ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Unsupervised Learning of Local Equivariant Descriptors for Point Clouds

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Marcon, M., Spezialetti, R., Salti, S., Silva, L., Di Stefano, L. (2022). Unsupervised Learning of Local Equivariant Descriptors for Point Clouds. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 44(12), 9687-9702 [10.1109/TPAMI.2021.3126713].

Availability:
This version is available at: https://hdl.handle.net/11585/864730 since: 2022-12-06

Published:
DOI: http://doi.org/10.1109/TPAMI.2021.3126713

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

08 October 2026

This is the final peer-reviewed accepted manuscript of:

M. Marcon, R. Spezialetti, S. Salti, L. Silva and L. D. Stefano, "Unsupervised Learning of Local Equivariant Descriptors for Point Clouds," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 12, pp. 9687-9702, 1 Dec. 2022

The final published version is available online at ***https://dx.doi.org/10.1109/TPAMI.2021.3126713***

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Unsupervised Learning of Local Equivariant Descriptors for Point Clouds

Marlon Marcon, Riccardo Spezialetti, Samuele Salti, Luciano Silva and Luigi Di Stefano,

Abstract—Correspondences between 3D keypoints generated by matching local descriptors are a key step in 3D computer vision and graphic applications. Learned descriptors are rapidly evolving and outperforming the classical handcrafted approaches in the field. Yet, to learn effective representations they require supervision through labeled data, which are cumbersome and time-consuming to obtain. Unsupervised alternatives exist, but they lag in performance. Moreover, invariance to viewpoint changes is attained either by relying on data augmentation, which is prone to degrading upon generalization on unseen datasets, or by learning from handcrafted representations of the input which are already rotation invariant but whose effectiveness at training time may significantly affect the learned descriptor. We show how learning an *equivariant* 3D local descriptor instead of an invariant one can overcome both issues. LEAD (Local EquivAriant Descriptor) combines Spherical CNNs to learn an equivariant representation together with plane-folding decoders to learn without supervision. Through extensive experiments on standard surface registration datasets, we show how our proposal outperforms existing unsupervised methods by a large margin and achieves competitive results against the supervised approaches, especially in the practically very relevant scenario of transfer learning.

Index Terms—Deep learning on point clouds, local features, equivariant, unsupervised, feature learning, registration.

1 Introduction

Matching 3D local *features* is a well-established approach to find correspondences between shapes which help addressing 3D Computer Vision tasks like surface registration, shape classification and retrieval, object recognition, and more. Effective pipelines leveraging the feature-matching paradigm hinge upon compact representations of the local geometry referred to as *descriptors*. Descriptors must be designed to be either *invariant* or *robust* to the nuisances encountered in 3D Computer Vision scenarios, such as viewpoint changes, sensor noise, point density variations, occlusions and clutter. Delineating *handcrafted* functions to extract robust and distinctive features from 3D data has a long history in Computer Vision [1], [2], [3], [4], [5], [6]. Yet, due to the challenging requirements mentioned above, designing an effective local descriptor turns out a pretty complex endeavour. Indeed, as highlighted in the most recent evaluation in the field [7], it is still unclear which surface attributes are most conducive to good representations, any choice leading to diverse limits and merits across the considered datasets.

More recently, in the wake of the striking results achieved by data-driven models in countless 2D vision tasks, surface description started to be addressed through deep neural networks. In the migration process from handcrafted to *deep* descriptors, a peculiar challenge concerning the unstructured nature of the main 3D data representations, such as, in particular, point clouds, has emerged. Indeed, deep learning for 2D vision owes a good deal of its success to convnets deploying strong priors suited to data structured as tensors. Thus, scholars have been experimenting with a variety of input structures ranging from *voxel* grids [8], [9] to high-dimensional embeddings [10] akin to traditional handcrafted descriptors [2], [3]. We argue that feeding deep neural networks with carefully engineered input representations may hinder their ability to learn from data those best suited to solve the problem at hand, as learning rather than designing features has been one key reason behind the deep learning success. For instance, as vouched by the experiments in [7], the *Point Pair Features* [11] utilized by PPFNet [12] and PPF-FoldNet [13] may not be the best representation when data are acquired by a spectrum of diverse sensing modalities. Such a choice, thus, would limit the performance of PPF-FoldNet on those datasets where the chosen input representation is sub-optimal. Likewise, the CGF features proposed in [10] rely on the handcrafted descriptor presented in [3], which renders the former inherently less amenable to settings where the latter cannot capture the key shape traits effectively.

Most learned 3D descriptors are trained in a supervised framework [9], [10], [14], [15]. Training data usually comes in the form of partial scans of a 3D scene or object whose aligning transformations are provided by human annotators. Such ground-truth transformations can be used to define corresponding keypoints across overlapping views according to some proximity threshold or similar criteria. Corresponding keypoints form the positive examples to train the descriptors, while a representative subset of the negative examples is usually mined with some heuristic rules from the large set of pairs of non-corresponding points [16], [17]. Beside the time-consuming and cumbersome process to annotate the training data, all these hand-crafted decisions, and, in particular, the effectiveness of the negative mining strategy, play an important role in the resulting performance of the learned descriptors [9].

Finally, invariance to viewpoint changes is paramount to 3D

- *M. Marcon is with the Department of Software Engineering, Federal University of Technology - Paraná, Brazil. E-mail: marlonmarcon@utfpr.edu.br*
- *L. Silva is with the Department of Computer Science, Federal University of Paraná, Brazil.*
- *R. Spezialetti, S. Salti and L. Di Stefano are with the Department of Computer Science and Engineering (DISI), University of Bologna, Italy.*

descriptors. Yet, some learned approaches are not design to be robust or invariant to such changes and exhibit a performance drop when trained and tested on different 3D rotations [8], [12], [18]. The remain learned descriptors, instead, achieve rotation invariance by expressing the 3D coordinates of the points belonging to the input patch w.r.t. a coordinate system centered at the feature point and defined according to a *local reference frame* (LRF) [9], [10] or a *reference axis* (RA) [13], *i.e*. the same strategy followed also by hand crafted methods [5]. For instance, CGF [10] and 3DSmoothNet [9] rely on the LRFs proposed by [5] and [6], respectively. However, these are, again, handcrafted choices that may cause noisy supervision to be injected into the training process due to the imperfect repeatability of the actual algorithm deployed to compute the LRF. As a matter of fact, and similarly to descriptors, the literature on LRFs vouches for the lack of a gold standard approach, with different algorithms behaving differently across datasets [19]. With the advent of new 3D convolution operators [20], [21], some of the most recent approaches in the field [14], [15] employ fully-convolutional architectures [22] to densely learn descriptors across the input cloud in one forward pass. Robustness to rotations in these methods is achieved by augmenting the training data by a random set of rotations, though, as pointed out previously, this approach may fail to generalize to rotations unseen at training time, in particular when training and testing happen on different datasets [15].

To overcome the limitations of existing approaches, in this paper we propose a novel unsupervised framework to learn a *rotation-equivariant* local surface descriptor directly from the raw input data. To do so, we combine Spherical CNNs [18], [23], a recently introduced deep learning machinery which extends the correlation operator to signals living in S^2 and $\mathrm{SO}(3)$, *i.e*. the space of 3D rotations defined formally in section 3, together with *plane-folding* operators [24], [25]. In our training architecture, a spherical encoder [23] learns to compress the geometric traits of the input 3D patch into a low-dimensional latent space while a folding decoder [24] warps a 2D grid to reconstruct the input patch. As usually done in unsupervised learning through encoder-decoder architectures, at inference time the decoder is discarded and the low-dimensional representation computed by the encoder provides the patch descriptor. Due to its unsupervised nature, our learning framework does not require ground-truth annotations or the choice of a non-corresponding pairs mining strategy [9], [14], [15] to supervise the network.

Encoder-decoder architectures have already been used to learn 3D descriptors [13]. Yet, while the encoder-decoder architecture proposed in [13] compresses and reconstructs pose-invariant Point Pair Features [11], ours does so for raw 3D coordinates under arbitrary poses. As it will be shown in section 5, pose information is needed to correctly reconstruct a patch of 3D points by a plane folding decoder under an arbitrary pose. To encode pose into the latent space, we rely on the unique *rotation-equivariance* property of Spherical CNNs. In fact, a spherical encoder consists of layers which compute feature maps defined on $\mathrm{SO}(3)$ that are equivariant with respect to a 3D rotation of the raw input data. Thus, we do not need to rely on any handcrafted choice to either express the input 3D patch in a canonical pose or remap it into a pose-invariant representation, Rather, we learn a rotation-equivariant bottleneck from the raw training data. Then, to pursue pose-invariant descriptor matching at test time, the equivariant representation provided by our spherical encoder can be canonicalized by applying a 3D rotation provided by an external LRF algorithm.

We prove the effectiveness of our approach through an extensive experimental evaluation carried out on the main benchmark datasets used in the field, which deal with *pairwise* surface registration in both indoor (3DMatch [8]) and outdoor (ETH [26]) environments. Despite the different nature of the surfaces present in the data as well as the diverse type of sensors used to capture them, in both datasets our method correctly aligns most of the scans and provides the new state-of-the-art results among local 3D descriptors based on unsupervised learning.

We can summarize our contributions as follows:

- We propose the first rotation-equivariant local descriptor which operates on point clouds and can be learned without any supervision from raw data. Purposely, we realize the first spherical auto-encoder for point clouds, *i.e*. a neural network architecture featuring a Spherical CNN that computes a rotation-equivariant bottleneck used to steer a plane folding decoder.
- To pursue our design, we carry out a thorough ablation study on the hyperparameters of its main architectural components, namely the filters used to perform correlation operations within the Spherical Encoder.
- Our optimal design yields state-of-the-art performance among local 3D descriptors based on unsupervised learning. In particular, our descriptor excels in generalization to unseen scenarios, a key feature to address many practical point clouds registration applications.

2 Related Works

In this section, we review the literature concerning 3D local descriptors. We start with a brief discussion on hand-crafted methods and then move on to more modern data-driven proposals. For the sake of completeness, we also mention recent approaches to apply deep learning on raw point cloud coordinates.

Handcrafted 3D Local Descriptors. Scholars have designed several hand-crafted functions to abstract the salient structural information of a 3D keypoint's neighborhood into a low-dimensional representation. To this end, some chosen attributes are typically quantised and summarized into *histograms*. The main proposals differ for the employed geometric or topological attributes [7]. Methods such as *Spin Images* [1], *Unique Shape Context (USC)* [3] and *RoPs* [4] deploy the distribution of point coordinates while others, like *FPFH* [2] and *SHOT* [5] leverage on geometric properties of the surface such as normals and curvatures. To handle viewpoint variations and attain invariance to rotation, the above mentioned descriptors rely on either a Local Reference Frame (LRF) or a reference axis (RA).

Learned 3D Local Descriptors. Deep learning is currently the most successful approach to analyze almost any kind of 2D visual data. This success has drawn attention toward learning *deep* local descriptors for 3D data [8], [10], [12], [13], [27]. The typical supervised workflow for descriptor learning entails the adoption of a Siamese architecture [28] alongside a loss [29], [30] amenable to pull similar features together, *i.e*. descriptors for the same 3D point acquired under different viewpoints, while pushing dissimilar ones apart. To manage the unstructured nature of point clouds, a 3D keypoint's neighbourhood is converted into a suitable structured representation. Purposely, while 3DMatch [8] and 3DSmoothNet [9] rely on TSDF [31] and smoothed density value voxel grids, respectively, *CGF* [10] adopts an high-dimensional representation that closely resembles the USC descriptor [3].

A multi-view approach can also be adopted: in [32] Li *et al*. integrate a differentiable renderer into a 2D neural network so to optimize a multi-view representation in order to learn a local feature descriptor. Unsupervised approaches, instead, propose to employ the latent codeword of an encoder-decoder architecture as a 3D feature descriptor. PPF-FoldNet [13] and 3D-PointCapsNet [33] learn to reconstruct the 4 dimensional Point Pair Feature [11], [34] of a local patch by a FoldingNet [25] decoder. While we use the same encoder-decoder formulation to perform unsupervised learning, we do not rely on a pre-defined and pose-invariant input representation as discussed in the introduction.

Other recent proposals employ fully convolutional networks [22] and data augmentation to learn a rotation-invariant 3D local descriptor from point clouds by a supervised approach. The first work in this direction concerns the *Fully Convolutional Geometric Features* (FCGF) proposed in [14], that deploy sparse convolution [21] to manage the unorganized structure of point clouds and densely extract a compact local embedding. Similarly, *D3Feat* [15] leverages KPConv [20] to perform convolutions on raw 3D coordinates and predicts both a detection score and a feature descriptor at each 3D location in the input cloud. These methods are highly efficient due to the ability to extract dense features in just one forward pass. Yet, they generalize poorly to novel data dealing with scene and geometries exhibiting different traits wrt those observed at training time, as we will show in section 6. This weakness in transfer learning is particularly critical for descriptors learned by supervised approaches because it limits applicability to datasets for which ground truth information is available, which, in turn, is unlikely the case in many practical settings.

Eventually, we point out that this manuscript consolidates and extends the preliminary results presented in [27].

Deep Learning on point clouds. Processing point clouds through deep neural networks is challenging due to the lack of a grid-like structure. Early works in the field of *Shape Classification* proposed to represent 3D objects as a collection of 2D views [35], [36] or quantize point coordinates into 3D voxel grids [17], [37]. Alternatively, more scalable indexing structures, such as k-d trees [38] and octrees [39], were deployed to efficiently manage the sparsity of non-empty voxels. Differently, Qi *et al*., with PointNet [40] and PointNet++ [41], introduced a cutting-edge deep learning framework based on a *Multi-Layer Perceptron* (MLP) and a symmetry function which allows for directly consuming point clouds. To achieve invariance to rigid transformations, PointNet employs a transformation network [42] to predict an affine motion that canonicalizes the 3D point coordinates. Howeever, PointNet fails to generalize to unseen rotations, as demonstrated in [18], and the learned local descriptors built upon a PointNet backbone, *i.e*. PPFNet [12], are not rotation invariant. Recently, a few interesting works have attempted to define point convolution operators [20], [43], [44], [45], [46], [47], [48]. However, these methods mainly operate on 3D coordinates extracted from synthetic and watertight meshes. As such, there is no evidence in literature on whether they may effectively withstand the typical nuisances, e.g. noise, occlusions, missing regions and point density variations, that affect point clouds sensed in real-world scenarios.

3 Spherical CNNs

As described in [23], the basic intuition behind Spherical CNNs deals with lifting the notion of the classical planar correlation deployed in standard CNNs: the value of a feature map at $x \in \mathbb{Z}^2$ is computed as an inner product between an input signal and a learned filter *shifted* by x. In Spherical CNNs, feature maps live in $\mathrm{SO}(3)$ and the value of a feature map at $R \in \mathrm{SO}(3)$ is computed as the inner product between an input signal and a learned filter *rotated* by R. As a result, the major difference between standard CNNs and Spherical CNNs concerns the nature of the receptive field associated with a feature map. While in the former both the input image and the feature maps live in $\mathbb{Z}^2$, in the latter a feature map is a signal defined on $\mathrm{SO}(3)$. Hence the value of a $\mathrm{SO}(3)$ feature map at a certain location represents the filter response for a specific rotation, not for a region in the input signal (a surface area within a point cloud in our settings). In the following, we report the key definitions and properties related to Spherical CNNs.

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

Spherical Signals. A spherical signal is a continuous K-valued function defined on S^2, $f : S^2 \rightarrow \mathbb{R}^K$, K being the number of channels.

Rotations. A rotation in three dimensions lives in a three-dimensional manifold called *Special Orthogonal Group*, usually denoted as $\mathrm{SO}(3)$. According to [23], a suitable parameterization of $\mathrm{SO}(3)$ is given by the ZYZ-Euler angles, $\alpha \in [0, 2\pi], \beta \in [0, \pi]$ and $\gamma \in [0, 2\pi]$, Rotations can be expressed by 3×3 matrices that preserve distance (i.e. $\|Rx\| = \|x\|$) and orientation ($det(R) = +1$). If we represent the points in S^2 as 3D unit vectors x, the matrix-vector product Rx rotates x by R.

Rotations of Spherical Signals. The definition of the spherical correlation operation requires to define the rotation of a filter, which is itself a spherical signal. Thus, [23] introduces the L_R operator, that takes a spherical signal 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. Given a K-valued spherical signal f and a filter ψ, *i.e*. $f, \psi : S^2 \rightarrow \mathbb{R}^K$, and denoting the inner product on the vector space of spherical signals as $\langle \psi, f \rangle$ [23], the spherical correlation between them can be defined 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 operation will be referred to hereinafter as S^2 correlation. As Equation 2 can be computed for any $R \in \mathrm{SO}(3)$, the output of a spherical correlation is a signal living in $\mathrm{SO}(3)$. To further process such a signal in a Spherical CNN one has to define a correlation operation for $\mathrm{SO}(3)$ signals, which, in turn, requires introduction of a rotation operator.

Rotation of $\mathrm{SO}(3)$ Signals. Similarly to Equation 1, given a K-valued $\mathrm{SO}(3)$ signal $h : \mathrm{SO}(3) \rightarrow \mathbb{R}^K$, and $R, Q \in \mathrm{SO}(3)$, the L_R operator rotates h by R:

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

with $R^{-1}Q$ denoting the composition of rotations.

Rotation Group Correlation. Akin to Equation 2, the correlation between a K-valued $\mathrm{SO}(3)$ signal, h, and filter, ψ, *i.e*. $h, \psi : \mathrm{SO}(3) \rightarrow \mathbb{R}^K$, can be defined as the inner product between the signal and the rotated filter:

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

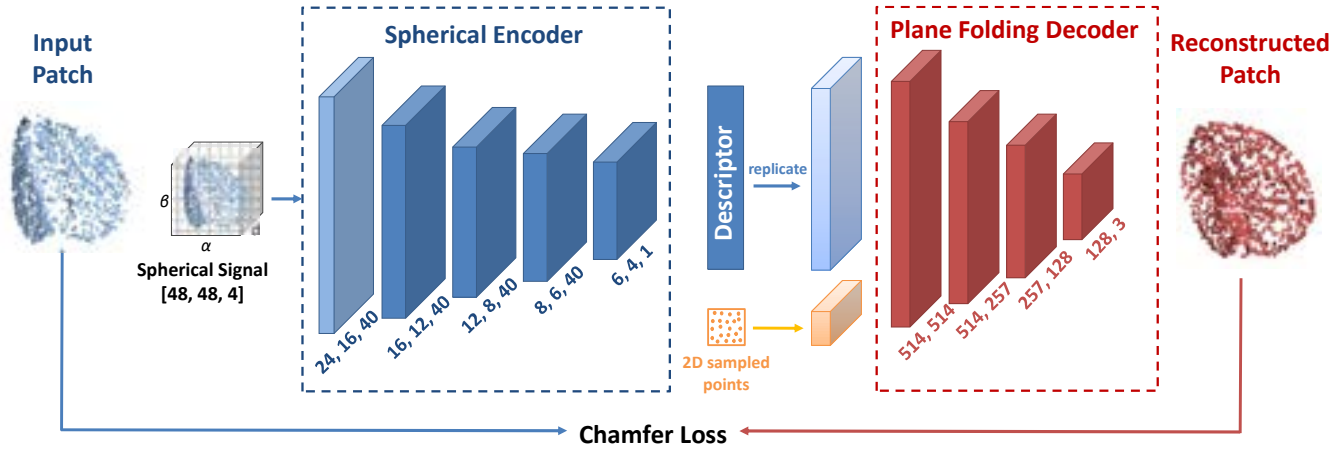

Fig. 1: **Network Architecture and Training Workflow**. The local patch around the given feature point is first converted into a discrete Spherical Signal. The triplet associated with the Spherical signal reports the number of bins along the azimuth (α), inclination (β) and radial (d) coordinates. The Spherical signal is processed through a Spherical Encoder to obtain a rotation-equivariant patch descriptor. The triplets associated with the encoder layers denote the input bandwidth, output bandwidth and number of channels, respectively. The Plane Folding Decoder is an MLP that reconstructs the input patch by deforming a 2D grid according to the computed descriptor. The pairs associated with the decoder layers denote the number of input and output units, respectively.

The integration measure dQ is the invariant measure on $\mathrm{SO}(3)$, which may be expressed in ZYZ-Euler angles as $d\alpha \sin(\beta) d\beta d\gamma / (8\pi^2)$. This operation will be referred to hereinafter as $\mathrm{SO}(3)$ correlation.

Equivariance. Both S^2 (Equation 2) and $\mathrm{SO}(3)$ correlation (Equation 4) are equivariant w.r.t. a 3D rotation, *i.e.* if the input (either a Spherical or $\mathrm{SO}(3)$ signal, respectively) is rotated by $Q \in \mathrm{SO}(3)$, the output undergoes the same rotation. Formally, given $g(\cdot)$, $\psi(\cdot)$, $Q, R \in \mathrm{SO}(3)$:

$$[\psi \star [L_Q g]](R) = [L_Q [\psi \star g]](R). \tag{5}$$

Signal Flow. In Spherical CNNs, the input signal, *e.g.* an image or, as it is the case of our settings, a point cloud, is first transformed into a k-valued spherical signal. Then, the first network layer (S^2 layer) computes feature maps by spherical correlations (Equation 2). As the computed feature maps are SO(3) signals, the successive layers ($\mathrm{SO}(3)$ layers) compute deeper feature maps by $\mathrm{SO}(3)$ correlations (Equation 4). Due to equivariance (Equation 5), if the input spherical signal rotates by $Q \in \mathrm{SO}(3)$ so do the outputs of all the network layers, *i.e.* the feature maps computed by the first S^2 layer as well as by the successive $\mathrm{SO}(3)$ layers.

Computation of S^2 and $\mathrm{SO}(3)$ correlations. Correlations and convolutions for Euclidean signals can be efficiently computed by the Fast Fourier Transform. For signals living on S^2 and $\mathrm{SO}(3)$ one can leverage on the Generalized Fourier Transform (GFT), which is defined according to the representation theory of groups [49], [50]. The GFT can be seen as a linear projection of a function onto a set of orthogonal basis functions. For S^2 and $\mathrm{SO}(3)$ signals the basis functions are the Spherical Harmonics and the Wigner D-functions, respectively [23]. In Spherical CNNs the $\mathrm{SO}(3)$ correlation (Equation 4) is implemented by the $\mathrm{SO}(3)$ Fourier Transform, known as *SOFT*, which, in turn, is computed via the discrete formulation referred to as DSOFT [51]. This entails sampling functions through a finite grid defined on a chosen parametrization of $\mathrm{SO}(3)$. Thus, as proposed in [23], the ZYZ-Euler angles, α, β and γ, are discretized into a grid of $2B \times 2B \times 2B$ bins, with B referred to as *bandwidth*. Along the same line of reasoning, the S^2 correlation (Equation 2) is computed via a discrete GFT, with the sampling grid required to discretize functions defined on the spherical coordinates, azimuth (α) and inclination (β) that parametrize the unit sphere S^2. Further information on the computation of discrete GFTs can be found in [52], [53].

4 Proposed Learning Framework

As anticipated in section 1, we propose to learn unsupervisedly a local surface descriptor via an encoder-decoder architecture which, peculiarly, deploys a Spherical Encoder, *i.e.* an encoder realized through a Spherical CNN, in order to learn a rotation-equivariant bottleneck. We start with providing an overview of our network architecture and training workflow with the aid of Figure 1.

As required by Spherical CNNs, the input 3D patch around the given feature point is converted into a discrete spherical signal which is then fed into a spherical encoder consisting of multiple layers. The first layer (brighter blue in Figure 1) computes S^2 correlations while the following ones are $\mathrm{SO}(3)$ layers (darker blue in Figure 1). As upon a 3D rotation of the input patch the spherical signal undergoes the same 3D rotation, due to the equivariance property of the feature maps calculated in a Spherical CNN it follows that the low-dimensional representation computed by the encoder is equivariant w.r.t. a 3D rotation of the input patch. At training time, an ensemble of random points forming a 2D grid is concatenated to the equivariant representation computed by the encoder so to form the input to the decoding section of the network (depicted in red in Figure 1)). This is realized as a plane folding decoder [24], that is a Multi-Layer Perceptron (MLP) that warps the 2D grid according to the information encoded into the bottleneck in order to reconstruct the input 3D patch. The Chamfer distance between the input and reconstructed patches provides the loss function to train the whole network.

As customary in unsupervised learning through encoder-decoder networks, the decoder is dismissed at inference time, the low-dimensional representation computed by the encoder providing our rotation-equivariant patch descriptor. To pursue pose-invariant descriptor matching, this equivariant representation is canonicalized by applying a 3D rotation computed by an external LRF algorithm.

In the following subsections, we provide more details on the network architecture as well as the computation performed at both training and inference time.

4.1 From Point Clouds to Spherical Signals

As discussed in section 3, the spherical correlation operator is defined for signals living on the unit sphere and, thus, to process a 3D patch by a Spherical CNN, the geometry around a feature point has to be converted into a spherical signal. The approach adopted in [18], [23] consists in projecting a 3D mesh onto an enclosing discretized sphere using a raycasting scheme. However, in our settings the input data are provided as a cloud of points in a patch around the *keypoint* we wish to describe, not as a watertight mesh. Similarly to [54], thus, we first express the 3D Euclidean coordinates of the points in the given patch according to a spherical coordinate system and then construct a quantized grid in this new coordinate system. Accordingly, a cell in the grid is identified by three quantized spherical coordinates $(\alpha[i], \beta[j], d[k]) \in S^2 \times D$, where $\alpha[i]$, $\beta[j]$ and $d[k]$ represent the quantized azimuth, inclination and radial distance, respectively. The K-valued spherical signal, $f : S^2 \rightarrow \mathbb{R}^K$, is then built by encoding for each pair of azimuth and inclination bins $(\alpha[i], \beta[j])$ the density of points for all bins $(\alpha[i], \beta[j], d[k]), k = 1, \dots, K$ laying along a radius of the quantized sphere. As proposed in [54], the density is estimated robustly with respect to quantization effects and, to consider the non-uniform spacing in the spherical space, cells near the south or north pole are wider in spherical coordinates. This process is applied to the neighbourhood of every keypoint $\mathbf{p}$ of the surface we choose to describe: before being quantized in the spherical signal, the Euclidean coordinates of the points in the neighbourhood $\mathbf{p}_i$ are expressed in a reference frame centered on the keypoint $\mathbf{p}$ to achieve translation invariance, *i.e.* the coordinates to be quantized are the translation-invariant coordinates $\hat{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}$. Finally, we wish to point out that the above described procedure is equivariant w.r.t. a 3D rotation of the input up to quantization noise. Indeed, if we extend the definition of the rotation operator L_R to point clouds, it follows immediately that if the input point cloud rotates by R, then the spherical signal rotates accordingly. Formally,

$$f(L_R x) = f(R^{-1} x) = [L_R f](x) \tag{6}$$

where

$$L_R x = R^{-1} x \tag{7}$$

is the rotation operator for point clouds.

4.2 Network Architecture

The procedure described in subsection 4.1 converts a set of 3D points around a given keypoint into a K-valued signal defined on S^2. Thus, we need an S^2 correlation layer as the first layer of our Spherical Encoder. Unlike the standard definition of Spherical convolution [55], which outputs a signal on the sphere S^2, we rely on Equation 2, which outputs a signal defined on $\mathrm{SO}(3)$. In fact, the use of a conventional convolution definition would have limited the network expressive capacity due to the symmetry along the Z-axis of the learned filters [23]. To process the resulting $\mathrm{SO}(3)$ feature map, we stack an ensemble of $SO(3)$ correlation layers, where the last one outputs the final equivariant feature descriptor. The encoder should learn to compress the salient shape information within the local neighborhood of the given keypoint, $\mathbf{p}$, into a compact representation so as to produce a robust and distinctive descriptor. According to a popular unsupervised learning procedure, we accomplish this by employing a decoder capable of reconstructing the input neighbourhood starting from the compressed representation computed by the encoder. The key tool to realize learning via input-output reconstruction with point clouds is given by the *plane folding* operator proposed in [24] and [25]. Following [24], our decoder is an MLP that tries to deform points in $\mathbb{R}^2$ into surface points in $\mathbb{R}^3$ according to the computed descriptor. Let $\mathcal{A}$ be a set of points sampled in the unit square $[0, 1]^2$. Given a feature representation $\mathbf{d}$ for a 3D surface, we concatenate the descriptor $\mathbf{d}$ with the sampled point coordinates $(a_x, a_y) \in \mathcal{A}$ and then forward the concatenated vectors through the fully-connected layers of the MLP, as shown in Figure 1.

4.3 Loss

Our loss minimizes the dissimilarity between the input and reconstructed patches, both given as sets of 3D points. Purposely, we rely on the Chamfer distance, that can assess the dissimilarity between two sets of points. Let $\mathcal{S}$ be the set of 3D points belonging to the input neighborhood and $\mathcal{S}^\star$ the set of points generated by the decoder, the Chamfer distance can be formulated as follows:

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

This symmetric formulation drives both the distance from $\mathcal{S}$ to $\mathcal{S}^\star$ as well as that from $\mathcal{S}$ to $\mathcal{S}^\star$ to be small. The term $\min_{\mathbf{x}^\star \in S^\star} \|\mathbf{x} - \mathbf{x}^\star\|_2$ enforces that each 3D point $\mathbf{x}$ in the input patch has a nearby matching 3D point $\mathbf{x}^\star$ in the reconstructed one, whereas the term $\min_{\mathbf{x} \in S} \|\mathbf{x}^\star - \mathbf{x}\|_2$ constrains each point in the reconstructed patch $\mathbf{x}^\star$ to lay close to a point $\mathbf{x}$ in the input one. In other words, neither any portion of the input patch should be missing in the reconstructed one nor the decoder should hallucinate regions not present in the input.

4.4 Hyperparameters

As shown in Figure 1, the input patch is converted into a Spherical signal discretized with bandwidth $B = 24$ (*i.e.* $48 = 2 \cdot 24$ bins) along the azimuth and inclination coordinates, while we use $K = 4$ bins along the radial dimension. Hence, the encoder is fed with a 4-valued Spherical signal. Our spherical CNN consists of an initial S^2 layer and four $\mathrm{SO}(3)$ layers. As discussed in section 3, in Spherical CNNs correlation operations are computed by the Generalized Fourier Transform, which requires signals to be discretized according to a bandwidth parameter, B. For the first S^2 layer the input and output bandwidths are $B = 24$ (as discussed above) and $B = 16$, respectively. Then, in each of the four $\mathrm{SO}(3)$ layers, the input bandwidth is given by the output bandwidth of the previous layer. As shown in Figure 1, we reduce the signal bandwidth along the cascade of the four $\mathrm{SO}(3)$ layers,

the output bandwidths being $B = 12$, $B = 8$, $B = 6$ and, in the the final layer, $B = 4$. All the correlation layers but the last one have 40 feature maps (*i.e.* channels), while the last layer features a single channel, *i.e.* our equivariant descriptor living in $\mathrm{SO}(3)$. As the output of the last layer is a $\mathrm{SO}(3)$ feature map with bandwidth $B = 4$, our descriptor consists of $(2 \cdot 4)^3 = 512$ entries. In each layer, we apply *BatchNorm* step and *ReLU* non-linearities. In subsection 6.3 we present the model selection study that lead us to define the Spherical CNN design detailed in this sub-section. As for the plane folding decoder, it consists of four fully-connected layers, with number of input and output units reported in Figure 1. We use *BatchNorm* and *ReLU* in the first three layers, whilst we adopt $\tanh$ non-linearities in the output layer. We trained the whole network with mini-batches of size 100 using ADAM [56] and a fixed learning rate of 0.001.

4.5 Test-Time Invariant Feature Descriptor

At test-time, descriptor matching has to be pursued invariantly w.r.t. 3D rotations of the input clouds. As already pointed out, unlike state-of-the-art methods that rotate input patches to bring them into a canonical orientation, we apply the canonicalizing rotations to descriptors, *i.e.* $\mathrm{SO}(3)$ feature maps computed from unoriented input patches. Signals in $\mathrm{SO}(3)$ can be rotated by remodulating the Wigner D-functions employed in their Generalized Fourier Transform. A thorough treatment of the topic and the mathematical details of this procedure can be found in [57]. Please, note that the dimensionality of the final descriptor stays unchanged upon such a rotation. In a preliminary version of this work [27] we investigated on two approaches to determine the canonicalizing rotation. One tries to exploit the special properties of Spherical CNNs to determine such a rotation by finding a feature point, *i.e.* a repeatably detectable bin, within an $\mathrm{SO}(3)$ feature map computed by the Spherical Encoder. The other relies on deploying an off-the-shelf LRF estimation algorithm. In this extended manuscript we focus on the latter approach as it provides neatly superior results [27]. In particular, following the same setup as in [27], we experiment with the LRF algorithm proposed in [58], that we dub FLARE according to its publicly available PCL [59] implementation.

5 Equivariance Is What You Need

The main contribution of our work consists in proposing the first rotation-equivariant learned descriptor for 3D keypoints. We argue that this design choice yields several advantages, as validated by the experiments reported in section 6. Thanks to the equivariance property, we do not need to feed the network with rotation-invariant representations at training time, as done in previous work [9], [10], [13], and we can delay the canonicalization step to test time, thereby achieving important benefits. First, in contrast to existing proposals, not having to choose a specific LRF at training time allows us to train the network from raw rather than pre-processed input data. This better adheres to the likes of end-to-end representation learning, a key success factor of the deep learning paradigm. Secondly, by avoiding to canonicalize the training data with a chosen algorithm we avoid injecting into the training process the unavoidable mistakes the chosen algorithm shall make. Indeed, it is well-known that LRF algorithms are far from perfect and may behave differently on data acquired by different 3D sensors as well as when dealing with different nuisances [19]. As a matter of fact, the domain shift issue is particularly critical when learning from 3D data because of the heterogeneous acquisition techniques and sensing principles. Therefore, not being tied to a specific LRF at training time enables us to choose the most appropriate approach to obtain a canonical representation at test time without having to retrain the network. This flexibility endows our proposal with excellent generalization capabilities, as demonstrated in the transfer learning experiments reported in subsection 6.6. The reader may notice that within our framework whom may perform test-time canolicalization either in the descriptor space (our choice) or in the input space, obtaining equivalent results up to quantization effects. In fact, as our descriptors are $\mathrm{SO}(3)$ signals, they can be brought into a canonical orientation by a 3D rotation operation (Equation 3).

In the following subsections, we provide more details regarding our choices on the design of the LEAD architecture. In subsection 5.1 we demonstrate how an equivariant encoder can be more effective than an invariant one when trained with unoriented data. In subsection 5.2, we explore the reconstruction attainable by plane folding decoders in the same scenario.

5.1 Equivariant encoder

In this subsection we wish to point out how equivariance is not only distinct but, indeed, key to the effectiveness of our method. As motivated so far, we aim at learning a 3D descriptor unsupervisedly, *i.e.*, by an encoder-decoder architecture, from raw and unoriented training data. Thus, an interesting study concerns exploring whether a standard encoder architecture, such as, for instance, the popular PointNet architecture that consumes raw 3D coordinates, could be deployed in our framework in place of the Spherical encoder. Due to the requirement of relying on unoriented training data, with such an alternative design one may pursue learning a rotation-invariant descriptor only by augmenting the input data at training time by random 3D rotations. Thus, we replace the Spherical encoder in the architecture illustrated in Figure 1 with a standard PointNet and aim at learning a rotation invariant embedding without applying a canonical orientation to the input data, that is having the network learn such invariance by observing randomly rotated versions of the same neighborhood at training time. To verify whether the learned descriptor has achieved invariance to rotation we measure the distance between descriptors computed at the same keypoint under different 3D rotations. Figure 2 compares a PointNet encoder, trained on the 3DMatch dataset (presented in subsection 6.1) and a Spherical encoder with randomly initialized weights. In fact, due to equivariance being a built-in property of Spherical CNNs, it is not necessary to train the Spherical encoder to perform this study. We rotate a neighborhood around a random axis by an increasing angle, whose value is reported along the horizontal axis in Figure 2. For every rotation, we forward the rotated neighborhood through a PointNet and Spherical encoder: the former would compute an invariant descriptor, the latter an equivariant bottleneck to be later oriented by a rotation. Hence, afterwards computation, we rotate the output of the Spherical encoder by the inverse of the applied rotation (simulating availability of a perfect LRF algorithm). For both encoders we plot the distance between the descriptor obtained from the rotated and the unrotated neighborhood. Figure 2 shows that PointNet cannot learn an invariant descriptor in our unsupervised learning framework, while the equivariant representation provided by a Spherical CNN can achieve almost perfect invariance when properly rotated.

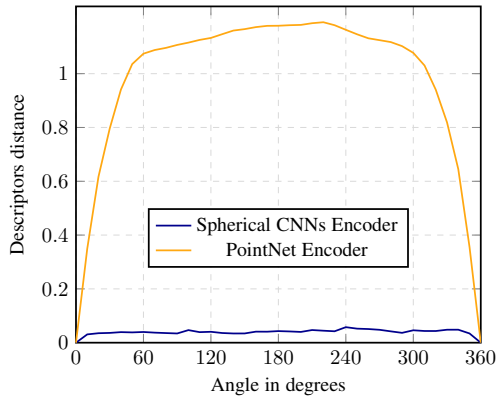

Fig. 2: Comparison between a PointNet and Spherical encoder when learning without supervision from unoriented data.

5.2 Plane folding decoder

Why, thus, the PointNet encoder fails in yielding an invariant representation in our framework? The reason is that for plane folding decoders, like AtlasNet or FoldingNet, to be able to reconstruct arbitrarily rotated input patches, 3D rotation information must be encoded into the bottleneck. Indeed, should the encoder produce an invariant bottleneck, the decoder would be given no information to determine under which 3D rotation it should reconstruct the input so to minimize the loss. Thus, to fulfill the learning objective, the PointNet encoder entangles rotation information into the bottleneck, which, therefore, as vouched by Figure 2, is not rotation-invariant. Yet, unlike our equivariant descriptor living in $\mathrm{SO}(3)$, one cannot apply a canonicalizing 3D rotation to the PointNet bottleneck, namely the output of an MLP, so to turn it into an invariant descriptor.

We found it worth investigating also on how would learning proceed in our framework should we *constrain* the encoder bottleneck to be rotation-invariant. Spherical CNNs have been successfully exploited [18], [23], [54] to learn an invariant global shape embedding for 3D Object Classification under random rotations. To this end, a *max-pooling* layer is inserted between the chain of S^2-$\mathrm{SO}(3)$ correlation layers and the last *fully-connected* layer performing classification, so as to select the strongest response in each feature map regardless of the rotation under which the object may appear. Indeed, recalling the analogy between standard and Spherical CNNs, as in the former global max pooling provides translation invariance, in the latter it does so for 3D rotations. Hence, we slightly modify our Spherical encoder to obtain an invariant feature descriptor by removing the last $\mathrm{SO}(3)$ correlation layer, which yields the equivariant descriptor, and adding a max pooling layer followed by a fully connected layer to expand the codeword dimensionality to 512. In Figure 3, we compare the reconstructions yielded by our framework when the decoder is fed with either the equivariant or invariant bottleneck computed by the Spherical encoder. Clearly, the AtlasNet [24] decoder cannot converge to sensible reconstructions when no information about the input rotation is incorporated into the latent space, *i.e.* when the bottleneck is rotation invariant. Indeed, in this experimental setting, the decoder is asked to output different reconstructions, *i.e.* the rotated versions of a given input neighbourhood, based on the same invariant representation. The last column of Figure 3 highlights how, in order to try to fulfill this *one-to-many* learning objective, the decoder can only average out across all inputs, *i.e.* it can output reconstructions that try to match in as much as possible all the possible rotations of the input, resulting in cloud of points that do not resemble the input patches. This is clearly an underfitting regime for our unsupervised learning framework, which prevents learning distinctive representations of the input patches. Thus, for all the reasons discussed above, to learn via input-output reconstruction from unoriented point clouds one needs an equivariant representation.

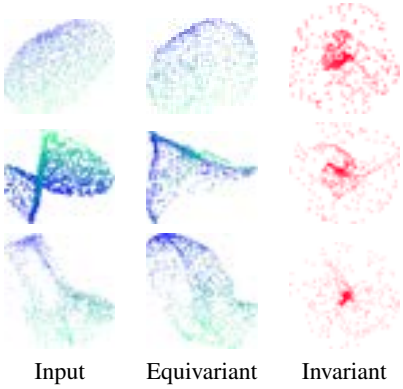

Fig. 3: Comparison between the reconstructions obtained when using the Spherical CNN encoder to learn an equivariant versus an invariant bottleneck. Results after 10K training iterations.

6 Experimental Study

According to the established experiment design in the field of local 3D descriptors, in this section we consider pairwise surface registration scenarios dealing with indoor and outdoor datasets for which accurate ground-truth transformations between views are available. We present an initial set of experiments aimed at validating our design choices (Sec. 6.3). Then, we carry out a comparative performance evaluation by considering LEAD as well as the most prominent proposals concerning handcrafted and learned descriptors.

6.1 Datasets

As for indoor data, we use the 3DMatch benchmark [8], which is the *de-facto* standard for evaluation of learned 3D descriptors, alongside its *Rotated* version proposed to assess upon invariance to 3D rotation of learned methods in [13]. As for outdoor data, we rely on the ETH dataset [26], which has been recently deployed in [9] to evaluate how well learned descriptors trained on 3DMatch can generalize to different sensing modalities and environments. Both datasets are organized into a collection of *scenes*, where a scene consists in a set of 2.5D scans, each depicting a small fraction of the whole environment and therefore referred to as *fragment*.

3DMatch includes 62 scenes taken from the publicly available Analysis-by-Synthesis [60], 7-Scenes [61], SUN3D [62], RGB-D Scenes v.2 [63] and Halberand Funkhouser [64] datasets. According to the standard protocol adopted to evaluate learned descriptors, we train and validate on 54 scenes, leaving the remaining 8 scenes for testing. The point clouds available for training and testing are obtained by fusing 50 consecutive depth frames sensed with an RGB-D sensor [13]. The *Rotated* 3DMatch benchmark is generated by randomly rotating the fragments of the test split in order to sample the space of 3D rotations [13].

The ETH dataset [26] is a challenging outdoor dataset featuring 8 sequences of sparse and dense vegetation (*e.g.*, trees and bushes) acquired by a *laser scanner* under seasonal changes. Similarly to [9], we consider only 4 scenes, namely: *Gazebo-Summer*, *Gazebo-Winter*, *Wood-Autumn* and *Wood-Summer*. We realize the setup proposed in [13]: each fragment is downsampled by a voxel grid filter with a leaf of 2 cm and surface normals are estimated using a 17-point neighborhood [65].

Regarding the radius of the supporting patches to compute descriptors, akin to [13] and [9] we use 0.3 m with 3DMatch and 1.0 m with ETH, so as to handle the different scales of the represented geometries.

6.2 Methodology

We follow the standard evaluation protocol adopted in previous works [9], [13], [15], [32]. For each scene, we consider all the fragment pairs which exhibit at least 30% overlap and describe the 5000 uniformly sampled keypoints per fragment made available by the authors of [8] (3DMatch) and [9] (ETH). Correspondences for each pair of fragments are established by finding reciprocal nearest neighbours in the descriptor space [8]. Once correspondences are found, we compute metrics, such as *Recall*, *Relative Rotation Error* and *Relative Translation Error*, aimed at evaluating the effectiveness of a descriptor when deployed as a component within a robust pairwise registration pipeline.

The *Recall* is defined as the percentage of fragment pairs for which the registration transformation is likely to be estimated correctly by a robust pipeline [13]. According to this metric, a pair of fragments can be registered correctly if the percentage of correctly matched keypoints is higher than an *inlier ratio* threshold, τ_2, customarily set to 5%. A match between two keypoints is considered correct if, upon application of the ground-truth transformation between the fragments, their distance is less than a threshold τ_1, *e.g.* 10 cm.

For those pairs of fragments which can be correctly registered according to the above definition of *Recall*, the *Relative Rotation Error* (RRE) and *Relative Translation Error* (RTE) measure the accuracy of the 6 DOF registration transformation estimated by a standard RANSAC-based pipeline given the set of matches provided by the descriptor. Hence, given the estimated $(\hat{R}, \hat{T})$ and ground-truth (R^*, T^*) rotation and translation, the RRE and RTE are calculated as follows:

$$RRE = \arccos\left(\frac{(tr(\hat{R}^T R^*) - 1)}{2}\right) \tag{9}$$

$$RTE = \|\hat{T} - T^*\|_2 \tag{10}$$

Moreover, by considering only the keypoints detected in the overlap area, we compute the average percentage of correct correspondences per fragment pair, which is a task-agnostic performance figure directly related to the effectiveness of the descriptor matching process.

As for the comparison w.r.t. other methods, we consider the popular handcrafted descriptors: FPFH [2], Spin Images [1], SHOT [5] and USC [3]. Besides, we compare LEAD against the current state-of-the-art in learned 3D feature descriptors. In particular, we consider 3DMatch [8], CGF [10], PPFNet [12], 3DSmoothNet [9], FCGF [14], D3Feat [15] and Li *et al.* [32] as supervised methods, while PPFFoldNet [13], 3DPointCaps [33] and the preliminary formulation of our method [27], referred to here as Ours ICCV, as unsupervised methods. For handcrafted descriptors, we used the implementations available in the PCL library [66]. As regards learned descriptors, we report the results provided by the authors in their respective papers for the 3DMatch (Table 3) and 3DMatch rotated (Table 4) benchmarks, while on ETH (Table 7) we use the publicly available code provided by the authors where the results are not reported in their papers. As for D3Feat, which may jointly learn a 3D keypoint detector and descriptor, to establish a fair comparison w.r.t. the other methods we only evaluate descriptors on the sets of randomly sampled keypoints defined in the 3DMatch and ETH benchmarks.

6.3 Model Selection

As described in [23], an important choice in the design of a Spherical CNN deals with the type of filters used to perform correlations. In particular, one may rely on two types of spherical grids, referred to as near identity and equatorial grids. The former defines spatially localized filters initialized at the north pole and rotated over the sphere via the action of $S^2/SO(3)$ correlations, the latter ring-shaped kernels localized around the equator. Besides, other key hyper-parameters of Spherical CNNs concern the input bandwidth and number of channels (*i.e.*, filters) in each $SO(3)$ layer as well as the number of layers. In our preliminary work [27] we defined and tested a Spherical CNN architecture (denoted as A in Table 1) based on equatorial filters and 3 $SO(3)$ layers, with (input bandwidth, number of channels) equal to (24,40), (24,40) and (4,1), respectively. We present here an experimental study aimed at selecting the optimal configuration of the key hyper-parameters of our Spherical CNN so as to improve both accuracy and speed compared to our previous design.

As detailed in Table 1, we consider 14 architectures, including our previous proposal (A), by varying the type of spherical grid, number of $\mathrm{SO}(3)$ layers, input bandwidth and number of channels per layer. Each of these Spherical CNN architectures is used as encoder to train our encoder-decoder network (Figure 1) on the 3DMatch dataset. Then, all the 14 learned descriptors are evaluated on the test split of 3DMatch by considering a reduced number of keypoints, *i.e.* 500 instead of 5000. To select the optimal architecture, we compute both the average *Recall* across the test scenes as well the computation time required for a forward pass of the encoder on a mini-batch of 25 samples. As we are interested in a comparative analysis relative to our previous design [27], we consider as time unit the computation time yielded by the architecture denoted as A in Table 1.

The last two columns in Table 1 report the two performance metrics. We can notice a clear trend: the key parameter to obtain significant speed-ups is the bandwidth used to discretize the rotation manifold, as shown by the clear gap between architectures A to E, which uses the same bandwidth used in [27], and the others, which decrease it while moving deeper into the network. With the new bandwidth scheme, near-identity filters achieve higher recall, as vouched for instance by architecture I versus H or N versus M. Yet, as shown in the Table, none of the considered architectures can optimize both metrics. Hence, in Figure 4 we report the Pareto analysis concerning our model selection experiment. Accordingly, focusing on the data laying on the Pareto frontier, we select the architecture refereed to as N as our preferred trade-off between accuracy and speed. Indeed, only architectures N and D can yield a higher *Recall* than our previous design A, but while D is slower than A, N turns out much faster. Therefore, in the experiments

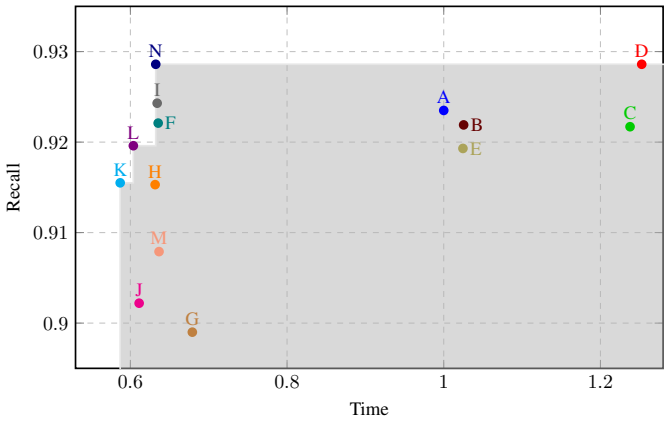

Fig. 4: Scatter plot of the model selection experiment. The light gray area highlights the Pareto frontier.

discussed hereinafter we will use the Spherical Encoder denoted as N in Table 1. Indeed, as anticipated in subsection 4.4, this is the Spherical Encoder architecture depicted in Figure 1.

TABLE 1: Model selection experiment on the 3DMatch benchmark (500 keypoints): considered architectures and associated performance metrics. All architectures have the same number of channels (reported in the fifth column) in all but the last layer, which consists in a single channel. Best values of Recall and normalized time are shown in bold.

Network	Grid Eq.	Grid NI	$SO(3)$ layers	Input BW	Channels	Recall	Time
A [27]	✓		3	[24, 24, 4]	40	0.924	1.000
B		✓	3	[24, 24, 4]	40	0.922	1.025
C	✓		3	[24, 24, 24]	40	0.922	1.238
D		✓	3	[24, 24, 24]	40	**0.929**	1.253
E		✓	2	[24, 24]	40	0.919	1.025
F		✓	3	[12, 8, 6]	40	0.922	0.636
G		✓	3	[16, 12, 8]	60	0.899	0.679
H	✓		3	[16, 12, 8]	40	0.915	0.632
I		✓	3	[16, 12, 8]	40	0.924	0.634
J		✓	3	[16, 12, 8]	30	0.902	0.611
K		✓	3	[16, 12, 8]	20	0.916	**0.587**
L		✓	2	[16, 8]	40	0.920	0.604
M	✓		4	[16, 12, 8, 6]	40	0.908	0.637
N (LEAD)		✓	4	[16, 12, 8, 6]	40	**0.929**	0.632

6.4 Equivariance vs. Invariance

As pointed out in section 1 and discussed in section 5, a distinctive choice underpinning our proposal concerns learning from unoriented as opposed to canonicalized 3D patches. To validate this choice, in our architecture (Figure 1) we now consider as encoder the same three layers PointNet as in section 5 and train the whole network on 3DMatch by feeding it with patches rotated according to the canonical orientation computed by an LRF algorithm. Thereby, the PointNet encoder can learn a rotation-invariant descriptor, though, according to our intuition, the learning process may turn out sub-optimal due to the non-ideality of the chosen canonicalization algorithm. To compare fairly the invariant descriptor and LEAD, at training time we fed the PointNet encoder with patches canonicalized by the same algorithm (FLARE [58]) we rely upon to rotate our equivariant descriptor at test time (subsection 4.5).

Table 2 reports the average *Recall* yielded by LEAD and the invariant descriptor across the test scenes of the 3DMatch and 3DMatch Rotated datasets. These results show that LEAD can provide substantially better performance, the margin being almost 10% on both datasets, and validate our intuition on the benefits attainable by learning end-to-end an equivariant descriptor from unoriented data in order to avoid injecting noise into the learning process in the attempt of pursuing invariance at training time.

TABLE 2: Learning a rotation equivariant vs. rotation-invariant descriptor: average *Recall* on 3DMatch and 3DMatch Rotated.

	Equivariant (LEAD)	Invariant
3DMatch	**0.9584**	0.8680
3DMatch Rotated	**0.9601**	0.8741

6.5 Evaluation on the 3DMatch Benchmark dataset

Table 3 reports the *Recall* measurements dealing with the comparative evaluation on the 3DMatch benchmark. Firstly, these results confirm the findings of the model selection experiment (subsection 6.3) due to LEAD turning out more accurate than our previous proposal [27] also in the standard evaluation set-up defined for 3DMatch, *i.e.* testing with the given 5000 keypoints. Our proposal outperforms all previous unsupervised proposals, LEAD yielding remarkably higher accuracy than both PointCaps3D [33] and PPF-FoldNet [13]. As the authors of [33] provide their results, and those yielded by PPFFoldnet alike, by testing on 2000 keypoints, we run LEAD also in this setting (first three rows in Table 6): our descriptor outperforms both PointCaps3D and PPFFoldnet by a large margin, *i.e.* $\approx 16\%$ and 27%, respectively. The large gap ($\approx 24\%$) between LEAD and PPFFoldnet is confirmed by the measurements dealing with the standard 3DMatch setting (5000 keypoints) reported in other rows. Finally, with an average *Recall* of 95.84% LEAD outperforms also most of the recent *supervised* approaches, *i.e.* FCGF [14], D3Feat [15] and 3DSmoothNet [9], and it offers the runner-up performance of all tested methods on the standard benchmark for this field, surpassed only by the current state of the art *supervised* 3D descriptors proposed very recently by Li *et al.* [32]. It is also interesting to note how classical handcrafted descriptors, leaded by SHOT [5] and USC [3], can yield competitive results compared to descriptors learned unsupervisedly, like PointCaps3D and PPFFoldnet, as well as earlier proposals based on supervised learning such as 3DMatch, CGF and PPFNet.

In Figure 5, we plot the average *Recall* as a function of the *inlier ratio* threshold (τ_2) used to establish whether a fragment pair may be aligned [13]. Thus, as we move towards the right of the horizontal axis the registration task set forth by the experiment gets more challenging as any given descriptor has to deliver more correct correspondences for a pair of fragments to be considered as aligned. Figure 5 shows that LEAD compares favourably w.r.t. all previous methods across the whole range of *inlier ratio* thresholds, again with the exception of Li *et al.* [32]. It is also worth highlighting how the performance gain provided by LEAD w.r.t. 3DSmoothNet [9], FCGF [14] and D3Feat [15] gets larger as the registration task gets more challenging. In particular, at the highest *inlier ratio*, which requires matching correctly at least 20% of the extracted keypoints, LEAD can yield a *Recall* about 4% higher than D3Feat, 8% higher than 3DSmoothNet and 12 % higher than FCGF [14].

In Table 4 we report the results dealing with the rotated 3DMatch benchmark [13]. As expected, while earlier learned

TABLE 3: Results on the 3DMatch benchmark. Test data are from SUN3D [62], except for *Kitchen* data which is from 7-scenes [61]. Best result on each column highlighted in bold.

	#points	learned	unsupervised	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
PPFFoldNet	2K	✓	✓	0.7352	0.7564	0.6250	0.6593	0.6058	0.8889	0.5753	0.5974	0.6804
PointCaps3D	2K	✓	✓	0.8518	0.8333	0.7740	0.7699	0.7308	0.9444	0.7397	0.6494	0.7867
LEAD	2K	✓	✓	0.9822	0.9679	0.9087	0.9956	0.9519	0.9815	0.9281	0.8961	0.9515
FPFH	5K			0.7391	0.7885	0.6442	0.8142	0.7115	0.8889	0.7432	0.7013	0.7539
SHOT	5K			0.8893	0.8974	0.8221	0.9336	0.8750	0.8889	0.8630	0.8312	0.8751
SI	5K			0.6561	0.7564	0.6731	0.6770	0.6346	0.7407	0.4692	0.4545	0.6327
USC	5K			0.9308	0.9103	0.7788	0.9204	0.8462	0.8889	0.8664	0.8052	0.8684
CGF	5K	✓		0.4605	0.6154	0.5625	0.4469	0.3846	0.5926	0.4075	0.3506	0.4776
3DMatch	5K	✓		0.5810	0.7244	0.6154	0.5442	0.4808	0.6111	0.5171	0.5065	0.5726
PPFNet	5K	✓		0.8972	0.5577	0.5913	0.5796	0.5769	0.6111	0.5342	0.6364	0.6231
3DSmoothNet	5K	✓		0.9700	0.9550	0.8940	0.9650	0.9330	0.9820	0.9450	0.9350	0.9474
FCGF	5K	✓		0.9860	0.9620	0.9330	0.9780	0.9420	0.9820	0.9350	0.8960	0.9518
D3Feat	5K	✓		0.9802	0.9808	0.9038	0.9779	0.9231	0.9815	**0.9589**	**0.9221**	0.9535
Li *et al.*	5K	✓		**0.9940**	**0.9872**	**0.9470**	**0.9960**	**1.0000**	**1.0000**	0.9550	**0.9221**	**0.9750**
PPFFoldNet	5K	✓	✓	0.7866	0.7628	0.6154	0.6814	0.7115	0.9444	0.6199	0.6234	0.7182
Ours ICCV	5K	✓	✓	0.9802	0.9615	0.8942	0.9823	0.9519	0.9815	0.9144	0.8701	0.9420
LEAD	5K	✓	✓	0.9901	0.9808	0.9135	**0.9960**	0.9808	0.9815	0.9418	0.8831	0.9584

TABLE 4: Results on the rotated 3DMatch benchmark. Test data are from SUN3D [62], except for *Kitchen* data which is from 7-scenes [61]. Best result on each column highlighted in bold.

	#points	learned	unsupervised	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
PointCaps3D	2K	✓	✓	0.8498	0.8525	0.7692	0.8141	0.7596	0.9259	0.7602	0.7272	0.8073
PPFFoldNet	2K	✓	✓	0.7352	0.7692	0.6202	0.6637	0.6058	0.9259	0.5616	0.6104	0.6865
LEAD	2K	✓	✓	0.9862	0.9744	0.8942	0.9956	0.9615	0.9815	0.9315	0.8571	0.9478
FPFH	5K			0.7451	0.7949	0.6587	0.8142	0.7212	0.9259	0.7260	0.7530	0.7674
SHOT	5K			0.8794	0.8910	0.8317	0.9425	0.8654	0.9074	0.8493	0.8312	0.8747
SI	5K			0.6502	0.7628	0.6635	0.6903	0.6635	0.7222	0.4692	0.4935	0.6394
USC	5K			0.9170	0.9103	0.7548	0.9292	0.8558	0.9074	0.8836	0.8571	0.8769
CGF	5K	✓		0.4466	0.6667	0.5288	0.4425	0.4423	0.6296	0.4178	0.4156	0.4987
3DMatch	5K	✓		0.0040	0.0128	0.0337	0.0044	0.0000	0.0096	0.0000	0.0260	0.0113
PPFNet	5K	✓		0.0020	0.0000	0.0144	0.0044	0.0000	0.0000	0.0000	0.0000	0.0026
3DSmoothNet	5K	✓		0.9720	0.9620	0.9090	0.9650	0.9230	**0.9815**	0.9452	0.9351	0.9491
FCGF	5K	✓		0.9783	**0.9744**	0.9183	0.9735	**0.9712**	**0.9815**	0.9452	0.8831	0.9532
D3Feat	5K	✓		0.9684	**0.9744**	0.8846	0.9735	0.9327	**0.9815**	**0.9486**	**0.9610**	0.9531
Li *et al.*	5K	✓		**0.9921**	0.9679	**0.9375**	**0.9956**	**0.9904**	**0.9815**	**0.9486**	0.9351	**0.9686**
PPFFoldNet	5K	✓	✓	0.7885	0.7821	0.6442	0.6770	0.6923	0.9630	0.6267	0.6753	0.7311
Ours ICCV	5K	✓	✓	0.9763	0.9679	0.8894	0.9779	0.9615	**0.9815**	0.9110	0.8442	0.9387
LEAD	5K	✓	✓	**0.9921**	**0.9744**	0.8990	**0.9956**	0.9712	**0.9815**	0.9452	0.9221	0.9601

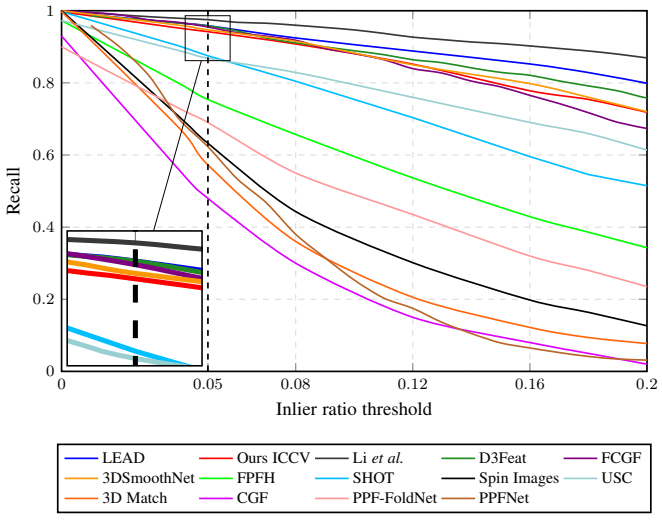

Fig. 5: Average *Recall* while varying the *inlier ratio* ratio threshold. The measurements dealing with learned descriptors are either available in the original papers or were kindly provided to us by the authors. The results for handcrafted descriptors have been computed by the implementations available in PCL [66].

descriptors (3DMatch, PPFnet) not geared to handle pose variations exhibit a dramatic performance drop, all rotation-invariant methods yield results quite similar to those shown in Table 3. Our equivariant descriptor oriented at test time by FLARE [58] can now provide performance very close to Li *et al.*, the difference in average *Recall* turning out less than 1%

In Table 5 we report the RRE (degrees) and RTE (meters) as defined in subsection 6.2. As errors are computed only on the fragment pairs actually registered by a method, we consider only those methods that succeed in registering at least 80% of all the fragment pairs according to the *Recall* figures shown in Table 3. This avoids over-rewarding methods that tend to register mostly the "easy" (*i.e.* featuring a large overlap) pairs. The results in Table 5 show that Li *et al.* is the most accurate method while still being able to register the highest number of pairs. Yet, LEAD, in spite of its unsupervised nature, outperforms in terms of both accuracy and number of registered pairs all other learned descriptors which rely on supervision. It is also worth noticing that SHOT and the previous formulation of our proposal can deliver better accuracy than LEAD, though based on smaller sets of registered pairs.

As pointed out in subsection 6.2, a more direct assessment of the effectiveness of the descriptor matching process may be achieved by measuring the percentage of correct correspondences

TABLE 5: Registration errors (RRE in degrees, RTE in meters) for methods yielding at least 80% *Recall* on 3DMatch. The average number of registered pairs is also reported in the last column. Best results are highlighted in bold.

	Kitchen		Home 1		Home 2		Hotel 1		Hotel 2		Hotel 3		Study		MIT Lab		Average		
	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	Pairs
SHOT	2.24	**0.06**	5.22	0.17	5.05	0.18	2.10	0.07	3.98	0.12	4.22	0.08	4.65	0.15	2.64	0.10	3.76	0.12	182
USC	3.82	0.09	2.01	**0.07**	5.87	0.17	4.74	0.13	8.59	0.25	**3.52**	**0.06**	6.50	0.22	5.08	0.10	5.02	0.14	178
3DSmoothNet	2.43	**0.06**	5.46	0.18	3.06	**0.12**	**1.67**	**0.06**	2.97	0.12	6.65	0.08	5.44	0.17	6.13	0.25	4.23	0.13	194
FCGF	2.23	0.07	2.60	0.11	6.94	0.15	2.24	0.07	3.84	**0.08**	7.21	0.14	6.71	0.25	4.91	0.21	4.59	0.14	194
D3Feat	2.73	0.10	5.64	0.18	10.68	0.31	2.61	0.09	3.35	0.12	16.79	0.26	7.75	0.25	9.85	0.30	7.43	0.20	195
Li *et al.*	**2.13**	**0.06**	**1.80**	**0.07**	**2.78**	0.14	1.77	0.07	4.31	0.12	4.98	0.08	**3.27**	**0.12**	3.20	0.12	**3.03**	**0.10**	**198**
Ours ICCV	2.85	0.08	2.48	0.09	3.77	0.13	2.03	0.07	**2.88**	0.09	6.94	0.10	3.62	0.13	**2.39**	**0.09**	3.37	**0.10**	190
LEAD	2.87	0.08	4.45	0.14	4.50	0.16	2.04	0.07	4.52	0.14	5.17	0.09	4.64	0.17	3.02	0.10	3.90	0.12	196

per fragment pair, reported in Table 6. The Table does not include PPFNet, PPFFoldNet and PointCaps3D as these figures are not provided in the papers and the code is not publicly available. These results show how Li *et al*. can provide a significantly larger number of good correspondences per fragment pair. However, LEAD and D3Feat turn out basically on-par as second-best methods, outperforming all other methods by a large margin.

TABLE 6: Percentage of correctly matched keypoints per fragment pair on 3DMatch. Best result in each column are highlighted in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	0.1663	0.2370	0.2192	0.1604	0.1524	0.1983	0.1068	0.1348	0.1719
SHOT	0.2549	0.3272	0.2981	0.2643	0.2184	0.2549	0.1935	0.2309	0.2553
SI	0.1016	0.1207	0.1379	0.0903	0.0762	0.0930	0.0445	0.0562	0.0900
USC	0.3330	0.4043	0.3655	0.3429	0.2546	0.3051	0.2494	0.2638	0.3148
CGF	0.1125	0.1418	0.1482	0.0840	0.0993	0.1627	0.0549	0.0667	0.1088
3DMatch	0.0881	0.1192	0.1227	0.0661	0.0668	0.0777	0.0601	0.0645	0.0831
3DSmoothNet	0.3279	0.4286	0.3822	0.3503	0.3350	0.4226	0.3000	0.3361	0.3603
FCGF	0.2593	0.3011	0.2942	0.2919	0.2915	0.3169	0.2417	0.3171	0.2892
D3Feat	0.3436	0.4608	0.4296	0.3915	0.3823	0.4599	0.3555	**0.4160**	0.4049
Li *et al.*	**0.4265**	**0.4963**	**0.4817**	**0.5104**	**0.4781**	**0.5268**	**0.3703**	0.3864	**0.4596**
Ours ICCV	0.3374	0.4111	0.4010	0.3706	0.3303	0.3617	0.2767	0.3027	0.3489
LEAD	0.4229	0.4695	0.4576	0.4510	0.3940	0.4115	0.3220	0.3367	0.4081

6.6 Evaluation on the ETH dataset

Here we evaluate descriptors in an outdoor scenario and focus on the ability to generalize to new environments, which we argue to be key for learned 3D descriptors. Indeed, in practical settings, one would wish to train on a certain, possibly large, data corpus and get a model that could be effectively deployed on a variety of novel target datasets. In particular, it is so when addressing pairwise registration, *i.e*. the reference task to evaluate 3D descriptors, by models learned through supervision: should the ground-truth poses between views needed to train the descriptor be available in the target dataset, there would be no need to perform any pairwise registration on that dataset.

Hence, in this subsection we evaluate learned 3D descriptors by a transfer learning experiment: we get the models trained on 3DMatch and test them on the ETH dataset. Akin to Table 6 we do not consider PPFNet, PFFFoldNet and PointCaps3D due lack of reported data and unavailability of code. For the sake of completeness, we report also the results provided by handcrafted descriptors. The *Recall* figures yielded by the considered methods are collected in Table 7. When applied to unseen data lacking any ground-truth poses, the setting most likely to be faced in a real registration application, LEAD delivers by far the best performance, *i.e*. 97.5% average recall, surpassing all other methods by vast margins. We deem it worth pointing out, in particular, how the margin is as high as 17.6% compared to Li *et al*., the method which provides the best performance on 3DMatch, and about 18.5% w.r.t. 3DSmoothNet. On the other hand, the other two learned descriptors showing excellent performance on 3DMatch, namely FCGF and D3Feat, cannot generalize effectively to the new environments set forth by ETH dataset. Interestingly, these methods try to learn their invariance to rotations via data augmentation at training time, which further validates our intuition on the limitations of such an approach compared to theoretical equivariance of the descriptor.

TABLE 7: *Recall* on the *ETH* dataset. Best result in each column shown in bold.

Method	Gazebo		Wood		Average
	Summer	Winter	Summer	Autumn	
FPFH	0.3860	0.1420	0.1480	0.2080	0.2210
SHOT	0.7450	0.4530	0.6320	0.6170	0.6118
SI	0.6957	0.3979	0.5520	0.5043	0.5375
USC	0.7065	0.2872	0.6160	0.6348	0.5611
CGF	0.3750	0.1380	0.1920	0.1040	0.2023
3DMatch	0.2280	0.0830	0.2240	0.1390	0.1685
3DSmoothNet	0.9130	0.8410	0.7280	0.6780	0.7900
FCGF	0.2554	0.1661	0.2348	0.3040	0.2410
D3Feat	0.4570	0.2390	0.1300	0.2240	0.2620
Li *et al.*	0.8530	0.7200	0.8400	0.7830	0.7990
Ours ICCV	0.6739	0.4844	0.5920	0.5304	0.5702
LEAD	**0.9239**	**0.9862**	**0.9913**	**1.0000**	**0.9753**

Similarly to subsection 6.5, in Table 8 we report the percentage of matched keypoints per fragment pair, while in Table 9 the RRE (degrees) and RTE (meters) for methods that can register at least 70% of the fragment pairs according to the *Recall* figures in Table 7.

TABLE 8: Percentage of correctly matched keypoints per fragment pair on the ETH dataset. Best result in each column highlighted in bold.

Method	Gazebo		Wood		Average
	Summer	Winter	Summer	Autumn	
FPFH	0.0453	0.0235	0.0212	0.0254	0.0289
SHOT	0.1170	0.0781	0.0824	0.0980	0.0939
SI	0.1000	0.0621	0.0772	0.0917	0.0828
USC	**0.1907**	0.0602	**0.1509**	0.1543	0.1391
CGF	0.0511	0.0250	0.0212	0.0241	0.0304
3DMatch	0.0308	0.0115	0.0206	0.0317	0.0236
3DSmoothNet	0.1795	0.1267	0.0959	0.1147	0.1292
FCGF	0.0451	0.0252	0.0405	0.0320	0.0357
D3Feat	0.0664	0.0384	0.0234	0.0281	0.0391
Li *et al.*	0.1452	0.1040	0.1023	0.1281	0.1199
Ours ICCV	0.1041	0.0753	0.0692	0.0825	0.0828
LEAD	0.1654	**0.1380**	0.1360	**0.1558**	**0.1488**

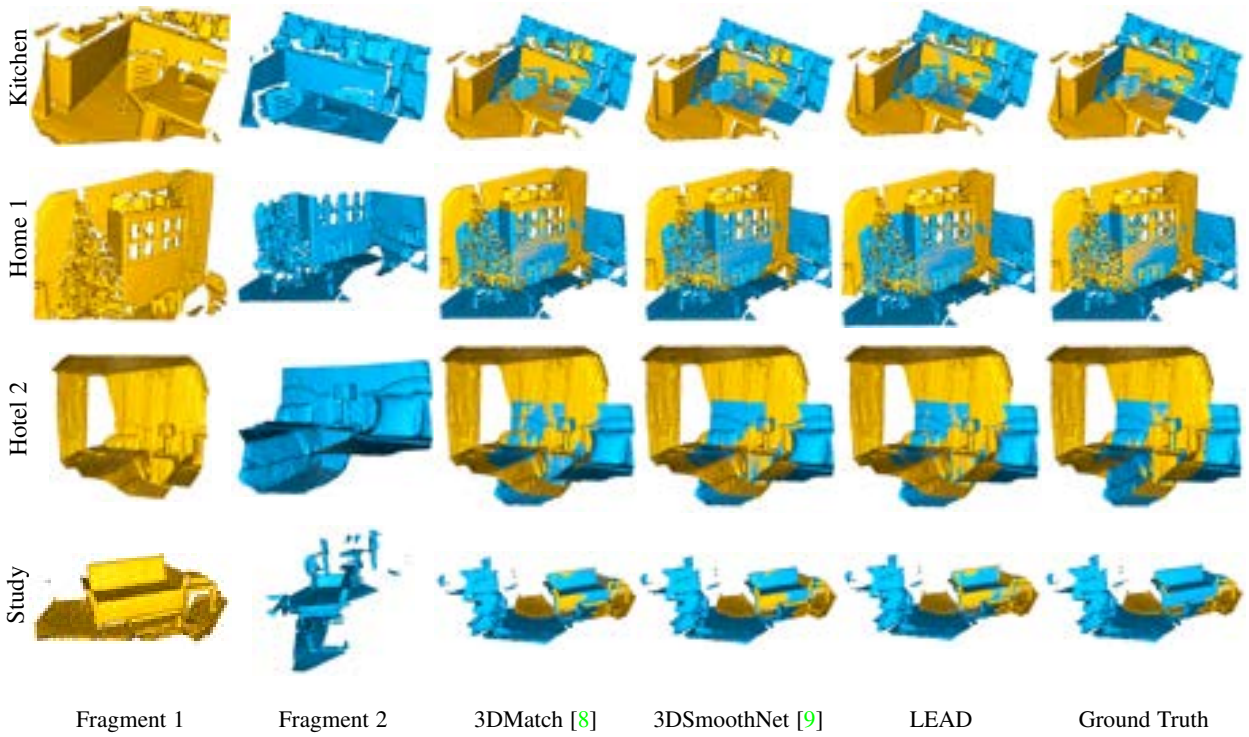

Fig. 6: Registration results on the 3DMatch Benchmark after RANSAC.

TABLE 9: Registration errors (RRE,RTE) for methods yielding at least 70% *Recall* on ETH. The average number of registered pairs is also reported in the last column. Best results highlighted in bold.

Method	Gazebo				Wood				Average		
	Summer		Winter		Summer		Autumn				
	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	RRE	RTE	Pairs
3DSmoothNet	1.52	0.07	**30.82**	**0.86**	1.38	0.38	1.06	0.20	**8.70**	0.38	145
Li *et al*.	20.49	0.50	43.12	1.19	0.88	**0.05**	1.06	0.09	16.39	0.46	140
LEAD	**1.29**	**0.05**	35.00	0.99	**0.74**	0.10	**0.90**	**0.08**	9.48	**0.31**	**174**

TABLE 10: Point cloud density experiment on 3DMatch. We test LEAD trained on uniformly sampled clouds at 2 *cm* on denser (0.5 and 1.0 *cm*) and sparser (3.0 and 4.0 *cm*) versions of the test set.

Sampling leaf radius (*cm*)	Average *Recall*	*Recall* decay
0.5	0.9547	0.0037
1.0	0.9499	0.0085
2.0	0.9584	-
3.0	0.9421	0.0163
4.0	0.9398	0.0186

Table 8 vouches again for the superiority of LEAD w.r.t. the other two learned descriptors that can withstand the domain change set forth by the transfer learning experiment, namely Li *et al*. and 3DSmoothNet. As for registration accuracy (Table 9), we argue that LEAD compares favourably to the above methods since it yields the lowest average translation error and almost the same rotation error (+0.78 degrees) as 3DSmoothNet in spite of a much larger number or registered pairs.

6.7 Robustness to cloud density variations

This section explores LEAD's robustness against cloud density variations. In particular, we test the network trained on the 3DMatch training set uniformly sampled with $2cm$ radius, *i.e*. the same network trained for the previous experiments, on different versions of the test set, comprising either denser (0.5 and $1cm$ radius) or sparser clouds (3 and $4cm$ radius). In Table 10 we report the results, which shows the robustness of LEAD to this nuisance.

6.8 Computation time

We measure the run time of our algorithm on a system equipped with an i7 3.2 GHz CPU, an RTX 2080Ti GPU and 64 GB of RAM. On this hardware platform, the average keypoint description time is 5.30 ms, of which the radius search to collect the neighbours in the patch takes 0.09 ms, the creation of the Spherical signal 1.35 ms and the inference time on GPU 3.87 ms. This time is comparable to the figures reported in literature for 3DMatch [8] and 3DSmoothNet [9], that is 5.0 and 4.6 ms respectively. Conversely, much shorter description times have been reported for other proposals like PPF-FoldNet [13] (0.794ms), PointCaps3D [33] (1.208 ms) and FCGF [14] (0.009 ms). We believe that this paper has demonstrated solidly the superior effectiveness attainable by LEAD compared to the above-mentioned more efficient methods. In particular, despite the very remarkable speed, FCGF seems to struggle to generalize when used in transfer learning scenarios, a quite desirable trait in practical scenarios.

6.9 Qualitative results

To visually assess upon the quality of the pairwise registrations yielded by the considered methods, in this section we show some qualitative results dealing with the 3DMatch (Figure 6) and ETH (Figure 7) datasets. Here we compare LEAD to two descriptors learned supervisedly, *i.e*. 3DMatch [8], as a baseline method, and 3DSmoothNet [9], a more recent approach that can deliver excellent performance on both datasets. Figure 6 and Figure 7

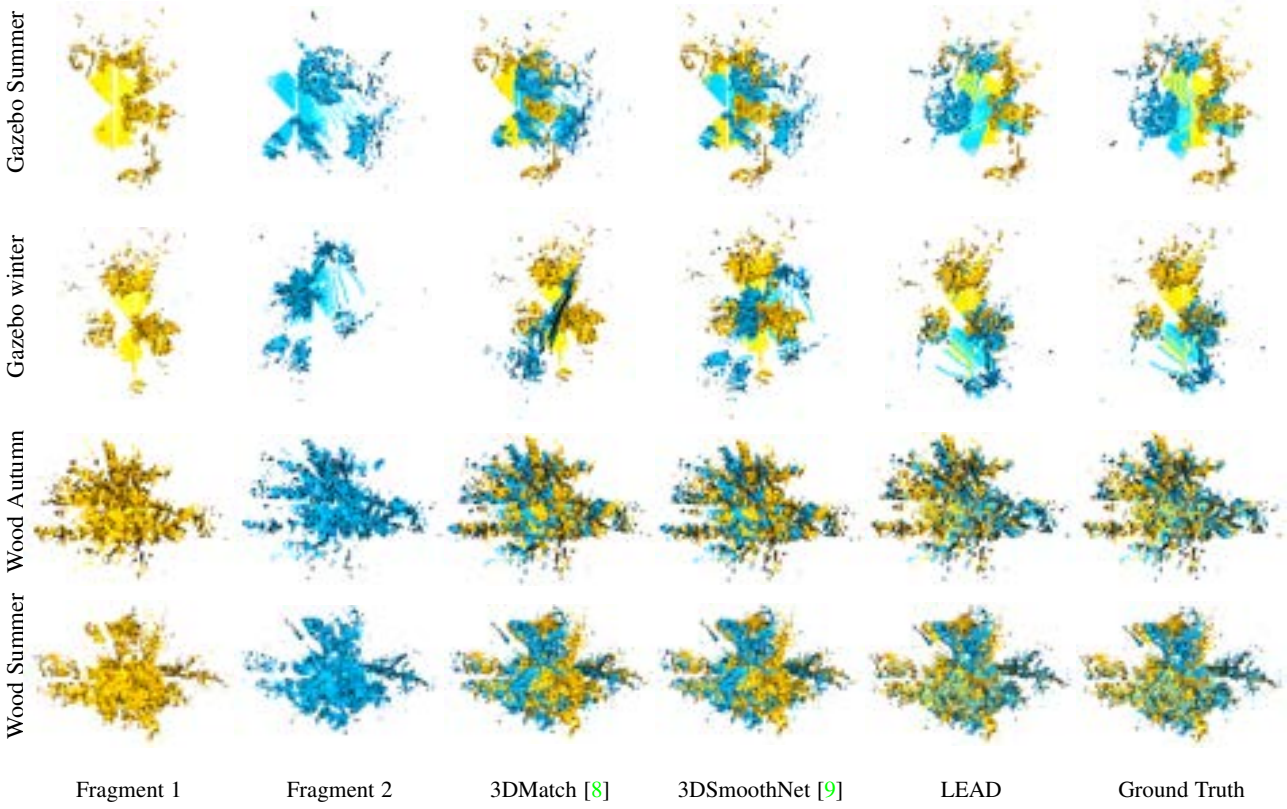

Fig. 7: Registration results on the ETH Benchmark after RANSAC.

highlight how LEAD produces very correct alignments that, in general, look more similar to the given ground-truths, especially in transfer learning settings (Figure 7).

7 Conclusions

We have presented LEAD, an effective methodology to learn a local 3D descriptor from raw point cloud data without supervision. Differently from the classical path taken to learn descriptors, we seek invariance to viewpoint changes only at test time, and learn instead a rotation equivariant representation at training time, by leveraging Spherical CNNs. Learning an equivariant descriptor frees ourselves from the need to choose a specific invariant input representation or to resort to data augmentation, as done in previous work to achieve invariance to viewpoint changes. Moreover, LEAD can be conveniently learned in an unsupervised framework. We empirically validated how our innovative design decision results in a very effective descriptor, which outperforms by a large margin all unsupervised proposals and most of the supervised approaches, all of which deployed one of the other two alternatives to achieve viewpoint invariance. We also provide a thorough study on the impact of different design decisions for the spherical encoder, discovering an architecture that is faster and more effective than the one used in the preliminary version of our work [27].

The ideas and results presented in this paper let us believe that it would be worth investigating how Spherical CNNs may be deployed to build a complete feature matching pipeline without relying on an external LRF. Indeed, our concurrent work shows how Spherical CNNs can be effectively deployed also to learn to rotate raw point clouds so to achieve viewpoint invariance [67]. Hence, the capability of operating on the $\mathrm{SO}(3)$ domain could be interestingly leveraged to develop a data-driven solution to jointly learn a canonical orientation alongside a local 3D feature descriptor. As a further extension, it will be worth to investigate how to deploy a recent improvement in the space of Spherical CNNs, referred to as Icosahedral CNNs [68], to speed-up LEAD by reducing the computation time of spherical correlations.

Acknowledgments

We gratefully acknowledge NVIDIA Corporation with the donation of the Titan RTX 2080Ti GPU used in this work. We also thank the 3DSmoothNet, D3Feat, PPFNet, and Li *et al*. authors for providing their results as well as the FCGF ones for their code. We would like to thank Injenia srl and UTFPR for partly supporting this research work.

References

[1] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 21, no. 5, pp. 433–449, 1999. 1, 2, 8

[2] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 3212–3217. 1, 2, 8

[3] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European conference on computer vision*. Springer, 2010, pp. 356–369. 1, 2, 8, 9

[4] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan, "Rotational projection statistics for 3D local surface description and object recognition," *International journal of computer vision*, vol. 105, no. 1, pp. 63–86, 2013. 1, 2

[5] S. Salti, F. Tombari, and L. Di Stefano, "SHOT: Unique signatures of histograms for surface and texture description," *Computer Vision and Image Understanding*, vol. 125, pp. 251–264, 2014. 1, 2, 8, 9

[6] J. Yang, Q. Zhang, Y. Xiao, and Z. Cao, "Toldi: An effective and robust approach for 3d local shape description," *Pattern Recognition*, vol. 65, pp. 175–187, 2017. 1, 2

[7] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, "A comprehensive performance evaluation of 3D local feature descriptors," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016. 1, 2

[8] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1802–1811. 1, 2, 7, 8, 12, 13

[9] Z. Gojcic, C. Zhou, J. D. Wegner, and W. Andreas, "The perfect match: 3d point cloud matching with smoothed densities," in *International conference on computer vision and pattern recognition (CVPR)*, 2019. 1, 2, 6, 7, 8, 9, 12, 13

[10] M. Khoury, Q.-Y. Zhou, and V. Koltun, "Learning compact geometric features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 153–161. 1, 2, 6, 8

[11] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*. Ieee, 2010, pp. 998–1005. 1, 2, 3

[12] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 195–205. 1, 2, 3, 8

[13] ——, "PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 602–618. 1, 2, 3, 6, 7, 8, 9, 12

[14] C. Choy, J. Park, and V. Koltun, "Fully convolutional geometric features," in *ICCV*, 2019. 1, 2, 3, 8, 9, 12

[15] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6359–6367. 1, 2, 3, 8, 9

[16] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017. 1

[17] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848. 1, 3

[18] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) equivariant representations with spherical CNNs," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–68. 2, 3, 5, 7

[19] J. Yang, Y. Xiao, and Z. Cao, "Toward the repeatability and robustness of the local reference frame for 3d shape matching: An evaluation," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3766–3781, 2018. 2, 6

[20] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6411–6420. 2, 3

[21] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084. 2, 3

[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440. 2, 3

[23] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," *arXiv preprint arXiv:1801.10130*, 2018. 2, 3, 4, 5, 7, 8

[24] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3D surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224. 2, 4, 5, 7

[25] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 206–215. 2, 3, 5

[26] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *The International Journal of Robotics Research*, vol. 31, no. 14, pp. 1705–1711, 2012. 2, 7, 8

[27] R. Spezialetti, S. Salti, and L. D. Stefano, "Learning an effective equivariant 3d descriptor without supervision," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2, 3, 6, 8, 9, 13

[28] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 539–546. 2

[29] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Advances in neural information processing systems*, 2004, pp. 41–48. 2

[30] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of Machine Learning Research*, vol. 10, no. 2, 2009. 2

[31] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312. 2

[32] L. Li, S. Zhu, H. Fu, P. Tan, and C.-L. Tai, "End-to-end learning local multi-view descriptors for 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1919–1928. 3, 8, 9

[33] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 8, 9, 12

[34] T. Birdal and S. Ilic, "Point pair features based object detection and pose estimation revisited," in *2015 International Conference on 3D Vision*. IEEE, 2015, pp. 527–535. 3

[35] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953. 3

[36] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1544–1553. 3

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

[38] R. Klokov and V. 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*, 2017, pp. 863–872. 3

[39] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2088–2096. 3

[40] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 652–660. 3

[41] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017, pp. 5099–5108. 3

[42] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025. 3

[43] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1607–1616. 3

[44] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 984–993. 3

[45] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102. 3

[46] F. Groh, P. Wieschollek, and H. P. Lensch, "Flex-convolution," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 105–122. 3

[47] Y. Rao, J. Lu, and J. Zhou, "Spherical fractal convolutional neural networks for point cloud recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 452–460. 3

[48] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung, "Rotation invariant convolutions for 3d point clouds deep learning," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 204–213. 3

[49] M. Sugiura, *Unitary representations and harmonic analysis: an introduction*. Elsevier, 1990. 4

[50] G. B. Folland, *A course in abstract harmonic analysis*. CRC press, 2016, vol. 29. 4

[51] P. J. Kostelec and D. N. Rockmore, "Soft: So (3) fourier transforms," *Department of Mathematics, Dartmouth College, Hanover, NH*, vol. 3755, 2007. 4

[52] D. K. Maslen, "Efficient computation of fourier transforms on compact groups," *Journal of Fourier Analysis and Applications*, vol. 4, no. 1, pp. 19–52, 1998. 4

[53] D. K. Maslen and D. N. Rockmore, "Generalized ffts—a survey of some recent results," in *Groups and Computation II*, vol. 28. American Mathematical Soc., 1997, pp. 183–287. 4

[54] Y. You, Y. Lou, Q. Liu, L. Ma, W. Wang, Y. Tai, and C. Lu, "PRIN: Pointwise rotation-invariant network," *arXiv preprint arXiv:1811.09361*, 2018. 5, 7

[55] J. R. Driscoll and D. M. Healy, "Computing Fourier transforms and convolutions on the 2-sphere," *Advances in applied mathematics*, vol. 15, no. 2, pp. 202–250, 1994. 5

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 6

[57] T. Risbo, "Fourier transform summation of Legendre series and D-functions," *Journal of Geodesy*, vol. 70, no. 7, pp. 383–396, 1996. 6

[58] A. Petrelli and L. Di Stefano, "A repeatable and efficient canonical reference for surface matching," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, 2012, pp. 403–410. 6, 9, 10

[59] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4. 6

[60] J. Valentin, A. Dai, M. Nießner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 323–332. 7

[61] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937. 7, 10

[62] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1625–1632. 7, 10

[63] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3050–3057. 7

[64] M. Halber and T. Funkhouser, "Fine-to-coarse global registration of RGB-D scans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1755–1764. 7

[65] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface reconstruction from unorganized points*. ACM, 1992, vol. 26, no. 2. 8

[66] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012. 8, 10

[67] R. Spezialetti, F. Stella, M. Marcon, L. Silva, S. Salti, and L. Di Stefano, "Learning to orient surfaces by self-supervised spherical cnns," *Advances in Neural Information Processing Systems*, vol. 33, 2020. 13

[68] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral CNN," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 1321–1330. [Online]. Available: http://proceedings.mlr.press/v97/cohen19d.html 13

Marlon Marcon received the PhD degree in Computer Science from the Federal University of Paraná, Brazil in 2020. He spent a year (2019–2020) in a doctoral internship stage at the Computer Vision Laboratory (CVLab), University of Bologna, Italy. He is currently an Assistant Professor at the Department of Software Engineering, Federal University of Technology - Paraná - Brazil. His research interests include image processing, computer vision, and deep learning methods applied on 2D and 3D data.

Riccardo Spezialetti received his PhD degree in Computer Science and Engineering from University of Bologna in 2020. Currently, he is a Post-doc researcher at Department of Computer Science and Engineering, University of Bologna. His research interest concerns machine/deep learning for 3D computer vision problems.

Samuele Salti is currently assistant professor at the Department of Computer Science and Engineering (DISI) of the University of Bologna, Italy. Before joining the University of Bologna, he was leading the Data Science team at Verizon Connect. His main research interest is computer vision, in particular 3D computer vision, and machine/deep learning applied to computer vision problems. Dr. Salti has co-authored 42 publications in international conferences and journals and 8 international patents. In 2020, he co-founded the start-up eyecan.ai. He was awarded the best paper award runner-up at 3DIMPVT 2011, the top international conference on 3D computer vision, and was outstanding reviewer at CVPR 2020,2021 and NeurIPS 2020.

Luciano Silva received the BS and MSc degrees in informatics from the Universidade Federal do Parana (UFPR), Brazil, in 1998 and 2000, respectively, and the PhD degree in electrical engineering and industrial informatics from the Universidade Tecnológica Federal do Parana (UTFPR), Brazil, in 2003. He joined the IMAGO Research Group in 1996. From 2002 to 2003, he did doctoral studies from the Signal Analysis and Machine Perception Laboratory at Ohio State University. Since 2004, he has been with the Informatics Department of UFPR, where he is currently an associate professor. His research interests include computer vision, 2D/3D image analysis, and assistive technologies. He is a member of the IEEE and the Brazilian Computer Society.

Luigi Di Stefano received the PhD degree in electronic engineering and computer science from the University of Bologna in 1994. He is currently a full professor at the Department of Computer Science and Engineering, University of Bologna, where he founded and leads the Computer Vision Laboratory (CVLab). His research interests include image processing, computer vision and machine/deep learning. He is the author of more than 150 papers and several patents. He has been scientific consultant for major companies in the fields of computer vision and machine learning. He is a member of the IEEE Computer Society and the IAPR-IC.