network accepts a point cloud P as input, and it contains three modules adapted

from PointNet++ [19] segmentation architectures. The prediction by the network consists of four types of information: rigid part segmentation, per-part NPCS coordinates, global transformation from each NPCS to NAOCS, and joint parameters in NAOCS.

The first module deals with part segmentation and NPCS coordinate regression. We assume that objects from the category of interest all share the same kinematic chain with M parts, and we simply associate each point $\mathbf{p}_i$ with one of the chain parts $S^{(j)}$ for segmentation purposes. Since we normalize each part with a separate NPCS, the network predicts all possible NPCS coordinates $\{\mathbf{c}_i^{(j)} \in \mathbb{R}^3 | \, j = 1, ..., M\}$ for point $\mathbf{p}_i$ instead of regressing a unique one. The final NPCS coordinates $\mathbf{c}_i$ will be selected using the predicted segmentation label. The segmentation branch and NPCS regression branch share the same feature backbone because the two tasks are closely related. They only differ in the heads, which have one fully-connected layers.

The second module predicts global transformations $\{G^{(j)}\}$ which place NPCSs properly in the NAOCS. We design NPCS and NAOCS so that the part orientation remains the same in both spaces. This allows us to only estimate a global translation $G_t^{(j)}$ and a global scaling $G_s^{(j)}$ for the j-th NPCS. We use this approach through all of our experiments. The input to this module is a concatenation of the original point cloud $\{\mathbf{p}_i\}$ and the final NPCS coordinates $\{\mathbf{c}_i\}$ predicted from the first module. Instead of predicting a unique $G_t^{(j)}$ and $G_s^{(j)}$ for the j-th NPCS, we perform a dense regression of a per-point global translation $G_{ti}^{(j)}$ and global scaling $G_{si}^{(j)}$, which tends to provide more reliable NAOCS coordinates in practice. The final per-point global translation G_{ti} and global scaling G_{si} will be selected based on the predicted segmentation label. The NAOCS coordinates can be represented as $\{\mathbf{g}_i | \, \mathbf{g}_i = G_{si}\mathbf{c}_i + G_{ti}\}$.

The last module infers joint parameters ϕ_k' for each joint in the NAOCS space. (We use the symbol "$'$" to distinguish NAOCS-space parameters from camera-space parameters.) As we mentioned before, we are mainly interested in two types of joints: 1D revolute joint whose parameters include rotation axis direction $\mathbf{u}_k^{(r)\prime}$ and pivot point position $\mathbf{q}_k'$, 1D prismatic joint whose parameters are translation axis direction $\mathbf{u}_k^{(t)\prime}$. Joints serve as an auxiliary structure to the object geometry, and the fact that they are spatially sparse introduces challenges to the problem of parameter estimation. To cope with these challenges, we estimate joint heap maps on the point cloud data to localize joints, and we leverage voting schemes to estimate the joint parameters. To be specific, for input point cloud $P = \{\mathbf{p}_i \in \mathbb{R}^3 | i = 1, ..., N\}$, we use the corresponding NAOCS coordinates $P' = \{\mathbf{p}_i' \in \mathbb{R}^3 | i = 1, ..., N\}$ to compute a per-point heat map H_{k1} associated with a revolute joint $\phi_{k1}' = (\mathbf{u}_{k1}'(r)\prime, \mathbf{q}_{k1}')$ as follows:

$$H_{k1}(\mathbf{p}_i) = \max(0, 1 - \frac{||(\mathbf{p}_i' - \mathbf{q}_{k1}') \times \mathbf{u}_{k1}^{(r)\prime}||}{\sigma})$$

where σ is a distance threshold which we set as 0.1 in all the experiments. For a prismatic joint $\phi_{k2}' = (\mathbf{u}_{k2}^{(t)\prime})$, assuming its child part is S_j (we assume a known kinematic chain for each category so this is easy to obtain), we define the per-point heapmap H_{k2} as:

$$H_{k2}(\mathbf{p}_i) = \begin{cases} 1 & \mathbf{p}_i \in S_j \\ 0 & \text{otherwise} \end{cases}$$

We regress one heat map for each joint and at the same time, we also perform a dense regression of the joint axis orientation $\mathbf{u}_k^{(r)\prime}$ or $\mathbf{u}_k^{(t)\prime}$. The final predictions of the joint axis orientation will simply be the average prediction over all the points with a heat score larger than 0.5. To predict the pivot point of a 1D revolute joint which is not uniquely defined (it could move arbitrarily along the rotation axis), points are again filtered according to their heat scores to vote for a possible answer. We perform a dense regression of a unit direction vector from each point to the rotation axis. Since the heat map for a revolute joint already indicates the distance from each point to the rotation axis, together with the predicted direction vectors, we can project each point onto the rotation axis. We simply average these projections to get a possible pivot point.

Loss functions: We use relaxed IoU loss [29] L_{seg} for part segmentation. We use mean-square loss L_{NPCS} for NPCS coordinate regression. Instead of directly supervising per-point global translation $G_{ti}^{(j)}$ and scaling $G_{si}^{(j)}$ for each part $S^{(j)}$, we compose NPCS coordinates from all different parts, leveraging the predicted global transformations and minimizing the mean-square difference L_{NAOCS} from the ground truth NAOCS coordinates. This trick stabilizes the prediction of global transformations and improves the predicted NAOCS coordinates. We also use mean-square loss L_{joint} for all the joint-related predictions. Our total loss is given by $L = \lambda_1 L_{\text{seg}} + \lambda_2 L_{\text{NPCS}} + \lambda_3 L_{\text{NAOCS}} + \lambda_4 L_{\text{joint}}$, and we set the multiplication factors to 1, 10, 10, 1 heuristically in our experiments.

Training data generation: To train this network, we generate synthetic depth rendering using the object 3D model provided in the Shape2Motion dataset [27]. In this dataset, each object instance data contains descriptions of the object's 3D geometry and its articulation parameters, which are both necessary for training our network. During rendering, the program automatically generates random articulation poses for each object instance, according to its joint limits. Then the depth images and corresponding

ground truth masks are rendered from a set of random camera viewpoints. We also filter out camera poses where some parts of the object are completely occluded. On average, 30,000 training images for 40 different object instances are generated for each object category.

4.3. Pose Optimization with Kinematic Constraints

In the previous section, we have introduced our A-NCSH network which not only segments an 3D point cloud $\{\mathbf{p}_i\}$ into rigid parts $S^{(j)}$, but also predicts the NPCS coordinates $\{\mathbf{c}_i\}$, per-point global translation $\{G_{ti}\}$ and global scaling $\{G_{si}\}$, the NAOCS coordinates $\{\mathbf{g}_i\}$, and the joint parameters $\{\phi'_k\}$ in the NAOCS. To fully describe the pose of an articulated object, we still need to estimate the 6D poses and sizes $\{R^{(j)}, \mathbf{t}^{(j)}, s^{(j)}\}$ for all the parts, as well as the joint states $\{\theta_k\}$ and joint parameters $\{\phi_k\}$ in the camera space.

Since for each part we have its NPCS coordinates and camera space point coordinates in correspondence, we could follow [26] to estimate its 6D pose and size. In [26], the Umeyama algorithm [24] is adopted within a RANSAC [8] framework to robustly estimate the 6D pose and size of a single rigid object. However, a naive extension of the approach to each individual part in our setting would easily lead to physically impossible poses. Since the degree of freedom for each part is not independent in a kinematic chain, a part-based pose estimation approach with noisy NPCS coordinates can easily break the kinematic constraints. To cope with this issue, we need to introduce kinematic constraints while estimating the part poses. Without the kinematic constraints, the energy function E_{vanilla} regarding all part poses can be written as $E_{\text{vanilla}} = \sum_j e_j$, where

$$e_j = \frac{1}{|S^{(j)}|} \sum_{\mathbf{p}_i \in S^{(j)}} ||\mathbf{p}_i - (s^{(j)} R^{(j)} \mathbf{c}_i + \mathbf{t}^{(j)})||^2$$

We then introduce the kinematic constraints by adding an energy term e_k for each joint to the energy function. In concrete terms, our modified energy function is $E_{\text{constrained}} = \sum_j e_j + \lambda \sum_k e_k$, where e_k is defined differently for each type of joint. For a revolute joint with parameters $\phi'_k = (\mathbf{u}_k^{(r)\prime}, \mathbf{q}'_k)$ in the NAOCS, assuming it connects part $S^{(k1)}$ and part $S^{(k2)}$, we define e_k as:

$$e_k = ||R^{(k1)} \mathbf{u}_k^{(r)\prime} - R^{(k2)} \mathbf{u}_k^{(r)\prime}||^2$$

For a prismatic joint with parameters $\phi'_k = (\mathbf{u}_k^{(t)\prime})$ in the NAOCS, again assuming it connects part $S^{(k1)}$ and part $S^{(k2)}$, we define e_k as:

$$e_k = \sum_{i=1,2} ||[R^{(ki)} u_k^{(r)\prime}]_\times \delta_{k1,k2}||^2 + \mu ||R^{(k1)} (R^{(k2)})^T - I||^2$$

where $[\cdot]_\times$ converts a vector into the matrix for conducting cross product with other vectors, and $\delta_{k1,k2}$ is defined as:

$$\delta_{k1,k2} = \mathbf{t}^{(k2)} - \mathbf{t}^{(k1)} + s^{(k1)} R^{(k1)} G_t^{(k1)} - s^{(k2)} R^{(k2)} G_t^{(k2)}$$

To minimize our energy function $E_{\text{constrained}}$, we can no longer separately solve different part poses using the Umeyama algorithm. Instead, we first minimize E_{vanilla} using the Umeyama algorithm to initialize our estimation of the part poses. Then we fix $\{s^{(j)}\}$ and adopt a non-linear least-squares solver to further optimize $\{R^{(j)}, \mathbf{t}^{(j)}\}$, as is commonly done for bundle adjustment [1]. Similar to [26], we also use RANSAC for outlier removal.

After estimating the pose and size of each part, the joint states and parameters can also be deduced. For a revolute joint k connecting parts $S^{(k1)}$ and $S^{(k2)}$, we compute its parameters $\phi_k = (\mathbf{u}_k^{(r)}, \mathbf{q}_k)$ in the camera space as:

$$\mathbf{u}_k^{(r)} = \frac{(R^{(k1)} + R^{(k2)}) \mathbf{u}_k^{(r)\prime}}{||(R^{(k1)} + R^{(k2)}) \mathbf{u}_k^{(r)\prime}||}$$

$$\mathbf{q}_k = \frac{1}{2} \sum_{i=1,2} R^{(ki)} \left(\frac{s^{(ki)}}{G_s^{(ki)}} \mathbf{q}'_k - G_t^{(ki)} \right) + \mathbf{t}^{(ki)}$$

where $G_s^{(ki)}$ and $G_t^{(ki)}$ are computed through averaging the per-point global scaling and translation over all the points on part $S^{(ki)}$. The joint state θ_k can be computed as:

$$\theta_k = \arccos((\text{trace}(R^{(k2)} (R^{(k1)})^T) - 1)/2)$$

For a prismatic joint k connecting parts $S^{(k1)}$ and $S^{(k2)}$, we compute its parameters $\phi_k = (\mathbf{u}_k^{(t)})$ in the camera space as:

$$\mathbf{u}_k^{(t)} = \frac{(R^{(k1)} + R^{(k2)}) \mathbf{u}_k^{(t)\prime}}{||(R^{(k1)} + R^{(k2)}) \mathbf{u}_k^{(t)\prime}||}$$

and its state θ_k is simply $||\delta_{k1,k2}||$.

5. Evaluation

5.1. Experimental Setup

Metrics. We use the following to evaluate and compare our algorithm's performance.

- **Per-part metrics.** For each part, we evaluate rotation error measured in degrees, translation error in normalized part coordinate space, and 3D intersection over union (IoU) [20] of the predicted amodal bounding box. For each case, we normalize the translation so that the translation errors are comparable among parts with different sizes.
- **Joint state.** For each revolute joint, we find the joint angle errors in degrees. For each prismatic joint, we compute the error of relative translation amounts. The relative translation is defined in the NAOCS.

Category	Method	Part-based Metrics			Joint State	Joint Parameter	
		rotation error ↓	translation error ↓	3D IoU % ↑	error ↓	angle error ↓	distance error ↓
Eye-glasses	NPCS	3.9, 11.0,	0.047, 0.100, 0.109	89.2 , 34.8 , 24.4	11.8° , 14.2°	-	-
	NAOCS	3.8, 30.0, 24.1	0.090 , 0.496 , 0.232	-	30.6°, 25.3°	-	-
	A-NCSH	**1.3 , 2.3 , 6.0**	**0.020 , 0.030 , 0.060**	**91.4, 42.1, 25.6**	**1.9° , 6.0 °**	**0.54 , 0.17**	**0.015 , 0.009**
Oven	NPCS	1.2 , 2.9	**0.030 , 0.043**	75.8 , 89.0	3.0°	-	-
	NAOCS	1.7, 4.7	0.036 , 0.090	-	5.1°	-	-
	A-NCSH	**1.1, 2.2**	0.033 , **0.043**	**75.9 , 89.5**	**2.1°**	**0.21**	**0.016**
Washing Machine	NPCS	1.0, 1.9	0.042 , 0.055	86.9 , 88.1	2.2 °	-	
	NAOCS	1.1 , 3.3	0.072 , 0.119	-	3.1 °	-	-
	A-NCSH	**1.0 , 1.4**	**0.045 , 0.037**	**91.5 , 92.0**	**1.5 °**	**0.12**	**0.006**
Laptop	NPCS	11.6, 4.4	0.098, **0.044**	35.7, 93.6	14.4 °	-	-
	NAOCS	12.4, 4.9	0.110, 0.049	-	15.2 °	-	-
	A-NCSH	**6.7, 4.3**	**0.062, 0.044**	**41.1, 93.0**	**9.7 °**	**0.17**	**0.011**
Drawer	NPCS	1.9, 3.5, 2.4, 1.8	0.032, 0.038, 0.024, **0.025**	82.8, 71.2, 71.5, **79.3**	0.026, 0.031, 0.046	-	-
	NAOCS	1.5, 2.5, 2.5, 2.0	0.044, 0.045, 0.073, 0.054	-	0.043, 0.066, 0.048	-	-
	A-NCSH	**1.0, 1.1, 1.2, 1.5**	**0.024, 0.021, 0.021**, 0.033	**84.0,72.1, 71.7**, 78.6	**0.011, 0.020, 0.030**	**0.13, 0.13, 0.13**	-

Table 1. Performance comparison on *unseen* object instances. The categories eyeglasses, oven, washing machine, and laptop contain only revolute joints and the drawer category contains three prismatic joints.

- **Joint parameter.** For each revolute joints, we evaluate the orientation error of the rotation axis in degrees, and the location error using the minimum line-to-line distance in NAOCS. For each prismatic joint, we compute the orientation error of the translation axis.

Datasets. We have evaluated our algorithm using both synthetic and real-word datasets. To generate the synthetic testset, we used a different set of the object instances from [27] that do not overlap with our training data. Following the same rendering pipeline with random camera viewpoints, we generated on average 3000 testing images of unseen object instances for each object category. For the real data, we evaluated our algorithm on the dataset provided by Michel *et al*. [16], which contains depth images for 4 different objects captured using the Kinect.

Baselines. There are no existing methods for category-level articulated object pose estimation. We therefore used ablated versions of our system for baseline comparison.

- **NPCS.** This algorithm predicts part segmentation and NPCS for each part (without the joint parameters). The prediction allows the algorithm to infer part pose, amodal bounding box for each part, and joint state for revolute joint by treating each part as a independent rigid body. However, it is not able to perform combined optimization with the kinematic constraints.
- **NAOCS.** This algorithm predicts part segmentation for each part and NAOCS representation for the whole object instance. The prediction allows the algorithm to infer part pose and joint state, but not the amodal bounding boxes for each part since the amodal bounding boxes is not defined in the NAOCS alone.
- **Direct joint regression.** This algorithm directly regresses joint parameters, including location and orientation for each joint from the point cloud using PointNet++ [19]. TODO: [what is the final algorithm we use?]

Our final algorithm predicts the full A-NCSH representation that includes NPCS, joint parameters, and per-point global scaling and translation value that can be used together with the NPCS prediction for computing NAOCS.

5.2. Experimental Results

Figure 4 presents some qualitative results. Tables 1 to 3 summarize the quantitative results. Following paragraphs provide our analysis and discussion of the results.

Effect of global optimization. First, we want to examine how global optimization would influence the accuracy of articulated object pose estimation, using both predicted joint parameters and predicted part poses. To see this, we compare the algorithm performance between NPCS and A-NCSH, where NPCS performs a per-part pose estimation and A-NCSH performs a combined optimization using the full kinematic chain to constrain the result. The results in Table 1 show that the combined optimization of joint parameters and part pose consistently improve the algorithm's accuracy for almost all object categories and evaluation metrics. The improvement is particularly salient for thin object parts such as the two temples of eyeglasses (the parts that extend over the ears), where the per-part based method produces large pose error due to limited point observation and shape ambiguity. This result demonstrates that the joint parameters predicted in the NPCS can regularize the part poses based on kinematic chain constraints during the combined pose optimization step and improve the pose estimation accuracy.

Comparison to direct regression of joint parameters. Direct regression of exact location and accurate orientation

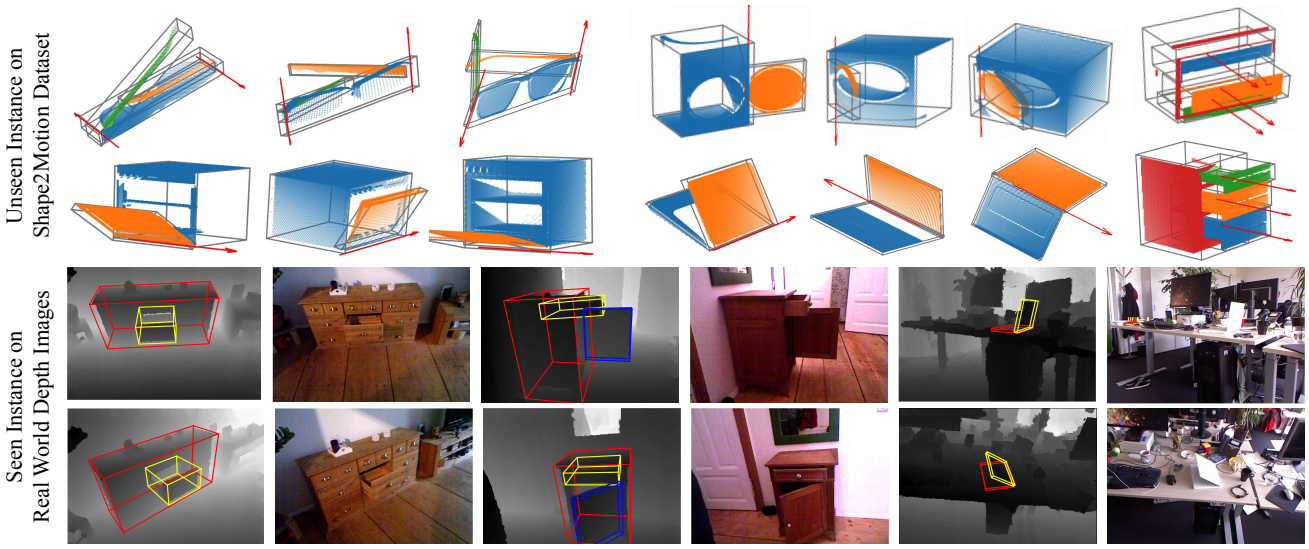


Figure 4. **Qualitative Results.** Top tow rows show test results on unseen object instances from the Shape2Motion dataset [27]. Bottom two rows show test result on seen instances in the real-world dataset [16]. Here we visualize the predicted amodal bounding box for each parts. Color images are for visualization only.

with all degrees of freedom is challenging. In our approach, we choose to predict the joint parameters in NPCS since it provides a canonical representation where the joint axes usually have a strong orientation prior, which simplified prediction considerably. We further use a voting-based scheme to reduce prediction noise. Based on the high-quality prediction of part poses, we can then transform the joint parameter predictions from NPCS into the 3D reference frame of the camera. Comparing to the direct regression baseline using PointNet++, Table 5 shows that our approach significantly improves joint axis prediction on unseen instances of eyeglasses. TODO: [xiaolong]

Category	Methods	angle error	distance error
Eye-glass	PointNet++	15.4°, 5.1°	**0.012**, **0.011**
	A-NCSH	**0.5°**, **0.2°**	0.018, **0.011**
Oven	PointNet++	27.0°	0.017
	A-NCSH	**0.21°**	**0.016**
Washing Machine	PointNet++	8.67°	0.012
	A-NCSH	**0.12°**	**0.006**
Laptop	PointNet++	29.5°	**0.011**
	A-NCSH	**0.17°**	**0.011**
Drawer	PointNet++	4.9°,5.0°,5.1°	
	A-NCSH	**0.13°**,**0.13°**,**0.13°**	-

Table 2. **A comparison of joint parameter predictions.**

Handling cases under severe occlusion. To further investigate our algorithm's robustness under occlusion, we manually controlled the occlusion levels in the test image and evaluated the algorithm's performance. The occlusion level is defined as the ratio of number of visible points of a specific object part, compared to the full part size. We divided the occlusion level into three categories: $> 80\%$ visible, $40\% - 80\%$ visible, and $< 40\%$ visible. Table 3 show the test results for the temple parts of unseen eyeglass instances. We can observe that while the occlusion level increases, the performance of the NPCS algorithm drops drastically (-4.9% in 3D IoU). By comparison, our algorithm is still able to preform reasonably well, with a slight drop of 1.4%.

	Occlusion (Low to High)		
NPCS	43.4	40.4	38.5
A-NCSH	**44.6**	**43.7**	**43.2**

Table 3. Performance (IoU%) under occlusion.

Generalization to real depth images. We have also tested our algorithm's ability to generalize to real-world depth images on the dataset provided in [16]. The dataset is designed for instance-level articulated object pose estimation, and it contains four different object instances: laptop, cupboard, 4-part toy train, and cabinet. Each has two test sequences, captured using a Kinect. Following the same training protocol, we train the algorithm with synthetically rendered depth images of object instances using the URDF files provided in [16]. Then we test the pose estimation accuracy on the real world depth image. Although our algorithm has not been designed for instance-level pose estimation and was never trained using any real-word depth images, it is able to achieve reasonable performance in the real-world cases despite the large domain gap. Table 6 and

Figure 4 show example pose estimation results. Please refer to our supplementary material for more results.

6. Conclusion

This paper has presented an approach for category-level pose estimation of articulated objects from a single depth image. To accommodate unseen object instances and different articulation types, we proposed a new representation scheme known as Articulation-Aware Normalized Coordinate Space Hierarchy (A-NCSH). The system formulates articulated pose fitting from the A-NCSH space as a combined optimization problem, taking both rigid parts pose fitting and joint constraints into consideration. Our experiments demonstrate that the A-NCSH representation and the global optimization approach significantly improve algorithm accuracy in both part pose prediction and joint parameter estimation.

7. Acknowledgement

This research used resources provided by Advanced Research Computing within the Division of Information Technology at Virginia Tech. We are also grateful for financial and hardware support from Google. TODO: [add Acknowledgement]

References

[1] Sameer Agarwal, Noah Snavely, Steven M Seitz, and Richard Szeliski. Bundle adjustment in the large. In *European conference on computer vision*, pages 29–42. Springer, 2010. 6

[2] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018. 3

[3] Paul J Besl and Neil D McKay. A method for registration of 3-d shapes. In *PAMI*, 1992. 2

[4] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551. Springer, 2014. 2, 10, 14

[5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3364–3372, 2016. 2

[6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2018. 10

[7] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. Factored pose estimation of articulated objects using efficient nonparametric belief propagation. *arXiv preprint arXiv:1812.03647*, 2018. 2

[8] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 6

[9] Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-point regression pointnet for 3d hand pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 475–491, 2018. 3

[10] Karol Hausman, Scott Niekum, Sarah Osentoski, and Gaurav S Sukhatme. Active articulation model estimation through interactive perception. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3305–3312. IEEE, 2015. 1, 2

[11] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *2008 IEEE International Conference on Robotics and Automation*, pages 272–277. IEEE, 2008. 1, 2

[12] Dov Katz, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In *2013 IEEE International Conference on Robotics and Automation*, pages 5003–5010. IEEE, 2013. 1, 2

[13] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6050–6059, 2017. 3

[14] Roberto Martín-Martín, Sebastian Höfer, and Oliver Brock. An integrated approach to visual perception of articulated objects. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5091–5097. IEEE, 2016. 1, 2

[15] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. 3

[16] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, pages 181–1, 2015. 2, 7, 8, 10, 13, 14

[17] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G Derpanis, and Kostas Daniilidis. Coarse-to-fine volumetric prediction for single-image 3d human pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7025–7034, 2017. 3

[18] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 459–468, 2018. 3

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas 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. 4, 5, 7

[20] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings*

Object	Sequence		Brachmann et al.[4]	Frank et al.[16]	A-NCSH (Ours)
Laptop	1	all	8.9%	64.8%	**94.1%**
		parts	29.8% 25.1%	65.5% 66.9%	**97.5% 94.7%**
	2	all	1%	65.7%	**98.4%**
		parts	1.1% 63.9%	66.3% 66.6%	**98.9% 99.0%**
Cabinet	3	all	0.5%	**95.8%**	90.0%
		parts	86% 46.7% 2.6%	98.2% 97.2% **96.1%**	**98.9% 97.8%** 91.9%
	4	all	49.8%	**98.3%**	94.5%
		parts	76.8% 85% 74%	98.3% 98.7% **98.7%**	**99.5% 99.5%** 94.9%
Cupboard	5	all	90%	95.8%	93.9%
		parts	91.5% 94.3%	95.9% **95.8%**	**99.9%** 93.9%
	6	all	71.1%	99.2%	**99.9%**
		parts	76.1% 81.4%	99.9% 99.2%	**100% 99.9%**
Toy train	7	all	7.8%	**98.1%**	68.4%
		parts	90.1% 17.8% 81.1% 52.5%	**99.2% 99.9% 99.9%** 99.1%	92.0% 68.5% 99.3% **99.2%**
	8	all	5.7%	**94.3%**	91.1%
		parts	74.8% 20.3% 78.2% 51.2%	100% 100% 97% **94.3%**	**100% 100% 100%** 91.1%

Table 4. **Instance-level real-world depth benchmark.** While not designed for instance-level articulated object pose estimation, our algorithm is able to achieve comparable performance compare to the state-of-the-art approach and improves the performance for challenging cases such as laptops. TODO: [should we include it in appendix?]

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 808–816, 2016. 6

[21] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. Compositional human pose regression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2602–2611, 2017. 3

[22] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018. 1

[23] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018. 1

[24] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):376–380, 1991. 6

[25] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dense 3d regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2018. 3

[26] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019. 1, 2, 3, 4, 6

[27] Xiaogang Wang, Bin Zhou, Yahao Shi, Xiaowu Chen, Qinping Zhao, and Kai Xu. Shape2motion: Joint analysis of motion parts and attributes from 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8876–8884, 2019. 5, 7, 8, 11

[28] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 1

[29] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. In *SIGGRAPH Asia 2018 Technical Papers*, page 209. ACM, 2018. 2, 5

A. Real-world instance-level benchmark.

Table 6 shows quantitative comparison of AD accuracy on ICCV2015 Articulated Object Challenge [16], which contains RGB-D data with 4 articulated objects: laptop, cabinet, cupboard and toy train. Qualitative results are visualized in Figure 8. This dataset provides 2 testing sequences for each object. Each sequence contains around 1000 images captured by having a RGB-D camera slowly moving around the object. Objects maintain the same articulation state within each sequence. Each part of the articulated object is annotated with its 6D pose with respect to the known CAD model. Since no training data is provided, we use the provided CAD models to render synthetic depth data, with 10 groups of random articulation status considered. We render object masks for the testing sequences with Pybullet[6]. Following the original paper [16], we adopt 10% of the object part diameter as the

threshold to compute Averaged Distance(AD) accuracy, and test the performance on each sequence separately.

B. Additional results

Figure 7 shows additional results on the Shape2Motion[27] dataset.

C. Comparison on joints estimation.

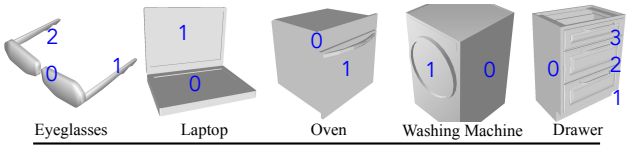
Table 5 shows the result comparison on joint parameter estimation for all object categories. The result shows that our approach consistently outperforms the PointNet++ regression baseline on both joint orientation and location accuracy. Note that the prismatic joints in drawers don't have a pivot point so there is no need for evaluating distance error.

Category	Methods	angle error	distance error
Eye-glasses	PointNet++	15.4°, 5.1°	**0.012**, **0.011**
	A-NCSH	**0.5°**, **0.2°**	0.018, **0.011**
Oven	PointNet++	27.0°	0.017
	A-NCSH	**0.21°**	**0.016**
Washing Machine	PointNet++	8.67°	0.012
	A-NCSH	**0.12°**	**0.006**
Laptop	PointNet++	29.5°	**0.011**
	A-NCSH	**0.17°**	**0.011**
Drawer	PointNet++	4.9°,5.0°,5.1°	
	A-NCSH	**0.13°**,**0.13°**,**0.13°**	-

Table 5. **A comparison of joint parameter predictions.**

D. Part definition

Figure 5 shows the index definitions of parts for each object category used in the main paper.



Category	Part 0	Part 1	Part 2	Part 3
Eyeglasses	center	left temple	right temple	-
Oven	base	door	-	-
Washing Machine	base	door	-	-
Laptop	base	display	-	-
Drawer	base	lowest	middle	top

Figure 5. **Part definitions for each object category.**

E. Limitation and failure cases

Figure 6 shows typical failure cases of our algorithm. A typical failure mode of our algorithm is the inaccurate prediction under heavy occlusion where one of the object parts is almost not observed. Figure 6 shows one of such cases where one of the eye-glasses temples is almost completely occluded. Also, under the situation of heavy occlusion for prismatic joints, there is considerate ambiguity for A-NCSH prediction on the size of the heavily occluded parts, as shown in Figure 6. However, NAOCS representation does not suffer from the size ambiguity, thus leading to a more reliable estimation of the joint state (relative translation distance compare to the rest state) and joint parameters (translation axis).

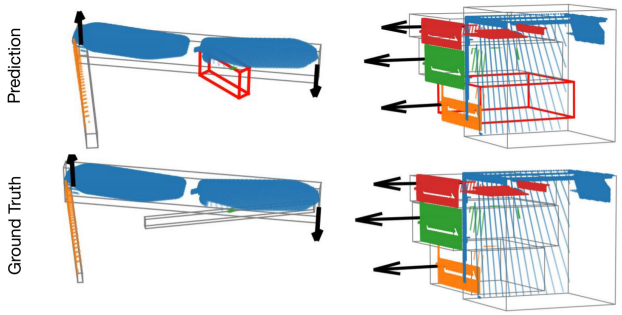


Figure 6. **Failure cases.** Left column shows failure cases on unseen eyeglasses instances, when a part is under heavy occlusion and barely visible. Right column shows the failure case on unseen drawers, when there are shape variations on parts and only the front area of the drawer is visible. The predicted drawer size is bigger than the real size. Although the box prediction is wrong, our method can reliably predict the joint state and joint parameters by leveraging the NAOCS representation.

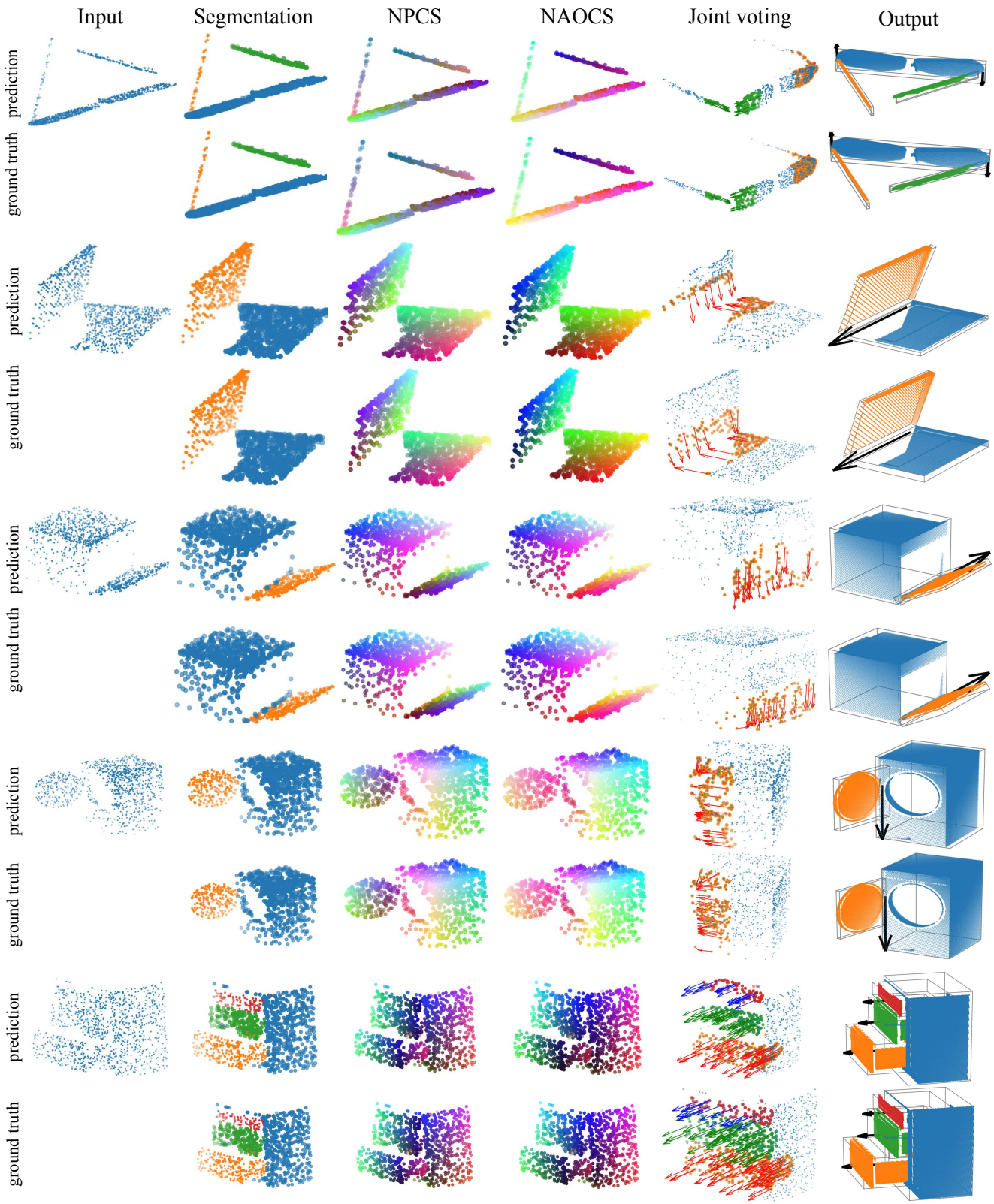


Figure 7. **Additional results on category-level Shape2Motion dataset.** The first column shows the input point clouds; the second column shows our prediction and ground truth part segmentation mask; the third and fourth column show our prediction and ground truth NPCS and NAOCS, where the RGB channels encode the coordinates; the fifth column visualizes joint voting, where the arrows represent offset vectors to rotational hinge for revolute joints and the direction of joint axis for prismatic joints; the sixth column visualizes per part 3D bounding boxes, together with joint parameters.

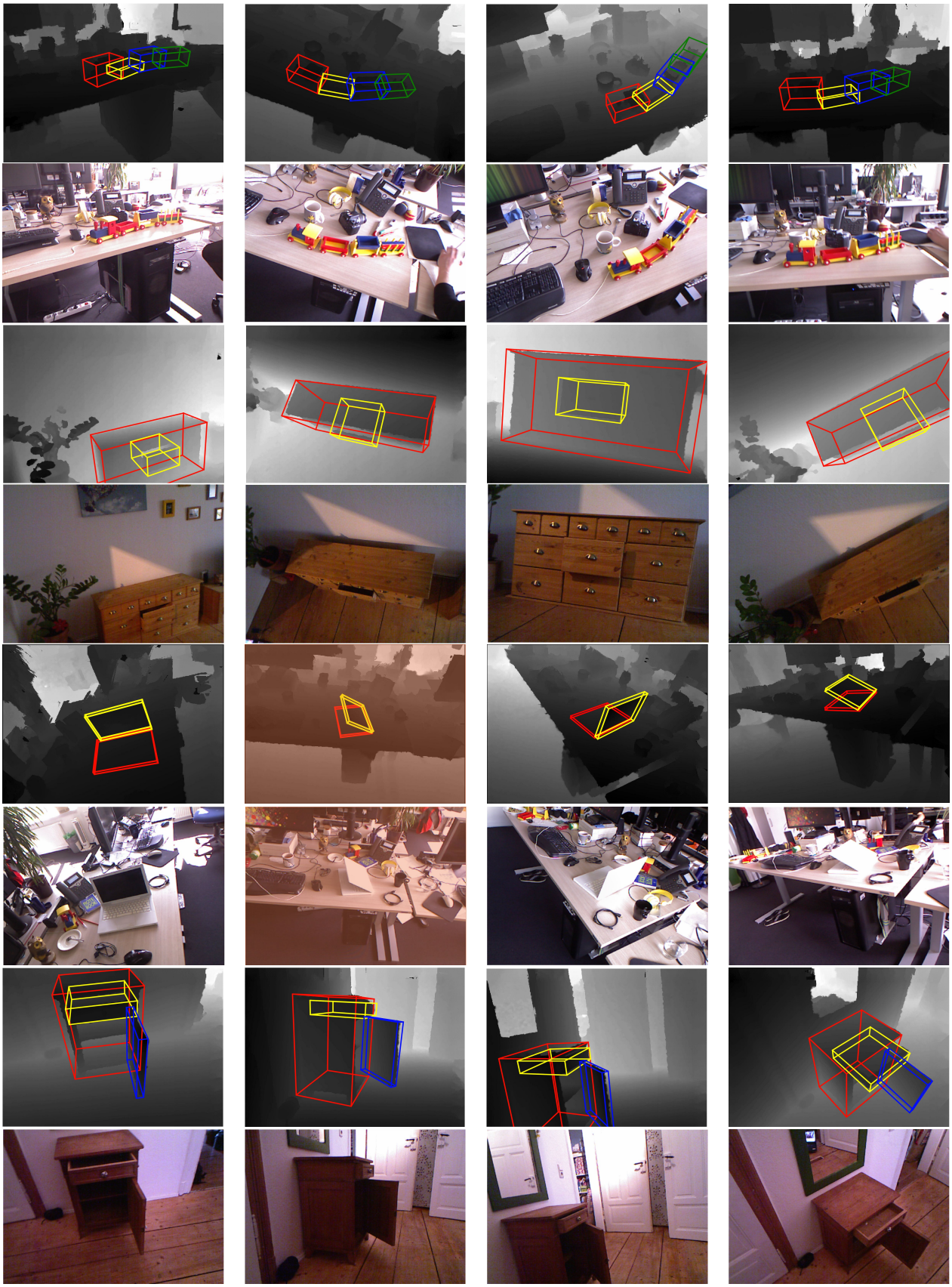


Figure 8. **Additional results on real-world instance-level depth dataset.** More qualitative results on all 4 objects from ICCV2015 Articulated Object Challenge [16] are shown here, with toy train, cupboard, laptop, cabinet from up-pest row to lowest row in order. Only depth images are used for pose estimation, RGB images are shown here for better reference. For each object, we estimate 3D tight bounding boxes to all parts on the kinematic chain, and project the predicted bounding boxes back to the depth image.

Object	Sequence		Brachmann et al.[4]	Frank et al.[16]	A-NCSH (Ours)
Laptop	1	all	8.9%	64.8%	**94.1%**
		parts	29.8% 25.1%	65.5% 66.9%	**97.5% 94.7%**
	2	all	1%	65.7%	**98.4%**
		parts	1.1% 63.9%	66.3% 66.6%	**98.9% 99.0%**
Cabinet	3	all	0.5%	**95.8%**	90.0%
		parts	86% 46.7% 2.6%	98.2% 97.2% **96.1%**	**98.9% 97.8%** 91.9%
	4	all	49.8%	**98.3%**	94.5%
		parts	76.8% 85% 74%	98.3% 98.7% **98.7%**	**99.5% 99.5%** 94.9%
Cupboard	5	all	90%	95.8%	93.9%
		parts	91.5% 94.3%	95.9% **95.8%**	**99.9%** 93.9%
	6	all	71.1%	99.2%	**99.9%**
		parts	76.1% 81.4%	99.9% 99.2%	**100% 99.9%**
Toy train	7	all	7.8%	**98.1%**	68.4%
		parts	90.1% 17.8% 81.1% 52.5%	**99.2% 99.9% 99.9%** 99.1%	92.0% 68.5% 99.3% **99.2%**
	8	all	5.7%	**94.3%**	91.1%
		parts	74.8% 20.3% 78.2% 51.2%	100% 100% 97% **94.3%**	**100% 100% 100%** 91.1%

Table 6. **Instance-level real-world depth benchmark.** While not designed for instance-level articulated object pose estimation, our algorithm is able to achieve comparable performance compare to the state-of-the-art approach and improves the performance for challenging cases such as laptops.