Learning an Effective Equivariant 3D Descriptor Without Supervision

Riccardo Spezialetti, Samuele Salti, Luigi di Stefano
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{riccardo.spezialetti, samuele.salti, luigi.distefano }@unibo.it

Abstract

Establishing correspondences between 3D shapes is a fundamental task in 3D Computer Vision, typically addressed by matching local descriptors. Recently, a few attempts at applying the deep learning paradigm to the task have shown promising results. Yet, the only explored way to learn rotation invariant descriptors has been to feed neural networks with highly engineered and invariant representations provided by existing hand-crafted descriptors, a path that goes in the opposite direction of end-to-end learning from raw data so successfully deployed for 2D images.

In this paper, we explore the benefits of taking a step back in the direction of end-to-end learning of 3D descriptors by disentangling the creation of a robust and distinctive rotation equivariant representation, which can be learned from unoriented input data, and the definition of a good canonical orientation, required only at test time to obtain an invariant descriptor. To this end, we leverage two recent innovations: spherical convolutional neural networks to learn an equivariant descriptor and plane folding decoders to learn without supervision. The effectiveness of the proposed approach is experimentally validated by outperforming hand-crafted and learned descriptors on a standard benchmark.

1. Introduction

Surface matching is a challenging problem in 3D Computer Vision. It has a large number of applications such as 3D Object Recognition, 3D Object Retrieval, 3D Registration and Reconstruction. The definition of compact and effective representations of the local geometry of a surface, usually referred to as *descriptors*, plays a key role in surface matching. Indeed, performance of the algorithms proposed to tackle the above mentioned applications is often largely determined by the effectiveness of the chosen descriptor. This has fostered intensive research in the area of local 3D descriptors in the last decades [29, 25, 13, 10, 24].

The success of deep neural networks in image recognition has motivated a recent paradigm shift from handcrafted algorithms to data-driven approaches also in the design of local 3D descriptors [36, 14, 3, 4]. However, state-of-the art proposals do not actually learn new local 3D descriptors from the input data but from existing handcrafted 3D descriptors, which are already rotation-invariant by design: e.g., CGF [14] starts from a high-dimensional input parameterization which closely resembles the Unique Shape Context (USC) descriptor [29], while PPF-FoldNet [3] relies on the well-known Point Pair Features (PPF) [6]. In other words, due to the difficulty of feeding neural networks with unorganized input data [14], these approaches create new descriptors by actually learning how to robustly *compress* a specific invariant handcrafted descriptor.

We argue that the drawbacks of relying on invariant handcrafted descriptors as input data to feed neural networks are twofold. On one hand, there not exist an optimal handcrafted descriptor across applications and datasets, as vouched by recent evaluations [9]. Therefore, for instance, performance of PPF-FoldNet are limited on some scenarios and datasets by the handcrafted design decision of using PPF as input representation. On the other hand, to achieve rotation invariance, existing handcrafted descriptors used in deep learning pipelines rely either on the normal at the point as a reference axis [3] or on a local reference frame (LRF) [14] to express point coordinates and angles with respect to a canonically oriented reference frame. Repeatability of the axis or the LRF directly affects the invariance and robustness of the input descriptor [20, 19] and, in turn, of the descriptor learned from such representations. However, parameters used to obtain such canonical orientation (e.g. the number of neighbours to estimate the normals, how to establish a reference direction on the tangent plane in a LRF, *etc*.) are again handcrafted design decisions and are not optimized during training.

Reliance on rotation-invariant handcrafted descriptors as input representations deviates significantly from the end-to-end learning paradigm so successfully applied to images. Therefore, in this paper we investigate on whether leaving the model free to learn an optimal descriptor from a non-canonically oriented input representation may unleash the untapped potential of deep learning also in this scenario. To this end, we exploit the paradigm recently proposed in FoldingNet [34] and AtlasNet [8] to realize unsupervised learning of an embedding space from 3D data, which learns to deform, according to the latent representation, points sampled from a plane so as to reconstruct the input surface. This concept has already been deployed to obtain an invariant 3D descriptor by reconstructing the Point Pair Features of the input data [3]. In our proposal, however, the learned latent space has to encode pose information in order to be able to reconstruct the input under arbitrary poses, as it will be shown later (Sec. 3.3). We argue that the ability to learn an embedding *equivariant* with respect to rotations of the input is the most sound approach to include pose information in the latent space. To this end, we leverage recent work on Spherical CNNs [2, 7], which have enhanced the deep learning machinery by enabling it to learn also rotation-equivariant representations from 3D spherical signals by means of correlations defined for the $\mathrm{SO}(3)$ group of rotations. Hence, in our architecture, a Spherical CNN encoder learns to summarize the geometry around a feature point into a rotation-equivariant embedding and a decoder warps a 2D grid in order to reconstruct the raw input data. This enables learning of an equivariant embedding without using noisy and arbitrary canonical orientations at training time.

To perform pose invariant descriptor matching at test time, we have investigated two alternative ways to orient our equivariant descriptor: we can again exploit the peculiar nature of the Spherical CNN output, which is a signal living in $\mathrm{SO}(3)$, to define a canonical orientation directly from the computed embedding; or we can orient the descriptor according to a canonical orientation provided by an external local reference frame computed on the input data. While the first approach enables end-to-end learning of the descriptor and the LRF, we have so far obtained better results with the second one. In particular, we have validated our claim on the superiority of learning a local descriptor from raw unoriented input data by comparing the two variants against handcrafted and learned methods on the popular 3DMatch benchmark data set [36]. Our proposal improves the state-of-the art by a remarkable margin, outperforming the method based on the same unsupervised learning framework, but applied to an invariant descriptor, by more than 0.23 points of fragments registation recall (31% increase).

2. Related Work

This section provides a review of the main proposals in the field of local descriptors, starting from early hand-crafted methods up to novel approaches based on deep learning.

Hand-crafted 3D Local Descriptors A local 3D descriptor creates a compact representation of a 3D surface by collecting geometric or topological measurements into histograms. Approaches such as *Spin Images* [13], *Unique Shape Context* [29] and *RoPs* [10] rely on the spatial distribution of the points on the surface, while others like *FPFH* [24] and *SHOT* [25] exploit geometric properties of the surface such as normals or curvatures. Rotation invariance is achieved using either a Local Reference Frame or a Reference Axis.

Learned 3D Local Descriptors The impressive progress in image recognition yielded by deep learning has inspired similar approaches to learn descriptor from 3D data. However, the unorganized nature of point clouds makes this extension not straightforward. As a consequence, several parallel tracks regarding the representation of the input data have emerged. Early works represent a 3D object as a collection of 2D views [27, 31]. Another approach concerns dense 3D voxel grids, with voxels containing either a binary occupancy grid [18, 32] or an alternative representation of the surface [36]. To limit the memory occupancy of voxel grids, researchers either rely on coarse spatial resolutions, which, however, introduce artifacts and hinder the ability to learn fine geometric structures, or on space partition methods like k-d trees or octrees [16, 28]. Other methods, differently, deploy high-dimensional hand-crafted features to parameterize the input point cloud and then use deep learning to project it into lower dimensional spaces [14, 3].

Learning from Raw 3D Data PointNet [21] and PointNet++ [22] are pioneering works presenting a general framework to learn features directly from raw point clouds data. Although yielding excellent performance in point cloud segmentation and classification tasks, these architectures have not been used yet to perform local surface description, likely due to the inability of providing rotation invariance. Nonetheless, PointNet is the core building block of PPFNet [4], which relies on raw point coordinates, normals and Point-Pair Features in order to learn a local feature descriptor. Indeed, due to the reliance on the PointNet architecture, PPFNet is not rotation invariant.

3. Proposed Method

In this section we present the whole pipeline of our method, graphically illustrated in Figure 1. Please note that our encoder only contains correlation layers, *i.e*. it does not include a max pooling layer at the end to learn a pose-invariant descriptor, which is instead present in the architectures proposed in [2].

3.1. Background

As we rely on Spherical CNNs, to make the paper self-contained we provide a brief overview of the mathematical model behind it. For more details, please refer to [2].

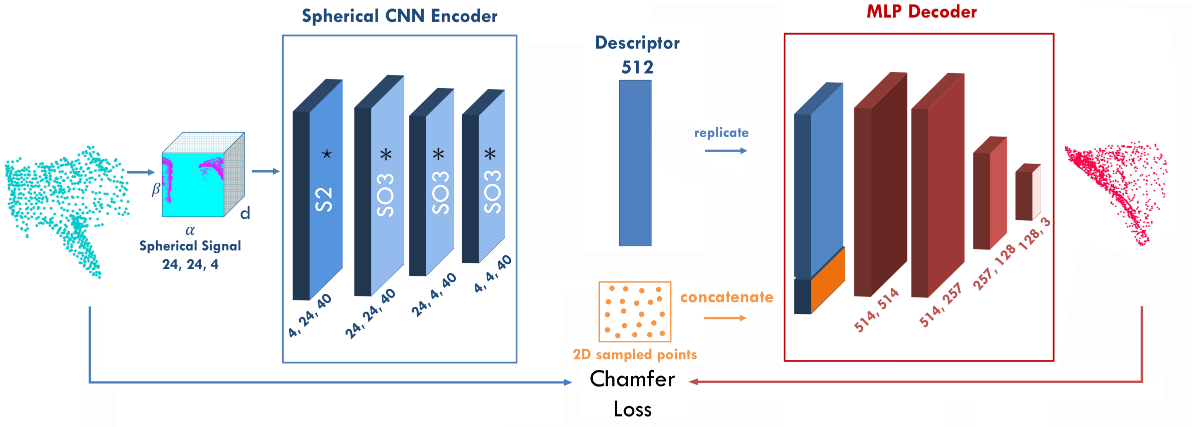

Figure 1: Architecture of the proposed method. The points within the local support of a given feature point $\mathbf{p}$ are converted into a spherical signal representation, and then sent through the spherical encoder to get an equivariant descriptor. The numbers below the spherical signal indicate the number of cells along α, β and d. The decoder reconstructs the original point cloud deforming sampled 2D points according to the descriptor. Operations in the encoder are implemented through the Generalized Fourier Transform with signals discretized according to a bandwidth parameter [2]. The triplets below the encoder layers indicate input bandwidth, output bandwidth and number of channels. As for the decoder, the pairs indicate the number of input and output channels, respectively.

The basic intuition behind Spherical CNNs can be grasped by analogy with the classical planar correlation used by traditional CNNs. As explained in [2], the value of the output feature map at $x \in \mathbb{Z}^2$ in a planar correlation can be understood as the inner product between the input feature map and the learned filter shifted by x. By analogy, the value of the output feature map at $R \in \mathrm{SO}(3)$ in a spherical correlation can be understood as the inner product between the input feature map and the learned filter, *rotated* by R.

A source of confusion when switching from traditional to spherical CNNs is that the space where input signals, for instance point clouds, and feature maps live is different: the former live in $\mathbb{R}^3$, while the latter live in $\mathrm{SO}(3)$. Therefore, when we read the value of a feature map, we are getting the response of the filter for a specific rotation, not for a location in the input cloud. This is not the case with traditional correlations, where both the input images and the feature maps live in $\mathbb{Z}^2$, and the concept of receptive field of a feature map is more intuitive.

Some useful definitions to understand spherical CNNs also from a formal point of view are given below.

The Unit Sphere S^2 can be defined as the set of points $x \in \mathbb{R}^3$ with norm 1. It is a two-dimensional manifold, which can be parameterized by spherical coordinates $\alpha \in [0, 2\pi]$ (azimuth) and $\beta \in [0, \pi]$ (inclination).

Spherical Signals the kernels of our Spherical encoder are designed as continuous K-valued functions: $f : S^2 \to \mathbb{R}^K$, where K is the number of channels.

Rotations A rotation in three dimensions lives in a three-dimensional manifold called $\mathrm{SO}(3)$, the "special orthogonal group". As in [2] the rotation group $\mathrm{SO}(3)$ can be parameterized by ZYZ-Euler angles $\alpha \in [0, 2\pi], \beta \in [0, \pi]$, and $\gamma \in [0, 2\pi]$. Rotations can be represented by 3×3 matrices that preserve distance (i.e. $\|Rx\| = \|x\|$) and orientation ($det(R) = +1$). If we represent points on the sphere as 3D unit vectors x, a rotation can be performed by using the matrix-vector product Rx.

Rotations of Spherical Signals The spherical correlation operator needs to rotate the filters on the sphere. For this purpose, [2] introduces the operator L_R that takes a function f and produces a rotated function $L_R f$ by composing f with the rotation R^{-1}:

$$[L_R f](x) = f(R^{-1}x) \tag{1}$$

Spherical Correlation Denoting with $\langle \psi, f \rangle$ the inner product on the vector space of spherical signals defined as in [2], the correlation between a K-valued spherical signal f and a filter ψ, $f, \psi : S^2 \to \mathbb{R}^K$ can be formalized as:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{2}$$

This is the operation performed by the first layer of our encoder (Figure 1). Unlike the standard definition of spherical convolution [5], which gives as output a function on the sphere S^2, the spherical correlation yield a signal on $\mathrm{SO}(3)$. The use of a conventional convolution definition would limit the expressive capacity of the network due to the symmetry along the Z axis of the learned filters.

Rotation of $\mathrm{SO}(3)$ Signals Similarly to what has been defined for spherical correlation in Eq. (2), to define a correlation in $\mathrm{SO}(3)$ the operator in Eq. (1) must be generalized so that it can act on $\mathrm{SO}(3)$. For a signal $h : \mathrm{SO}(3) \to \mathbb{R}^K$, and $R, Q \in \mathrm{SO}(3)$:

$$[L_R h](Q) = h(R^{-1}Q). \tag{3}$$

The term $R^{-1}Q$ in Eq. (3) denotes the composition of rotations.

Rotation Group Correlation Likewise in Eq. (2), we can define the correlation between a signal and a filter on the rotation group, $h, \psi : \mathrm{SO}(3) \to \mathbb{R}^K$, as follows:

$$[\psi * h](R) = \langle L_R \psi, f \rangle = \int_{\mathrm{SO}(3)} \sum_{k=1}^{K} \psi_k(R^{-1}Q) h_k(Q) dQ. \tag{4}$$

This is the operation performed by the all the layers of our encoder but the first one (Figure 1). The integration measure dQ is the invariant measure on $\mathrm{SO}(3)$, which may be expressed in ZYZ-Euler angles as $d\alpha \sin(\beta) d\beta d\gamma/(8\pi^2)$. Please note that unlike in [2] for better clarity we denote as $\star$ the spherical correlation (2) while with $*$ the rotation group correlation (4).

3.2. Learning from Spherical Signals

Our feature encoder operates on signals defined in a spherical domain. Hence, the local geometry surrounding a feature point needs to be converted into a spherical representation. A common strategy adopted by [2, 7] is to project a 3D mesh onto an enclosing discretized sphere using a ray-casting scheme. Since our input data is not a regular water-tight mesh, but a point cloud corresponding to the neighborhood of the point we wish to describe, we first convert 3D points into a spherical coordinate system and then construct a quantization grid in this new coordinate system, similarly to [35]. The i-th cell in the quantization is identified with three spherical coordinates $(\alpha[i], \beta[i], d[i]) \in S^2 \times D$ where $\alpha[i]$ and $\beta[i]$ represent the azimuth and inclination angles of its center and $d[i]$ is the distance from the sphere center. The K-valued spherical signal $f : S^2 \to \mathbb{R}^K$ is then composed by K concentric spheres corresponding to the number of subdivisions along the distance axis, each sphere encoding the density of the points within each cell $(\alpha[i], \beta[i])$ at a given distance $d[k]$. To take into account the non-uniform spacing in the spherical space, cells near the south or north pole are wider in spherical coordinates, as discussed in [35].

A spherical signal is computed on the local neighborhood of every input point we wish to describe (i.e., every *keypoint*). The signal then goes through our architecture to learn an equivariant bottleneck layer, which can then be used as a descriptor of the local geometry around the keypoint.

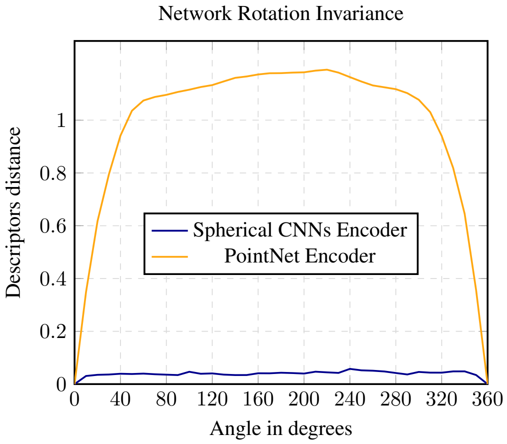

Figure 2: Comparison between PointNet and Spherical CNN used as encoders in our framework.

3.3. Rotation-Equivariant Descriptor

The main novelty of our approach is the use of Spherical CNNs as encoder to learn an equivariant bottleneck layer.

Learning an equivariant bottleneck removes the requirement to have invariant representations as input to the network at training time as the only way to achieve rotation invariance, the standard approach in existing proposals [3, 14]. In our framework, instead, we can delay the choice on how to canonically orient the descriptor at test time, which brings in two important benefits. On one hand, we do not have to choose a specific way to orient the input, e.g. a specific LRF, at training time, which means that we can train the network to learn the descriptor from less pre-processed input data than existing proposals, moving a step closer toward end-to-end descriptor learning. On the other hand, not using a LRF at training time frees our method from unavoidable errors of the LRF itself, which in turn inject noise in the training process. We expect both benefits to concur to increase the effectiveness of the learned descriptor.

Moreover, from a practical point of view, being able to train our descriptor without tying it to a specific LRF enables us to choose the best way to define a canonical representation at test time without training the network from scratch. Finally, it also opens up the possibility to use different LRFs for different test data, although we have not explored this property in the experimental results reported in this paper.

Please note that a truly rotation-equivariant CNN like Spherical CNNs is mandatory in our framework, as anticipated in the introduction. Indeed, only a descriptor that lives in $\mathrm{SO}(3)$ can be rotated after having been computed, i.e. only the output of a Spherical CNN to date. All the other standard representations, e.g. the output of a Multi Layer

Perceptron (MLP) as used in PointNet, cannot be rotated after having been computed. Therefore, if we want to use them in our framework where the input is not canonically oriented for the reasons discussed above, we can only hope the network learns to obtain directly a rotation-invariant descriptor by observing rotated versions of the same neighborhood during training without explicit supervision, which is however a harder task in our setup than learning an equivariant descriptor.

We have validated how harder this is experimentally, by using a standard PointNet encoder instead of the spherical one to learn an invariant descriptor. Results of the comparison are shown in Figure 2. Please note that equivariance is a theoretical property of a Spherical CNN, regardless of whether it is trained or not. Indeed, in the results in Figure 2, the Spherical encoder has **not** been trained, while the PointNet encoder has been trained on the 3DMatch Benchmark presented in Section 4.1. Given a neighborhood, we rotate it around a random axis by a growing angle, whose value is reported along the horizontal axis of the chart. For every rotation, we pass the rotated neighborhood through a Spherical CNN encoder and a PointNet encoder. The output of the Spherical CNN is then rotated by the inverse of the applied rotation (simulating the availability of a perfect LRF) and the distance between the descriptor obtained from the rotated neighborhood and the descriptor obtained from the un-rotated neighborhood is plotted. We can clearly see that PointNet fails to learn an invariant descriptor in our setup, while the equivariant representation provided by a Spherical CNN can achieve almost perfect invariance when properly rotated.

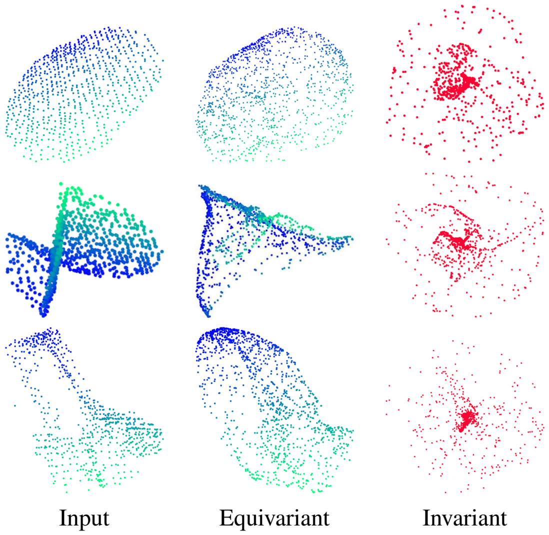

Figure 3: Comparison between the reconstructions obtained when using the Spherical CNN encoder to learn an equivariant versus an invariant bottleneck. Results after 10K training iterations.

Moreover, even if PointNet were able to learn a perfectly invariant bottleneck, we have found experimentally that this would result in low quality reconstructions. The reason is that it is not possible for frameworks like FoldingNet/AtlasNet to converge to sensible reconstructions if the learned bottleneck does not contain any pose information, i.e. it is almost perfectly invariant. This is shown in Figure 3. where we compare the quality of the reconstructions produced by our framework when using an equivariant bottleneck layer versus an invariant one. The invariant one in this case is obtained by removing the last $\mathrm{SO}(3)$ correlation layer from our encoder, which produces the equivariant descriptor in our architecture, and adding a max pooling layer selecting the maximum of each one of the now top-level 40 feature maps, followed by a fully connected layer to expand the codeword dimensionality to 512. As shown in the figure, if the encoder produces an invariant descriptor, the decoder doesn't have enough information to know in which pose it should reconstruct the input so as to minimize the loss. The best it can do is to produce reconstructions trying to account for all possible rotations of the input, e.g. the atom-like structures depicted in the last column, almost ignoring the invariant bottleneck layer.

3.4. Invariant Feature Descriptor

To obtain an invariant descriptor at test time, which can be matched across poses, we have to compute a canonical orientation for the equivariant descriptor. We have investigated two ways of doing it.

The first is the most intellectually satisfying, and leverages again the peculiar properties of Spherical CNNs. Indeed, every bin of a feature map in a Spherical CNN represents an element of $\mathrm{SO}(3)$, *i.e.* a potential LRF. This has been already exploited to align full shapes in [7], by finding the $\arg\max$ of the correlation between two feature maps. Note that we cannot use the same approach in the context of invariant descriptor matching, as this would require a costly computation to compute the distance between every pair of source and target descriptors.

However, because of the equivariance property, we can recover an aligning pose by processing the two descriptors separately. Let $[\psi * h](R)$ be the descriptor, *i.e.* a feature map, obtained when processing the input signal f, and let $[\psi * m](R)$ the one obtained when we process a rotated version of f, $g(x) = [L_Q f](x) = f(Q^{-1}x)$. Due to equivariance, the same rotation exists between inner feature maps h and m, *i.e.* $m(R) = [L_Q h](R) = h(Q^{-1}R)$, and recursively between descriptors, *i.e.*

$$\begin{aligned}[\psi * m](R_m) &= [\psi * [L_Q h]](R_m) \\ &= \langle L_{R_m}\psi, L_Q h\rangle \\ &= \langle L_{Q^{-1}R_m}\psi, h\rangle \\ &= [\psi * h](Q^{-1}R_m) \coloneqq [\psi * h](R_h) \qquad (5)\end{aligned}$$

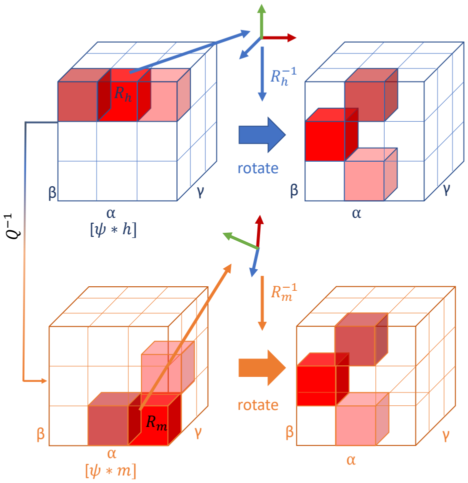

Figure 4: Self-orienting property of the learned equivariant descriptor. Every bin of our bottleneck layer corresponds to three Euler angles which define a rotation. If the descriptor is computed starting from a rotated input (second row), the values shifts in the feature maps. By finding two corresponding bins in the two descriptors and rotating them by the inverse of the corresponding rotations, the descriptors can be aligned, i.e. become pose invariant.

In other words, chosen an entry in a descriptor $[\psi * h]$ obtained when processing f, e.g. R_h, if, when the input is rotated by Q, we are able to find the same entry independently in the rotated descriptor $[\psi * m]$, we will find it at rotation $R_m = QR_h$. Therefore, given the two descriptors, we can align them to a common pose by applying the inverse of such rotations

$$\begin{aligned} [L_{R_m^{-1}}[\psi * m]](R) &= [\psi * m](R_m R) \\ &= [\psi * [L_Q h]](R_m R) \\ &= [\psi * h](Q^{-1} R_m R) \qquad (6) \\ [L_{R_h^{-1}}[\psi * h]](R) &= [\psi * h](R_h R) \\ &= [\psi * h](Q^{-1} R_m R) \qquad (7) \end{aligned}$$

as shown by last terms of the transformations being equal (and graphically in Figure 4). Please note that all transformations are applied to the descriptor (which is a feature map) obtained from the unrotated input and not to the input itself, i.e. we can rotate the descriptor computed from an unoriented input to achieve rotation invariance. The dimensionality of ours descriptor does not change under rotations, as it is rotated by remodulating the spherical harmonics functions resulting from its Fourier transform. A through treatment of this topic can be found in [23].

The problem of defining a repeatable LRF then translates into that of finding the same bin under rotations given a feature map. A simple choice could be the maximum of the feature map. Under perfect equivariance, the maximum would provide a repeatable anchor point across rotations, and therefore a repeatable rotation to obtain invariant descriptors. However, the network is not perfectly equivariant, due to numerical approximations and the use of non linearities (ReLUs) between layers, and also the feature map of the same keypoint seen in two different views changes due to other nuisances (occlusions, clutter, sampling). We have verified experimentally that the maximum of a feature map alone is not robust enough to define a repeatable LRF.

We have investigated several strategies to identify the same location of the feature map under rotations. The one that has given the best results so far starts by analyzing only the k bins corresponding to the top k values of the feature map, including the maximum. We then compute the density of top values in a $3 \times 3 \times 3$ neighborhood of every such bin. The bin with the maximum density is used to compute the required rotation. In the case of ties, we select the neighborhood with the largest value within it. Once we have selected a neighborhood, we average all the bins corresponding to the top values within it to get the final rotation. By means of the proposed algorithm, we have been able to define a *self-orienting* descriptor, an original trait of our proposal.

As our tests indicate the repeatability of the above defined LRF to be far from the optimal performance attainable with the equivariant descriptor, we have also assessed its performance when we make it invariant at test time by computing the canonicalizing rotation with the help of an external local reference frame extracted from the input cloud. We stress here that, although we compute an LRF on the input data, we again rotate the computed descriptor and not the input data. Moreover, even in this case, we perform LRF extraction only at test-time, as discussed above, so the chosen LRF algorithm does not affect the quality of the training data.

3.5. Decoder and Loss

Differently from [4], our goal is to reconstruct the whole set of points representing the local neighborhood of a given feature point $\mathbf{p}$. Inspired by [8] and [34], our decoder will try to deform points in $\mathbb{R}^2$ to surface points in $\mathbb{R}^3$ according to the learned descriptor. Given a feature representation $\mathbf{d}$ for a 3D surface, let $\mathcal{A}$ be a set of points sampled in the unit square $[0, 1]^2$, the descriptor $\mathbf{d}$ is concatenated with the sampled point coordinates $(a_x, a_y) \in \mathcal{A}$ and then forwarded through a stack of MLP layers as shown in Figure 1. We then minimize the Chamfer loss between the set of generated 3D points and the input points.

In particular, let $\mathcal{S}$ be the set of 3D input points belonging to the neighbohood of $\mathbf{p}$ and $\mathcal{S}^\star$ the set of points reconstructed by the decoder. During training, we minimize the following loss

$$\mathcal{L}(\mathcal{S},\mathcal{S}^\star)_\theta = \frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}}\min_{\mathbf{x}^\star\in\mathcal{S}^\star}\|\mathbf{x}-\mathbf{x}^\star\|_2 + \frac{1}{|\mathcal{S}^\star|}\sum_{\mathbf{x}^\star\in\mathcal{S}^\star}\min_{\mathbf{x}\in\mathcal{S}}\|\mathbf{x}^\star-\mathbf{x}\|_2. \tag{8}$$

The term $\min_{\mathbf{x}^\star\in\mathcal{S}^\star}\|\mathbf{x}-\mathbf{x}^\star\|_2$ enforces that any 3D point $\mathbf{x}$ in the original point cloud has a matching 3D point $\mathbf{x}^\star$ in the reconstructed point cloud, and the term $\min_{\mathbf{x}\in\mathcal{S}}\|\mathbf{x}^\star-\mathbf{x}\|_2$ enforces the matching viceversa. The overall loss is the sum of the two terms to enforce that the distance from $\mathcal{S}$ to $\mathcal{S}^\star$ and the distance viceversa have to be small simultaneously.

3.6. Network and training parameters

To learn our descriptor, we use one S^2 convolution layers and three SO(3) convolution layers with constant number of channels, 40, while the bandwidths is set to 24 for the first three layer and 4 for the last one, which results in a descriptor with 512 entries. The architecture of our decoder is made of 4 fully-connected layers, with ReLU nonlinearities on the first three layers and tanh on the final output layer. The network is trained with mini-bacthes of size 32 by using ADAM [15]. The starting learning rate is set to 0.001 and is decayed every 4000 iterations. We train the network for 14 epochs.

4. Experimental Results

4.1. Experimental setup

To test our proposal, we use the standard benchmark for the evaluation of learned 3D descriptors, the 3DMatch benchmark [36]. This benchmark addresses registration of unordered 3D views and the dataset has been put together by merging a large part of the publicly available datasets such as Analysis-by-Synthesis [30], 7-Scenes [26], SUN3D [33], RGB-D Scenes v.2 [17] and Halberand Funkhouser [11]. It contains 62 scenes in total, and, following [3], we use 54 for training and validation, while 8 scenes are used only at test time to run comparisons. The dataset already provides so-called fragments, *i.e.* the point clouds resulting from the fusion of 50 consecutive depth frames, for the test scenes, and we obtained the training fragments generated by the same methodology as the authors of [3]. We also procure the rotated version of the 3D Match benchmark, generated by the same authors by rotating all the fragments in the 3DMatch benchmark with randomly sampled axes and angles over the whole rotation space.

We use the same setup proposed in [3]: we downsample the fused fragments with a voxel grid filter of size 2 cm and compute surface normals using [12] in a 17-point neighborhood; we consider a radius of 30 cm to define the neighborhood of a keypoint.

4.2. Evaluation methodology

As for metrics, following the evaluation methodology proposed by [3], we consider the *recall* of pairs of fragments correctly registered among those with at least 30% overlap. A pair of fragments is considered correctly registered if the number of correctly matched keypoints is greater than the inlier ratio threshold τ_2, set to 5% of the extracted keypoints. Two keypoints are correctly matched if their l_2 distance is below a threshold $\tau_1 = 10$ cm. For each fragment, the descriptors are computed on 5000 uniformly sampled points, provided with the benchmark [36]. For handcrafted descriptors we used the implementation in PCL [1], while for learned descriptors results were taken from [3].

4.3. Quantitative results

Results of the tests on the 3D Match benchmark in terms of recall are reported in Table 1. With Ours SO we refer to the self orienting descriptor introduced in section 3.4, while with Ours LRF we refer to the descriptor oriented with an external local reference frame. In particular for this experiments we have used the LRF algorithm proposed in [20], which we will denote as FLARE according to the acronym used in its PCL implementation [1].

The first outcome of our experiments is that the use of an external LRF outperforms the self-orienting variant of our algorithm. Note that the two columns describe exactly the same equivariant descriptor under two different ways to compute a canonical orientation. Hence, the highest one is indicative of the quality of the learned descriptor itself. Although the performance of the self-orienting variant of our method is inferior to our descriptor oriented by an external LRF, it is remarkable that it delivers the second best recall on the dataset, *i.e.* it would provide state-of-the-art performance if we were not to orient our equivariant descriptor also with an external LRF. Our self-orienting variant is closely followed by SHOT and USC, *i.e.* two handcrafted descriptors, while the other tested methods deliver significantly lower recalls. The best learned approach is PPFFoldNet. The better performance of SHOT and USC with respect to PPFFoldNet offers support to the inspiring ideas behind this work: deep learning alone, if constrained to learn from highly engineered representations, cannot be a guarantee of superior performance. It is also interesting to analyze these results in light of our main claim: to learn an equivariant descriptor and then orient it to achieve invariance instead of learning directly an invariant one boosts its quality. If we compare the performance of our method when oriented with both tested variants against methods learning from invariant representations, like PPFFoldNet and CGF, we can interpret the large gap in performance (0.23 and 0.47 points of recall from the external LRF variant, respectively), as a validation of the drawbacks of existing learned descriptors discussed in the introduction. In Figure 5, we

Table 1: Results on the 3DMatch benchmark. Test data are from SUN3D [33], except for *Red Kitchen* data which is from 7-scenes [26]. Best result on each row is in bold.

	FPFH [24]	Spin Image [13]	SHOT [25]	USC [29]	3DMatch [36]	CGF [14]	PPFNet [4]	PPFFoldNet [3]	Ours SO	Ours LRF
Kitchen	0.7391	0.6561	0.8893	0.9308	0.5810	0.4605	0.8972	0.7866	0.8854	**0.9763**
Home 1	0.7885	0.7564	0.8974	0.9103	0.7244	0.6154	0.5577	0.7628	0.9487	**0.9615**
Home 2	0.6442	0.6731	0.8221	0.7788	0.6154	0.5625	0.5913	0.6154	0.8654	**0.8942**
Hotel 1	0.8142	0.6770	0.9336	0.9204	0.5442	0.4469	0.5796	0.6814	0.9204	**0.9823**
Hotel 2	0.7115	0.6346	0.8750	0.8462	0.4808	0.3846	0.5796	0.7115	0.8462	**0.9519**
Hotel 3	0.8889	0.7407	0.8889	0.8889	0.6111	0.5926	0.6111	0.9444	0.9630	**0.9815**
Study	0.7432	0.4692	0.8630	0.8664	0.5171	0.4075	0.5342	0.6199	0.8870	**0.9178**
MIT Lab	0.7013	0.4545	0.8312	0.8052	0.5065	0.3506	0.6364	0.6234	0.8182	**0.8701**
Average	0.7539	0.6327	0.8751	0.8684	0.5726	0.4776	0.6231	0.7182	0.8918	**0.9420**

Table 2: Results on the rotated 3DMatch benchmark. Test data are from SUN3D [33], except for *Red Kitchen* data which is from 7-scenes [26]. Best result on each row is in bold.

	FPFH [24]	Spin Image [13]	SHOT [25]	USC [29]	3DMatch [36]	CGF [14]	PPFNet [4]	PPFFoldNet [3]	Ours SO	Ours LRF
Kitchen	0.7451	0.6502	0.8794	0.9170	0.004	0.4466	0.002	0.7885	0.8893	**0.9783**
Home 1	0.7949	0.7628	0.8910	0.9103	0.0128	0.6667	0.0000	0.7821	0.9423	**0.9679**
Home 2	0.6587	0.6635	0.8317	0.7548	0.0337	0.5288	0.0144	0.6442	0.8413	**0.8894**
Hotel 1	0.8142	0.6903	0.9425	0.9292	0.0044	0.4425	0.0044	0.6770	0.9204	**0.9779**
Hotel 2	0.7212	0.6635	0.8654	0.8558	0.0000	0.4423	0.0000	0.6923	0.8558	**0.9615**
Hotel 3	0.9259	0.7222	0.9074	0.9074	0.0096	0.6269	0.0000	0.9630	0.9074	**0.9815**
Study	0.7260	0.4692	0.8493	0.8836	0.0000	0.4178	0.0000	0.6267	0.8733	**0.9110**
MIT Lab	0.7532	0.4935	0.8312	**0.8571**	0.0026	0.4156	0.0000	0.6753	0.7922	0.8442
Average	0.7674	0.6394	0.8747	0.8769	0.0113	0.4776	0.0026	0.7311	0.8778	**0.9387**

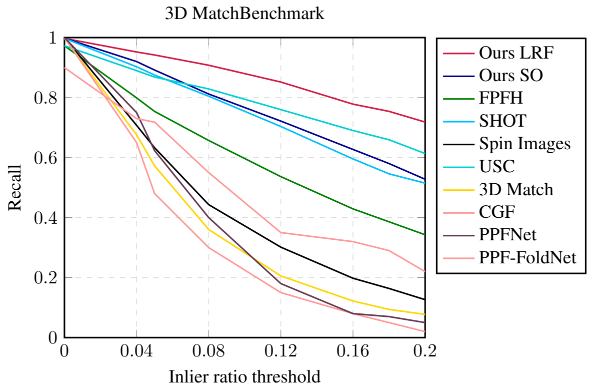

Figure 5: Results under varying inlier ratio threshold τ_2.

report results when varying the threshold τ_2 on the percentage of correct matches to consider a pair as correctly registered, as done in [3]. Our proposal oriented with an external LRF outperforms the others for all thresholds, and our self-orienting variant attains again recall values similar to SHOT, and slightly inferior to USC for the largest thresholds.

Finally, results of the tests on the rotated 3D Match benchmark are reported in Table 2. The dataset was proposed in [3] to test robustness against large rotations, not present in the original benchmark. As expected, all rotation-invariant methods obtain performance similar to the results reported in Table 1, and our equivariant descriptor oriented with the external LRF still delivers by far the best performance.

5. Conclusions

In this study, we have shown how the problem of learning an effective descriptor can be separated into the orthogonal problems of learning a robust equivariant representation and defining a good canonical orientation to make it invariant at test time. Our proposal to learn an equivariant representation in an unsupervised way leverages as encoder the recently proposed Spherical CNNs and turns out highly effective in tackling the first problem. When coupled with a robust algorithm to compute a local reference frame from the input cloud, it significantly advances the state of the art on a challenging benchmark.

We have also shown how the very same framework could be used to define a canonical orientation by exploiting the peculiar nature of the feature maps computed by the Spherical CNNs. Although this approach delivers performance on par with the state of the art, it is so far inferior to the use of an external LRF. Yet, we believe the elegance and potential implications of this technique were valid reasons to also communicate it and call for further studies along this line of research, with the aim of defining an end-to-end learned solution to the problem of invariant 3D description.

6. Acknowledgments

We gratefully acknowledge NVIDIA Corporation with the donation of the Titan V GPU used in this work.

References

[1] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation. *IEEE Robotics & Automation Magazine*, 19(3):80–91, 2012. 7

[2] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. *arXiv preprint arXiv:1801.10130*, 2018. 2, 3, 4

[3] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018. 1, 2, 4, 7, 8

[4] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 195–205, 2018. 1, 2, 6, 8

[5] James R Driscoll and Dennis M Healy. Computing Fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 15(2):202–250, 1994. 3

[6] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010. 1

[7] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018. 2, 4, 5

[8] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3D surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 2, 6

[9] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3D local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016. 1

[10] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3D local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013. 1, 2

[11] Maciej Halber and Thomas Funkhouser. Fine-to-coarse global registration of RGB-D scans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1755–1764, 2017. 7

[12] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992. 7

[13] Andrew E. Johnson and Martial 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. 1, 2, 8

[14] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 153–161, 2017. 1, 2, 4, 8

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[16] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 2

[17] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3D scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014. 7

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

[19] Alioscia Petrelli and Luigi Di Stefano. On the repeatability of the local reference frame for partial shape matching. In *2011 International Conference on Computer Vision*, pages 2244–2251. IEEE, 2011. 1

[20] Alioscia Petrelli and Luigi Di Stefano. A repeatable and efficient canonical reference for surface matching. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 403–410. IEEE, 2012. 1, 7

[21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 2

[23] Torben Risbo. Fourier transform summation of Legendre series and D-functions. *Journal of Geodesy*, 70(7):383–396, 1996. 6

[24] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009. 1, 2, 8

[25] Samuele Salti, Federico Tombari, and Luigi Di Stefano. SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014. 1, 2, 8

[26] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew 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. 7, 8

[27] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks

for 3D shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[28] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2088–2096, 2017. 2

[29] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010. 1, 2, 8

[30] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 323–332. IEEE, 2016. 7

[31] Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1553, 2016. 2

[32] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017. 2

[33] Jianxiong Xiao, Andrew Owens, and Antonio 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. 7, 8

[34] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 2, 6

[35] Yang You, Yujing Lou, Qi Liu, Lizhuang Ma, Weiming Wang, Yuwing Tai, and Cewu Lu. PRIN: Pointwise rotation-invariant network. *arXiv preprint arXiv:1811.09361*, 2018. 4

[36] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1802–1811, 2017. 1, 2, 7, 8