Distinctive 3D local deep descriptors

Fabio Poiesi and Davide Boscaini
Technologies of Vision, Fondazione Bruno Kessler, Trento, Italy
{poiesi, dboscaini}@fbk.eu

***Abstract*—We present a simple but yet effective method for learning distinctive 3D local deep descriptors (DIPs) that can be used to register point clouds without requiring an initial alignment. Point cloud patches are extracted, canonicalised with respect to their estimated local reference frame and encoded into rotation-invariant compact descriptors by a PointNet-based deep neural network. DIPs can effectively generalise across different sensor modalities because they are learnt end-to-end from locally and randomly sampled points. Because DIPs encode only local geometric information, they are robust to clutter, occlusions and missing regions. We evaluate and compare DIPs against alternative hand-crafted and deep descriptors on several indoor and outdoor datasets consiting of point clouds reconstructed using different sensors. Results show that DIPs (i) achieve comparable results to the state-of-the-art on RGB-D indoor scenes (3DMatch dataset), (ii) outperform state-of-the-art by a large margin on laser-scanner outdoor scenes (ETH dataset), and (iii) generalise to indoor scenes reconstructed with the Visual-SLAM system of Android ARCore. Source code: https://github.com/fabiopoiesi/dip.**

I. Introduction

Encoding local 3D geometric information (e.g. coordinates, normals) into compact descriptors is key for shape retrieval [1], face recognition [2], object recognition [3] and rigid (six degrees-of-freedom) registration [4]. Learning such encoding from examples using deep neural networks has outperformed hand-crafted methods [4]–[12]. These approaches have been designed to encode geometric information either from meshes [13], [14] or from point clouds [10]–[12]. Our method belongs to the latter category and can be used to rigidly register point clouds without requiring an initial alignment (Fig. 1).

Existing solutions to compute compact 3D descriptors can be categorised into one-stage [4], [6], [11], [12], [16] and two-stage [3], [10], [17] methods. Although both categories share the objective of making descriptors invariant to point-cloud rigid transformations, one-stage methods encode the local geometric information of a patch (collection of locally sampled points) using the points of the patch directly. Differently, two-stage methods firstly estimate a local reference frame (LRF) from the points within the patch to rigidly transform the patch to a canonical frame, then they encode the information of the canonicalised points into a compact descriptor. Given two corresponding patches of two non-aligned point clouds, if we canonicalise them through their respective LRFs we should obtain two identical overlapping patches. Hence, the encoding method should be simpler to design than that of one-stage methods. However, noise, occlusions and point clouds reconstructed with different sensors make the LRF estimation challenging [18], [19]. Modelling descriptors to be robust to different sensors (e.g. RGB-D, laser scanner) and to different environments (e.g. indoor, outdoor) is also a challenge [10], [12]. One-stage learning-based methods can achieve rotation invariance by encoding the local geometric information with a set of geometric relationships, such as points, normals and point pair features [6], and then by learning descriptors via a PointNet-based deep network in order to achieve permutation invariance with respect to the set of the input points [15]. Alternatively, 3D convolutional neural networks (ConvNets) can be used locally, to process patches around interest points [4], or globally, to process whole point clouds [11]. In two-step methods, LRFs can be computed with hand-crafted [20] or learning-based [18] methods. After LRF canonicalisation, points can be transformed into a voxel grid, where each voxel encodes the density of the points within [10]. Then, descriptors can be encoded from this voxel representations using a 3D ConvNet learnt with a Siamese approach [21].

In this paper we present a novel two-stage method where compact descriptors are learnt end-to-end from canonicalised patches. To mitigate the problem of incorrectly estimated LRFs,

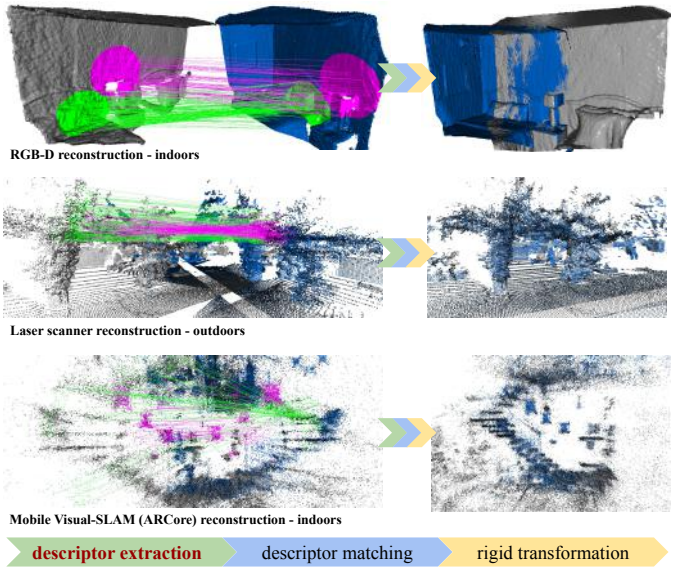

Fig. 1. Point cloud registration typically involves three steps: description extraction, descriptor matching and rigid transformation. We focus on the first step. DIPs can be used to register point clouds reconstructed with different sensor modalities in different environments. DIPs are local, rotation-invariant and compact descriptors that are learnt from examples using a PointNet-based deep neural network [15]. The correspondences between points of corresponding patches (magenta and green) are implicitly learnt by the network. The key aspect of DIPs is their generalisation ability.

we learn an affine transformation that refines the canonicalisation operation by minimising the Euclidean distance between points through the Chamfer loss [9]. Similarly to [6], we learn descriptors with a PointNet-based deep neural network through a Siamese approach, but differently from [6] (i) we use LRFs to canonicalise patches, (ii) our descriptors encode local information only, thus promoting robustness to clutter, occlusions, and missing regions, and (iii) we use a hardest contrastive loss to mine for quadruplets [11], thus improving metric learning. Differently from [6] and [10], points are consumed directly by our network without adding augmented hand-crafted features or performing prior voxelisations. We train our network using the 3DMatch dataset that consists of indoor scenes reconstructed with RGB-D sensors [4]. We achieve state-of-the-art results on the 3DMatch test set and on its augmented version, namely 3DMatchRotated [7], employed to assess descriptor rotation invariance. We significantly outperform existing approaches in terms of generalisation ability to different sensor modalities (RGB-D $\rightarrow$ laser scanner) and to different environments (indoor bedrooms $\rightarrow$ outdoor forest) using the ETH dataset [22]. Moreover, we validate DIP generalisation ability to another sensor modality (RGB-D $\rightarrow$ smartphone) by capturing three overlapping indoor point clouds with the Visual-SLAM system [23] of an ARCore-based App [24] we have developed to reconstruct the environment. Notably, DIPs can successfully and robustly be used also to align these point clouds. The source code and the reconstruction App are publicly available.

II. Our approach

Given a point cloud $\mathcal{P} \subset \mathbb{R}^3$, we define a local *patch* $\mathcal{X} = \{\mathbf{x}\} \subset \mathcal{P}$ as an unordered set of 3D points $\mathbf{x}$ with cardinality $|\mathcal{X}| = n$. We design a deep neural network $\Phi_{\mathbf{\Theta}}$ that generates DIPs such that $\mathbf{f} = \Phi_{\mathbf{\Theta}}(\mathcal{X})$, where $\mathbf{f} \subset \mathbb{R}^d$ and $\mathbf{\Theta}$ is the set of learnable parameters. Without loss of generality we use the 3D coordinates of the points as input to $\Phi_{\mathbf{\Theta}}$, i.e. $\mathbf{x} = (x, y, z)$.

A. Network architecture

Fig. 2 shows our PointNet-based architecture [15], where the three main modifications that allow us to produce DIPs are in the Transformation Network, the Bottleneck and the Local Response Normalisation layer.

Transformation Network The patch $\mathcal{X}$ is firstly passed through the Transformation Network (TNet) that predicts the affine transformation $\mathbf{A} \in \mathbb{R}^{3\times3}$ and that is applied to each $\mathbf{x} \in \mathcal{X}$. In our experiments we explored the possibility of constraining TNet to be close to an orthogonal matrix, hence a rigid transformation, via the regularisation term $\ell_{\text{reg}} = \|\mathbf{I} - \mathbf{A}\mathbf{A}^\top\|_F^2$ [15], [25]. We observed that while $\det(\mathbf{A}) \rightarrow 1$, which is a necessary condition for $\mathbf{A}$ to be a rigid transformation, $\mathbf{A} \rightarrow \mathbf{I}$ as well, thus making the contribution of $\mathbf{A}$ negligible. We empirically observed performance improvements without constraining TNet to be orthogonal. There also exists the possibility of using an Iterative Transformer Network, to perform patch canonicalisation iteratively through a series of

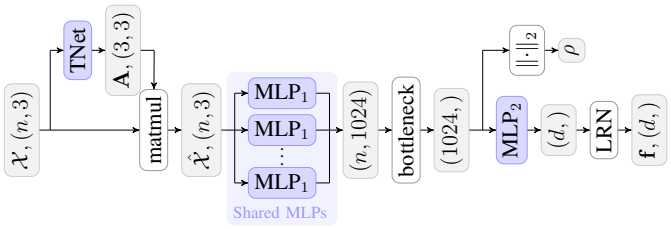

Fig. 2. PointNet-based architecture to encode an input patch $\mathcal{X}$ into a unitary-length d-dimensional descriptor $\mathbf{f}$. TNet learns the affine transformation $\mathbf{A}$. The first Multilayer Perceptron (MLP$_1$) block consists of three shared MLP layers of size (256,512,1024). The Bottleneck is a max-pooling layer that produces a 1024-dimension global signature, which is then processed by three MLPs of size (512,256,d) (MLP$_2$). ρ is the norm of the global signature. The Local Response Normalisation (LRN) layer performs a L2 normalisation of the output. Except for the last MLP, Batchnorm is used for all layers together with ReLU. Dropout is used for the last MLP. Colour key: grey = input/output tensors, blue = parametric layer, white = non-parametric layer.

3D rigid transformations [26]. This could be an interesting extension of our architecture, but it is out of the scope of our paper. TNet has a similar architecture to the rest of the network, except for the last MLP layer that outputs nine values that are opportunely reshaped to form $\mathbf{A}$. Although TNet is also used in the original version of PointNet [15], we have to train it carefully to achieve the desired behaviour for DIPs. We explain how we train TNet in Sec. II-B and III-B. Applying $\mathbf{A}$ to $\mathbf{x}$ results in $\hat{\mathbf{x}} = \mathbf{A}\mathbf{x}$. $\hat{\mathcal{X}} = \{\hat{\mathbf{x}}\}$ is then processed through the MLP layers with shared weights before reaching the bottleneck.

Bottleneck The bottleneck is modelled as a symmetric function that produces permutation-invariant outputs [15]. Although modelling this function with an average pooling operation has shown to be effective for 6DoF registration applications [27], we empirically found that max pooling provides superior performance [15]. Let m be the number of channels in output from the layer before the bottleneck, the max pooling operation is defined as $\max\colon \mathbb{R}^{n\times m} \rightarrow \mathbb{R}^m$ such that

$$\boldsymbol{\gamma} = \max_{\mathcal{X}} \Big(\Phi_{\mathbf{\Theta}_j}(\mathcal{X}) \Big), \tag{1}$$

where $\boldsymbol{\gamma} = (g_1, g_2, \ldots, g_m)$ is a *global signature* of the input patch, g_i is the i^{th} element of $\boldsymbol{\gamma}$ and $\Phi_{\mathbf{\Theta}_j}$ is an intermediate output of the network at the j^{th} layer. We observed that $\boldsymbol{\gamma}$ can be used to predict how informative DIP is. Let us define the function that returns the indices of the $\boldsymbol{\gamma}$ values as $\text{argmax}\colon \mathbb{R}^{n\times m} \rightarrow [1, n]^m$ such that

$$\boldsymbol{\alpha} = \underset{\mathcal{X}}{\text{argmax}} \Big(\Phi_{\mathbf{\Theta}_j}(\mathcal{X}) \Big). \tag{2}$$

Next, let us take two corresponding patches $\mathcal{X}$ and $\mathcal{X}'$ extracted from two overlapping point clouds $\mathcal{P}$ and $\mathcal{P}'$, and then compute $\boldsymbol{\gamma}, \boldsymbol{\alpha}$, and $\boldsymbol{\gamma}', \boldsymbol{\alpha}'$, respectively. $\boldsymbol{\alpha}$ ($\boldsymbol{\alpha}'$) will be the same regardless of the permutations of the points in $\mathcal{X}$ ($\mathcal{X}'$). We observed that the corresponding values of $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}'$ can be interpreted as the correspondences between points in $\mathcal{X}$ and $\mathcal{X}'$. Accordingly, their corresponding max values $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}'$ quantify how reliable these correspondences are. Then, we found that the norm of $\boldsymbol{\gamma}$ can be effectively used to quantify the reliability of $\boldsymbol{\gamma}$, e.g. to lower the importance of, or discard,

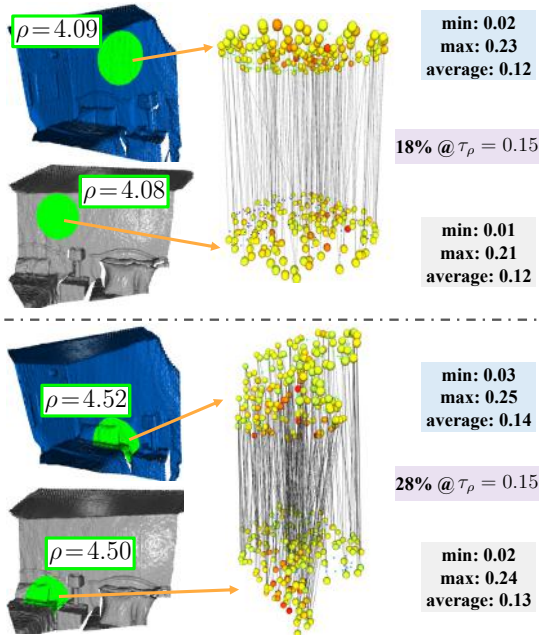

Fig. 3. Point correspondences computed from the global signatures of two pairs of corresponding patches (green) that are extracted from two overlapping point clouds (blue and grey) from the 3DMatch dataset [4]. (top) Patches extracted from flat surfaces. (bottom) Patches extracted from structured surfaces. 256 points are randomly sampled from each patch, which are given to our deep neural network as input to produce the global signature. Only the correspondences that satisfy the condition $g_i > \tau_\rho = 0.15$ are drawn. The percentage of these correspondences is reported. Points are colour-coded based on their respective g_i value. Minimum, maximum and average $g_i's$ values are reported for each case.

patches extracted from flat surfaces. It turns out that good descriptors can be selected imposing the condition

$$\rho = \|\boldsymbol{\gamma}\|_2 > \tau_\rho, \tag{3}$$

where τ_ρ is a threshold.

Fig. 3 shows an example of global signatures computed from two pairs of corresponding patches (green) that are extracted from two overlapping point clouds (blue and grey) from the 3DMatch dataset [4]. The first case shows two patches extracted from a flat surface (wall), whereas the patches in the second case are extracted from more structured surfaces (bed). From each patch we randomly sample 256 points and pass them through the network to obtain their respective global signatures (Eq. 1). This figure shows the correspondences between the points of the corresponding patches such that $g_i > \tau_\rho \; \forall i = 1, \ldots, m$, where $\tau_\rho = 0.15$. There are a few things we can observe in this example. First, values of $\boldsymbol{\gamma}$ are on average higher when the patches are extracted on structured surfaces. Second, the percentage of correspondences above the threshold is larger when the patches are extracted on structured surfaces (28% vs. 18%). Lastly, we can see that the patches extracted on flat surfaces have lower ρ. Fig. 4 shows the distribution of the ρ values for 20K patches randomly sampled from three point clouds. We can see that low values of ρ are distributed on flat surfaces (poor information) and along borders (incomplete

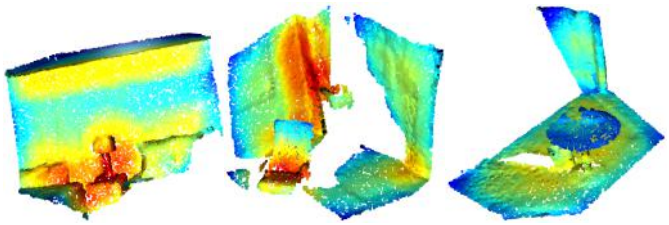

Fig. 4. Heatmaps of the ρ values for 20K patches randomly sampled from three point clouds of the 3DMatch dataset [4]. Each point is the centre of a patch with a radius of $0.3\sqrt{3}$m [10]. The more structured the surface enclosed in a patch is, the higher the value of ρ is.

information). Differently, ρ has higher value near corners and on objects.

It may be impractical to find a single τ_ρ that generalises across different input data or different network architecture. We thereby propose to infer it from the distribution of the ρ values. Let $\mathcal{R}$ be the set of ρ values computed from the patches extracted from $\mathcal{P}$, and let $f(\rho)$ be the probability density function of the ρ values. We determine τ_ρ as the $\mathsf{p}_\rho^{\text{th}}$ percentile through the cumulative density function, such that

$$\mathsf{p}_\rho = 100 \cdot \int_{-\infty}^{\tau_\rho} f(\rho) d\rho, \tag{4}$$

i.e. the area under the probability density function $f(\rho)$ to the left of τ_ρ is $\mathsf{p}_\rho/100$.

Metric layers and Local Response Normalisation After max pooling, $\boldsymbol{\gamma}$ is processed by a series of MLP layers acting as metric layers to learn distinctive embeddings for our descriptors. We use a Local Response Normalisation (LRN) layer to produce unitary-length descriptors as we found it works well in practice [10], [11], [21]. LRN consists of a L2 normalisation of the last MLP layer's d-dimensional output.

B. Loss functions

The objective of our training is to produce descriptors whose reciprocal distance in the embedding space is minimised for corresponding patches of different point clouds. To this end, we train our network following a Siamese approach that processes pairs of corresponding descriptors using two branches with shared weights [6], [10], [11]. Each branch independently calculates a descriptor for a given patch. We learn the parameters of the network by minimising the linear combination of two losses, aiming at two different goals. The first goal is to geometrically align two patches under the learnt affine transformation. The second goal is to produce compact and distinctive descriptors via metric learning.

Chamfer loss Given two patches $\mathcal{X}, \mathcal{X}'$, we want to minimise the distance between each point $\mathbf{x} \in \mathcal{X}$ and its nearest neighbour $\mathbf{x}' \in \mathcal{X}'$. Therefore we use the Chamfer loss [9], [29] on the output of TNet as

$$\ell_c(\mathcal{X}) = \frac{1}{2n} \Big(\sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{x}' \in \mathcal{X}'} \|\mathbf{A}\mathbf{x} - \mathbf{A}'\mathbf{x}'\|_2 + \sum_{\mathbf{x}' \in \mathcal{X}'} \min_{\mathbf{x} \in \mathcal{X}} \|\mathbf{A}\mathbf{x} - \mathbf{A}'\mathbf{x}'\|_2 \Big). \tag{5}$$

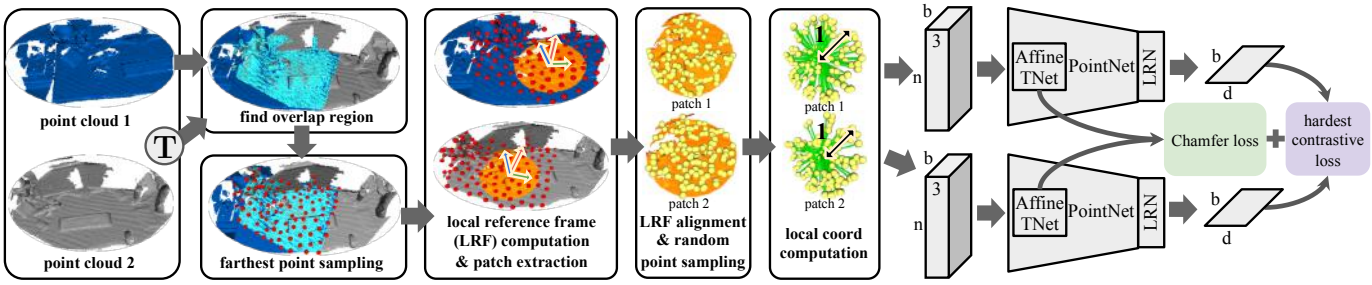

Fig. 5. DIP's training pipeline. Two overlapping point clouds are aligned using the ground-truth transformation. A set of b points (red) belonging to the overlap region (cyan) is sampled using the Farthest Point Sampling method [28]. We use a Siamese approach to train two deep neural networks with shared parameters concurrently. For each branch, we perform the following operations: (i) for each point a patch (orange) with radius τ_r is extracted and the corresponding Local Reference Frame (LRF) [10] is computed using the points of the patch; (ii) this patch is rigidly transformed using the LRF and n points are randomly sampled from the patch (yellow points); (iii) the coordinates of these n points are expressed relative to the patch centre and normalised in order to have a unitary radius; (iv) these n points are given to the deep network as input to learn the descriptor. We compute the final loss as the linear combination of the Chamfer loss [9] applied to the TNet's output and of the hardest-contrastive loss [11] applied to the network's output.

Hardest-contrastive loss Our metric learning is performed through negative mining using the hardest-contrastive loss [11]. Given a pair of anchors $(\mathbf{f}, \mathbf{f}')$, we mine the hardest-negatives $(\mathbf{f}_-, \mathbf{f}'_-)$ and define the loss as

$$\ell_h = \frac{1}{b} \sum_{(\mathbf{f},\mathbf{f}') \in \mathcal{C}_+} \left(\frac{1}{|\mathcal{C}_+|} [d(\mathbf{f},\mathbf{f}') - m_+]_+^2 + \frac{1}{2|\mathcal{C}_-|} [m_- - \underbrace{\min_{\tilde{\mathbf{f}} \in \mathcal{C}_-} d(\mathbf{f},\tilde{\mathbf{f}})}_{d(\mathbf{f},\mathbf{f}_-)}]_+^2 + \frac{1}{2|\mathcal{C}_-|} [m_- - \underbrace{\min_{\tilde{\mathbf{f}} \in \mathcal{C}_-} d(\mathbf{f}',\tilde{\mathbf{f}})}_{d(\mathbf{f}',\mathbf{f}'_-)}]_+^2 \right), \tag{6}$$

where $\mathcal{C}_+$ is the set of the anchor pairs and $\mathcal{C}_-$ is the set of descriptors (opportunely sampled) used for the hardest-negative mining extracted from a minibatch. m_+ and m_- are the margins for positive and negative pairs, respectively. $[\cdot]_+$ takes the positive part of its argument.

III. Experimental validation

We evaluate the distinctiveness of DIPs using the indoor 3DMatch dataset [4], and assess DIP generalisation on the outdoor ETH dataset [22] and on a new indoor dataset we have collected with a smartphone. We explain how patches are extracted and given as input to our deep network. The training pipeline is shown in Fig. 5. Our method is developed in Pytorch 1.3.1 [30]. We compare our method with 14 state-of-the-art methods and carry out an extensive ablation study.

A. Patch extraction

DIPs are learnt from patches that are extracted from point cloud pairs $(\mathcal{P}, \mathcal{P}')$ whose overlap region is greater than a threshold τ_o. Let $\mathcal{O} \subset \mathcal{P}$ and $\mathcal{O}' \subset \mathcal{P}'$ be the overlap regions. During training we know the ground-truth transformation $\mathbf{T} \in SE(3)$ that register $\mathcal{P}'$ to $\mathcal{P}$. Point correspondences between $\mathcal{O}$ and $\mathcal{O}'$ can be determined either by using the 3DMatch toolbox [4], or by using a nearest neighbourhood search after applying $\mathbf{T}$ to $\mathcal{P}'$ [10]. We use the latter approach by setting a threshold of $10cm$ to seek nearest points from $\mathcal{O}$ to $\mathcal{O}'$. Corresponding points in $\mathcal{O}$ and $\mathcal{O}'$ are the candidate anchors used by the hardest-contrastive loss (Eq. 6).

Farthest Point Sampling Anchor sampling is key to allow for an effective minimisation of Eq. 6, and typically this is carried out with random sampling [10], [11]. Such random sampling may lead to cases where anchors and negatives are sampled spatially close to each other. A solution can be discarding negatives within a certain radius from an anchor by computing the Euclidean distances amongst all the anchors in a minibatch to determine whether to penalise for the distance between descriptors in the embedding space (Eq. 5 in [11]). Not discarding them would force the network to learn distinctive descriptors for patches that may belong to the same region, thus making training unstable. Therefore to avoid computing the Euclidean distances amongst all the anchors in a minibatch [10], [11], we efficiently sample anchors with the largest distance amongst themselves using Farthest Point Sampling (FPS) [28]. Specifically, we sample b points within $\mathcal{O}$ using FPS and then search for the nearest neighbour counterparts in $\mathcal{O}'$. These points are the anchors that construct a minibatch on which hardest-negative mining is performed. In our experiments we use $b = 256$.

Local patch Given a point $\mathbf{c} \in \mathcal{O}$ sampled with FPS, and its nearest neighbour $\mathbf{c}' \in \mathcal{O}'$, we build the set $\mathcal{Y} = \{\mathbf{y} \in \mathcal{X} : \|\mathbf{y} - \mathbf{c}\|_2 \leq \tau_r\}$, and similarly the set $\mathcal{Y}'$ for $\mathbf{c}'$. We use the points in $\mathcal{Y}$ and $\mathcal{Y}'$ to compute their own Local Reference Frame (LRF) [10], [20]. Each LRF is constructed independently by computing the three orthogonal axes: the z-axis is computed as the normal of the local surface defined by the points of $\mathcal{Y}$ ($\mathcal{Y}'$); the x-axis is computed as a weighted sum of the vectors constructed as the projection of vectors between $\mathbf{c}$ and the points in $\mathcal{Y} \backslash \mathbf{c}$ on the plane orthogonal to the z-axis; the y-axis is computed as the cross-product between the z-axis and the x-axis. We implement the LRF following [10]. Let $\mathbf{L}, \mathbf{L}' \in \mathbb{R}^{3\times3}$ be the LRFs of $\mathcal{Y}$ and $\mathcal{Y}'$, respectively. $\mathcal{Y}$ and $\mathcal{Y}'$ may contain a large number of points, typically a few thousands, thus it is

impractical to process them all with a deep network. Similarly to [15], we randomly sample $n=256$ points. This also helps regularisation during training and generalisation. Next, we recalculate the coordinate of each of the n points relative to their patch centre and normalise the radius of the sphere that contains them. Formally, let $Q(\mathcal{Y}) = \{\hat{\mathbf{y}} \colon \hat{\mathbf{y}} = (\mathbf{y} - \mathbf{c})/\tau_r, \mathbf{y} \in \mathcal{Y}\}$ be the set of randomly-sampled and normalised points from $\mathcal{Y}$ where $|Q(\mathcal{Y})| = n$. Lastly, we apply $\mathbf{L}$ to $Q(\mathcal{Y})$ to rotate the points with respect to their LRFs, such that

$$\mathcal{X} = \mathbf{L} \otimes Q(\mathcal{Y}), \tag{7}$$

where the operation $\otimes$ defines the application of $\mathbf{L}$ to each element of $Q(\mathcal{Y})$ such that $\mathbf{x} = \mathbf{L}\hat{\mathbf{y}}$. Analogously, the same operations are performed for $Q(\mathcal{Y}')$.

B. Datasets, training and testing setup

3DMatch dataset We learn DIPs from the point clouds of the 3DMatch dataset [4]. 3DMatch is composed of 62 real-world indoor scenes collected from Analysis-by-Synthesis [31], 7-Scenes [32], SUN3D [33], RGB-D Scenes v.2 [34], and Halber and Funkhouser [35]. The official split consists of 54 scenes for training and 8 for testing. Each scene is split into partially overlapping and registered point cloud pairs. As in [10], we train our deep network with the pairs whose overlap is more than $\tau_o=30\%$. The $b = 256$ points are sampled from each of these overlap regions and we centre the patches on these points to construct each minibatch. As in [11], we set $m_+ = 0.1$ and $m_- = 1.4$ (Eq. 6). Each epoch consists of 16602 iterations. Each iteration is for a point cloud pair. We train for 40 epochs. We use Dropout with probability 0.3 at the last MLP layer. We subsample point clouds using a voxel size of 0.01m. As [10], we set $\tau_r = 0.3\sqrt{3}$m. Our training aims to minimise the linear combination of ℓ_h (Eq. 6) and ℓ_c (Eq. 5) as

$$\ell = \ell_h + \frac{1}{b} \sum_{\mathcal{X} \in \mathcal{P}} \ell_c(\mathcal{X}). \tag{8}$$

We use Stochastic Gradient Descent with an initial learning rate of 10^{-3} that decreases by a factor 0.1 every 15 epochs. The eight test scenes consists of 1117 point cloud pairs. As in [10]–[12], testing is performed by randomly sampling 5K points from each point cloud. To evaluated DIP's rotation invariance ability, we follow the evaluation of [10] and create an augmented version of 3DMatch, namely 3DMatchRotated: each point cloud is rotated by an angle sampled uniformly between $[0, 2\pi]$ around all the three axes independently. Unless otherwise stated we use $\mathsf{p}_\rho = 5$.

ETH dataset We use the ETH dataset to assess the ability of DIPs to generalise across sensor modalities (RGB-D $\rightarrow$ laser scanner) and on different scenes (indoor $\rightarrow$ outdoor) [22]. To this end we use the same model trained on the 3DMatch dataset (no fine tuning). The ETH dataset consists of four outdoor scenes, namely *Gazebo-Summer*, *Gazebo-Winter*, *Wood-Summer* and *Wood-Autumn*, containing partially overlapping, sparse and dense vegetation point clouds. Differently from the 3DMatch dataset we subsample point clouds using a voxel size of 0.06m. We set the patch kernel size $\tau_r = 0.6\sqrt{3}$m. For a fair comparison, the evaluation procedure follows verbatim [10], i.e. random sampling of 5K points.

VigoHome dataset To evaluate DIPs on another sensor modality (RGB-D $\rightarrow$ smartphone RGB), we have created a new dataset, namely VigoHome, by reconstructing the inside of a house using a Visual-SLAM smartphone App we developed with ARCore (Android) [24]. We captured three zones, namely *livingroom-downstairs* (94K points), *bedroom-upstairs* (43K points), and *bathroom-upstairs* (83K points). The stairs between the three zones is the overlapping region of the point clouds. We calculated their transformations to a common reference frame and determined the point correspondences to evaluate the registration: two points of a point cloud pair are corresponding if they are nearest neighbours within a 0.1m-radius. We subsample point clouds using voxels of 0.01m and set $\tau_r = 0.6\sqrt{3}$m.

C. Comparison and ablation study setup

We compare DIPs against 14 alternative descriptors: Spin [3], SHOT [17], FPFH [16], USC [36], CGF [37], 3DMatch [4], Folding [5], PPFNet [6], PPF-FoldNet [7], DirectReg [8], CapsuleNet [9], PerfectMatch [10], FCGF [11], and D3Feat [12]. Then, we carry out an ablation study to assess our implementation choices. To this end we train the model for five epochs on a subset of 3DMatch's scenes, i.e. *Chess* and *Fire*, and test on *Home2* and *Hotel3*. We chose these test scenes because we found them to be sufficiently challenging.

D. Evaluation metrics

Feature-matching recall We use the feature-matching recall (FMR) to quantify the descriptor quality [6]. FMR does not require RANSAC [38] as it directly averages the number of correctly matched point clouds across datasets. Only recall is measured, as the precision can be improved by pruning correspondences [6], [39]. FMR is defined as

$$\Xi = \frac{1}{|\mathcal{F}|} \sum_{s=1}^{|\mathcal{F}|} \mathbb{1}\Big(\underbrace{\Big[\frac{1}{|\Omega_s|} \sum_{(\mathcal{P},\mathcal{P}') \in \Omega_s} \mathbb{1}(\|\mathcal{P} \ominus \mathbf{T} \otimes \mathcal{P}'\|_2 < \tau_1) \Big]}_{\xi_{\Omega_s}} > \tau_2 \Big), \tag{9}$$

where $|\mathcal{F}|$ is the number of matching point cloud pairs having at least 30% overlap with each other. $(\mathcal{P}, \mathcal{P}')$ is a point cloud pair in the set of point cloud pairs Ω_s. $\ominus$ is the operator that subtracts each point of $\mathcal{P}$ to each corresponding and transformed point of $\mathcal{P}'$. For each pair $(\mathcal{P}, \mathcal{P}') \in \Omega_s$, the points in the Euclidean space are corresponding if their descriptors in the embedding space are mutually nearest neighbours [10]. $\mathbb{1}(\cdot)$ is the indicator function. $\tau_1 = 10$cm and $\tau_2 = 0.05$ are set based on the theoretical analysis that RANSAC will find at least three corresponding points that can provide the correct $\mathbf{T}$ with probability 99.9% using no more than $\approx 55K$ iterations [6], [10]. In addition to Ξ, we also report mean (μ_ξ) and standard deviation (σ_ξ) of ξ_{Ω_s} before applying τ_2.

Registration recall We measure the registration recall for the transformation estimated with RANSAC [11]. The registration

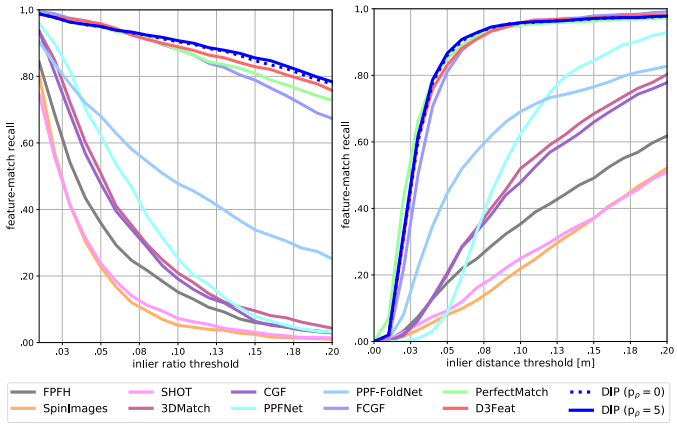

Fig. 6. Feature-matching recall as a function of (left) τ_2 and (right) τ_1.

recall quantifies the miss-rate by measuring the distance between corresponding points for each point cloud pair using the estimated transformation based on ground-truth point correspondence information. The registration recall is defined as

$$\Upsilon = \frac{1}{|\mathcal{F}|}\sum_{s=1}^{|\mathcal{F}|}\mathbb{1}\left(\sqrt{\frac{1}{|\Omega_s|}\sum_{(\mathcal{P},\mathcal{P}')\in\Omega_s}\|\mathcal{P}\ominus\tilde{\mathbf{T}}\otimes\mathcal{P}'\|_2^2} < 0.2\text{m}\right), \tag{10}$$

where $\tilde{\mathbf{T}}$ is the estimated transformation. Only point clouds pairs with at least 30% overlap are evaluated. 0.2m is set as in [4], [11], [12]. We configure RANSAC based on the FMR formulation and use its Open3D implementation [40].

E. Quantitative analysis and comparison

Tab. I reports DIP results in comparison with alternative descriptors on 3DMatch and 3DMatchRotated datasets. Results show that DIPs achieve state-of-the art results and that are rotation invariant as FMR is almost the same for both datasets. When FMR is measured at different values of τ_2, DIPs are more distinctive than the alternatives (Fig. 6). Interestingly, DIPs largely outperform PPFNet descriptors that are also computed with a PointNet-based backbone. We believe that this occurs on the one hand as a result of DIP's LRF canonicalisation, in fact PPFNet's FMR drops in 3DMatchRotated as no canonicalisation is performed, and on the other hand because DIPs encode only the local geometric information, which makes them more generic and distinctive across different scenes as opposed to PPFNet descriptors that instead encode contextual information too. Amongst the eight tested scenes, the worst performing one is *Lab* that will analyse in detail later. Our Python implementation takes 4.87ms to process a DIP using an i7-8700 CPU at 3.20GHz with a NVIDIA GTX 1070 Ti GPU and 16GB RAM. 93% of the execution time is for the LRF estimation. Once a patch is canonicalised the deep network processes the descriptor in 0.37ms. Note that the LRF estimation can be parallelised and implemented in C++ to reduce the execution time.

Tab. II reports the registration recall results, where the descriptor distinctiveness we have observed in Fig. 6 is reflected on the estimated transformations. On average, DIPs outperform all the other descriptors. We can see that the *Lab* scene mentioned before is the worst performing one. This occurs because *Lab* contains several point clouds of partially reconstructed objects and flat surfaces. Two examples are shown in Fig. 7. The first one is a failed registration due to the lack of informative geometries in the scene. The second one is a successful registration, where the kitchen appliances produced more distinctive descriptors than the first case.

We further evaluate the registration recall and assess DIP robustness following the comparative ablation study proposed in [12], where the registration recall is measured as a function of a decreasing number of sampled points used by RANSAC to estimate the transformation. Tab. III shows that DIPs on average have a superior robustness than the alternatives.

F. Ablation study

Tab. IV reports the results of our ablation study on the implementation choices. Here we can see how the three modules, i.e. TNet, LRF and LRN, affect FMR. TNet learns to compensate for incorrectly estimated LRFs. But we can see that without LRF, TNet cannot learn the complete transformation to canonicalise the patches (FMR drops on 3DMatchRotated). LRF is key to make DIPs rotation invariant. Following [10] and [11], we can notice how learning unitary-length descriptors improve FMR. Lastly, as expected, the more the capacity to encode the information in descriptors of larger dimension, the better the performance. However, we used 32-dimensional

TABLE I
FEATURE-MATCHING RECALL ON THE 3DMATCH DATASET [4].

Method	3DMatch Ξ	3DMatch std	3DMatchRotated Ξ	3DMatchRotated std	Feat. dim.	Time [ms]
Spin [3]	.227	.114	.227	.121	153	.133
SHOT [17]	.238	.109	.234	.095	352	.279
FPFH [16]	.359	.134	.364	.136	33	.032
USC [36]	.400	.125	-	-	1980	3.712
CGF [37]	.582	.142	.585	.140	32	1.463
3DMatch [4]	.596	.088	.011	.012	512	3.210
Folding [5]	.613	.087	.023	.010	512	.352
PPFNet [6]	.623	.108	.003	.005	64	2.257
PPF-FoldNet [7]	.718	.105	.731	.104	512	.794
DirectReg [8]	.746	.094	-	-	512	.794
CapsuleNet [9]	.807	.062	.807	.062	512	1.208
PerfectMatch [10]	.947	.027	.949	.024	32	5.515
FCGF [11]	.952	.029	.953	.033	32	.009
D3Feat [12]	**.958**	.029	**.955**	.035	32	-
DIP	.948	.046	.946	.046	32	4.870

TABLE II
REGISTRATION RECALL ON THE 3DMATCH DATASET [4].

Method	Kitchen	Home1	Home2	Hotel1	Hotel2	Hotel3	Study	Lab	Average
FPFH [16]	.36	.56	.43	.29	.36	.61	.31	.31	.40
USC [36]	.52	.35	.47	.53	.20	.38	.46	.49	.43
CGF [37]	.72	.69	.46	.55	.49	.65	.48	.42	.56
3DMatch [4]	.85	.78	.61	.79	.59	.58	.63	.51	.67
PPFNet [6]	.90	.58	.57	.75	.68	.88	.68	.62	.71
FCGF [11]	.93	.91	.71	.91	.87	.69	.75	**.80**	.82
DIP	**.98**	**.94**	**.85**	**.98**	**.92**	**.89**	**.80**	.75	**.89**

TABLE III
ABLATION STUDY USING THE REGISTRATION RECALL AS A FUNCTION OF THE NUMBER OF SAMPLED POINTS ON THE 3DMATCH DATASET [4].

Method	# sampled points					Average
	5000	2500	1000	500	250	
PerfectMatch [10]	.803	.775	.734	.648	.509	.694
FCGF [11]	.873	.858	.858	.810	.730	.826
D3Feat [12]	.822	.844	.849	.825	**.793**	.827
DIP	**.889**	**.890**	**.878**	**.866**	.774	**.859**

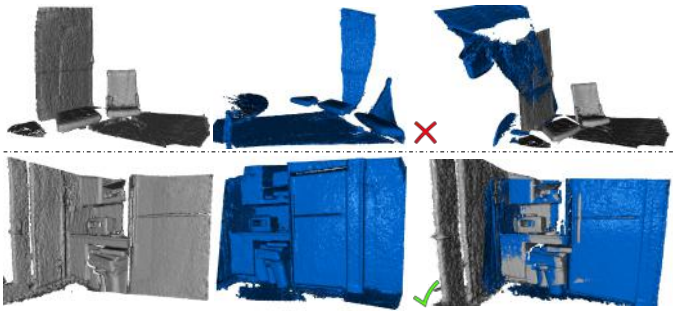

Fig. 7. Estimated rigid transformations of two point cloud pairs from the 3DMatch dataset [4]. (top) Incorrectly estimated transformation due to the lack of structured surfaces. (bottom) Correctly estimated transformation thanks to the structured elements given by the kitchen appliances.

TABLE IV
ABLATION STUDY ON DIP'S IMPLEMENTATION CHOICES.

d	TNet	LRF	LRN	3DMatch			3DMatchRotated		
				Ξ	μ_ξ	σ_ξ	Ξ	μ_ξ	σ_ξ
32		✓	✓	.886	.272	.207	.877	.271	.207
32	✓	✓		.831	.215	.175	.834	.214	.174
32	✓		✓	.824	.237	.195	.215	.039	.054
16	✓	✓	✓	.903	.264	.193	.906	.264	.193
32	✓	✓	✓	.919	.318	.218	.926	.317	.217
64	✓	✓	✓	.916	.327	.221	.928	.325	.220
128	✓	✓	✓	.933	.335	.222	.924	.334	.222

TABLE V
FEATURE-MATCHING RECALL ON THE ETH DATASET [22].

Method	Gazebo		Wood		Average
	Summer	Winter	Autumn	Summer	
FPFH [16]	.386	.142	.148	.208	.221
SHOT [17]	.739	.457	.609	.640	.611
3DMatch [4]	.228	.083	.139	.224	.169
CGF [37]	.375	.138	.104	.192	.202
PerfectMatch [10]	**.913**	.841	.678	.728	.790
FCGF [11]	.228	.100	.148	.168	.161
D3Feat [12]	.859	.630	.496	.480	.563
DIP ($\mathsf{p}_\rho = 0$)	.897	.869	.957	.944	.916
DIP ($\mathsf{p}_\rho = 5$)	.908	**.886**	**.965**	**.952**	**.928**

descriptors throughout all the experiments in order to compare results with existing descriptors.

G. Generalisation ability: comparison and analysis

Tab. V reports the results obtained on the ETH dataset [22]. We can see that DIPs on average largely outperform the alternative descriptors. Second to DIPs are PerfectMatch's descriptors [10] that, as DIPs, use LRF canonicalisation. However, differently from DIPs, PerfectMatch's descriptors are learnt from hand-crafted representations, namely *voxelised smoothed density value* [10]. We argue that letting the network learn the encoding from the points directly (end-to-end), leads to a much greater robustness and generalisation ability. We can also observe that the application of p_ρ improves the performance. Fig. 8 shows an example of result from Gazebo-Summer. Although the sensor modality and the structure of the environment is very different from that of 3DMatch, DIPs maintain their distinctiveness and can be successfully used to register two point clouds reconstructed with a laser scanner.

Fig. 9 shows results on our dataset, i.e. VigoHome. In each point cloud we included the corresponding reference frame, which is where each mapping session started. The result of a successful registration estimated using DIPs is shown in the bottom-right corner. We can notice that the structure of the environment largely differs from that of 3DMatch and ETH datasets, and that the distribution of the points on the surfaces is much noisier that that in the 3DMatch dataset. To quantify the registration results, we used a similar evaluation of that used in Sec. III-F. For each number of sampled points we run RANSAC 100 times and compute the registration recall. Tab. VI shows that with only 5K points sampled from each point cloud, 85% of the times the three point clouds are correctly registered. A correct registration takes about $54s$ to be processed. We deem this a great result because it is achieved with DIPs learnt on the 3DMatch dataset. As additional comparison, we have also quantified the registration recall using FPFH descriptors [16]. However, we found that the registration fails regardless the parameters used. So we have intentionally not included the results obtained with FPFH in the table.

IV. CONCLUSIONS

We presented a novel approach to learn local, compact and rotation invariant descriptors end-to-end through a PointNet-based deep neural network using canonicalised patches. The affine transformation embedded in our network is learnt with the specific goal of improving patch canonicalisation. We showed the importance of this step through our ablation study. Results showed that DIPs achieve comparable performance to the state-of-the-art on the 3DMatch dataset, but that outperform the state-of-the-art by a large margin in terms of generalisation to different sensors and scenes. We further confirmed this by capturing a new indoor dataset using the Visual-SLAM system of ARCore (Android) running on an off-the-shelf smartphone. We observed that we can achieve good generalisation because DIPs are learnt end-to-end from the points without any hand-crafted preprocessing after canonicalisation. Our future research direction is to improve the canonicalisation operation [18].

ACKNOWLEDGMENT

This research has received funding from the *Fondazione CARITRO - Ricerca e Sviluppo* programme 2018-2020.

Fig. 8. Estimated rigid transformation of a point cloud pair from the ETH dataset [22]. This example shows that DIPs learnt on the 3DMatch dataset (RGB-D reconstructions) can be successfully used to register point clouds reconstructed with a laser scanner in outdoor scenes.

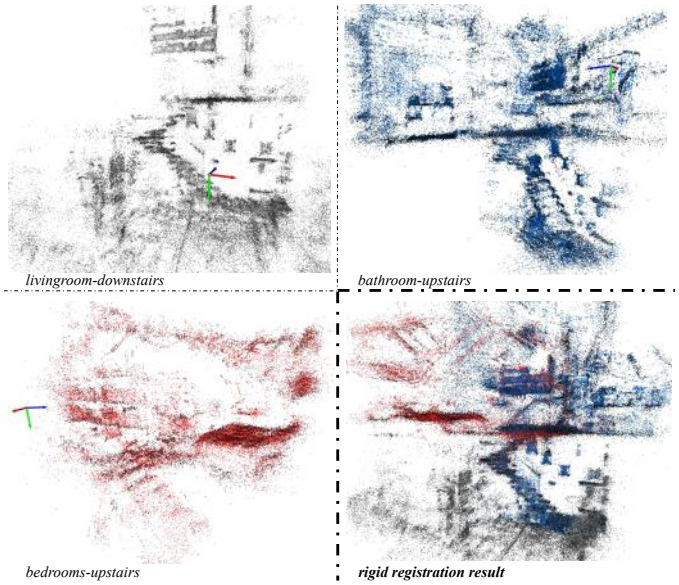

Fig. 9. Estimated rigid transformations of three point clouds from our VigoHome dataset. DIPs can also generalise to point clouds reconstructed with the Visual-SLAM system of ARCore (Android) running on an off-the-shelf smartphone (Xiaomi Mi8). Three different overlapping zones of the inside of a house have been reconstructed. The reference frame of each point cloud, that is where the reconstruction session has started, is shown for each zone.

TABLE VI
REGISTRATION RECALL ON THE VIGOHOME DATASET AS A FUNCTION OF THE NUMBER OF SAMPLED POINTS.

Method	# sampled points					
	15000	10000	5000	2500	1000	500
DIP	.97	.90	.85	.68	.44	.16

REFERENCES

[1] M. Ovsjanikov, Q. Merigot, F. Memoli, and L. Guibas, "One point isometric matching with the heat kernel," in *SGP*, 2010.
[2] Y. Lei, Y. Guo, M. Hayat, M. Bennamoun, and X. Zhou, "A two-phase weighted collaborative representation for 3D partial face recognition with single sample," *Pattern Recognition*, vol. 52, 2016.
[3] A. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *PAMI*, vol. 21, 1999.
[4] A. Zeng, S. Song, M. Niener, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning the matching of local 3D geometry in range scans," in *CVPR*, 2017.
[5] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Interpretable unsupervised learning on 3D point," in *CVPR*, 2017.
[6] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *CVPR*, 2018.
[7] ——, "PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors," in *ECCV*, 2018.
[8] ——, "3D local features for direct pairwise registration," in *CVPR*, 2019.
[9] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *CVPR*, 2019.
[10] Z. Gojcic, C. Zhou, J. Wegner, and W. Andreas, "The perfect match: 3D point cloud matching with smoothed densities," in *CVPR*, 2019.
[11] C. Choy, J. Park, and V. Koltun, "Fully Convolutional Geometric Features," in *ICCV*, 2019.
[12] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, "D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features," in *CVPR*, 2020.
[13] D. Boscaini, J. Masci, E. Rodola, M. Bronstein, and D. Cremers, "Anisotropic Diffusion Descriptors," *Computer Graphics Forum*, vol. 35, 2016.
[14] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *CVPR*, 2017.
[15] C. Qi, H. Su, K. Mo, and L. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *CVPR*, 2017.
[16] R. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *ICRA*, 2009.
[17] S. Salti, F. Tombari, and L. di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *CVIU*, vol. 125, 2014.
[18] S. Melzi, R. Spezialetti, F. Tombari, M. Bronstein, L. D. Stefano, and E. Rodolá, "GFrames: Gradient-Based Local Reference Frame for 3D Shape Matching," in *CVPR*, 2019.
[19] A. Zhu, J. Yang, C. Zhao, K. Xian, Z. Cao, and X. Li, "LRF-Net: Learning Local Reference Frames for 3D Local Shape Description and Matching," *arXiv:2001.07832*, 2020.
[20] J. Yang, Q. Zhang, Y. Xiao, and Z.-G. Cao, "TOLDI: An effective and robust approach for 3D local shape description," *Pattern Recognition*, vol. 65, 2016.
[21] Y. Tian, B. Fan, and F. Wu, "L2-Net: deep learning of discriminative patch descriptor in Euclidean space," in *CVPR*, 2017.
[22] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *IJRR*, vol. 31, 2012.
[23] R. Mur-Artal, J. Montiel, and J. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *TRO*, vol. 31, 2015.
[24] "ARCore," https://developers.google.com/ar, Accessed: Apr 2020.
[25] D. Huynh, "Metrics for 3D rotations: comparison and analysis," *J. Math Imaging Vis.*, vol. 35, 2009.
[26] W. Yuan, D. Held, C. Mertz, and M. Hebert, "Iterative Transformer Network for 3D Point Cloud," in *CVPR Workshop*, 2019.
[27] C. Wang, D. Xu, Y. Zhu, R. Martn-Martn, C. Lu, L. Fei-Fei, and S. Savarese, "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion," in *CVPR*, 2019.
[28] C. Qi, L. Yi, H. Su, and L. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *NIPS*, 2017.
[29] C. Fernandez-Labrador, A. Chhatkuli, D. Paudel, J. Guerrero, C. Demonceaux, and L. V. Gool, "Unsupervised Learning of Category-Specific Symmetric 3D Keypoints from Point Sets," *arXiv:2003.07619*, 2020.
[30] Pytorch, http://pytorch.org, last accessed: Apr 2020.
[31] J. Valentin, A. Dai, M. Niener, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *3DV*, 2016.
[32] 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 *CVPR*, 2013.
[33] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A Database of Big Spaces Reconstructed using SfM and Object Labels," in *ICCV*, 2013.
[34] V. G. Kim, S. Chaudhuri, L. Guibas, and T. Funkhouser, "Shape2Pose: human-centric shape analysis," *TOG*, vol. 33, no. 4, 2014.
[35] M. Halber and T. Funkhouser, "Structured global registration of RGB-D scans in indoor environments," *arXiv:1607.08539*, 2016.
[36] F. Tombari, S. Salti, and L. D. Stefano, "Unique shape context for 3D data description," in *ACM Workshop on 3D Object Retrieval*, 2010.
[37] M. Khoury, Q.-Y. Zhou, and V. Koltun, "Learning compact geometric features," in *ICCV*, 2017.
[38] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 6, 1981.
[39] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *CVPR*, 2015.
[40] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.