2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Fully Convolutional Geometric Features for Category-level Object Alignment

Qiaojun Feng   Nikolay Atanasov

Abstract— **This paper focuses on pose registration of different object instances from the same category. This is required in online object mapping because object instances detected at test time usually differ from the training instances. Our approach transforms instances of the same category to a normalized canonical coordinate frame and uses metric learning to train fully convolutional geometric features. The resulting model is able to generate pairs of matching points between the instances, allowing category-level registration. Evaluation on both synthetic and real-world data shows that our method provides robust features, leading to accurate alignment of instances with different shapes.**

I. Introduction

Meaningful and detailed environment reconstruction is an enabling capability for autonomous robot operation in various tasks, including safe navigation, object manipulation, or human-robot interaction. As embodied agents have limited storage and computation capabilities, developing compressed, yet, expressive environment models is a key problem. Online object-based simultaneous localization and mapping (SLAM) methods [1], [2] are promising in generating efficient and semantically meaningful maps composed of sparse object landmarks. These methods work by detecting, segmenting and tracking object instances online and using the semantic information to estimate the object poses and shapes. Since the object instances observed online are always different from the stored models, semantic SLAM methods need to perform cross-instance alignment. Another challenge is that only partial and potentially low-resolution observations may be available due to the limited field-of-view or resolution of the onboard sensors. See Fig. 1 for an example.

Recently, learning-based 3D feature extraction algorithms have shown promising performance in point-cloud registration task. Convolutional neural network architectures can learn powerful descriptors from pre-aligned matching patches. However, generating matching patches for different instances in the same category may not be easy. While same-category objects share similar overall structure, individual parts may be quite different (e.g., armchair vs an office chair). Existing works rely on human annotations of sparse semantically meaningful object parts (e.g., chair legs, back support, seat) to obtain such matchings across instances [3].

The main **contribution** of this paper is a learning-based method for generating dense matching pairs across different instances from the same category. We leverage the idea of normalized canonical coordinates (NCC) [4] to align the different instances during training and automatically generate positive and negative examples of matching pairs. We use metric learning to train a sparse convolutional neural network to predict cross-instance geometric features [5] suitable for registration. Our approach enables improved object pose estimation versus the state-of-the-art on both synthetic and real-world data.

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181 and ONR SAI 00014-18-1-2828.

The authors are with Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093, USA {qjfeng,natanasov}@ucsd.edu

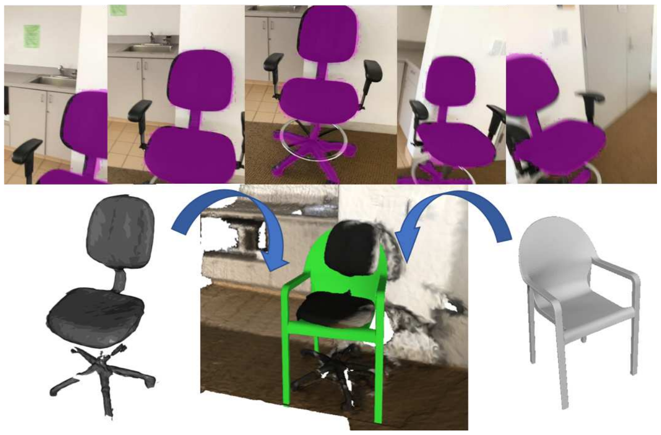

Fig. 1: A sequence of RGB-D images (top) may be used for object-level mapping. This paper focuses on the important subproblem of category-level registration. A partial object point cloud (bottom, left) may be generated from object detection, segmentation, and tracking (top, purple). A not-identical CAD model (bottom, right) needs to be aligned to the observed point cloud (bottom, middle) in order to generate an object-level map.

II. Related Work

3D point cloud features: Traditional methods of hand-crafted 3D feature include Spin Image [6], FPFH [7], SHOT [8]. They mainly rely on simple local geometric information like normals and histogram. Recently learning-based methods have gained more attention. 3DMatch [9] applies 3D ConvNet on fixed-size 3D patch represented in truncated distance function. 3DFeat-Net [10] collects local points in a radius-fixed ball, uses a detector module to estimate the local orientation and generates the descriptor after alignment. 3DSmoothNet [11] proposes the idea of smoothed density value voxelization as the preprocessed input to reduce the sparsity of the input patch. FCGF [5] leverages the sparse convolution [12] to build fully-convolutional network in 3D space and use metric learning to learn the feature. The point cloud coordinates and associated features are used for registration to estimate the rigid transformation.

Pose estimation with known model: Assume the object shape is given, we can estimate its 6DoF pose from RGB or RGB-D images. PVNet [13] estimates sparse object keypoints on the RGB image via voting mechanism. DPOD [14] predicts a dense object classification and 2D-3D correspondence mask. There are also algorithms doing a straightforward pose regression without explicit associations such as [15]. [16] trains a CNN to predict the object pose using pixel surface normals, followed by model selection and alignment using ICP. The Scan2CAD [17] dataset contains 9DoF alignment annotation of CAD models from ShapeNet [18] w.r.t. the indoor scene reconstruction of ScanNet [19]. They voxelize the RGBD scan in the form of signed distance field (SDF). A 3D CNN network is trained to predict sparse keypoints correspondence with a fixed-size patch input.

Category-level object pose estimation: Objects in the same category usually share similar structure with relatively consistent distribution of semantic keypoints [20], such as the wheels of the car and the chair legs. Category-level semantic keypoints [3] are predicted on RGB images and a deformable shape model is fitted to recover the pose. StarMap [21] extends to predict category-agnostic keypoints by predicting the keypoint's normalized canonical coordinate from RGB image observation. NOCS [4] also uses the idea of normalized canonical coordinate and generate a dense annotation covering the whole object surface instead of sparse semantic keypoints. Our work is most similar to NOCS. The main difference is that we work on point cloud data and the point feature is predicted instead of the normalized coordinate.

III. Problem Formulation

Consider an RGBD camera moving in an unknown environment. Suppose that a convolutional neural network, such as Mask R-CNN [22], is used to detect and segment objects in each RGB image. Suppose that an object tracking algorithm, such as SiamMask [23], tracks the detections over time. Suppose also that the camera pose is tracked using a SLAM algorithm, such as ORB-SLAM2 [24]. Given the camera trajectory and the segmented RGBD pixels associated over time, we can construct a point cloud $\mathbf{X} \in \mathbb{R}^{N\times 3}$ of the object in the world frame by accumulating the partial views and projecting them using the estimated camera poses.

Assume we have a small database of object models for the detected category $\mathcal{Y} = \{\mathbf{Y}_1, \dots, \mathbf{Y}_k\}$, where $\mathbf{Y}_i \in \mathbb{R}^{M_i\times 3}$ is a point cloud. Given a query point cloud $\mathbf{X}$ observed online, we want to find a similar model $\mathbf{Y} \in \mathcal{Y}$ and estimate the pose $\mathbf{T} := [\mathbf{R}\ \mathbf{p};\ \mathbf{0}^\top\ 1] \in SE(3)$ that aligns it with $\mathbf{X}$. Define a distance function between an incomplete point cloud $\mathbf{X}$ and complete point cloud $\mathbf{Y}$ as $d(\mathbf{X},\mathbf{Y}) : \mathbb{R}^{N\times 3}\times\mathbb{R}^{M\times 3} \to \mathbb{R}^+$. Our objective is to find an existing model $\mathbf{Y}^*$ and its pose $\mathbf{T}^*$ such that it fits an incomplete observation $\mathbf{X}$ well:

$$\mathbf{Y}^*, \mathbf{T}^* = \operatorname*{arg\,min}_{\mathbf{Y}\in\mathcal{Y},\mathbf{T}\in SE(3)} d(\mathbf{T}(\mathbf{X}), \mathbf{Y}), \tag{1}$$

where the transformation is defined as:

$$\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N]^\top \quad \mathbf{T}(\mathbf{X}) = [\mathbf{R}\mathbf{x}_1 + \mathbf{p} \cdots \mathbf{R}\mathbf{x}_N + \mathbf{p}]^\top. \tag{2}$$

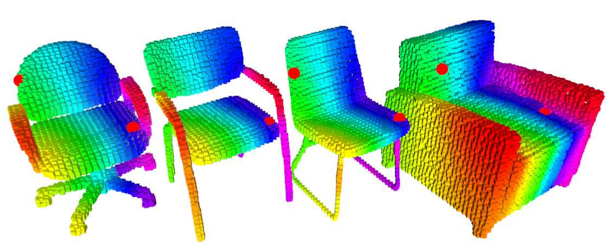
Fig. 2: Visualization of different chairs in Normalized Canonical Coordinates (NCCs). Colors indicate the coordinates. The red dots annotate the same coordinate on different objects. Note that the red dot on the second chair is occluded by the back. Although not all points can be associated well in the NCCs, many positive matching pairs can be discovered.

IV. Background

A. Sparse Convolution

Convolutional neural networks (CNNs) can be applied to 3D data directly by extending all 2D modules by one dimension. However, this not only increases the computation needed for training but also ignores the sparse structure of 3D data such as point clouds. Sparse convolution [12] is an approach for efficient feature extraction on structured 3D data. Suppose $\mathbf{c} \in \mathbb{Z}^n$ is a coordinate in an n-dimensional quantized space and its associated feature vector is $\mathbf{x_c}$. Define $\mathcal{V}^n(K)$ as the set of unit coordinate offsets around $\mathbf{0}$ in n-D space with size K in each dimension. For example, $\mathcal{V}^1(3) = \{-1, 0, 1\}, \mathcal{V}^2(3) = \{\mathcal{V}^1(3) \times \mathcal{V}^1(3)\}$. Standard dense convolution combines features from all the region nearby:

$$\mathbf{x}^{\text{out}}_{\mathbf{c}} = \sum_{\mathbf{i}\in\mathcal{V}^n(K)} \mathbf{W_i}\mathbf{x}^{\text{in}}_{\mathbf{c}+\mathbf{i}}. \tag{3}$$

For sparse convolution, the selected offset $\mathbf{i}$ is based on the non-empty locations of the input. Suppose we can define the non-empty coordinate space $\mathcal{C}^{\text{in}}$ for $\mathbf{c}^{\text{in}}$ and $\mathcal{C}^{\text{out}}$ for $\mathbf{c}^{\text{out}}$. Then the sparse convolution is defined as

$$\mathbf{x}^{\text{out}}_{\mathbf{c}} = \sum_{\mathbf{i}\in\mathcal{U}(\mathbf{c},\mathcal{C}^{\text{in}})} \mathbf{W_i}\mathbf{x}^{\text{in}}_{\mathbf{c}+\mathbf{i}} \text{ for } \mathbf{c} \in \mathcal{C}^{\text{out}} \tag{4}$$

, where $\mathcal{U}(\mathbf{c},\mathcal{C}^{\text{in}}) = \{\mathbf{i}|\mathbf{c}+\mathbf{i} \in \mathcal{C}^{\text{in}}\}$ only consider the existing adjacent feature value.

A fully convolutional network (FCN) [25] uses convolution and deconvolution to build the network structure without maxpooling or upsampling layers. FCN is suitable for dense prediction and its kernels have larger reception fields. The idea of fully convolutional layers can be combined with sparse convolution by using only convolutional layers.

B. Normalized Canonical Coordinate

The task of matching different objects is challenging. When scans are obtained from different object instance, the definition of good alignment becomes vague since a rigid transformation cannot eliminate the intrinsic shape difference. If the problem is restricted to aligning objects within the same category, we can leverage the conventions of object canonical pose. For example, a chair always has a seat and often a back and a canonical chair pose can be defined accordingly. The Normalized Object Coordinate Space (NOCS) is proposed in [4] and here we call this concept as Normalized Canonical Coordinate (NCC). The normalized object bounding box has a unit-length diagonal and is centered at the zero. Fig. 2 visualizes some examples of the chairs in the category NCC. The NCC of a object can be recovered given its pose and scale. NCC can bridge different objects with different poses and shapes as a intermediate pose-invariant shape representation.

V. Technical Approach

Given a source point cloud $\mathbf{X}$ and target point cloud $\mathbf{Y}$, we determine the transformation $\mathbf{T}$ that aligns $\mathbf{X}$ to $\mathbf{Y}$ in two steps. First, we extract features from both point clouds and establish point-to-point correspondences based on the features. Next, given the correspondences, we solve the alignment problem using a robust registration algorithm such as RANSAC [26] or TEASER [27].

A. Feature Extraction Model

Given a point cloud $\mathbf{X}$, we define a neural network model $f(\mathbf{X};\boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$ to generate a set of point cloud features:

$$\mathbf{F}_{\mathbf{X}} := f(\mathbf{X};\boldsymbol{\theta}) = \{\mathbf{f}_1, \dots, \mathbf{f}_N\} \tag{5}$$

where $\mathbf{f}_i \in \mathbb{R}^k$ is the feature associated with point $\mathbf{x}_i$. We build a feature extractor using the ResUNet network proposed in FCGF [5], which introduces residual blocks [28] into the UNet architecture [29] . One normalization layer is added before the output to generate unit-norm features. The network is trained using metric learning with training pairs generated as described in Sec. V-B. We set the feature dimension to $k = 32$.

B. Pairs Matching

The main idea of metric learning is to learn a distance metric between objects in order to establish similarity or dissimilarity, which aligns with our goal. The goal of 3D point cloud feature learning is to find a feature space where matching pairs should be close to each other while non-matching pairs should be far apart.

For the metric learning method, the generation of positive and negative pairs is at least as important as the loss design. 3DMatch [9] generates matching pairs between RGB-D scans from the camera poses recovered from different 3D reconstruction algorithm. Once two scans are aligned, positive pairs can be generated by local neighbor search, while the negative pairs are the remaining ones with larger distance. Define the positive matching pair set between two aligned point clouds $\mathbf{X}, \mathbf{Y}$ as:

$$p(\mathbf{X}, \mathbf{Y}) = \{(i,j) | \|\mathbf{x}_i - \mathbf{y}_j\| < \tau, i \le N, j \le M, i,j \in \mathbb{N}\}, \tag{6}$$

where τ is the local neighbor search region. During training we usually only keep a subset of the positive matching pairs. Negative pairs are sampled from the complement set of the positive pair set.

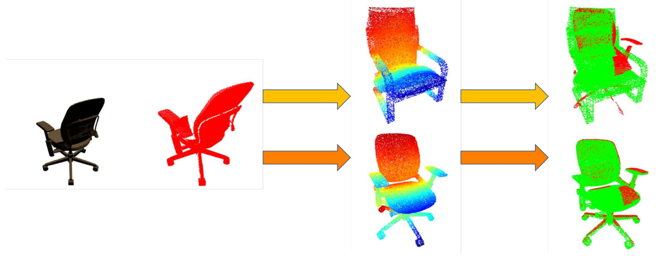
Fig. 3: The observed point cloud (left) can be aligned with different CAD models (middle) in normalized canonical coordinate (right) to generate matching pairs. In the right column the observed is in red and the model is in green.

We use the idea of NCC to bridge the connection between different object instances from the same category. We can convert an object point cloud $\mathbf{X}$ to the canonical frame with its pose $\mathbf{T}_{\mathbf{X}} \in SE(3)$ annotation and normalize it with the scale $s_{\mathbf{X}} \in \mathbb{R}$

$$\mathbf{X}_{\text{NCC}} = \text{NCC}(\mathbf{X}, s_{\mathbf{X}}, \mathbf{T}_{\mathbf{X}}) = s_{\mathbf{X}}^{-1} \cdot \mathbf{T}_{\mathbf{X}}^{-1}(\mathbf{X}) \tag{7}$$

where $\mathbf{T}_{\mathbf{X}}^{-1}(\cdot)$ follows the same definition in eq. (2). For two different object instances point clouds $\mathbf{X}, \mathbf{Y}$, we convert them both into the NCC and find the positive matching pair set

$$p'(\mathbf{X}, \mathbf{Y}) = p(\mathbf{X}_{\text{NCC}}, \mathbf{Y}_{\text{NCC}}) \tag{8}$$

as shown in Fig. 3.

Our main intuition is that though we cannot get most of the points perfectly aligned, there are still enough roughly-aligned parts. Especially if we focus on the model pairs that are more similar than a random pick, we can generate quite a lot reasonable positive matching pair by extending the local neighbor search radius τ in eq. (6). More details on matching pair generation and training on our customized dataset are introduced in Section VI.

C. Metric Learning Loss

Unlike other learning-based 3D feature extractors which usually have some hand-crafted pre-processing step to concatenate some local feature, we uses fully-convolutional sparse convolution to automatically extract features. We define contrastive loss function [30] used in metric learning for training. Assume $\mathbf{x}, \mathbf{y}$ are two points from different but aligned point clouds $\mathbf{X}, \mathbf{Y}$ and i, j are their associated indices respectively. The associated features generate through feature extractor model $f(;\boldsymbol{\theta})$ in eq. (5) are $\mathbf{fx}, \mathbf{fy}$. $m_{\mathbf{xy}}$ indicates the matching information.

$$m_{\mathbf{xy}} = \begin{cases} 1, & \text{if } (i,j) \in p(\mathbf{X}, \mathbf{Y}) \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

See eq. (6) for the definition of positive matching pairs $p(\mathbf{X}, \mathbf{Y})$. Define $d(\cdot,\cdot)$ as a distance function between features. The contrastive loss is defined as

$$L_{\text{con}}(\mathbf{fx}, \mathbf{fy}) = m_{\mathbf{xy}}(d(\mathbf{fx}, \mathbf{fy}) - p_{\text{pos}})^2 + \bar{m}_{\mathbf{xy}}(d(\mathbf{fx}, \mathbf{fy}) - p_{\text{neg}})^2 \tag{10}$$

where $\bar{m}_{\mathbf{xy}} = 1 - m_{\mathbf{xy}}$, $p_{\text{pos}}, p_{\text{neg}}$ are distance threshold for positive and negative pair. These thresholds should be design such that the positive pairs move closer and the negative pairs get separated when the contrastive loss is decreasing. We set $p_{\text{pos}} = 0.1, p_{\text{neg}} = 1.4$ for our normalized feature vectors. And the 2-norm is used for distance function between features.

D. Registration with Correspondences

After the training, we have a neural network model as the feature extractor function $f(\cdot;\boldsymbol{\theta})$. We fixing the $\boldsymbol{\theta}$, use the feature extraction model for inference and extract point-wise features $\mathbf{F}_{\mathbf{X}}, \mathbf{F}_{\mathbf{Y}}$ for the point clouds $\mathbf{X}, \mathbf{Y}$ as in eq. (5). We can perform the nearest-neighbor search for each point in $\mathbf{X}$ to generate correspondences, which is similar to eq. (6) but calculate the Euclidean distance between feature vectors instead of point coordinates.

$$\begin{aligned} p_f(\mathbf{F}_{\mathbf{X}}, \mathbf{F}_{\mathbf{Y}}) = \{(i, j_i) | j_i = \arg\min_j d(\mathbf{f x}_i, \mathbf{f y}_j), \\ i \leq N, j_i \leq M, i, j_i \in \mathbb{N}\} \end{aligned} \tag{11}$$

We then solve the pose estimation, or the point cloud registration problem with correspondences using robust methods like RANSAC [26]. In each iteration, RANSAC randomly samples k pairs of matching points $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{k}$ and solve the minimization problem

$$\mathbf{T}^* = \arg \min_{T \in SE(3)} \sum_{i=1}^{k} \|T(\mathbf{x}_i) - \mathbf{y}_i\|^2 \tag{12}$$

using Kabsch algorithm [31]. Then the $\mathbf{T}^*$ is evaluate on the whole set to see how many inliers it contains as the consensus set. The estimation with largest consensus set is maintained. We use the RANSAC implementation in Open3D [32].

VI. Experiments

A. Dataset

Our dataset is built from Scan2CAD [17]. We focus on the category of chair and we select the scenes from subcategory of Lobby and Conference Room in ScanNet [19]. Based on the appearance annotations of Scan2CAD we collect 137 chair models for training and 42 chair models for testing, both from ShapeNet [18]. We build a synthetic dataset from these ShapeNet models in canonical pose. In the synthetic dataset, for each CAD model we render depth images from 10 fixed viewpoints and convert them into point clouds.

To prepare for the non-identical model matching, we decide to annotate the model neighboring pairs instead of matching each pair of them. This step acts as providing an oracle for choosing the $\mathbf{Y}^*$ in eq. (1). We choose the 3-nearest-neighbors for each model's point cloud based on the Earth Mover's distance (EMD) [33]. For two point sets $\mathbf{X}, \mathbf{Y}$ with same size, the Earth Mover's distance is defined as

$$d_{EMD}(\mathbf{X}, \mathbf{Y}) = \min_{\Phi : \mathbf{X} \to \mathbf{Y}} \sum_{\mathbf{x} \in \mathbf{X}} \|\mathbf{x} - \Phi(\mathbf{x})\|_2 \tag{13}$$

where $\Phi : \mathbf{X} \to \mathbf{Y}$ is a one-to-one correspondence. We sample 2048 points on each CAD model to calculate the EMD. For both training and testing set the neighbors are annotated. Notice for the CAD models in the testing set their neighbors are selected only in the training set.

For the real-world data, we look into the scene reconstruction meshes in ScanNet. From the 117 training scenes we segment out 974 chair meshes and convert them into point clouds. Also for the 45 testing scenes 331 chairs are selected. For each of these chair the Scan2CAD provides its associated ShapeNet CAD model and its pose. We use this to evaluate the performance of real-world point cloud pose estimation. Besides, some RGB-D images with object segmentations are collected for quantitative evaluation in Sec. VI-D.

B. Evaluation Metric

We borrow the idea of matched fragments from [34] and evaluate the accuracy of matching pairs. Assume $\mathbf{X}$ and $\mathbf{Y}$ are two associated point clouds with the matching between the same index. The groundtruth transformation from $\mathcal{X}$ to $\mathcal{Y}$ is $\mathbf{T}^*$. The matching accuracy evaluates how many matching pairs can be aligned within some threshold τ_1 after applying the groundtruth transformation.

$$\text{MatchAcc}(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left(\|\mathbf{T}^*(\mathbf{x}_i) - \mathbf{y}_i\|_2 < \tau_1\right) \tag{14}$$

Here we set $\tau_1 = 0.05\ m$. In [34] the authors set the positive inlier ratio to be 0.05, indicating that as long as there are 5% of the matching pairs we can get a good registration result using some robust algorithms to remove the outliers.

For cross-instance matching we cannot have a general matching distance due to the shape variations. Complementarily, we can also estimate the relative pose and measure the relative error. The pose can be represented by the rotation part and translation part. Define the groundtruth pose as $(\mathbf{R}^*, \mathbf{p}^*)$ and the estimated pose as $(\hat{\mathbf{R}}, \hat{\mathbf{p}})$, where $\mathbf{R}^*, \hat{\mathbf{R}} \in SO(3), \mathbf{p}^*, \hat{\mathbf{p}} \in \mathbb{R}^3$. We can decouple them to define the relative rotation error (RRE)

$$\text{RRE}(\hat{\mathbf{R}}, \mathbf{R}^*) = \arccos\{[\mathbf{tr}\,(\hat{\mathbf{R}}^T \mathbf{R}^*) - 1]/2\} \tag{15}$$

and the relative translation error (RTE)

$$\text{RRE}(\hat{\mathbf{p}}, \mathbf{p}^*) = \|\hat{\mathbf{p}} - \mathbf{p}^*\|_2 \tag{16}$$

C. Synthetic Dataset: ShapeNet

From the synthetic dataset, each CAD model in the training set and its 10 point clouds observations at different poses are used for model training. We use the curriculum learning [35] to verify our cross-instance matching data augmentation idea so that the training procedure is more stable. We first train on the same-instance pairs. Each CAD model can pair with its own incomplete point clouds to generate 10 same-instance pairs. The baseline model is trained on the pairs for 100 epochs.

Afterwards we introduce cross-instance matching pairs for the training. Based on the EMD neighbors we compute, we generate pairs between each CAD model and their EMD-neighboring CAD models. Also we pair the incomplete depth image point cloud with its EMD-neighboring CAD models. We keep a few same-instance pairs as used in the baseline

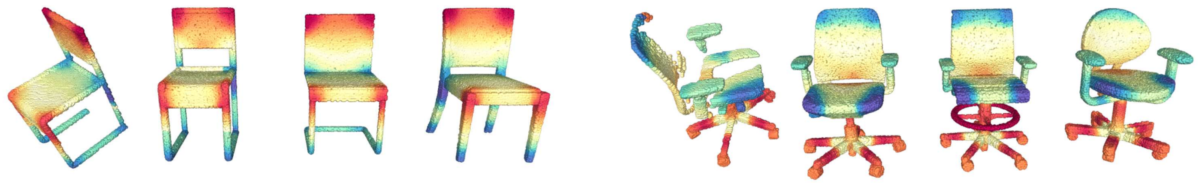
Fig. 4: Examples of point feature embedded by t-SNE. Column 1: pointcloud from single depth scan. Column 2: identical CAD model. Column 3&4: neighbor models.

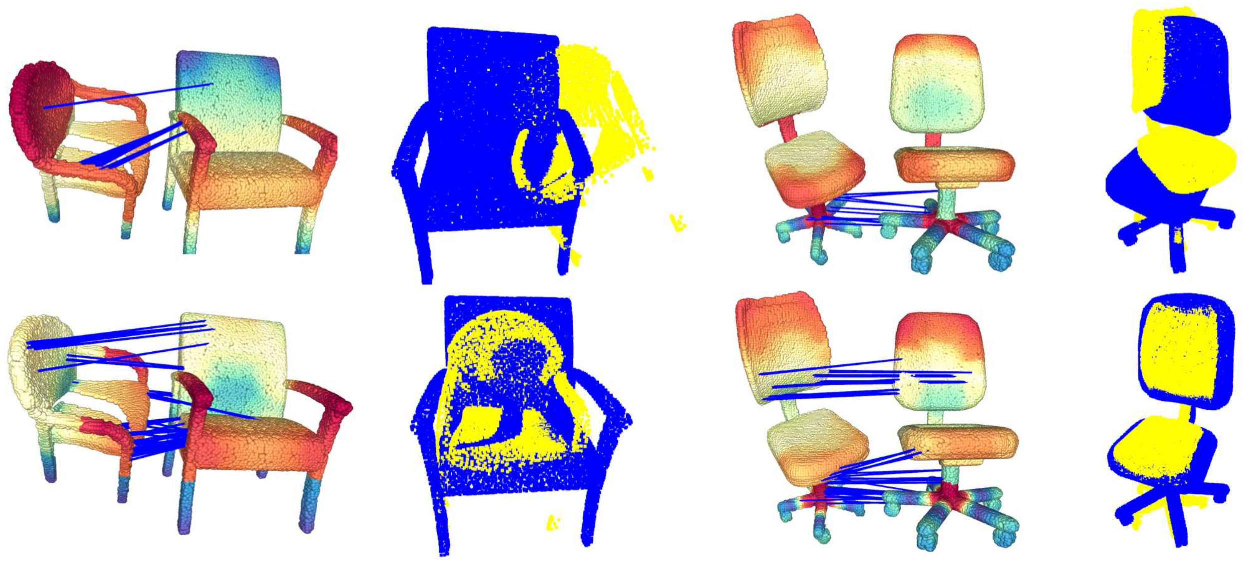
Fig. 5: Comparing the models trained with (bottom row) and without (top row) cross-instance matching data on synthetic data. Column 1 and 3 show the point feature embedded by t-SNE. The depth scan is on the left and the CAD model is on the right. The blue lines indicate the accurate match among the total 1000 sampled matches. Column 2 and 4 show the final registration results based on 1000 matching pairs. The CAD model is in blue and the depth scan is in yellow. The negative matching pairs are not shown, which cause the pose estimation failure.

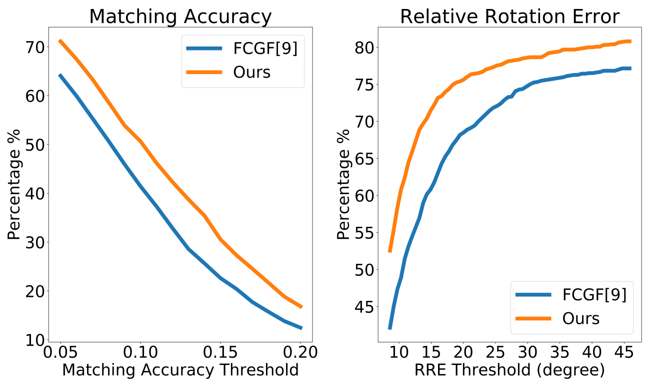

Fig. 6: Results on ShapeNet synthetic data. Left: matching accuracy. Right: relative rotation error. FCGF [5] is the baseline model w/o cross-instance matching.

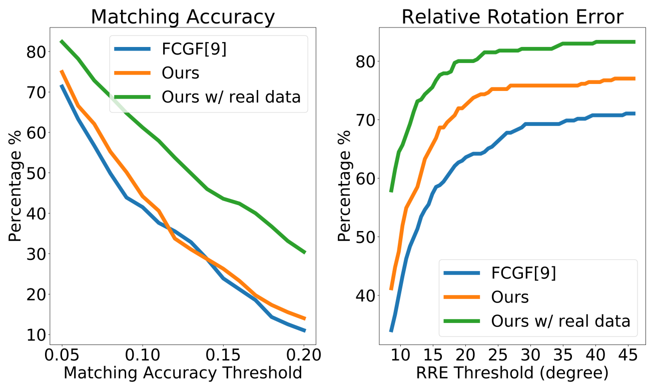

Fig. 7: Results on ScanNet real-world data. Left: matching accuracy. Right: relative rotation error. FCGF [5] is the baseline model w/o cross-instance matching. Our w/ real data is the model trained with real-world point clouds.

to make the training more stable. The model with cross-instance matching is trained for another 100 epochs. For a fair comparison, we also keep training the baseline model for another 100 epochs solely on same-instance pairs and name it as the model without cross-instance matching.

By pairing with EMD neighbor we can discover more meaningful and stable matching pairs. And we may derive implicit connections with more than the neighbors since the neighbor CAD also has its own neighbor. Notice that here the neighbor CADs are all included in the original training set. We are not introducing new data but generate new matching pairs in the original database. This can be viewed as a data augmentation strategy for this specific feature learning task.

TABLE I: The percentage of the testing data within the RRE/RTE threshold. The RRE values are aligned with Fig. 6, 7. Ours+ is the model trained on the real-world data.

Dataset	Method	RRE (degree)			RTE (cm)	
		10	20	30	5	10
Synthetic	FCGF[5]	48.02	68.49	74.84	47.38	72.78
	Ours	**59.21**	**75.56**	**78.57**	**55.32**	**78.49**
Real-world	FCGF[5]	41.49	63.58	69.25	45.97	68.06
	Ours	50.15	72.54	75.82	54.33	74.33
	Ours+	**65.07**	**80.00**	**82.09**	**69.55**	**83.88**

In Fig. 4 we visualize the point cloud features between the partial point cloud and the CAD models by embedding the high dimensional feature vector using t-SNE [36] and coloring the point cloud.

During testing, the incomplete point clouds associated with the CAD models in the test set are used as observation. We assume that we don't have access to the test models that generate these observation. Instead, we have the pre-selected EMD neighbors of each test CAD model from the training set. For each observed point cloud, three models from the training set is provided as candidates and we use the trained model for the task of cross-instance matching and pose estimation. The comparison result is shown in Fig. 6 and Table. I. We visualize some comparing cases on pose estimation in Fig. 5. We can see that with cross-instance matching data included for the training, the model can generate more consistent feature vectors across different instance, estimate more inlier matching pairs and also perform better on the pose estimation task.

D. Real-world Dataset: ScanNet

For the real-world dataset we are trying to match the CADs with the segmented point cloud from the scene reconstruction mesh. Here we only use the annotated CAD model from Scan2CAD but not the neighbor models because the potential sim-real domain gap. We do a transfer learning, training on the basis of the model learned on the synthetic dataset which has already learned from cross-instance data. Fig. 7 is showing the performance of different models and quantitative results are listed in Table. I. With cross-instance matching training the model can have a better performance on the real-world data even there is domain transfer gap and the model has never seen real-world data. Also with transfer learning on the real-world data the model can have higher matching accuracy and the lower rotation and translation error.

We visualize some qualitative result in Fig. 8. Here we extract the observed object point cloud from a segmented RGB-D image. The first and second rows show examples with relatively complete observations. The third example is incomplete due to image truncation. The fourth and the fifth observed objects are sharing the same CAD model for pose estimation. Most of them are showing reasonable performance. There are some failure cases shown in Fig. 9. When the inlier matching pairs are limited, the RANSAC algorithm provides unstable estimation. We observe that observation from the back of the chair is challenging for the model because it is very similar to a plane without specific geometric structure. Also sometimes the pose estimation

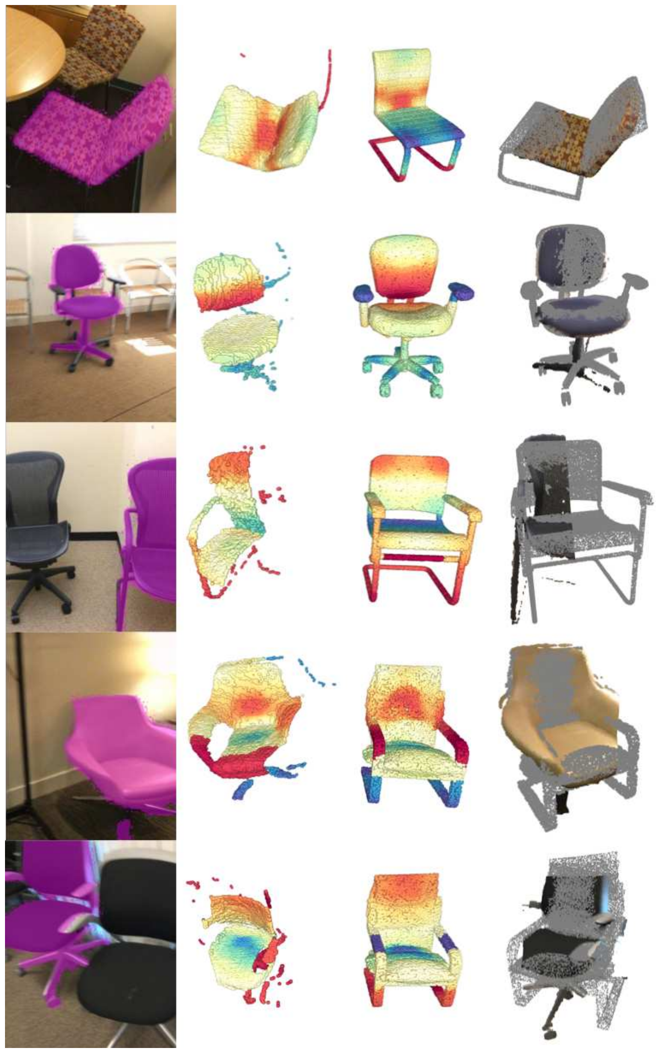
Fig. 8: Examples on matching the point cloud segmented from a single RGB-D image. Column 1: RGB images and object segmentatin masks in purple. Column 2 & 3: color-coded point cloud features of the observed objects and the CAD models. Column 4: Alignment results with the CAD models painted in grey.

of cube-shape sofa chair is noisy. One possible reason is that there are too much outlier matching pairs on the plane surfaces. One potential solution to this is to train a classifier to justify useful points and their features for pose estimation instead of using all of them or randomly sampling a subset.

VII. Conclusion

In this work, we perform the category-specific cross instance point cloud matching and pose estimation. We propose to convert different instances from the same category into the normalized canonical coordinate to make a good alignment, which is helpful for positive matching pairs generation in metric learning. We train a fully-convolutional sparse convolution model for point cloud feature extraction. During testing, we use RANSAC to estimate the pose with associations from the features. Our future work will focus on the prediction and optimization in the shape-deformation space. It is interesting to investigate more on the problem of CAD model retrieval given partial observations. Given a

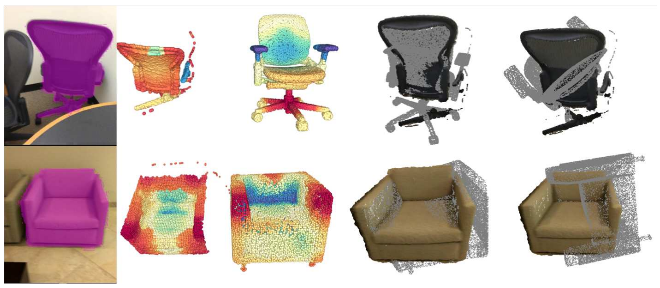

Fig. 9: Failure cases on RGB-D point cloud registration. Column 1: RGB images and object segmentatin masks in purple. Column 2 & 3: color-coded point cloud features of the observed objects and the CAD models. Column 4 & 5: Different alignment results after running RANSAC, with the CAD models painted in grey.

relative good pose and shape initialization, we want to refine the object pose and shape jointly.

References

[1] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013, pp. 1352–1359.

[2] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, “Probabilistic Data Association for Semantic SLAM,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1722–1729.

[3] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, “6-DoF Object Pose from Semantic Keypoints,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2011–2018.

[4] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, “Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 2637–2646.

[5] C. Choy, J. Park, and V. Koltun, “Fully Convolutional Geometric Features,” in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 8957–8965.

[6] A. E. Johnson and M. Hebert, “Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, May 1999.

[7] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 3212–3217.

[8] S. Salti, F. Tombari, and L. D. Stefano, “SHOT: Unique signatures of histograms for surface and texture description,” *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014.

[9] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, “3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 199–208.

[10] Z. J. Yew and G. H. Lee, “3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 630–646.

[11] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, “The Perfect Match: 3D Point Cloud Matching With Smoothed Densities,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 5540–5549.

[12] C. Choy, J. Gwak, and S. Savarese, “4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 3070–3079.

[13] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, “PVNet: Pixel-Wise Voting Network for 6DoF Pose Estimation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 4556–4565.

[14] S. Zakharov, I. Shugurov, and S. Ilic, “DPOD: 6D Pose Object Detector and Refiner,” in *IEEE/CVF International Conference on Computer Vision (CVPR)*, 2019, pp. 1941–1950.

[15] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, “PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Estimation,” in *Proceedings of Robotics: Science and Systems (R:SS)*, June 2019.

[16] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik, “Aligning 3D models to RGB-D images of cluttered scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 4731–4740.

[17] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner, “Scan2CAD: Learning CAD Model Alignment in RGB-D Scans,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 2609–2618.

[18] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, “ShapeNet: An Information-Rich 3D Model Repository,” *ArXiv*, vol. abs/1512.03012, 2015.

[19] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “Scannet: Richly-annotated 3d reconstructions of indoor scenes,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2432–2443.

[20] Y. Xiang, R. Mottaghi, and S. Savarese, “Beyond PASCAL: A benchmark for 3D object detection in the wild,” in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2014, pp. 75–82.

[21] X. Zhou, A. Karpur, L. Luo, and Q. Huang, “StarMap for Category-Agnostic Keypoint and Viewpoint Estimation,” in *European Conference on Computer Vision (ECCV)*, 2018, pp. 328–345.

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, Feb 2020.

[23] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. S. Torr, “Fast Online Object Tracking and Segmentation: A Unifying Approach,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 1328–1338.

[24] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct 2017.

[25] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[26] M. A. Fischler and R. C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, 1987, p. 726–740.

[27] H. Yang, J. Shi, and L. Carlone, “TEASER: Fast and Certifiable Point Cloud Registration,” *arXiv: 2001.07715*, 2020.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[29] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[30] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality Reduction by Learning an Invariant Mapping,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, June 2006, pp. 1735–1742.

[31] W. Kabsch, “A solution for the best rotation to relate two sets of vectors,” *Acta Crystallographica*, vol. 32, no. 5, pp. 922–923, 1976.

[32] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.

[33] H. Fan, H. Su, and L. Guibas, “A Point Set Generation Network for 3D Object Reconstruction from a Single Image,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2463–2471.

[34] H. Deng, T. Birdal, and S. Ilic, “PPFNet: Global Context Aware Local Features for Robust 3D Point Matching,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 195–205.

[35] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum Learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, p. 41–48.

[36] L. v. d. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.