This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

DeepPRO: Deep Partial Point Cloud Registration of Objects

Donghoon Lee Onur C. Hamsici Steven Feng Prachee Sharma Thorsten Gernoth
Apple
{donghoonlee,ohamsici,shuo_feng,prachee_sharma,tgernoth}@apple.com

Abstract

*We consider the problem of online and real-time registration of partial point clouds obtained from an unseen real-world rigid object without knowing its 3D model. The point cloud is partial as it is obtained by a depth sensor capturing only the visible part of the object from a certain viewpoint. It introduces two main challenges: 1) two partial point clouds do not fully overlap and 2) keypoints tend to be less reliable when the visible part of the object does not have salient local structures. To address these issues, we propose DeepPRO, a keypoint-free and an end-to-end trainable deep neural network. Its core idea is inspired by how humans align two point clouds: we can imagine how two point clouds will look like after the registration based on their shape. To realize the idea, DeepPRO has inputs of two partial point clouds and directly predicts the point-wise location of the aligned point cloud. By preserving the ordering of points during the prediction, we enjoy dense correspondences between input and predicted point clouds when inferring rigid transform parameters. We conduct extensive experiments on the real-world Linemod and synthetic ModelNet40 datasets. In addition, we collect and evaluate on the PRO1k dataset, a large-scale version of Linemod meant to test generalization to real-world scans. Results show that DeepPRO achieves the best accuracy against thirteen strong baseline methods, e.g., 2.2*mm *ADD on the Linemod dataset, while running 50 fps on mobile devices.*

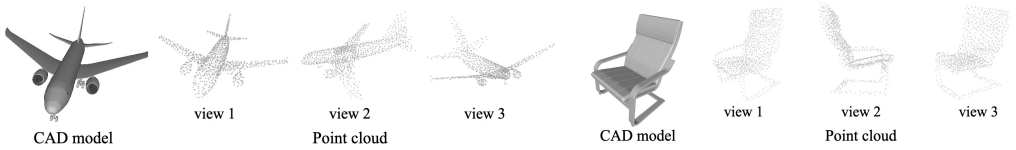
1. Introduction

In this work, we are interested in developing an online and a real-time point cloud registration algorithm for unseen real-world objects. In other words, as we move a depth sensor or an object, we aim to find 3D rotation and translation parameters between the depth sensor and object based on the captured point cloud at the current and previous frames. Estimated parameters by the registration algorithm is essential for a number of applications such as tracking and reconstruction of the 3D object. We list key challenges for this problem that are not fully addressed in existing literature.

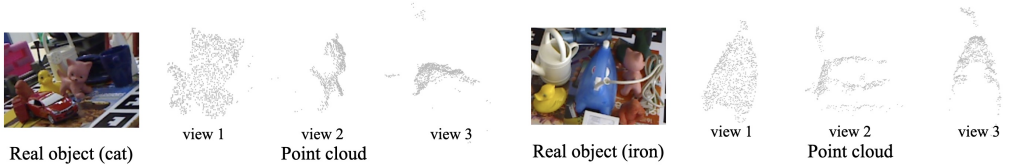
First, we observe a part of an object from a certain viewpoint at a time. Without knowing 3D model of unseen objects, the only input are two partial point clouds. Since input partial point clouds do not fully overlap with each other, unlike previous works [32, 39, 28], we cannot make one-to-one correspondence assumption between input points or point clusters. In addition, recent works on object point cloud registration rely on 3D CAD models in the ModelNet40 dataset [35] as shown in Table 1. Due to a big domain gap between real-world partial point clouds and synthetic 3D CAD models as shown in Figure 1, [2, 32, 33, 38] use different heuristic approaches to simulate the partial observation, e.g., pick a random point and gather its neighbors from the full point cloud. However, such methods do not properly model complex self-occlusions, object materials and sensor noises of real data. In our experiments, state-of-the-art models trained on the synthetic dataset do not generalize well on real-world data as shown in Figure 4 and Table 2.

Second, it is often difficult to extract meaningful number of precise keypoints for object point clouds. For outdoor-scale [12, 19] or indoor-scale [40] point clouds, it is relatively easy to extract keypoints as they have rich local structures and salient geometries [18, 9, 8]. However, for small objects, its partial point cloud might not always contain a number of distinguishable local structures for keypoints. In addition, irregular sensor noises around different object materials and depth edges exacerbate the reliability of keypoints. We empirically demonstrate that existing keypoint-based methods that work well on outdoor or indoor point clouds become less accurate on object point clouds in Table 2.

Third, the registration algorithm should be efficient and effective for a small baseline. We can focus on the small baseline since modern depth sensing devices typically capture at 30 fps or faster. Hence, the pose difference between two consecutive frames would be small. To achieve real-time performance, it is encouraged to avoid expensive multi-step iterative approaches such as the classical iterative closest point (ICP) [5] and its variants for global optimization [37]. However, these expensive methods are often adopted in offline point cloud registration systems [41, 1, 29] or used as an ad-hoc post-processing [32, 6] to improve the accuracy.



(a) Synthetic data with full 3D model.



(b) Real-world observation.

Figure 1: Here we show the difficulties of the point cloud registration for real-world objects. (a) Existing learning-based works rely on the synthetic CAD model of the object which can render point clouds without self-occlusion. (b) Our work focuses on real-world data. Captured partial point clouds only cover a part of the object. In addition, they have self-occlusion and irregular sensor noises (noises are better visible with a digital zoom). Shared content also gets smaller as the viewpoint changes.

Table 1: Existing learning-based point cloud registration methods on different input scales and data sources.

Input scale	Synthetic data	Real data
Outdoor, indoor	-	[4, 6, 7, 8, 13, 18, 39]
Object	[2, 15, 27, 32, 33, 38, 39]	Our interest

To address above issues, we first use a real-world dataset rather than augmenting synthetic data. We extensively evaluate 13 point cloud registration methods on the real-world Linemod dataset [14]. In addition, we collect and evaluate on a large-scale dataset, called PRO1k in this paper. Second, we propose a keypoint-free algorithm called Deep Partial point cloud Registration of Objects (DeepPRO). The algorithm predicts the registered location of each point in one point cloud in the other point cloud's coordinate using a point cloud generation network in a way that the predicted points preserve the ordering of the original point cloud. It establishes dense correspondences between points in two coordinates which allow us to effectively infer rigid transform parameters. Third, we train DeepPRO with point cloud pairs in the small baseline and estimate its runtime on a mobile device to make sure it works in real-time for our use case.

Experimental results on two real-world and one synthetic object databases show that DeepPRO works favorably against existing methods. For example, for unseen objects, we achieve $1.19°$ error in rotation, $1.46\,\mathrm{cm}$ error in translation, and $0.22\,\mathrm{cm}$ 3D distance error on the Linemod dataset. In addition, DeepPRO shows faster than 50 fps runtime on mobile devices without any optimization techniques such as network quantization or pruning.

2. Related Work

Geometric registration and pose estimation are gaining more attention with 1) the growing application in robotics, autonomous driving, and augmented reality and 2) widely available sensors from Lidar on autonomous vehicles to RGB-D sensor on mobile devices. Therefore, the problem setup and objectives are diverse across different applications. In this paper, we focus on reviewing existing works relevant to registering small spatial scale partial point clouds.

ICP [5] is a classical method in geometric registration, which iteratively extracts matching points between two point clouds and moves one point cloud closer to the other based on estimated rotation and translation parameters until convergence. In general, ICP works well when a reliable initialization of point clouds is available. To alleviate this issue, Go-ICP [37] setups a branch-and-bound framework with the higher order of computation cost. FGR [42] initializes the correspondence with FPFH feature [26] which are sensitive to occlusions and partial point clouds. These methods are generic so particular knowledge on objects are not required.

In recent years, as shown in Table 1, data-driven approaches are emerging for registering point clouds based on the development of point cloud encoders such as PointNet [25], DGCNN [34], KPConv [30], or FCGF [7]. They typically encode two point clouds using the Siamese-style network and compare the output features to find the rotation and translation between them. We briefly review recent object-scale point cloud registration methods.

Deep closest point (DCP) [32] approximates the combinatorial matching in ICP based on the attention module in Transformer [31] and a probabilistic matching of feature vectors. A differentiable SVD head is incorporated into the network to estimate the transform parameters. PointNetLK

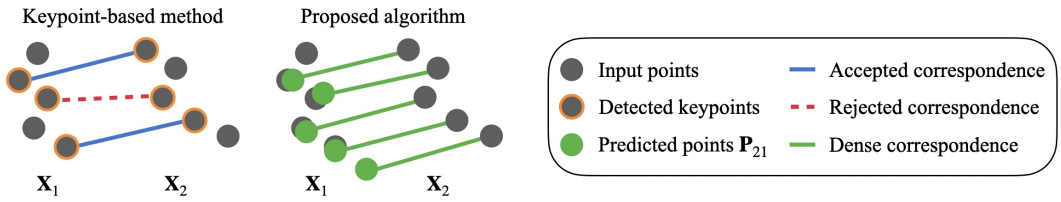


Figure 2: Key differences between the existing method and proposed algorithm. Existing keypoint-based approaches aim to find correspondences between two input point clouds $\mathbf{X}_1$ and $\mathbf{X}_2$. On the other hand, the proposed algorithm predicts $\mathbf{P}_{21}$, the registered location of each point in $\mathbf{X}_2$ in $\mathbf{X}_1$'s coordinate. It lets us to build keypoint-free dense correspondences.

[2] uses the idea of the Lucas-Kanade algorithm [21]. Its formulation is based on the Jacobian of globally pooled feature vectors and the framework is unrolled to recurrent neural network for training. PRNet [33] is a keypoint based method which aims to predict a sharp mapping between keypoints using Gumbel-Softmax [16, 22]. The temperature parameter, which controls the shape of the softmax probability distribution, is automatically determined based on the actor-critic method [23]. A feature-metric registration proposed in [15] minimizes the difference of encoded features of two point clouds for training the network.

3. Approach

The core idea of DeepPRO is shown in Figure 2. The goal is to find rotation and translation parameters, $\mathbf{R}_{21}$ and $\mathbf{t}_{21}$, which align the partial point cloud $\mathbf{X}_2 \in \mathbb{R}^{3\times N}$ observed at view 2 with $\mathbf{X}_1 \in \mathbb{R}^{3\times N}$ observed at view 1, i.e., $\mathbf{R}_{21}\mathbf{X}_2 + \mathbf{t}_{21}$, without any prior knowledge on the rigid object[1]. Unlike the mainstream keypoint-based methods, DeepPRO predicts $\mathbf{P}_{21}$ which describes how $\mathbf{X}_2$ will look like if it is registered with $\mathbf{X}_1$ in $\mathbf{X}_1$'s coordinate. Hence $\mathbf{X}_2$ and $\mathbf{P}_{21}$ inherit dense correspondences between two different coordinates which facilitates the point cloud registration.

We build a deep convolutional neural network and train it with the ground truth transform label as shown in Figure 3. It consists of four parts: a shared encoder, a conditional point cloud generation network, a transform estimation network, and a do-and-undo of the random rotation layer. Following the convention in literature [2, 32, 33, 38], we assume a known object region so that we can randomly sample N points from the object. From our experiments, DeepPRO works well for different number of N as shown in Table 5.

3.1. Network Architecture

Shared Encoder. We use a point cloud encoder which maps the input point cloud $\mathbf{X}_1$ to the per-point local feature $\mathbf{f}_1^l \in \mathbb{R}^{d\times N}$ and global feature $\mathbf{f}_1^g \in \mathbb{R}^{d\times 1}$. The same encoder is applied to $\mathbf{X}_2$ with shared parameters to get $\mathbf{f}_2^l$ and $\mathbf{f}_2^g$. In this work, we utilize DGCNN [34] architecture while it can be substituted with other point cloud encoders as long as one can get local and global features. Before we feed the point cloud to the encoder, both point clouds are zero-centered and normalized to a $[-0.5, 0.5]^3$ cube by dividing with a common scale factor. The scaling factor is determined by the maximum value of zero-centered point clouds. We store the mean of each point cloud, $\mu(\mathbf{X}_1)$ and $\mu(\mathbf{X}_2)$, and the scale factor $\mathbf{s}$ to de-normalize the point cloud later.

Conditional Point Cloud Generation Network. This module takes $\mathbf{f}_1^g$, $\mathbf{f}_2^g$, and $\mathbf{f}_2^l$ as input and outputs the predicted location of each point in $\mathbf{X}_2$ at viewpoint 1, i.e. $\mathbf{P}_{21}$. The global features are replicated and concatenated with the local features to form an input as shown in Figure 3. The network can be described as a generator which reconstructs $\mathbf{X}_2$ using $\mathbf{f}_2^g$ and $\mathbf{f}_2^l$ while conditioning on $\mathbf{f}_1^g$ to put it at the viewpoint 1, as $\mathbf{P}_{21}$. Note that, both $\mathbf{P}_{21}$ and $\mathbf{f}_2^l$ have the same ordering of points as in $\mathbf{X}_2$ since we use 1×1 convolutions on per point encoding which do not change the ordering of points. This embeds dense correspondences between $\mathbf{X}_2$ and $\mathbf{P}_{21}$ which will be useful for the transform estimation network to estimate rigid transform parameters. We use the hyperbolic tangent activation function after the last layer's output to contain predicted locations within $[-1, 1]^3$ cube. Then, we de-normalize the generated point cloud based on the previously stored normalization parameters $\mu(\mathbf{X}_1)$ and $\mathbf{s}$ to bring it to viewpoint 1, i.e., we obtain $\mathbf{P}_{21}$.

Transform Estimation Network. Given two point clouds $\mathbf{X}_2$ and $\mathbf{P}_{21}$ with dense correspondences, we estimate rotation and translation parameters. For a rigid transform, an SVD-based closed form solution [10] is widely used. However, we empirically found that putting SVD into the network makes end-to-end training unstable. Therefore, we build the transform estimation network to estimate parameters. Thanks to dense correspondences, we can simply concatenate $\mathbf{X}_2$ and $\mathbf{P}_{21}$ and feed it to the network which predicts the rotation in the form of quaternion. We parameterize the rotation axis $\mathbf{n} = (n_x, n_y, n_z)$ and rotation angle θ of a quaternion vector as $\mathbf{q} = \left(\cos(\frac{\pi\sigma(\theta)}{2}), \sin(\frac{\pi\sigma(\theta)}{2})\frac{\mathbf{n}}{\|\mathbf{n}\|}\right)$ where σ is the sigmoid function to represent the rotation $\mathbf{R}_{21}$

[1]For convenience, we explain transform from view 2 to 1 while the same logic can be applied to the other direction.

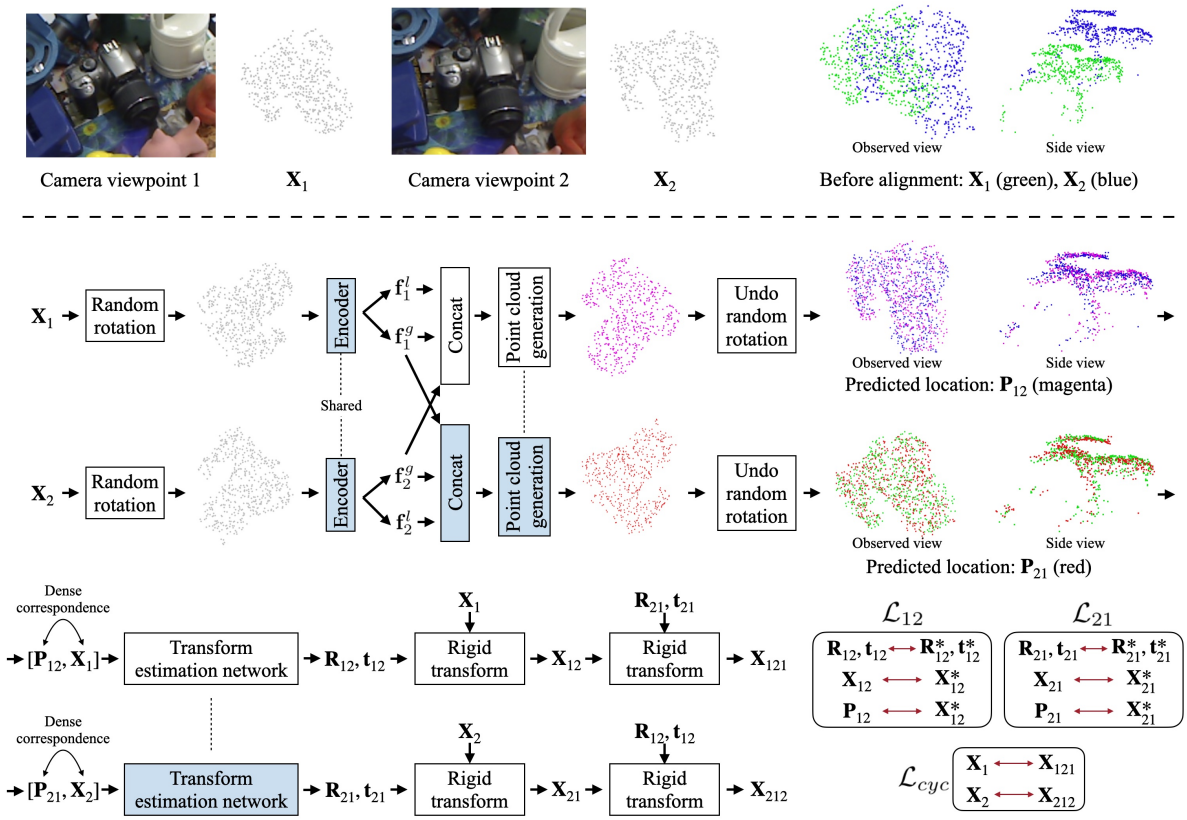


Figure 3: Overall flow of DeepPRO. A shared encoder outputs local and global features for each partial point cloud. Then, we predict the registered point-wise location of $\mathbf{X}_i$ in $\mathbf{X}_j$'s coordinate, i.e., $\mathbf{P}_{ij}$, using the shared point cloud generation network. As $\mathbf{P}_{ij}$ and $\mathbf{X}_i$ have the same ordering of points, we have dense correspondences between two coordinates. Therefore, we can concatenate $\mathbf{P}_{ij}$ and $\mathbf{X}_i$ and feed it to the transform estimation network to get $(\mathbf{R}_{ij}, \mathbf{t}_{ij})$. Random rotation is applied to augment data during training. For inference, we only execute boxes colored in blue. Red arrows show loss computation.

on the positive real hemisphere and $\|\mathbf{n}\| = \sqrt{n_x^2 + n_y^2 + n_z^2}$. Then, the estimated translation is calculated as $\mathbf{t}_{21} = \mu(\mathbf{P}_{21}) - \mathbf{R}_{21}\mu(\mathbf{X}_2)$. Using the predicted $\mathbf{R}_{21}$ and $\mathbf{t}_{21}$, we can actually move $\mathbf{X}_2$ to viewpoint 1, i.e., $\mathbf{X}_{21} = \mathbf{R}_{21}\mathbf{X}_1 + \mathbf{t}_{21}$. We normalize $\mathbf{X}_2$ and $\mathbf{P}_{21}$ based on the same normalization method used in the point cloud encoder.

Random Rotation Layer. The output of conditional point cloud generation is rotation and translation invariant since the alignment of $\mathbf{X}_1$ and $\mathbf{X}_2$ depends only on their shape. Therefore, during training, we exploit the transform invariant property by adding the random rotation layer as shown in Figure 3. The translation invariant part is handled during point cloud normalization, i.e., subtracting the mean. We undo this random rotation after the network predicts $\mathbf{P}_{21}$ so that $\mathbf{R}_{21}$ and $\mathbf{t}_{21}$ can be estimated with the original orientations of the input. For simplicity, we slightly abuse the notation $\mathbf{P}_{21}$ as the predicted point clouds after undoing the random rotation for the rest of the paper. In practice, we found that the network can be effectively trained with a limited perturbation, e.g., $\pm 5°$, in the random rotation layer.

3.2. Objective Function

The proposed network is trained with the following objective function to transform $\mathbf{X}_2$ to view 1:

$$\mathcal{L}_{21} = \mathcal{L}_{R,t}(\mathbf{R}_{21}, \mathbf{t}_{21}, \mathbf{R}^*_{21}, \mathbf{t}^*_{21}) + \mathcal{L}_{3D}(\mathbf{P}_{21}, \mathbf{X}_{21}, \mathbf{X}^*_{21}), \tag{1}$$

where $\mathcal{L}_{R,t}$ regulates the error in the rotation and translation parameter space and $\mathcal{L}_{3D}$ describes the distance in the 3D space. In addition, $\mathbf{X}^*_{21}$ is $\mathbf{X}_2$ transformed by ground truth rotation and translation parameters $\mathbf{R}^*_{21}$ and $\mathbf{t}^*_{21}$.

Rotation and Translation Loss. We minimize $\|(\mathbf{E}^*_{21})^{-1}\mathbf{E}_{21} - \mathbf{I}\|$ where $\mathbf{E} = [\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{4\times4}$ is an extrinsic matrix and $\mathbf{I}$ is the identity matrix. For the type of the norm, we tested L_1, L_2, and L_S (shrinkage loss [20]). We use L_S norm for regression losses in this paper, as we empirically achieve better results than for other norms.

Distance Loss in 3D Space. We consider two terms for the distance loss between point clouds as follows:

$$\sum_{\mathbf{p}_{21}\in\mathbf{P}_{21},\mathbf{x}_{21}\in\mathbf{X}_{21},\mathbf{x}^*_{21}\in\mathbf{X}^*_{21}} \frac{1}{N}(\|\mathbf{p}_{21}-\mathbf{x}^*_{21}\|+\|\mathbf{x}_{21}-\mathbf{x}^*_{21}\|), \tag{2}$$

where N is the number of points. The first term is designed to predict correct $\mathbf{P}_{21}$. Note that we can directly calculate $\mathbf{p}_{21}-\mathbf{x}^*_{21}$ without finding point-to-point matching as they already have dense correspondences as discussed in Section 3.1. The second term is the average distance of model points (ADD) [14] which measures the 3D distance between the ground truth and registered point clouds. It helps $\mathcal{L}_{R,t}$ to minimize the error of transform parameters.

Final Loss. The final loss considers bi-directional transform and a cyclic loss as follows:

$$\mathcal{L}=\mathcal{L}_{21}+\mathcal{L}_{12}+\mathcal{L}_{cyc}, \tag{3}$$

where $\mathcal{L}_{12}$ is defined similarly with $\mathcal{L}_{21}$ and $\mathcal{L}_{cyc}$ guides the network to have the inverse relationship between $(\mathbf{R}_{21},\mathbf{t}_{21})$ and $(\mathbf{R}_{12},\mathbf{t}_{12})$. It is defined as follows:

$$\mathcal{L}_{cyc}=\|\mathbf{X}_1-(\mathbf{R}_{21}\mathbf{X}_{12}+\mathbf{t}_{21})\|+\|\mathbf{X}_2-(\mathbf{R}_{12}\mathbf{X}_{21}+\mathbf{t}_{12})\|. \tag{4}$$

All loss functions and networks are end-to-end trainable.

4. Experimental Results

Due to space limit, we describe implementation details, baseline methods discussions, training stability, and more failure case analysis in the supplementary material.

Datasets. While outdoor and indoor point cloud registration methods are evaluated on real-world benchmark databases, such as KITTI [12], ETH [24], and 3DMatch [40], learning-based object-scale registration methods [32, 33, 27, 2, 15] are limited to a non-realistic synthetic ModelNet40 [35] dataset. To alleviate this problem, we train and test our algorithm on the real-world Linemod [14] and PRO1k dataset as well as synthetic ModelNet40 dataset.

The Linemod dataset is a well-known benchmark for 6-DOF object pose estimation. We gather the camera pose and object point cloud based on 2D masks of eleven asymmetric objects. Then, we build an input pair $(\mathbf{X}_1, \mathbf{X}_2)$ and calculate their ground truth label $(\mathbf{R}^*_{21}, \mathbf{t}^*_{21})$ using the camera pose. We collect pairs that at least have 512 object points and viewpoint difference within 10° and $10\,\mathrm{cm}$. We focused on the small baseline since viewpoint changes between neighboring frames are expected to be small for real-time applications. For more details of the Linemod dataset, we refer to [14].

For PRO1k, we choose 1,000 real-world objects that are diverse in size, category (electronics, hardware tools, boxes, foods, kitchenware, sculptures, decorations, toys, etc.), texture and shape. Then, we follow a similar protocol, e.g., maintain similar distance and angle to the object, as the Linemod dataset to collect data. With this large dataset, we are interested in demonstrating if DeepPRO can scale up and generalize on unseen real-world data.

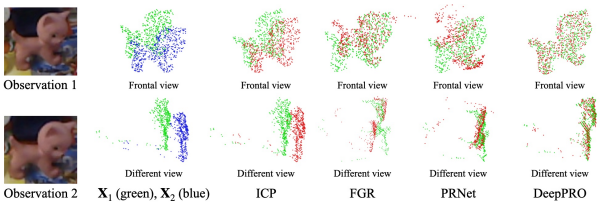


Figure 4: Visualization of point cloud registration results. Best viewed with digital zoom.

ModelNet40 is a widely used synthetic dataset in learning-based object point cloud registration field. It has 12,311 CAD models of 40 object categories. Although it is not our primary interest to evaluate on the unrealistic synthetic dataset, we can make head-to-head comparisons to existing works for large baseline point cloud registration. In our experiments, we follow the protocol in [33] to prepare partial point clouds and unseen test data.

Comparison to Existing Methods. We compare DeepPRO against thirteen strong baselines from classical methods [17, 5, 37, 42, 36, 11] to recent learning-based approaches [32, 4, 2, 33, 38, 6] as shown in Table 2[2]. We train learning-based approaches using the code released by authors while we found that some methods [32, 33] are unstable to train on real-world dataset as similarly observed in [6]. For those, we use pretrained model on ModelNet40 for evaluation.

The first thing that catches our attention from Table 2 is that classical methods interestingly show better results compared to recent learning based approaches. For example, ICP, FGR and GMMreg achieve the rotation error around 2.5° which is better than most of other learning based methods. It is because existing learning based algorithms have either an assumption which does not comply with real observations or low generalization ability on object-scale partial and noisy point clouds. Qualitative results in Figure 4 and 5 visualize results for 2D projections of 3D point clouds.

We scale up experiments using the PRO1k dataset as shown in Table 3. Results show that DeepPRO can be effectively trained on the large-scale dataset and generalize to unseen objects. Regarding runtime, the most comparable Go-ICP [37] takes about 20 seconds to run on a desktop computer. In contrast, DeepPRO runs faster than 50 fps on a mobile device without any network quantization or pruning.

[2]Recent DeepGMR [39] method is not included in our comparison since it inherently does not deal with partially overlapped point cloud pairs.

Table 2: Error of various point cloud registration methods on the Linemod dataset. ADD is defined in (2).

	Object	Ape	Vise	Cam	Can	Cat	Driller	Duck	Puncher	Iron	Lamp	Phone	Average
	Diameter (cm)	10.2	24.7	17.2	20.1	15.4	26.1	10.9	14.5	27.8	28.3	21.2	19.7
	Number of pairs	2,094	2,081	2,037	2,002	2,048	1,991	2,134	2,120	1,792	2,012	1,966	2,025
Rotation (°)	DCP v1 [32]	36.89	47.49	43.02	42.50	35.88	39.48	41.68	44.05	42.44	51.71	40.46	42.33
	DCP v2 [32]	56.99	70.76	61.24	66.74	59.66	48.01	64.36	67.64	70.02	72.50	65.31	63.93
	D3Feat [4]	25.86	53.15	42.69	37.11	44.29	51.68	24.87	39.31	34.97	46.46	62.73	41.10
	PointNetLK [2]	11.14	19.28	7.52	17.54	9.84	22.61	7.72	11.07	10.10	14.76	10.61	12.93
	PRNet [33]	7.37	7.02	8.51	6.58	6.12	7.08	7.59	7.03	5.24	5.74	5.64	6.72
	TEASER++ [36]	6.13	4.91	5.68	4.53	4.36	5.54	4.94	9.13	5.52	5.84	4.11	5.52
	JRMPC [11]	2.96	1.24	2.35	2.01	1.70	1.41	4.30	3.74	3.84	2.15	1.43	2.47
	DGR [6]	2.85	2.11	3.11	2.47	2.28	2.54	2.84	3.60	3.21	2.42	2.71	2.74
	RPM-Net [38]	2.97	1.56	2.14	2.24	2.29	1.81	2.75	2.50	1.81	1.65	1.97	2.15
	GMMreg [17]	2.42	2.07	2.21	2.28	2.10	2.50	2.43	2.61	2.51	2.43	2.38	2.36
	ICP [5]	2.14	2.35	2.02	2.11	2.14	2.25	2.06	2.15	2.03	2.45	1.90	2.15
	FGR [42]	2.49	2.85	2.32	2.57	2.37	2.68	2.71	2.41	2.62	3.02	2.36	2.58
	Go-ICP [37]	2.24	1.36	1.61	1.46	1.66	1.70	2.22	1.69	1.66	1.32	1.60	1.68
	DeepPRO	**1.28**	**1.31**	**1.25**	**1.16**	**1.19**	**1.23**	**1.25**	**1.40**	**1.01**	**1.09**	**0.97**	**1.19**
Translation (cm)	DCP v1 [32]	22.86	36.21	23.20	32.46	23.36	29.45	23.99	35.12	39.10	45.04	34.74	31.41
	DCP v2 [32]	36.79	54.95	35.17	51.40	37.91	32.64	35.01	55.03	60.58	58.43	51.56	46.32
	D3Feat [4]	20.65	43.75	38.81	36.34	39.47	47.97	21.82	30.89	31.85	41.50	58.94	37.45
	PointNetLK [2]	11.15	21.38	8.09	18.55	11.15	26.38	7.45	12.28	12.88	16.97	12.23	14.41
	PRNet [33]	8.44	7.46	9.79	7.83	7.68	7.84	8.55	8.41	6.33	5.94	6.89	7.74
	TEASER++ [36]	6.21	4.97	5.97	4.50	4.62	5.21	5.20	8.75	5.89	5.71	4.41	5.59
	JRMPC [11]	2.89	1.32	2.36	2.14	1.99	1.51	3.35	3.19	3.73	2.63	1.69	2.44
	DGR [6]	2.93	2.19	3.25	2.68	2.53	2.73	2.94	3.84	3.26	2.52	2.76	2.88
	RPM-Net [38]	3.11	1.74	2.49	2.43	2.74	2.21	3.06	2.86	2.03	1.81	2.12	2.42
	GMMreg [17]	5.53	4.32	4.78	4.28	4.45	4.94	4.86	5.22	5.44	4.70	5.20	4.88
	ICP [5]	2.53	2.89	2.67	2.60	2.71	2.94	2.61	2.52	2.41	2.99	2.38	2.66
	FGR [42]	3.21	4.40	3.64	3.92	3.48	4.57	3.37	3.32	3.73	4.22	3.35	3.75
	Go-ICP [37]	2.14	**1.54**	1.82	1.61	1.90	1.80	2.27	1.83	1.62	1.40	1.69	1.78
	DeepPRO	**1.48**	1.61	**1.61**	**1.45**	**1.47**	**1.58**	**1.48**	**1.64**	**1.22**	**1.31**	**1.20**	**1.46**
ADD (cm)	DCP v1 [32]	1.52	3.96	2.94	3.26	2.03	3.69	1.93	2.76	3.55	4.47	3.02	3.01
	DCP v2 [32]	2.22	5.39	4.06	4.75	3.08	4.39	2.83	4.04	5.14	5.83	4.43	4.20
	D3Feat [4]	3.70	5.58	4.72	4.67	4.49	7.49	2.85	3.83	5.17	5.83	5.45	4.89
	PointNetLK [2]	0.91	4.01	1.07	3.08	1.32	5.12	0.73	1.32	1.89	2.44	1.77	2.15
	PRNet [33]	0.49	0.89	0.88	0.81	0.52	1.21	0.57	0.68	0.81	0.85	0.67	0.76
	TEASER++ [36]	0.33	0.52	0.52	0.47	0.33	0.64	0.36	0.64	0.64	0.75	0.46	0.52
	JRMPC [11]	0.67	0.41	0.77	0.65	0.62	0.75	0.75	0.73	0.84	1.04	0.54	0.71
	DGR [6]	0.25	0.33	0.40	0.36	0.26	0.49	0.30	0.39	0.52	0.43	0.39	0.37
	RPM-Net [38]	0.20	0.27	0.28	0.30	0.22	0.35	0.25	0.26	0.34	0.31	0.28	0.28
	GMMreg [17]	0.32	0.52	0.36	0.37	0.31	0.47	0.30	0.39	0.42	0.68	0.36	0.41
	ICP [5]	2.32	2.35	2.86	2.36	2.55	2.53	2.34	2.68	2.52	2.83	2.55	2.54
	FGR [42]	0.97	2.03	1.46	1.74	1.24	2.44	1.02	1.12	1.52	1.78	1.47	1.53
	Go-ICP [37]	0.17	**0.20**	0.24	**0.22**	**0.17**	**0.31**	0.21	**0.21**	**0.26**	**0.25**	0.23	**0.22**
	DeepPRO	**0.14**	0.25	**0.23**	**0.22**	**0.17**	0.32	**0.17**	**0.21**	0.28	0.27	**0.21**	**0.22**

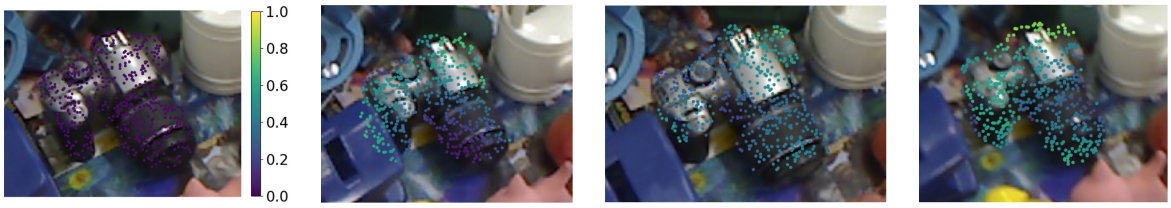
Table 3: Average error on the PRO1k dataset.

	Rotation (°)	Translation (cm)	ADD (cm)
ICP [5]	3.19	2.40	1.08
Go-ICP [37]	3.20	2.84	0.84
GMMreg [17]	2.98	3.51	0.83
FGR [42]	2.64	2.10	0.82
DeepPRO	1.07	0.93	0.41

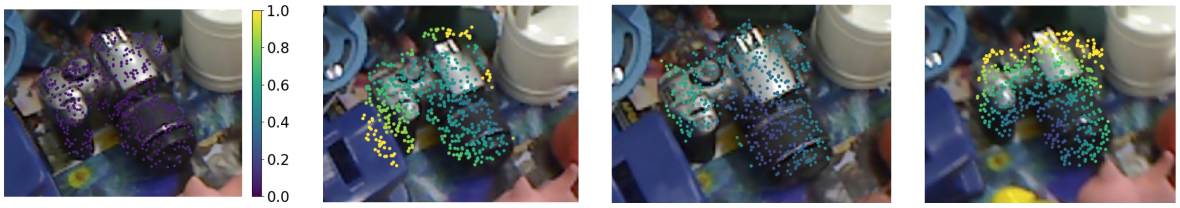
Synthetic Data and Large Baseline. Table 4 shows results of large baseline experiments on the ModelNet40 dataset. For fair comparison, we borrow results from [33] and use the same metric for evaluation. For DeepPRO, we use the same network architecture and hyper-parameters that we used for real data. Results show that DeepPRO consistently outperforms state-of-the-art methods. It also demonstrates that our approach is effective for both real and synthetic data from small to large baseline registration.

Ablation Study. To validate the effectiveness of each component in our algorithm, we carry out ablation studies as shown in Table 5. We first construct a naive deep network which bypasses the point cloud generation block. It uses the same encoder as DeepPRO, feeds concatenated global

(a) 3D point clouds $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$ are projected onto corresponding frames. Colorbar shows depth in cm.



(b) We first use ground truth pose to get $\mathbf{X}_{i1} = \mathbf{R}^*_{i1}\mathbf{X}_i + \mathbf{t}^*_{i1}$. Then, we visualize reprojected $\mathbf{X}_{i1i} = \mathbf{R}_{1i}\mathbf{X}_{i1} + \mathbf{t}_{1i}$ onto corresponding frames where $\mathbf{R}_{1i}$ and $\mathbf{t}_{1i}$ are predicted by DeepPRO. The colorbar indicates the distance between $\mathbf{X}_i$ and $\mathbf{X}_{i1i}$ in cm.



(c) Reprojected point clouds obtained with FGR pose.

Figure 5: Reprojection error for pairs with pose differences of $(10.00°, 29\,\text{cm})$, $(11.24°, 18\,\text{cm})$ and $(12.59°, 27\,\text{cm})$. While DeepPRO is only trained with pairs in $(10°, 10\,\text{cm})$ pose difference, results show the generalization to a wider range.

Table 4: Large baseline results of unseen partial point clouds with Gaussian noise on the sythetic ModelNet40 dataset.

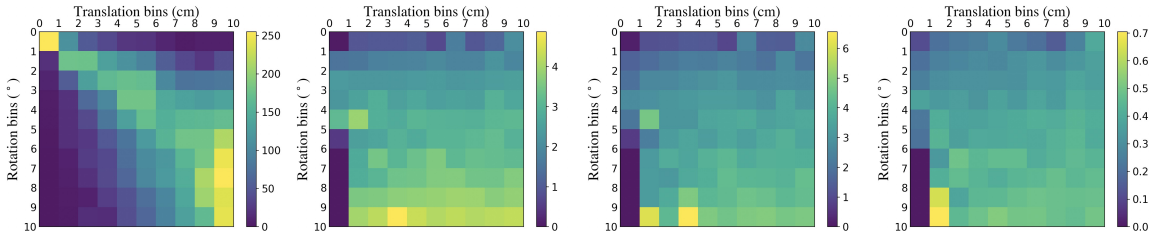
Model	MSE($\mathbf{R}$)↓	RMSE($\mathbf{R}$)↓	MAE($\mathbf{R}$)↓	R^2($\mathbf{R}$)↑	MSE($\mathbf{t}$)↓	RMSE($\mathbf{t}$)↓	MAE($\mathbf{t}$)↓	R^2($\mathbf{t}$)↑
ICP [5]	1229.670	35.067	25.564	-6.252	0.0860	0.294	0.250	-0.045
Go-ICP [37]	150.320	12.261	2.845	0.112	0.0008	0.028	0.029	0.991
FGR [42]	764.671	27.653	13.794	-3.491	0.0048	0.070	0.039	0.941
PointNetLK [2]	397.575	19.939	9.076	-1.343	0.0032	0.057	0.032	0.960
DCP v2 [32]	47.378	6.883	4.534	0.718	0.0008	0.028	0.021	0.991
PRNet [33]	18.691	4.323	2.051	0.889	0.0003	0.017	0.012	0.995
DeepPRO	**7.930**	**2.617**	**1.452**	**0.987**	**0.0002**	**0.013**	**0.007**	**0.998**

features to fully-connected layers, and is trained to minimize $\|(\mathbf{E}^*_{21})^{-1}\mathbf{E}_{21} - \mathbf{I}\|$. Results show that the naive deep network can crudely learn registering two point clouds. Second, we use conventional SVD-based pose estimation using the dense correspondence between $\mathbf{P_{12}}$ and $\mathbf{X_1}$. The proposed transform estimation network is more robust on noisy correspondences than SVD. Next, we study the efficacy of the random rotation layer, bidirectional training ((1) v.s. (3)), and ADD loss (2). It shows that each design choice in DeepPRO is effective. We also replace the DGCNN encoder with the PointNet and re-train the network with the same hyperparameters. Results show that DeepPRO is not sensitive to the choice of point cloud encoders. In addition, we show that DeepPRO can be trained well with the predicted object mask using [3]. Finally, we train DeepPRO with less number of points sampled from the point cloud. The accuracy degrades gracefully as the number of input points decreases. For example, with the half of the input points, the ADD error is increased only $0.04\,\text{cm}$.

Failure Cases. We visualize the distribution of training data and test errors for different range of rotations and translations in Figure 6. It shows that the average prediction error in each bin is mostly high when the input pair has a large rotation and a small translation differences, e.g., a user stands still and rotates the depth sensor. As shown in Figure 6(a), this type of data samples are limited in the Linemod dataset since a person circles around an object while capturing data.

Table 5: Ablation study of the proposed algorithm regarding rotation, translation, and ADD errors.

	Object	Ape	Vise	Cam	Can	Cat	Driller	Duck	Punch	Iron	Lamp	Phone	Average
Rotation (°)	Naive deep net	2.46	2.31	1.99	2.06	2.09	2.18	2.36	1.96	1.91	2.10	1.92	2.12
	SVD	2.05	1.91	2.11	1.83	1.95	1.74	2.04	2.31	2.03	1.70	1.70	1.94
	w/o random rotation	1.73	1.33	1.50	1.51	1.36	1.44	1.58	1.83	1.32	1.38	1.12	1.46
	w/o bidirectional path	1.37	1.41	1.51	1.41	1.29	1.26	1.43	1.68	1.36	1.32	1.12	1.38
	w/o ADD loss	1.42	1.38	1.34	1.41	1.30	1.19	1.49	1.62	1.23	**1.07**	1.11	1.32
	PointNet encoder	1.32	1.31	1.51	1.29	1.31	1.33	1.41	1.42	1.58	1.32	1.15	1.36
	Predicted mask [3]	1.31	1.50	1.45	1.34	1.39	1.22	1.34	1.48	1.08	1.21	1.22	1.32
	Less points N=128	1.44	1.48	1.60	1.55	1.46	1.23	1.42	1.57	1.21	1.29	1.16	1.40
	Less points N=256	1.41	**1.29**	1.36	1.51	1.22	**1.05**	1.37	**1.40**	1.08	1.11	1.06	1.26
	Full model N=512	**1.28**	1.31	**1.25**	**1.16**	**1.19**	1.23	**1.25**	**1.40**	**1.01**	1.09	**0.97**	**1.19**
Trans. (cm)	Naive deep net	2.34	2.27	1.75	2.11	2.09	2.15	2.38	1.72	1.70	1.79	1.65	2.00
	SVD	2.30	2.20	2.44	2.13	2.29	2.11	2.29	2.62	2.29	1.94	1.96	2.24
	w/o random rotation	2.04	1.74	1.89	1.86	1.77	1.89	1.88	2.11	1.59	1.68	1.44	1.81
	w/o bidirectional path	1.61	1.79	1.88	1.75	1.65	1.69	1.73	1.95	1.62	1.62	1.46	1.70
	w/o ADD loss	1.69	1.79	1.73	1.80	1.69	1.61	1.78	1.89	1.47	1.35	1.44	1.66
	PointNet encoder	1.55	1.62	1.92	1.64	1.64	1.71	1.69	1.67	1.83	1.56	1.45	1.66
	Predicted mask [3]	1.56	1.87	1.83	1.65	1.71	1.62	1.62	1.75	1.33	1.50	1.54	1.63
	Less points N=128	1.71	1.84	1.97	1.89	1.79	1.63	1.71	1.77	1.44	1.54	1.40	1.70
	Less points N=256	1.70	**1.60**	1.75	1.87	1.49	**1.37**	1.64	**1.59**	1.30	1.36	1.28	1.54
	Full model N=512	**1.48**	1.61	**1.61**	**1.45**	**1.47**	1.58	**1.48**	1.64	**1.22**	**1.31**	**1.20**	**1.46**
ADD (cm)	Naive deep net	1.40	1.17	1.17	1.14	1.12	1.32	1.28	1.15	1.12	1.16	1.15	1.20
	SVD	0.17	0.27	0.27	0.25	0.19	0.33	0.20	0.24	0.34	0.30	0.24	0.25
	w/o random rotation	0.18	0.32	0.27	0.28	0.20	0.40	0.20	0.25	0.34	0.32	0.24	0.27
	w/o bidirectional path	0.16	0.33	0.28	0.29	0.20	0.39	0.21	0.27	0.35	0.32	0.27	0.28
	w/o ADD loss	0.17	0.32	0.27	0.29	0.21	0.38	0.21	0.25	0.34	0.30	0.25	0.27
	PointNet encoder	0.15	0.26	0.26	0.24	0.18	0.34	0.18	0.23	**0.22**	0.30	0.22	0.23
	Predicted mask [3]	0.18	0.38	0.28	0.30	0.22	0.39	0.24	0.24	0.33	0.34	0.29	0.29
	Less points N=128	0.20	0.38	0.33	0.38	0.26	0.41	0.23	0.29	0.35	0.37	0.31	0.32
	Less points N=256	0.17	0.29	0.25	0.32	0.20	0.33	0.20	0.23	0.29	0.30	0.25	0.26
	Full model N=512	**0.14**	**0.25**	**0.23**	**0.22**	**0.17**	**0.32**	**0.17**	**0.21**	0.28	**0.27**	**0.21**	**0.22**



(a) Number of training data (b) Rotation error (°) (c) Translation error (cm) (d) ADD (cm)

Figure 6: Number of training data and test error distributions in the Linemod dataset for different rotation and translation bins.

In addition, we sort test pairs based on ADD error and reason out top 15 cases in the supplementary material. Results show that the error is large when the object is deformed or the point cloud mistakenly includes points from other objects or far away background.

5. Conclusion

We foray into the online and real-time registration of real-world object point clouds using deep networks while other learning-based methods focus on a synthetic database. We empirically show that, due to partial and noisy depth measurements, models trained on synthetic data and keypoint-based approaches do not generalize well on real objects. To circumvent the issue, we propose DeepPRO which generates dense correspondences based on input shape without keypoints. Experiments on two real-world (Linemod and PRO1k we collected) and one synthetic (ModelNet40) object databases show that DeepPRO is more accurate than existing methods and is fast (50 fps on a mobile device). For future work, we are interested in colored point clouds, dealing with inaccurate depth from transparent or reflective materials, and consistent multi-frame registration of the object point cloud.

References

[1] Khalil Al-Manasir and Clive S Fraser. Registration of terrestrial laser scanner data using imagery. *The Photogrammetric Record*, 21(115):255–268, 2006. 1

[2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. PointNetLK: Robust & efficient point cloud registration using PointNet. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 3, 5, 6, 7

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. 7, 8

[4] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3Feat: Joint learning of dense detection and description of 3D local features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 5, 6

[5] Paul J. Besl and Neil D. MacKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 1992. 1, 2, 5, 6, 7

[6] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 5, 6

[7] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *IEEE International Conference on Computer Vision*, 2019. 2

[8] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2

[9] Li Ding and Chen Feng. DeepMapping: Unsupervised map estimation from multiple point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[10] David W. Eggert, Adele Lorusso, and Robert B. Fisher. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine vision and applications*, 9(5-6), 1997. 3

[11] Georgios Dimitrios Evangelidis and Radu Horaud. Joint alignment of multiple point sets with batch and incremental expectation-maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1397–1410, 2017. 5, 6

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 1, 5

[13] Zan Gojcic, Caifa Zhou, Jan Dirk Wegner, and Wieser Andreas. The perfect match: 3D point cloud matching with smoothed densities. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[14] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, 2012. 2, 5

[15] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 3, 5

[16] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 3

[17] Bing Jian and Baba C. Vemuri. Robust point set registration using Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2011. 5, 6

[18] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepVCP: An end-to-end deep neural network for point cloud registration. In *IEEE International Conference on Computer Vision*, 2019. 1, 2

[19] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-Net: Towards learning based LiDAR localization for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[20] Xiankai Lu, Chao Ma, Bingbing Ni, Xiaokang Yang, Ian Reid, and Ming-Hsuan Yang. Deep regression tracking with shrinkage loss. In *European Conference on Computer Vision*, 2018. 4

[21] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, 1981. 3

[22] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. 3

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016. 3

[24] François Pomerleau, Ming Liu, Francis Colas, and Roland Siegwart. Challenging data sets for point cloud registration algorithms. *The International Journal of Robotics Research*, 31(14), 2012. 5

[25] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2

[26] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE international Conference on Robotics and Automation*, 2009. 2

[27] Vinit Sarode, Xueqian Li, Hunter Goforth, Yasuhiro Aoki, Rangaprasad Arun Srivatsan, Simon Lucey, and Howie Choset. PCRNet: Point cloud registration network using PointNet encoding. *arXiv preprint arXiv:1908.07906*, 2019. 2, 5

[28] Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey Hinton, and Kwang Moo Yi. Canonical capsules: Unsupervised capsules in canonical pose. *arXiv preprint arXiv:2012.04718*, 2021. 1

[29] Pascal Willy Theiler, Jan Dirk Wegner, and Konrad Schindler. Keypoint-based 4-points congruent sets–Automated marker-less registration of laser scans. *ISPRS journal of photogrammetry and remote sensing*, 96:149–163, 2014. 1

[30] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE International Conference on Computer Vision*, 2019. 2

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 2

[32] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE International Conference on Computer Vision*, 2019. 1, 2, 3, 5, 6, 7

[33] Yue Wang and Justin M Solomon. PRNet: Self-supervised learning for partial-to-partial registration. In *Advances in Neural Information Processing Systems*, 2019. 1, 2, 3, 5, 6, 7

[34] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 2019. 2, 3

[35] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 5

[36] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020. 5, 6

[37] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-ICP: Solving 3D registration efficiently and globally optimally. In *IEEE International Conference on Computer Vision*, 2013. 1, 2, 5, 6, 7

[38] Zi Jian Yew and Gim Hee Lee. RPM-Net: Robust point matching using learned features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 3, 5, 6

[39] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. DeepGMR: Learning latent Gaussian mixture models for registration. *European Conference on Computer Vision*, 2020. 1, 2, 5

[40] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 5

[41] Hongwei Zheng, Dietmar Saupe, Markus Roth, Andreas Böhler, and Peter Opuchlik. Efficient 3D shape acquisition and registration using hybrid scanning data. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2008. 1

[42] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, 2016. 2, 5, 6, 7