PPFNet: Global Context Aware Local Features for Robust 3D Point Matching

Haowen Deng[†,*,✠]   Tolga Birdal[†,*]   Slobodan Ilic[†,*]

[†] Technische Universitat München, Germany,   [*] Siemens AG, München, Germany
[✠] National University of Defense Technology, China,

`haowen.deng@tum.de, tolga.birdal@tum.de, slobodan.ilic@siemens.com`

Abstract

We present PPFNet - Point Pair Feature NETwork for deeply learning a globally informed 3D local feature descriptor to find correspondences in unorganized point clouds. PPFNet learns local descriptors on pure geometry and is highly aware of the global context, an important cue in deep learning. Our 3D representation is computed as a collection of point-pair-features combined with the points and normals within a local vicinity. Our permutation invariant network design is inspired by PointNet and sets PPFNet to be ordering-free. As opposed to voxelization, our method is able to consume raw point clouds to exploit the full sparsity. PPFNet uses a novel N-tuple loss and architecture injecting the global information naturally into the local descriptor. It shows that context awareness also boosts the local feature representation. Qualitative and quantitative evaluations of our network suggest increased recall, improved robustness and invariance as well as a vital step in the 3D descriptor extraction performance.
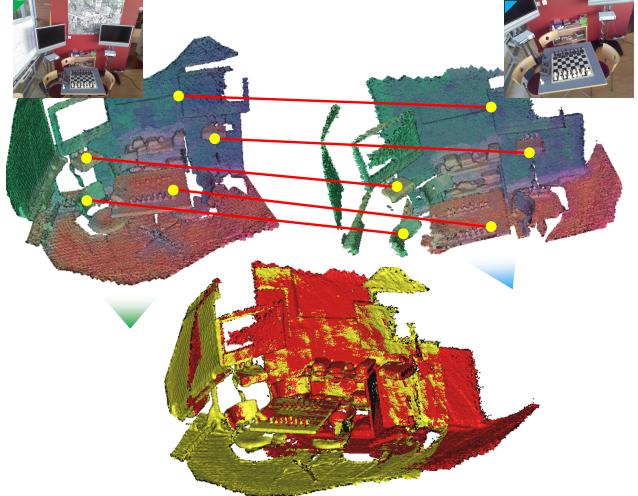
Figure 1. PPFNet generates repeatable, discriminative descriptors and can discover the correspondences simultaneously given a pair of fragments. Point sets are colored by a low dimensional embedding of the local feature for visualization. 3D data and the illustrative image are taken from 7-scenes dataset [39].

1. Introduction

Local description of 3D geometry plays a key role in 3D vision as it precedes fundamental tasks such as correspondence estimation, matching, registration, object detection or shape retrieval. Such wide application makes the local features amenable for use in robotics [7], navigation (SLAM) [37] and scene reconstruction for creation of VR contents and digitalization. Developing such a general-purpose tool motivated scholars to hand-craft their 3D feature descriptors/signatures for decades [38, 36, 35]. Unfortunately, we now notice that this quest has not been very fruitful in generating the desired repeatable and discriminative local descriptors for 3D point cloud data, especially when the input is partial or noisy [12, 25].

The recent trends carefully follow the paradigm shift to deep neural networks, but the latest works either base the representation on a hand-crafted input encoding [24] or try to naively extend the networks from 2D domain to 3D [49]. Both approaches are sub-optimal, as they do not address an end-to-end learning of the raw data, point sets.

In this paper, we present *PPFNet* network for deep learning of fast and discriminative 3D local patch descriptor with increased tolerance to rotations. To satisfy its desirable properties, we first represent the local geometry with an augmented set of simple geometric relationships: points, normals and point pair features (PPF) [35, 9]. We then design a novel loss function, which we term as *N-tuple loss*, to simultaneously embed multiple matching and non-matching pairs into a Euclidean domain. Our loss resembles the contrastive loss [15], but instead of pairs, we consider an N-combination of the input points within two scene

fragments to boost the separability. Thanks to this many-to-many loss function, we are able to inject the global context into the learning, i.e. PPFNet is aware of the other local features when establishing correspondence for a single one. Also because of such parallel processing, PPFNet is very fast in inference. Finally, we combine all these contributions in a new pipeline, which trains our network from correspondences in 3D fragment pairs. PPFNet extends PointNet [31] and thereby is natural for point clouds and neutral to permutations. Fig. 1 visualizes our features and illustrates the robust matching we can perform.

Our extensive evaluations show that PPFNet achieves the state of the art performance in accuracy, speed, robustness to point density and tolerance to changes in 3D pose.

2. Related Work

Hand-crafted 3D Feature Descriptors   Similar to its 2D counterpart, extracting meaningful and robust local descriptors from 3D data kept the 3D computer vision researchers busy for a long period of time [38, 35, 42, 22, 14]. Unfortunately, contrary to 2D, the repeatability and distinctiveness of 3D features were found to be way below expectations [11, 12, 25]. Many of those approaches try to discover a local reference frame (LRF), which is by its simplest definition, non-unique [13]. This shifts the attention to LRF-free methods such as the rotation invariant point pair features (PPF) to be used as basis for creating powerful descriptors like PPFH [36] and FPFH [35]. PPFs are also made semi-global to perform reasonably well under difficult scenarios, such as clutter and occlusions [9, 2, 17, 3]. Thus, they have been applied in many problems to estimate 3D poses or retrieve and recognize objects [35, 44]. Thanks to the simplicity and invariance properties of PPFs, along with the raw points and normals, we use it in PPFNet to describe the local geometry and learn a strong local descriptor.

Learned 3D Feature Descriptors   With the advent of deep learning, several problems like 3D retrieval, recognition, segmentation and descriptor learning have been addressed using 3D data [46, 40]. A majority of them operate on depth images [23, 45], while just a handful of works target point clouds directly. As we address the problem of 3D descriptor learning on point sets, we opt to review the data representations regardless the application and then consider learning local descriptors for point cloud matching.

There are many ways to represent sparse unstructured 3D data. Early works exploited the apparent dense voxels [30, 32], usually in form of a binary-occupancy grid. This idea is quickly extended to more informative encoding such as TDF [41], TSDF [49] (Truncated Signed Distance Field), multi-label occupancy [46] and other different ones [21, 48]. Since mainly used in the context of 3D retrieval, entire 3D objects were represented with small voxel

grids 30^3 limited by the maximal size of 3D convolutions kernels [30]. These representations have also been used for describing the local neighbours in the context of 3D descriptor learning. One such contemporary work is 3DMatch [49]. It is based on a robust volumetric TSDF encoding with a contrastive loss to learn correspondences. Albeit straightforward, 3DMatch ignores the raw nature of the input: sparsity and unstructured-ness. It uses dense local grids and 3D CNNs to learn the descriptor and thus can fall short in training/testing performance and recall.

A parallel track of works follow a view based scheme [10, 20], where the sub-spaces of 3D information in form of projections or depth map are learned with well studied 2D networks. Promising potential, these methods do not cover for sparse point sets. Another spectrum of research exploits graph networks [27, 8] also to represent point sets [34]. This new direction prospers in other domains but graph neural networks are not really suited to point clouds, as they require edges, not naturally arising from point sets. Khoury et. al. [24] try to overcome the local representation problem by a hand-crafted approach and use a deep-network only for a dimensionality reduction. Their algorithm also computes the non-unique LRF and taking such a path deviates from the efforts of end-to-end learning.

The low hanging vital step is taken by PointNet [31], a network designed for raw 3D point input. PointNet demonstrated that neural networks can be designed in a permutation invariant manner to learn segmentation, classification or keypoint extraction. It is then extended to PointNet++ to better handle the variations in point density [33]. However, this work in its original form, cannot tackle the problem of local geometry description as we successfully do.

As we will describe, PPFNet, trained on PPFs, points and normals of local patches, boosts the notion of global context in the semi-local features of PointNet by optimizing combinatorial matching loss between multitudes of patches, resulting in powerful features, outperforming the prior art.

3. Background

Motivation   Before explaining our local descriptor, let us map out our setting. Consider two 3D point sets $\mathbf{X} \in \mathbb{R}^{n \times 3}$ and $\mathbf{Y} \in \mathbb{R}^{n \times 3}$. Let $\mathbf{x}_i$ and $\mathbf{y}_i$ denote the coordinates of the ith points in the two sets, respectively. Momentarily, we assume that for each $\mathbf{x}_i$, there exists a corresponding $\mathbf{y}_i$, a bijective map. Then following [29] and assuming rigidity, the two sets are related by a correspondence and pose (motion), represented by a permutation matrix $\mathbf{P} \in \mathbb{P}^n$ and a rigid transformation $\mathbf{T} = \{\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3\}$, respectively. Then the L_2 error of point set registration reads:

$$d(\mathbf{X}, \mathbf{Y} | \mathbf{R}, \mathbf{t}, \mathbf{P}) = \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{R} \mathbf{y}_{i(\mathbf{P})} - \mathbf{t}\|^2 \quad (1)$$
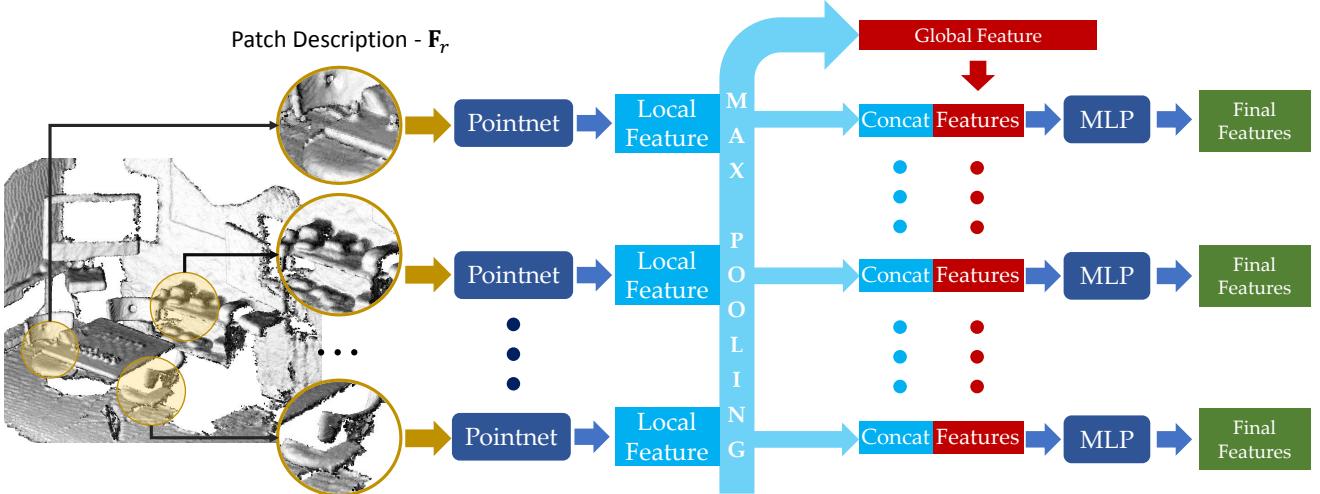
Figure 2. PPFNET, our inference network, consists of multiple PointNets, each responsible for a local patch. To capture the global context across all local patches, we use a max-pooling aggregation and fusing the output back into the local description. This way we are able to produce stronger and more discriminative local representations.

where $\mathbf{x}_i$ and $\mathbf{y}_{i(\mathbf{P})}$ are matched under $\mathbf{P}$. We assume $\mathbf{X}$ and $\mathbf{Y}$ are of equal cardinality ($|\mathbf{X}| = |\mathbf{Y}| = n$). In form of homogenized matrices, the following is equivalent:

$$d(\mathbf{X}, \mathbf{Y}|\mathbf{T}, \mathbf{P}) = \frac{1}{n}\|\mathbf{X} - \mathbf{PYT}^T\|^2 \qquad (2)$$

Two sets are ideally matching if $d(\mathbf{X}, \mathbf{Y}|\mathbf{T}, \mathbf{P}) \approx 0$. This view suggests that, to learn an effective representation, one should preserve a similar distance in the embedded space:

$$d_f(\mathbf{X}, \mathbf{Y}|\mathbf{T}, \mathbf{P}) = \frac{1}{n}\|f(\mathbf{X}) - f(\mathbf{PYT}^T)\|^2 \qquad (3)$$

and $d_f(\mathbf{X}, \mathbf{Y}|\mathbf{T}, \mathbf{P}) \approx 0$ also holds for matching points sets under any action of $(\mathbf{T}, \mathbf{P})$. Thus, for invariance, it is desirable to have: $f(\mathbf{Y}) \approx f(\mathbf{PYT}^T)$. Ideally we would like to learn f being invariant to permutations $\mathbf{P}$ and as intolerant as possible to rigid transformations $\mathbf{T}$. Therefore, in this paper we choose to use a minimally handcrafted point set to deeply learn the representation. This motivates us to exploit PointNet architecture [31] which intrinsically accounts for unordered sets and consumes sparse input. To boost the tolerance to transformations, we will benefit from point-pair-features, the true invariants under Euclidean isometry.

Point Pair Features (PPF) Point pair features are anti-symmetric 4D descriptors, describing the surface of a pair of oriented 3D points $\mathbf{x}_1$ and $\mathbf{x}_2$, constructed as:

$$\boldsymbol{\psi}_{12} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)) \qquad (4)$$

where $\mathbf{d}$ denotes the difference vector between points, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the surface normals at $\mathbf{x}_1$ and $\mathbf{x}_2$. $\|\cdot\|$ is the Euclidean distance and $\angle$ is the angle operator computed in a numerically robust manner as in [2]:

$$\angle(\mathbf{v}_1, \mathbf{v}_2) = \text{atan2}\left(\|\mathbf{v}_1 \times \mathbf{v}_2\|, \mathbf{v}_1 \cdot \mathbf{v}_2\right) \qquad (5)$$

$\angle(\mathbf{v}_1, \mathbf{v}_2)$ is guaranteed to lie in the range $[0, \pi)$. By construction, this feature is invariant under Euclidean transformations and reflections as the distances and angles are preserved between every pair of points.

PointNet PointNet [31] is an inspiring pioneer addressing the issue of consuming point clouds within a network architecture. It is composed of stacking independent MLPs anchored on points up until the last layers where a high dimensional descriptor is synthesized. This descriptor is weak and used in max-pooling in order to aggregate to a global information, which is then fed into task specific losses. Use of the max-pooling function makes the network inconsiderate of the input ordering and that way extends notions of deep learning to point sets. It showed potential on tasks like 3D model classification and segmentation. Yet, local features of PointNet are only suitable for the tasks it targets and are not generic. Moreover, the spatial transformer layer employed can bring only marginal improvement over the basic architectures. It is one aspect of PPFNet to successfully cure these drawbacks for the task of 3D matching. Note, in our work, we use the vanilla version of PointNet.

4. PPFNet

Overview We will begin by explaining our *input preparation* to compute a set describing the local geometry of a 3D point cloud. We then elaborate on *PPFNet architecture*, which is designed to process such data with merit. Finally, we explain our training method with a new loss function to solve the combinatorial correspondence problem in a global manner. In the end, the output of our network is a local descriptor per each sample point as shown in Fig. 2.
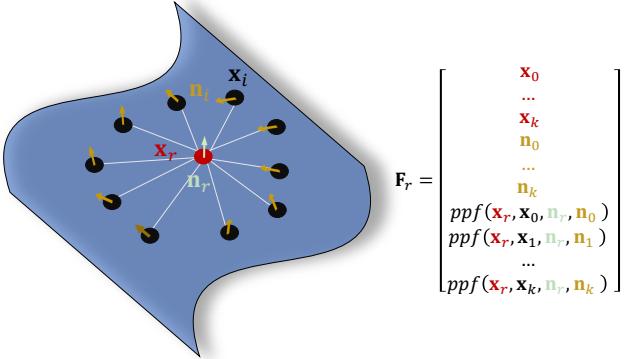
Figure 3. Simplistic encoding of a local patch.

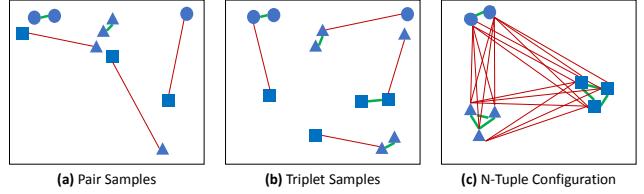(a) Pair Samples (b) Triplet Samples (c) N-Tuple Configuration

Figure 4. Illustration of N-tuple sampling in feature space. Green lines link similar pairs, which are coerced to keep close. Red lines connect non-similar pairs, pushed further apart. Without N-tuple loss, there remains to be some non-similar patches that are close in the feature space and some distant similar patches. Our novel N-tuple method pairs each patch with all the others guaranteeing that all the similar patches remain close and non-similar ones, distant.

Encoding of Local Geometry Given a reference point lying on a point cloud $\mathbf{x}_r \in \mathbf{X}$, we define a local region $\Omega \subset \mathbf{X}$ and collect a set of points $\{\boldsymbol{m}_i\} \in \Omega$ in this local vicinity. We also compute the normals of the point set [19]. The associated local reference frame [42] then aligns the patches with canonical axes. Altogether, the oriented $\{\boldsymbol{x}_r \cup \{\boldsymbol{x}_i\}\}$ represent a local geometry, which we term *a local patch*. We then pair each neighboring point i with the reference r and compute the PPFs. Note that complexity-wise, this is indifferent than using the points themselves, as we omit the quadratic pairing thanks to fixation of central reference point $\mathbf{x}_r$. As shown in Fig. 3, our final local geometry description and input to PPFNet is a combined set of points normals and PPFs:

$$\mathbf{F}_r = \{\mathbf{x}_r, \mathbf{n}_r, \mathbf{x}_i, \cdots, \mathbf{n}_i, \cdots, \boldsymbol{\psi}_{ri}, \cdots\} \quad (6)$$

Network architecture The overall architecture of PPFNet is shown in Fig. 2. Our input consists of N local patches uniformly sampled from a fragment. Due to sparsity of point-style data representation and efficient GPU utilization of PointNet, PPFNet can absorb those N patches concurrently. The first module of PPFNet is a group of mini-PointNets, extracting features from local patches. Weights and gradients are shared across all PointNets during training. A max pooling layer then aggregates all the local features into a global one, summarizing the distinct local information to the global context of the whole fragment. This global feature is then concatenated to every local feature. A group of MLPs are used to further fuse the global and local features into the final global-context aware local descriptor.

N-tuple loss Our goal is to use PPFNet to extract features for local patches, a process of mapping from a high dimensional non-linear data space into a low dimensional linear feature space. Distinctiveness of the resulting features are closely related to the separability in the embedded space. Ideally, the proximity of neighboring patches in the data space should be preserved in the feature space.

To this end, the state of the art seems to adopt two loss functions: contrastive [49] and triplet [24], which try to consider pairs and triplets respectively. Yet, a fragment consists of more than 3 patches and in that case the widely followed practice trains networks by randomly retrieving 2/3-tuples of patches from the dataset. However, networks trained in such manner only learn to differentiate maximum 3 patches, preventing them from uncovering the true matching, which is combinatorial in the patch count.

Generalizing these losses to N-patches, we propose N-tuple loss, an N-to-N contrastive loss, to correctly learn to solve this combinatorial problem by catering for the many-to-many relations as depicted in Fig. 4. Given the ground truth transformation $\mathbf{T}$, N-tuple loss operates by constructing a correspondence matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ on the points of the aligned fragments. $\mathbf{M} = (m_{ij})$ where:

$$m_{ij} = \mathbb{1}(\|\mathbf{x}_i - \mathbf{T}\mathbf{y}_j\|_2 < \tau) \quad (7)$$

$\mathbb{1}$ is an indicator function. Likewise, we compute a feature-space distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ and $\mathbf{D} = (d_{ij})$ where

$$d_{ij} = \|f(\mathbf{x}_i) - f(\mathbf{y}_j)\|_2 \quad (8)$$

The N-tuple loss then functions on the two distance matrices solving the correspondence problem. For simplisity of expression, we define an operation $\sum^*(\cdot)$ to sum up all the elements in a matrix. N-tuple loss can be written as:

$$L = \sum^* \left(\frac{\mathbf{M} \circ \mathbf{D}}{\|\mathbf{M}\|_2^2} + \alpha \frac{max(\theta - (1 - \mathbf{M}) \circ \mathbf{D}, 0)}{N^2 - \|\mathbf{M}\|_2^2} \right) \quad (9)$$

Here $\circ$ stands for Hadamard Product - element-wise multiplication. α is a hyper-parameter balancing the weight between matching and non-matching pairs and θ is the lower-bound on the expected distance between non-correspondent pairs. We train PPFNet via N-tuple loss, as shown in Fig. 5, by drawing random pairs of fragments instead of patches. This also eases the preparation of training data.
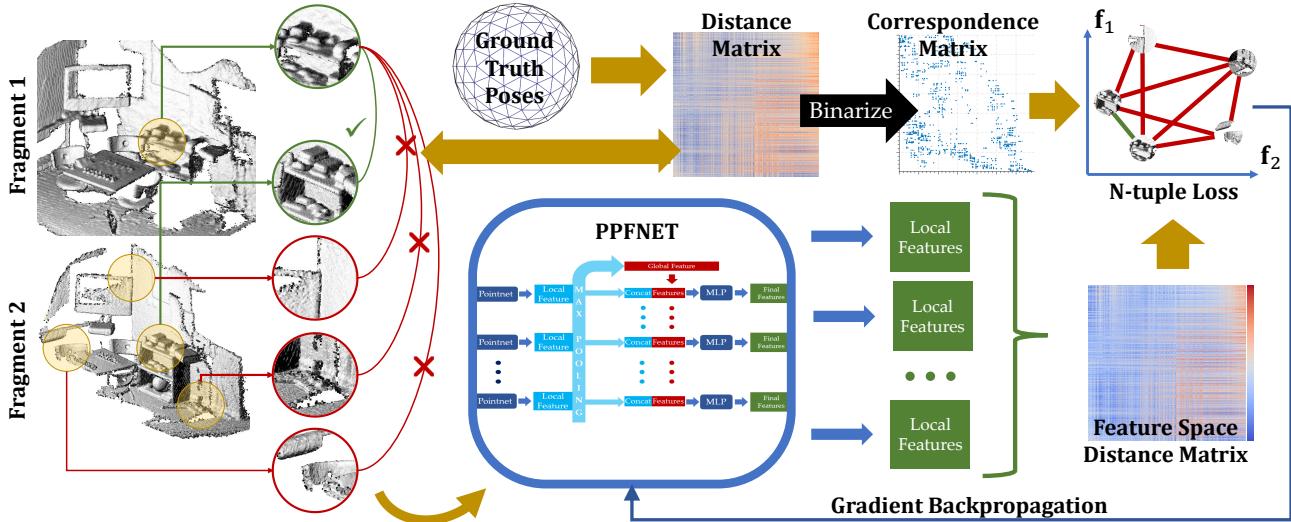
Figure 5. Overall training pipeline of PPFNet. Local patches are sampled from a pair of fragments respectively, and feed into PPFNet to get local features. Based on these features a feature distance matrix is computed for all the patch pairs. Meanwhile, a distance matrix of local patches is formed based on the ground-truth rigid pose between the fragments. By binarizing the distance matrix, we get a correspondence matrix to indicate all the matching and non-matching relationships between patches. N-tuple loss is then calculated by coupling the feature distance matrix and correspondence matrix to guide the PPFNet to find an optimal feature space.

Table 1. Our evaluations on the 3DMatch benchmark before RANSAC. *Kitchen* is from 7-scenes [39] and the rest from SUN3D [47].

	Spin Images [22]	SHOT [38]	FPFH [35]	USC [42]	PointNet [31]	CGF [24]	3DMatch [49]	3DMatch-2K [49]	PPFNet (ours)
Kitchen	0.1937	0.1779	0.3063	0.5573	0.7115	0.4605	0.5751	0.5296	**0.8972**
Home 1	0.3974	0.3718	0.5833	0.3205	0.5513	0.6154	**0.7372**	0.6923	0.5577
Home 2	0.3654	0.3365	0.4663	0.3077	0.5385	0.5625	**0.7067**	0.6202	0.5913
Hotel 1	0.1814	0.2080	0.2611	0.5354	0.4071	0.4469	0.5708	0.4779	**0.5796**
Hotel 2	0.2019	0.2212	0.3269	0.1923	0.2885	0.3846	0.4423	0.4231	**0.5769**
Hotel 3	0.3148	0.3889	0.5000	0.3148	0.3333	0.5926	**0.6296**	0.5185	0.6111
Study	0.0548	0.0719	0.1541	0.5068	0.4315	0.4075	**0.5616**	0.3904	0.5342
MIT Lab	0.1039	0.1299	0.2727	0.4675	0.5065	0.3506	0.5455	0.4156	**0.6364**
Average	0.2267	0.2382	0.3589	0.4003	0.4710	0.4776	0.5961	0.5085	**0.6231**

5. Results

Setup   Our input encoding uses a 17-point neighborhood to compute the normals for the entire scene, using the well accepted plane fitting [19]. For each fragment, we anchor 2048 sample points distributed with spatial uniformity. These sample points act as keypoints and within their $30cm$ vicinity, they form the patch, from which we compute the local PPF encoding. Similarly, we down-sample the points within each patch to 1024 to facilitate the training as well as to increase the robustness of features to various point density and missing part. For occasional patches with insufficient points in the defined neighborhood, we randomly repeat points to ensure identical patch size. PPFNet extracts compact descriptors of dimension 64.

PPFNet is implemented in the popular Tensorflow [1]. The initialization uses random weights and ADAM [26] optimizer minimizes the loss. Our network operates simultaneously on all 2048 patches. Learning rate is set at 0.001 and exponentially decayed after every 10 epochs un-

til 0.00001. Due to the hardware constraints, we use a batch size of 2 fragment pairs per iteration, containing 8192 local patches from 4 fragments already. This generates 2×2048^2 combinations for the network per batch.

Real Datasets   We concentrate on real sets rather than synthetic ones and therefore our evaluations are against the diverse 3DMatch RGBD benchmark [49], in which 62 different real-world scenes retrieved from the pool of datasets Analysis-by-Synthesis [43], 7-Scenes [39], SUN3D [47], RGB-D Scenes v.2 [28] and Halber et [16]. This collection is split into 2 subsets, 54 for training and validation, 8 for testing. The dataset typically includes indoor scenes like living rooms, offices, bedrooms, tabletops, and restrooms. See [49] for details. As our input consists of only point geometry, we solely use the fragment reconstructions captured by Kinect sensor and not the color.

Can PPFNet outperform the baselines on real data? We evaluate our method against hand-crafted baselines of
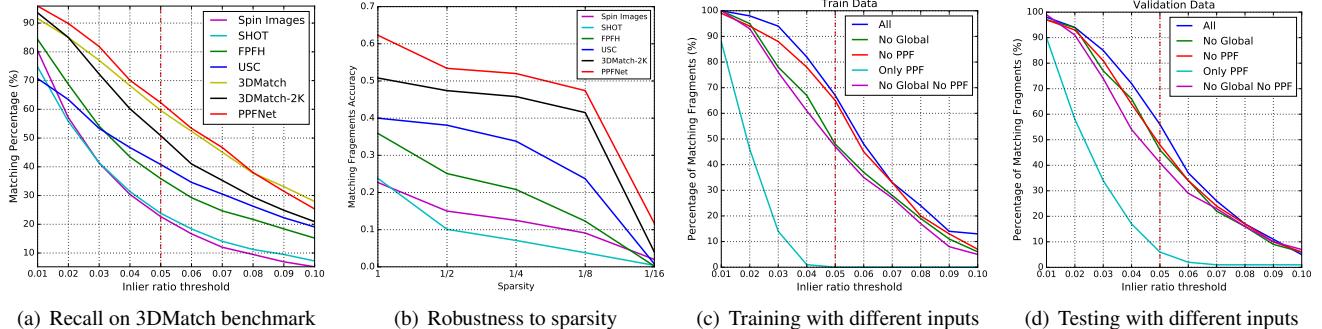
| (a) Recall on 3DMatch benchmark | (b) Robustness to sparsity | (c) Training with different inputs | (d) Testing with different inputs |

Figure 6. Evaluating PPFNet on real datasets: **(a)** Our method consistently outperforms the state-of-the-art on matching task (no RANSAC is used) in terms of recall. **(b)** Thanks to its careful design, PPFNet clearly yields the highest robustness to change in the sparsity of the input, even when only 6.25% of the input data is used. **(c, d)** Assessing different elements of the input on training and validation sets, respectively. Note that combining cues of global information and point pair features help the network to achieve the top results.

Spin Images [22], SHOT [38], FPFH [35], USC [42], as well as 3DMatch [49], the state of the art deep learning based 3D local feature descriptor, the vanilla PointNet [31] and CGF [24], a hybrid hand-crafted and deep descriptor designed for compactness. To set the experiments more fair, we also show a version of 3DMatch, where we use 2048 local patches per fragment instead of 5K, the same as in our method, denoted as 3DMatch-2K. We use the provided pre-trained weights of CGF [24]. We keep the local patch size same for all methods. Our evaluation data consists of fragments from 7-scenes [39] and SUN3D [47] datasets. We begin by showing comparisons without applying RANSAC to prune the false matches. We believe that this can show the true quality of the correspondence estimator. Inspired by [24], we accredit recall as a more effective measure for this experiment, as the precision can always be improved by better corresponding pruning [6, 5]. Our evaluation metric directly computes the recall by averaging the number of matched fragments across the datasets:

$$R = \frac{1}{M} \sum_{s=1}^{M} \mathbb{1}\left(\left[\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \mathbb{1}\big((\mathbf{x}_i - \mathbf{T}\mathbf{y}_j) < \tau_1 \big) \right] > \tau_2 \right)$$

(10)

where M is the number of ground truth matching fragment pairs, having at least 30% overlap with each other under ground-truth transformation $\mathbf{T}$ and $\tau_1 = 10cm$. (i,j) denotes an element of the found correspondence set Ω. $\mathbf{x}$ and $\mathbf{y}$ respectively come from the first and second fragment under matching. The inlier ratio is set as $\tau_2 = 0.05$. As seen from Tab. 1, PPFNet outperforms all the hand crafted counterparts in mean recall. It also shows consistent advantage over 3DMatch-2K, using an equal amount of patches. Finally and remarkably, we are able to show $\sim 2.7\%$ improvement on mean recall over the original 3DMatch, using only $\sim 40\%$ of the keypoints for matching. The performance boost from 3DMatch-2K to 3DMatch also indicates that having more keypoints is advantageous for matching.

Our method expectedly outperforms both vanilla PointNet and CGF by 15%. We show in Tab. 2 that adding more samples brings benefit, but only up to a certain level (< 5K). For PPFNet, adding more samples also increases the global context and thus, following the advent in hardware, we have the potential to further widen the performance gap over 3DMatch, by simply using more local patches. To show that we do not cherry-pick τ_2 but get consistent gains, we also plot the recall computed with the same metric for different inlier ratios in Fig. 6(a). There, for the practical choices of τ_2, PPFNet persistently remains above all others.

Table 2. Recall of 3DMatch for different sample sizes.

Samples	128	256	512	1K	2K	5K	10K	20K	40K
Recall	0.24	0.32	0.40	0.47	0.51	0.59	0.59	0.56	0.60

Application to geometric registration Similar to [49], we now use PPFNet in a broader context of transformation estimation. To do so, we plug all descriptors into the well established RANSAC based matching pipeline, in which the transformation between fragments is estimated by running a maximum of 50,000 RANSAC iterations on the initial correspondence set. We then transform the source cloud to the target by estimated 3D pose and compute the point-to-point error. This is a well established error metric [49]. Tab. 3 tabulates the results on the real datasets. Overall, PPFNet is again the top performer, while showing higher recall on a majority of the scenes and on the average. It is noteworthy that we always use 2048 patches, while allowing 3DMatch to use its original setting, 5K. Even so, we could get better recall on more than half of the scenes. When we feed 3DMatch 2048 patches, to be on par with our sampling level, PPFNet dominates performance-wise on most scenes with higher average accuracy.

Robustness to point density Changes in point density, a.k.a. sparsity, is an important concern for point clouds, as it

Table 3. Our evaluations on the 3D-match benchmark after RANSAC. *Kitchen* is from 7-scenes [39] and the rest from SUN3D [47].

	Spin Images [22]		SHOT [38]		FPFH [35]		USC [42]		PointNet [31]		CGF [24]		3DMatch [49]		3DMatch-2K		PPFNet	
	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.	recall	prec.
Red Kitchen	0.27	0.49	0.21	0.44	0.36	0.52	0.52	0.60	0.76	0.60	0.72	0.54	0.85	0.72	0.80	0.54	**0.90**	0.66
Home 1	0.56	0.14	0.37	0.13	0.56	0.16	0.35	0.16	0.53	0.16	0.69	0.18	**0.78**	0.35	0.79	0.21	0.58	0.15
Home 2	0.35	0.10	0.30	0.11	0.43	0.13	0.47	0.24	0.42	0.13	0.46	0.12	**0.61**	0.29	0.52	0.14	0.57	0.16
Hotel 1	0.37	0.29	0.28	0.29	0.29	0.36	0.53	0.46	0.45	0.38	0.55	0.38	**0.79**	0.72	0.74	0.45	0.75	0.42
Hotel 2	0.33	0.12	0.24	0.11	0.36	0.14	0.20	0.17	0.31	0.18	0.49	0.15	0.59	0.41	0.60	0.22	**0.68**	0.22
Hotel 3	0.32	0.16	0.42	0.12	0.61	0.21	0.38	0.14	0.43	0.11	0.65	0.16	0.58	0.25	0.58	0.14	**0.88**	0.20
Study Room	0.21	0.07	0.14	0.07	0.31	0.11	0.46	0.17	0.48	0.16	0.48	0.16	0.63	0.27	0.57	0.17	**0.68**	0.16
MIT Lab	0.29	0.06	0.22	0.09	0.31	0.09	0.49	0.19	0.43	0.14	0.42	0.10	0.51	0.20	0.42	0.09	**0.62**	0.13
Average	0.34	0.18	0.27	0.17	0.40	0.21	0.43	0.27	0.48	0.23	0.56	0.23	0.67	0.40	0.63	0.24	**0.71**	0.26

Table 4. Average per-patch runtime of different methods.

	input preparation	inference / patch	total
3DMatch	$0.31ms$ on GPU	$2.9ms$ on GPU	$3.21ms$
PPFNet	$2.24ms$ on CPU	$55\mu s$ on GPU	**2.25ms**

can change with sensor resolution or distance for 3D scanners. This motivates us to evaluate our algorithm against others in varying sparsity levels. We gradually decrease point density on the evaluation data and record the accuracy. Fig. 6(b) shows the significant advantage of PPFNet, especially under severe loss of density (only 6.5% of points kept). Such robustness is achieved due to the PointNet backend and the robust point pair features.

How fast is PPFNet? We found PPFNet to be lightning fast in inference and very quick in data preparation since we consume a very raw representation of data. Majority of our runtime is spent in the normal computation and this is done only once for the whole fragment. The PPF extraction is carried out within the neighborhoods of only 2048 sample points. Tab. 4 shows the average running times of different methods and ours on an NVIDIA TitanX Pascal GPU supported by an Intel Core i7 3.2GhZ 8 core CPU. Such dramatic speed-up in inference is enabled by the parallel-PointNet backend and our simultaneous correspondence estimation during inference for all patches. Currently, to prepare the input for the network, we only use CPU, leaving GPU idle for more work. This part can be easily implemented on GPU to gain even further speed boosts.

5.1. Ablation Study

N-tuple loss We train and test our network with 3 different losses: contrastive (pair) [15], triplet [18] and our N-tuple loss on the same dataset with identical network configuration. Inter-distance distribution of correspondent pairs and non-correspondent pairs are recorded for the train/validation data respectively. Empirical results in Fig. 7 show that the theoretical advantage of our loss immediately transfers to practice: Features learned by N-tuple are better separable, i.e. non-pairs are more distant in the embedding space and pairs enjoy a lower standard deviation.
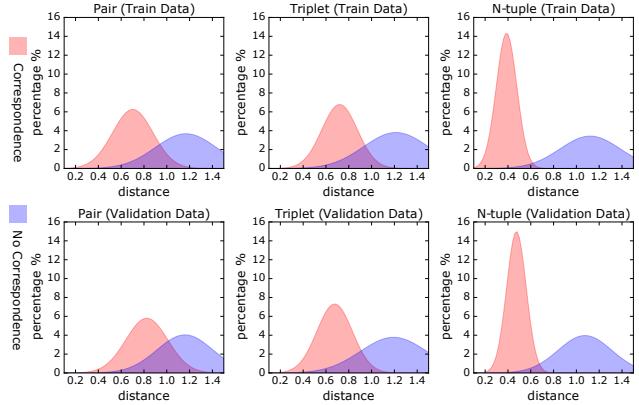

Figure 7. N-tuple Loss (c) lets the PPFNet better separate the matching vs non-matching pairs w.r.t. the traditional contrastive (a) and triplet (b) losses.

Table 5. Effect of different components in performance: Values depict the number of correct matches found to be 5% inlier ratio.

Method	Train	Validation
Without points and normals	0%	6%
Vanilla PointNet [31]	47%	41%
Without global context	48%	46%
Without PPF	65%	48%
All combined	**67%**	**56%**

N-tuples loss repels non-pairs further in comparison to contrastive and triplet losses because of its better knowledge of global correspondence relationships. Our N-tuple loss is general and thus we strongly encourage the application also to other domains such as pose estimation [45].

How useful is global context for local feature extraction? We argue that local features are dependent on the context. A corner belonging to a dining table should not share the similar local features of a picture frame hanging on the wall. A table is generally not supposed to be attached vertically on the wall. To assess the returns obtained from adding global context, we simply remove the global feature concatenation, keep the rest of the settings unaltered, and re-train and test on two subsets of pairs of fragments. Our results are shown in Tab. 5, where injecting global information into local features improves the matching by 18% in training and 7% in

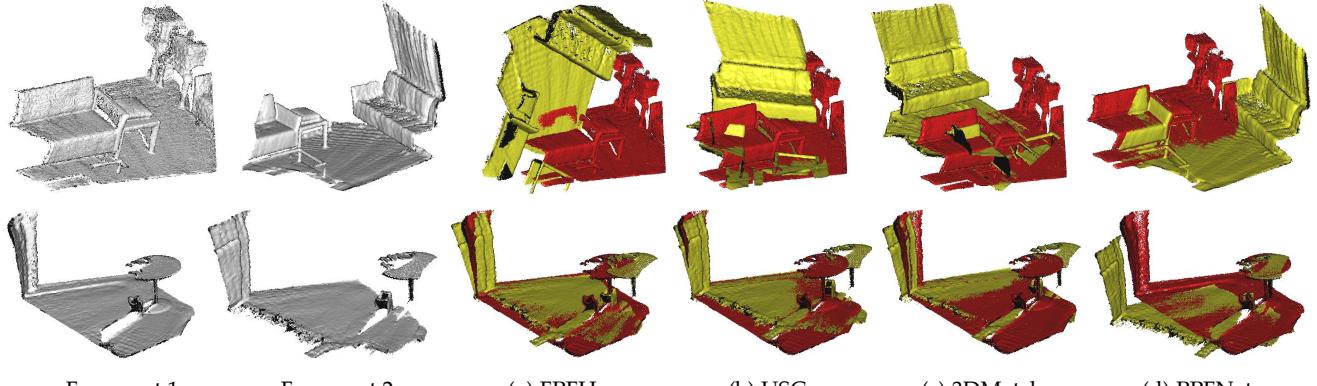| Fragment 1 | Fragment 2 | (a) FPFH | (b) USC | (c) 3DMatch | (d) PPFNet |

Figure 8. Visualization of estimated transformations. Thanks to its robustness and understanding of global information, PPFNet can operate under challenging scenarios with confusing, repetitive structures as well as mostly planar scenes with less variation in geometry.
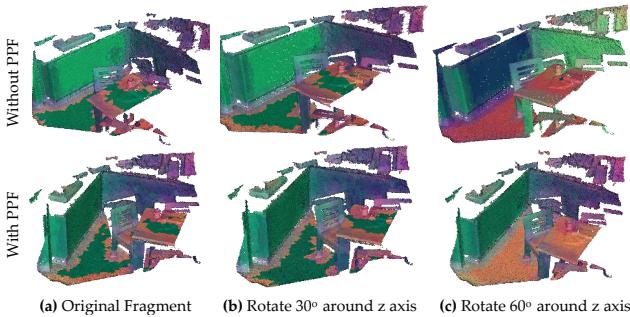

(a) Original Fragment   (b) Rotate 30° around z axis   (c) Rotate 60° around z axis

Figure 9. Inclusion of PPF makes the network more robust to rotational changes as shown, where the appearance across each row is expected to stay identical, for a fully invariant feature.

validation set as opposed to our baseline version of Vanilla PointNet •, which is free of global context and PPFs. Such significance indicates that global features aid discrimination and are valid cues also for local descriptors.

What does adding PPF bring? We now run a similar experiment and train two versions of our network, with/without incorporating PPF into the input. The contribution is tabulated in Tab. 5. There, a gain of 1% in training and 5% in validation is achieved, justifying that inclusion of PPF increases the discriminative power of the final features.

While being a significant jump, this is not the only benefit of adding PPF. Note that our input representation is composed of 33% rotation-invariant and 66% variant representations. This is already advantageous to the state of the art, where rotation handling is completely left to the network to learn from data. We hypothesize that an input guidance of PPF would aid the network to be more tolerant to rigid transformations. To test this, we gradually rotate fragments around z-axis to 180° with a step size of 30° and then match the fragment to the non-rotated one. As we can observe from Tab. 6, with PPFs, the feature is more robust to rota-

---

•Note that this doesn't 100% correspond to the original version, as we modified PointNet with task specific losses for our case.

Table 6. Effect of point pair features in robustness to rotations.

z-rotation	0°	30°	60°	90°	120°	150°	180°
with PPF	100.0%	53.3%	35.0%	20.0%	8.3%	5.0%	0.0%
w/o PPF	100.0%	38.3%	23.3%	11.7%	1.7%	0.0%	0.0%

tion and the ratio in matching performance of two networks opens as rotation increases. In accordance, we also show a visualization of the descriptors at Fig. 9 under small and large rotations. To assign each descriptor an RGB color, we use PCA projection from high dimensional feature space to 3D color space by learning a linear map [24]. It is qualitatively apparent that PPF can strengthen the robustness towards rotations. All in all, with PPFs we gain both accuracy and robustness to rigid transformation, the best of seemingly contradicting worlds. It is noteworthy that using only PPF introduces full invariance besides the invariance to permutations and renders the task very difficult to learn for our current network. We leave this as a future challenge.

A major limitation of PPFNet is quadratic memory footprint, limiting the number of used patches to 2K on our hardware. This is, for instance, why we cannot outperform 3DMatch on fragments of *Home-2*. With upcoming GPUs, we expect to reach beyond 5K, the point of saturation.

6. Conclusion

We have presented **PPFNet**, a new 3D descriptor tailored for point cloud input. By generalizing the contrastive loss to N-tuple loss to fully utilize available correspondence relatioships and retargeting the training pipeline, we have shown how to learn a globally aware 3D descriptor, which outperforms the state of the art not only in terms of recall but also speed. Features learned from our PPFNet is more capable of dealing with some challenging scenarios, as shown in Fig. 8. Furthermore, we have shown that designing our network suitable for set-input such as point pair features are advantageous in developing invariance properties.

Future work will target memory bottleneck and solving the more general rigid graph matching problem.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. 5

[2] T. Birdal and S. Ilic. Point pair features based object detection and pose estimation revisited. In *3D Vision*, pages 527–535. IEEE, 2015. 2, 3

[3] T. Birdal and S. Ilic. Cad priors for accurate and flexible instance reconstruction. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 133–142, Oct 2017. 2

[4] T. Birdal and S. Ilic. A point sampling algorithm for 3d matching of irregular geometries. In *International Conference on Intelligent Robots and Systems (IROS 2017)*. IEEE, 2017. 11

[5] D. Campbell, L. Petersson, L. Kneip, and H. Li. Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 6

[6] T.-J. Chin, P. Purkait, A. Eriksson, and D. Suter. Efficient globally optimal consensus maximisation with tree search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2421, 2015. 6

[7] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1724–1731. IEEE, 2012. 1

[8] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 2

[9] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. Ieee, 2010. 1, 2

[10] G. Elbaz, T. Avraham, and A. Fischer. 3d point cloud registration for localization using a deep neural network autoencoder. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[11] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016. 2

[12] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and J. Zhang. Performance evaluation of 3d local feature descriptors. In *Asian Conference on Computer Vision*, pages 178–194. Springer, 2014. 1, 2

[13] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013. 2

[14] Y. Guo, F. A. Sohel, M. Bennamoun, J. Wan, and M. Lu. Rops: A local feature descriptor for 3d rigid objects based on rotational projection statistics. In *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, pages 1–6. IEEE, 2013. 2

[15] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006. 1, 7

[16] M. Halber and T. Funkhouser. Fine-to-coarse global registration of rgb-d scans. 2017. 5

[17] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. In *European Conference on Computer Vision*, pages 834–848. Springer, 2016. 2

[18] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 7

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. *Surface reconstruction from unorganized points*, volume 26.2. ACM, 1992. 4, 5, 11

[20] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer. Learning local shape descriptors with view-based convolutional networks. *CoRR*, abs/1706.04496, 2017. 2

[21] J. Huang and S. You. Point cloud labeling using 3d convolutional neural network. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*, pages 2670–2675. IEEE, 2016. 2

[22] 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*, 21(5):433–449, 1999. 2, 5, 6, 7

[23] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *European Conference on Computer Vision*, pages 205–220. Springer, 2016. 2

[24] M. Khoury, Q.-Y. Zhou, and V. Koltun. Learning compact geometric features. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2, 4, 5, 6, 7, 8

[25] L. Kiforenko, B. Drost, F. Tombari, N. Krüger, and A. G. Buch. A performance evaluation of point pair features. *Computer Vision and Image Understanding*, 2017. 1, 2

[26] D. Kinga and J. B. Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[27] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[28] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014. 5

[29] H. Li and R. Hartley. The 3d-3d registration problem revisited. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. 2

[30] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 2

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2, 3, 5, 6, 7

[32] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016. 2

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2

[34] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun. 3d graph neural networks for rgbd semantic segmentation. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[35] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE, 2009. 1, 2, 5, 6, 7

[36] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Persistent Point Feature Histograms for 3D Point Clouds. In *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10), Baden-Baden, Germany*, 2008. 1, 2

[37] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013. 1

[38] S. Salti, F. Tombari, and L. Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. 1, 2, 5, 6, 7

[39] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013. 1, 5, 6, 7

[40] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 656–664, 2012. 2

[41] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 2

[42] F. Tombari, S. Salti, and L. Di Stefano. Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 57–62. ACM, 2010. 2, 4, 5, 6, 7

[43] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin. Learning to navigate the energy landscape. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 323–332. IEEE, 2016. 5

[44] E. Wahl, U. Hillenbrand, and G. Hirzinger. Surflet-pair-relation histograms: a statistical 3d-shape representation for

rapid classification. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 474–481. IEEE, 2003. 2

[45] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015. 2, 7

[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 2

[47] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013. 5, 6, 7

[48] D. Yarotsky. Geometric features for voxel-based surface recognition. *arXiv preprint arXiv:1701.04249*, 2017. 2

[49] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 1, 2, 4, 5, 6, 7
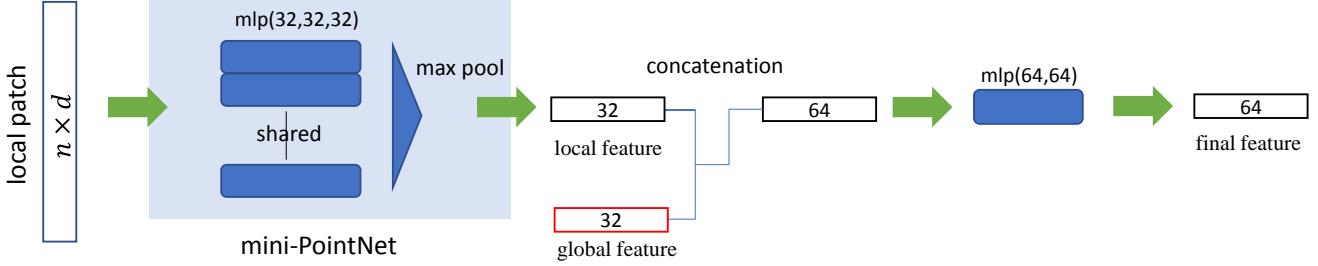
Figure 10. Pipeline for processing a single local patch.

A. Appendix

A.1. Further Architectural Details

Due to the constraint of GPU memory, we adopt a minimized version of vanilla PointNet in our implementation. Fig. A.2 demonstrates a pipeline for processing one single patch and more details of our network.

The size of a local patch is $n \times d$, where $n = 1024$ is number of points in the patch and d depends on the specific representation of local patch. For only point coordinates and normals, $d = 6$; for the one with extra PPF, $d = 10$.

A patch is first sent into a mini-PointNet with three layers, each has 32 nodes, and then a max pooling function aggregates all the information into a 32-dimensional local feature. After combining with the 32-dimensional global feature, it is further processed by a two-layer MLP, in which each layer has 64 nodes. The dimension of the final feature for the local patch is 64.

Fig. 11 demonstrates some qualitative fragment registration results in Section 5 in the paper, showing that learned features by PPFNet are able to cope with challenging point cloud matching problems under different situations.

A.2. Algorithmic Details

Sampling algorithm How we sample the point cloud down to 2048 keypoints (samples) plays an important role in learning. We try to be spatially as uniformly distributed as possible so that the samples are further apart and less dependent on one another. For this, we use the greedy algorithm given in Algorithm 1 inspired by [4]. Note that the algorithm involves a search over the so-far-sampled cloud, which we speed up using a voxel-grid.

Normal computation To achieve speed-up in point pair feature calculation, we pre-compute the normals of the input fragment. To compute a normal, the tangent plane to each local neighborhood is approximated by the least-square fitting as proposed in [19]. Computing the equation of the plane then boils down to a analysis of a covariance

---

Algorithm 1 Distance Constrained Sampling

Require: Source point cloud $\mathbf{X}$, Relative threshold τ
Ensure: Sampled point cloud $\mathbf{S}$ and its normals $\mathbf{N}$
  Compute normals for $\mathbf{X}$
  $\mathbf{S} \leftarrow []$
  $\mathbf{N} \leftarrow []$
  $R_d \leftarrow diameter(\mathbf{X})$
  for $\mathbf{x} \in \mathbf{X}$ **do**
    $d_{min} = \min_{(\mathbf{t} \in \mathbf{S})} |\mathbf{x} - \mathbf{t}|$
    if $(d_{min} > \tau R_d)$ **then**
      $\mathbf{S} \leftarrow \mathbf{S} \bigcup \mathbf{x}$
      $\mathbf{N} \leftarrow \mathbf{N} \bigcup \mathbf{n(x)}$ ▷ sample the normal as well
    end if
  end for

---

matrix created from the nearest neighbors:

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \qquad (11)$$

where $\bar{\mathbf{x}}$ denotes the mean, or the center of the local patch. The equation of the plane is then computed from the eigenvectors of $\mathbf{C}$. Due to the sign ambiguity in eigenvector analysis, the direction of the resulting normal is unknown. Thus, we use the convention where each surface normal is flipped towards the camera by ensuring the dot product between the viewpoint vector and surface normal be acute: $-\mathbf{p} \cdot \mathbf{n} < \pi/2$.
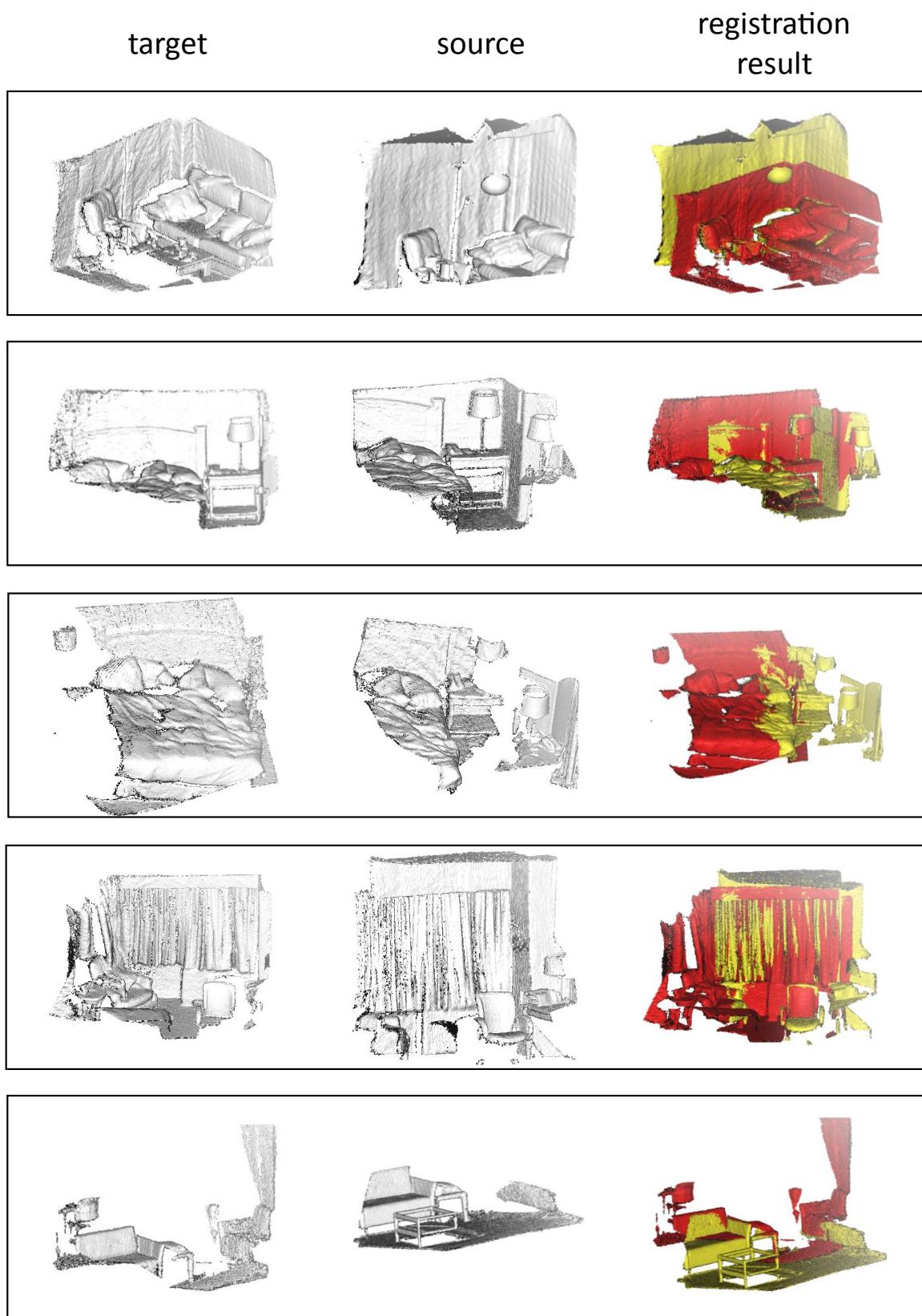
Figure 11. Qualitative registration results of 5 fragment pairs