3D Local Descriptors—from Handcrafted to Learned

3D Local descriptors - from hand-crafted to learned

Riccardo Spezialetti, Samuele Salti, Luigi Di Stefano and Federico Tombari

Abstract Surface matching is a fundamental task in 3D Computer Vision, typically tackled by describing and matching local features computed from the 3D surface. As a result, description of local features lays the foundations for a variety of applications processing 3D data, such as 3D object recognition, 3D registration and reconstruction, and SLAM. A variety of algorithms for 3D feature description exists in the scientific literature. The majority of them are based on different, hand-crafted ways to encode and exploit the geometric properties of a given surface. Recently, the success of deep neural networks for processing images has fueled also a data-driven approach to learn descriptive features from 3D data. This chapter provides a comprehensive review of the main proposals in the field.

1 Introduction

Nowadays surface matching is solved by establishing point-to-point correspondences obtained by matching *local 3D descriptors* [72, 13, 43]. Local 3D descriptors are also referred to as *local features* and the paradigm, illustrated in Figure 1, is referred to as *feature-based matching* . Examples of 3D computer vision tasks that leverage local 3D descriptors are:

Riccardo Spezialetti
University of Bologna, e-mail: riccardo.spezialetti@unibo.it

Samuele Salti
University of Bologna, e-mail: samuele.salti@unibo.it

Luigi Di Stefano
University of Bologna, e-mail: luigi.distefano@unibo.it

Federico Tombari
TU Munich, Google e-mail: tombari@in.tum.de

- **3D Object Detection:** identification within a scene of an object from a library of models and estimation of its 6D pose.
- **3D Object Reconstruction:** recreate the shape and appearance of objects by registering a set of partial views.
- **3D Object Retrieval:** retrieving three dimensional models in large databases given a similar object as query or given a textual description of the object.
- **SLAM (Simultaneous Localization and Mapping):** building the map of an unknown environment while simultaneously localizing the sensor therein.

Descriptors or features are a compact but rich representation of the underlying 3D data designed to be either *invariant* or *robust* to a large set of nuisances, like rotation, point density variations, sensor noise, view point changes, occlusions and clutter.

Depending on the area involved, we can classify descriptors as either *global* or *local* . *Global* methods encode the entire 3D model in the descriptor [49, 1, 39] and are usually applied after a segmentation step to limit the influence of clutter. *Local* descriptors, instead, encode a small part of the surface, small with respect to the overall scene or object being matched (*i.e.*a neighborhood around a keypoint). The main advantage of using local descriptors versus global ones is higher robustness to partial occlusions of the surface and to the presence of clutter. For this reason, the latter are currently the dominant approach. In this chapter we will focus uniquely on local descriptors, illustrating the main proposals in the literature and the way they have evolved from mostly hand-crafted proposals to those based on deep learning.

According to [19], two key aspects for a 3D local descriptor are *descriptiveness* and *robustness* : descriptiveness refers to the capacity to capture predominant traits of the surface; robustness is the ability to be not sensitive (*i.e.*to obtain a similar descriptor) in the presence of disturbances that can affect the data. Choosing the descriptor offering the right descriptiveness/robustness trade-off for a specific 3D computer vision application is a crucial step and still an open issue: usually a few prominent alternatives are tested and the best performing one is used.

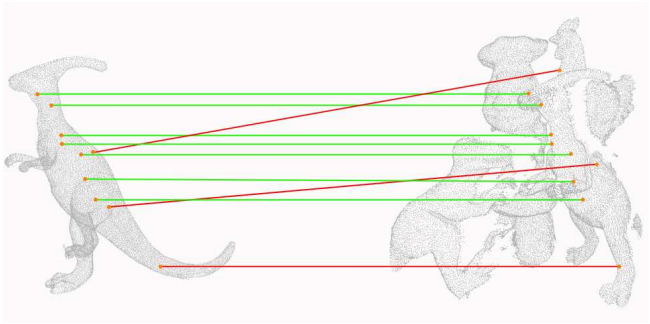


Fig. 1 Feature-based matching paradigm. Green lines correspond to good correspondences, while red lines correspond to mismatches.

A typical *feature-matching pipeline* consists of:

- **feature-detection:** the goal is to identify points of a surface particularly prominent with respect to a given saliency function computed on a local neighborhood of each point. These points are called *keypoints* .
- **feature-description:** as discussed, a descriptor is a compact representation of geometric properties of a point. Descriptors usually are designed to encode essential characteristic of points and to be invariant to nuisances like variations of point density and viewpoint. To speed up the overall feature matching process, typically just the keypoints of an object are described.
- **feature-matching:** the last phase of the pipeline yields correspondences between two or more surfaces by finding the most similar descriptors among the sets of descriptors computed at the keypoints of the surfaces. Several metrics can be used to compare descriptors, the most popular being the Euclidean distance in the descriptors space.

2 Background

Before we go into this chapter, let us provide some background definitions for a better understanding of the related works. We will start talking about the data structures most commonly used to represent 3D data, then we will introduce the basic concepts related to the computation of local features, and finally conclude with some definitions about the most widespread noise sources that can be observed in 3D data.

- **Point cloud**: A point cloud is a set of points, that describes the shape of 3D object, characterized by their position in a coordinate system expressed with euclidean coordinates. In addition to the single coordinates, it is also possible to store information about the color of the points, triple RGB, and their intensity (alpha channel). Cloud points are unorganized data structures, i.e. simple data collections, whose closeness in the data structure does not imply closeness in the observed space. The main limitation of this data structure is the high cost of searching for neighbours for a point in the 3D coordinate space.
- **Polygon Mesh**: A polygon mesh is a collection of vertices, edges and faces that defines the shape of 3D object. The faces usually consist of triangles, quadrilaterals, or other simple convex polygons. Unlike cloud points, meshes are organized data structures.
- **Feature point**: A feature point, denoted by $\mathbf{p}$, is a point belonging to a 3D shape $\mathcal{M}$, for which the descriptor is being computed. Typically feature points coincide with keypoints.
- **Support**: A support of a feature point $\mathbf{p}$, is the set of neighboring points lying within a spherical ball of radius $r > 0$ centered at $\mathbf{p}$, denoted by $B_r(p) = \{s \in \mathcal{M} : \|p - s\|_2 < r\}$. Alternatively, the support can be made up of a fixed number of neighbours. Usually to the search for nearby points, a space-partitioning data structure for organizing points in a k-dimensional space like k-trees [3] is

utilized. The support of the feature point is used the compute the associated descriptor.

- **Local Reference Frame**: A Local Reference Frame (LRF) is a local system of Cartesian coordinates at each point, with respect to which the local geometric structure around that point is encoded. An LRF $\mathcal{L}(p)$ at point $\mathbf{p} \in \mathcal{M}$ is an orthogonal set of unit vectors:

$$\mathcal{L}(p) = \{\hat{\mathbf{x}}(p), \hat{\mathbf{y}}(p), \hat{\mathbf{z}}(p)\} \tag{1}$$

  Invariance with respect to the object's pose is a fundamental trait of every 3D descriptor: state-of-the-art proposals achieve invariance using either a *Reference Axis* or a *Local Reference Frame (LRF)*, which co-variate with the object pose. Before descriptor computation, the support of the feature point is translated and oriented according to the Local Reference Frame or Reference Axis, effectively removing the pose change and making subsequent computations on geometric entities of the support pose invariant. As clear from this description and proven experimentally in [42], the repeatability of the reference system is key for the performance of the descriptor.
- **Clutter and cluttered scene**: The concept of clutter is relate to Object Detection. In this context we want to recognize a certain number of well-known objects within a scene. A clutter object is a distracting object that is not part of the set of known objects but can be present in the scene. Similarly, a scene containing a clutter object is defined as a cluttered-scene.
- **Missing part**: Once scanned, objects may present missing parts due to the sensor's limited field of view or the shadow generated by the presence of other objects in the scene.

3 Related works

This section reviews the main proposals for local 3D descriptors. In Figure 2 we show a temporal evolution of the main proposals for 3D descriptors. A large body of work has been dedicated to define new and more effective solutions to the surface matching problem based on local 3D descriptors in the last 20 years: *Point Signature*[10], *Spin Images*[28], *3D Shape Context*[16], *Local Surface Patch*[9], *Point Feature Histograms*[48], *Scale-Dependent Local Shape Descriptor*[38], *Heat Kernel Signature*[58], *Fast Point Feature Histograms*[47], *Intrinsic Shape Signature*[73], *MeshHog*[71], *3D Surf*[32], *SHOT*[51], *Unique Shape Context*[60], *Rops*[20], *Trisi*[21], *Toldi*[68], *3D Match*[72], *CGF*[29], *PPF-Net*[14] and *PPF-FoldNet*[13].

Local 3D descriptors can be further categorized into two main categories: those encoding hand-crafted traits of the support and those obtained as output of a (deep) learning algorithm. In the latter case, typically a deep neural network is trained purposely to infer the 3D descriptor given a portion or the entire surface as input.

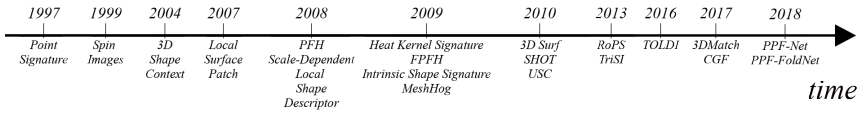


Fig. 2 Temporal evolution of 3D feature descriptors.

3.1 Handcrafted local 3D descriptors

A local 3D descriptor creates a compact representation of a 3D surface by collecting geometric or topological measurements into histograms. According to [19], hand-crafted local 3D descriptors can be grouped in *spatial distribution histogram* and *geometric attribute histogram*. Spatial distribution histogram count the number of points falling in each histogram bin taking into account the spatial distribution of the points on the surface. Conversely, geometric attribute histogram generate histograms exploiting geometric properties of the surface like normals or curvatures.
In the following we give an overview of methods belonging to both categories.

3.1.1 3D Shape Context

3D shape context (3DSC) directly extends the 2D shape context descriptor [2] to three dimensions. It was introduced by Frome *et al*. at **ECCV 2004** [16].
The descriptor captures the local shape of a point cloud at a feature point $\mathbf{p}$ using the distribution of points in a spherical grid support. The normal $\mathbf{n}$ at $\mathbf{p}$ serves as reference axis to align the north pole of the grid. Within the support region, a set of bins is constructed by equally dividing the azimuth and elevation dimensions, while the radial dimension is logarithmically spaced, as shown in Figure 3. Due to this binning scheme the descriptor is more robust to distortions in shape with distance from the feature point.

Each point p_i in the support is mapped to its corresponding bin using spherical coordinates, and the bin corresponding to them accumulates a weighted sum of local point density. The local point density for p_i is estimated as the number of the points in a sphere of radius δ around p_i. Relying only on a reference axis and not on a full 3D reference frame, the descriptor is invariant up to a rotation along the north pole. Therefore, multiple descriptors for a single feature point have to be described and matched across different views of a surface, which significantly slows down the performance of the overall pipeline. The size of the final descriptor is $d_a \times d_e \times d_r$, where d_a, d_e and d_r are respectively the numbers of bins along the azimuth, elevation and radial axes, and the number of descriptors to be computed for a feature point is equal to d_a.

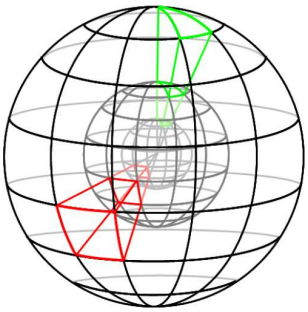


Fig. 3 Bins of 3D shape context. The grid is divided into several bins along the radial, azimuth and elevation dimensions.

3.1.2 Unique Shape Context

Unique Shape Context (USC) improves 3DSC by leveraging a definition of repeatable and unambiguous Local Reference Frame as pointed out in Figure 4. It was presented by Tombari *et al*. at the **ACM workshop on 3D object retrieval in 2010**.

One of the main drawbacks of 3DSC is the computation of multiple descriptors at a given feature points. Indeed, to take into account the degree of freedom on azimuth direction, a vector on the tangent plane at the normal $\mathbf{n}$ at the feature point $\mathbf{p}$ is randomly chosen and the grid support is then rotated about its north pole in to d_a positions. Each rotation defines a unique Local Reference Frame used to orientate the sphere grid and to computed the associated descriptor. As a consequence, d_a descriptors are estimated and stored for each feature point.

Differently from 3DSC, in USC the authors first estimate a Local Reference Frame using the same approach presented in 4.1.1 and proposed in [60], then construct the descriptor likewise 3DSC.

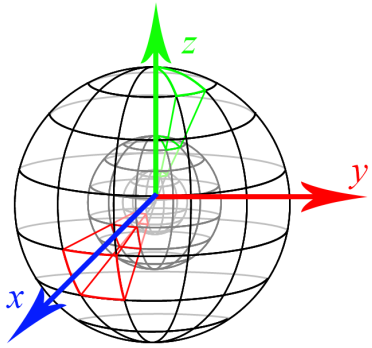


Fig. 4 USC unambiguous support. The spatial subdivision of 3DSC is aligned according to a Local Reference Frame to obtain a unique descriptor for each feature point.

Thanks to the adopted unique and repeatable Local Reference Frame, USC improves efficiency in terms of both performance and memory footprint with respect to 3DSC. The size of the final descriptor is the same as 3DSC.

3.1.3 Rotational Projection Statistics

Rotational Projection Statistics (RoPS) is a local 3D descriptor designed to operates on meshes. It was published on **International journal of computer vision** by Guo *et al*. [20] in 2013. The first contribution of RoPS is the definition of a novel Local Reference Frame which rely on the eigenvalue decomposition of the covariance matrix. Differently from pre-existing approaches, the covariance matrix for a given feature point $\mathbf{p}$ is estimated by aggregating covariance matrices computed for every single triangle within the support. Each single matrix is weighted differently to mitigate the effect of mesh resolution variations and enhance the robustness to clutter and occlusion. The obtained $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ axes are disambiguated in order to achieve a unique and repeatable canonical orientation. The third axis $\hat{\mathbf{y}}$, is derived as $\hat{\mathbf{z}} \times \hat{\mathbf{x}}$.

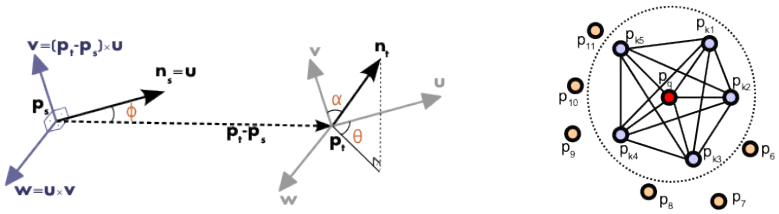
As for the histogram computation, the whole process is illustrated in Figure 4 in [20]. Given the $\hat{\mathbf{x}}$ axis the points within the support are rotated around it by a given angle, and projected onto three planes xy, yz and xz. For each projection a distribution matrix $\mathcal{D}$ is built by partitioning the plane into $L \times L$ bins and counting up the number of points falling into each bin. The number of bins represents the matrix dimension and is a parameter of the method. The distribution matrix $\mathcal{D}$ contains information about the local surface from a particular viewpoint. Hence, *five* statistics, including the central moments [12, 25] and Shannon entropy [54] are extracted from each distribution matrix. Different rotations along the same axis are taking into account to capture information from various viewpoints. The above mentioned process is repeated again for the $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ axes. The final descriptor is obtained by concatenating the statistics of all rotations. The size is $3 \times 3 \times 5 \times d_{rot}$, where d_{rot} is the number of rotations around each axis.

3.1.4 Point Feature Histogram

Point Feature Histogram (PFH) is designed to capture the surface variations based on the relationships between points in the local neighborhood and directions of the estimated normals. Hence, the performance is closely related to the quality of the surface normal estimations at each point. It was introduced by Rusu *et al*. [48] at **IROS 2008**.

As show in Figure 5, for every pair of points $\mathbf{p}_i$ and $\mathbf{p}_j$, in the neighborhood of $\mathbf{p}$, a Darboux frame is built by choosing one point as source $\mathbf{p}_s$, and the other as target $\mathbf{p}_t$:

$$\mathbf{u} = \mathbf{n}_s, \mathbf{v} = \mathbf{u} \times \frac{\mathbf{p}_t - \mathbf{p}_s}{\|\mathbf{p}_t - \mathbf{p}_s\|}, \mathbf{w} = \mathbf{u} \times \mathbf{v} \tag{2}$$



(a) (b)

Fig. 5 PFH overview. (a) The u, v and w vector of a Darboux frame computed for a pair of points. Given two points $\mathbf{p}_i$ and $\mathbf{p}_j$ together with their normal $\mathbf{n}_i$ and $\mathbf{n}_j$, one is chosen as source point, $\mathbf{p}_s$, and the other as target point $\mathbf{p}_t$. (b) PFH computation for a feature point $\mathbf{p}$ shown in red. All the points enclosed in the support region are interconnected in a dense mesh. The histogram is built taking into account the angular features between all pairs of points, leading to $\mathbf{O}(k^2)$ computational complexity for a point with k neighbor points. Images from [41].

Next, using the frame defined above, three angular features, expressing the relationship between normals $\mathbf{n}_t$ and $\mathbf{n}_s$, and between normals and the difference vector between $\mathbf{p}_t$ and $\mathbf{p}_s$ are computed for each pair of points:

$$\alpha = \langle \mathbf{v}, \mathbf{n}_t \rangle, \phi = \langle \mathbf{u}, \mathbf{p}_t - \mathbf{p}_s \rangle / d, \theta = arctan(\langle \mathbf{w}, \mathbf{n}_t \rangle, \langle \mathbf{u}, \mathbf{n}_t \rangle), d = ||\mathbf{p}_t - \mathbf{p}_s|| \tag{3}$$

Also the length of the difference vector d can be used as feature, although it is usually not considered as the distance between neighboring points increases with the distance from the viewpoint with standard 3D sensors. The PFH descriptor is obtained by binning each feature range with b bins and counting the occurrences for each bin. The final dimension is b^4, or b^3 if distance d is ignored.

3.1.5 Fast Point Feature Histogram

The computation of features for each pair of points makes PFH computationally expensive and not suitable for real-time application. Therefore, Rusu *et al*. introduced *Fast Point Feature Histogram* (FPFH) at **IROS 2009** [47]. FPFH is a simplified variant of PFH and operates in two steps. First a Simplified Point Feature Histogram (SPFH) is constructed using three angular features α, ϕ and θ, for each points within support of and its own neighbors. The computed features are then binned into three separate histograms and concatenated to form the SPFH descriptor for each point. Secondly the FPFH descriptor for $\mathbf{p}$ is obtained by summing up the SPFH descriptors belonging to each point within the support. The sum is weighted by the distance between the query point and a neighbor in the support region as shown in Figure 6. The FPFH descriptor can reduce the computational complexity from $O(nk^2)$ to $O(nk)$, where k is the number of neighboring points.

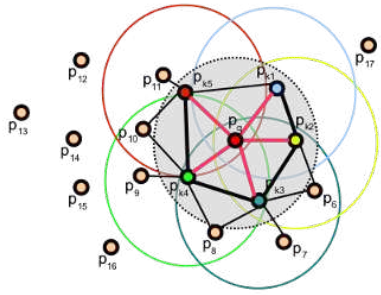


Fig. 6 FPFH overview. For a given feature point p_q, the algorithm first estimates its SPFH values by creating pairs between itself and its neighbors (illustrated using red lines). Then the SPFH values of p_q are weighted using the SPFH values of its p_k neighbors, thus creating the FPFH for p_q. The extra FPFH connections, resultant due to the additional weighting scheme, are shown with black lines. The connections marked with thicker lines will contribute to the FPFH twice. Image from [40].

3.2 Learned local 3D descriptor

The success of deep neural networks for processing images has motivated a data-driven approach to learning features from point clouds. Adaptation of deep learning to point cloud data is, however, far from straightforward. Indeed, the most effective deep neural networks deployed to process images, i.e. convolutional neural networks, are fed with input data arranged into a regular grid structure, but point clouds are inherently unstructured: a point cloud is a set of point coordinates without any obvious orderings between points.

3D deep learning approaches can be classified as:

- **View based methods**: a 3D object is represented as a collection of 2D views. Standard 2D CNNs operate on each single view and the learned features are merged together by a view pooling technique [57]. Moreover, these methods are suitable for application where the input data is a range image [64].
- **Voxel-based methods**: early works were based on 3D voxels, regular 3D grids of occupancy or density of points obtained from point clouds [36, 65, 72, 11, 56]. Although this representation is straightforward to obtain from point clouds, memory occupancy issues force the algorithm to operate with very small resolutions, which in turn introduces artifacts and makes it difficult to learn fine details of geometric structures. To address this limitation, some proposals rely on space partitions methods like k-d trees or octrees [31, 59]. [44] combine view-based and voxel approaches to address 3D shape classification.
- **Point-based methods**: these methods operate directly on unordered point sets. The first pioneering work in this direction was PointNet [43], where a symmetric function is applied to 3D coordinates to achieve invariance to permutation. However, PointNet is not able to capture local information around points. Hence,

the same authors introduced an improved multi-scale architecture, named PointNet++ [45], to exploit geometric features in a local points set.

- **Geometric Deep Learning**: these methods attempt to generalize deep learning model to non-Euclidean domains such as graphs and manifold [8]. The first formulation of neural networks on graphs was proposed in [52]. More recently, it has been extended to operate on non-rigid shape analysis [5, 6]. Finally, [63] proposes a graph convolutions network (GCN) directly on point sets by means of *edge convolution*.

When dealing with feature learning, two loss functions are usually taken into account to preserve in the feature space the proximity of neighboring patches in the Euclidean space: the contrastive loss [22] and the triplet loss [24], which consider pairs and triplets of patches, respectively. Usually triplet loss is more favourable than contrastive loss because takes an *anchor* example and tries to bring *positive* examples closer while also pushing away negative example. However, triplet loss performance heavily depends on the selection of the anchor, negative, and positive examples. Within this context we can think the anchor and the positive as the same 3D point captured from two different point of views, while the negative is from a viewpoint far from both. A graphic example of the difference between contrastive and triplet loss is show in Figure 4 in [14].

3.2.1 3DMatch

One of the first methods that learns a 3D local feature descriptor by means of deep learning algorithms is *3DMatch* [72]. It was introduced by Zeng *et al*. at **CVPR 2017**.

Early work [36, 44] in this field constructed voxel grid in the form of a binary-occupancy grid to represent sparse unstructured 3D data. 3DMatch extends this idea to more informative encoding based on the Truncated Signed Distance Function (TSDF) [62, 37].

In 3DMatch a standard Siamese 3D ConvNet is trained with a contrastive loss function [22] that minimizes the l_2 distance between descriptors generated from matching points, and maximizes l_2 distance between non-corresponding points. The architecture of the network is inspired from AlexNet [33]. An overview of the method is depicted in Figure 2 in [72]. The train set consists of RGB-D images of indoor scenes collected from existing popular datasets [62, 55, 66, 34, 23]. During the train stage, a local region is extracted around a randomly sampled interest point and voxelized with a grid of size $0.01m^3$. Every voxel in the $30 \times 30 \times 30$ grid thus obtained stores a TDF value that indicates the distance between the center of that voxel to the nearest 3D surface. Since TsDF can be computed from meshes, point clouds or depth maps, anyone of these 3D data structures can be used as input to the network. The size of the final descriptor is 512.

3.2.2 PPFNet: Point Pair Feature NETwork

Point Pair Feature NETwork aims to learn a local descriptor by working directly on point coordinates augmented with handcrafted features. It was originally proposed by Deng *et al*. [14] at **CVPR 2018**.
PPFNet combines the permutation invariant characteristic of PointNet [43] together with the high descriptive capacity of Point Pair Features (PPF) [15]. The network is fed by geometries containing information about the local neighborhood of a 3D point such as raw point coordinates, normals and PPFs. As depicted in Figure 2 in [14], PPFNet architecture consists of three parts. A first cluster of mini-PointNets, a concatenation block and a final group of MLPs.

The cluster of mini-PointNets processes N local patches uniformly sampled from a point cloud and extracts local features. Weights and gradients are shared within the cluster. With the aid of a max pooling layer acting on local features, a global context information is created and then concatenated to every local features. Max pool operation encapsulates the distinct local information capturing the global context of the whole point cloud. Finally, a group of MLPs merges the global and local features and creates the learned local descriptor.
PPFNet extends contrastive loss to N-patches by introducing a novel N-tuple loss, it operates on a set of partial scans belonging to the same environment where the ground truth transformation $\mathbf{T}$ between scans is known. The distance between learned features of corresponding patches is minimized acting on two distance matrices: a correspondence matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$, built on the points of the aligned scans, $\mathbf{M} = (m_{ij})$ with

$$m_{ij} = \mathbb{1}(\|\mathbf{x}_i - \mathbf{T}\mathbf{y}_j\|_2 < \tau) \tag{4}$$

and $\mathbb{1}$ is the indicator function; and a feature-space distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$, $\mathbf{D} = (d_{ij})$ with:

$$d_{ij} = \|f(\mathbf{x}_i) - f(\mathbf{y}_j)\|_2. \tag{5}$$

By defining the operator $\sum^*(\cdot)$ as the sum of all the elements in a matrix, the N-tuple loss can be formulated as:

$$L = \sum{}^* \left(\frac{\mathbf{M} \circ \mathbf{D}}{\|\mathbf{M}\|_2^2} + \alpha \frac{max(\theta - (1 - \mathbf{M}) \circ \mathbf{D}, 0)}{N^2 - \|\mathbf{M}\|_2^2} \right) \tag{6}$$

where $\circ$ is the Hadamard product (element-wise multiplication), α is a hyper-parameter balancing the weight between matching and non-matching pairs, and θ is the lower-bound on the expected distance between non-correspondent pairs.

PPFNet is trained using real data scenes from 3DMatch benchmark [72]. The dataset is described in Sec. 5. To explain how training data are sampled for PPFNet, however, it is important to anticipate that each of the 62 different real-words scenes in the dataset contains a variable number of *fragments*, *i.e*.partial views, whose registration reconstructs the full scene.

Rather than randomly detecting keypoints in each fragment, mini-batches in PPFNet are created by randomly extracting the fragments and each point in a fragment acts as a keypoint. To reduce the amount of training data each fragment is down-sampled to 2048 sample points. A radius search of 30 cm around each keypoints is performed to extract a local patch. Then, to increase robustness against point density variations each patch is down-sampled to 1024 points. If a patch contains less than 1024 points, points are randomly repeated. Due to memory limitations, each batch contains 2 fragment pairs with 8192 local patches. Number of combinations for the network at per batch is 2×2048^2.

4 Methods

Within this section, we provide a detailed presentation of some of the most popular methods for 3D local features description. The algorithms presented have been chosen on the basis of their heterogeneity, in order to provide a complete overview to the reader. We present two hand-crafted methods, **SHOT** [51] and **Spin Images** [28], and two methods based on deep learning techniques, **CGF** [29] and **PPF-FoldNet** [13].

4.1 SHOT: Unique Signatures of Histograms for Local Surface Description

The *SHOT* descriptor [51] can be used both for surface description, as originally presented in [60], and for combined shape and texture description [61], if RGB information are available.

The approach is based on the observation that local 3D descriptors can be categorized into *signatures* and *histograms* (Figure 7):

- **signatures**: these descriptors use one or more geometric attributes computed separately at each point within the support to describe the surface. The computed measurements are encoded according to the local coordinates defined by an invariant local reference frame. Signatures are highly descriptive but small errors in the definition of the local reference frame or small perturbations in the encoded trait can substantially modify the final descriptor;
- **histograms**: these descriptors use histograms to capture different characteristics of the surface. A specific domain of quantization (e.g. point coordinates, curvatures, normal angles), is discretized and topological entities (e.g. vertices, mesh triangle areas) are accumulated into each spatial bin. If the descriptor domain is based on coordinates, the definition of a local reference frame is again required to obtain a pose-invariant descriptor. Histogram-like descriptors loses descriptive power due to the quantization error but offers greater robustness to noise.

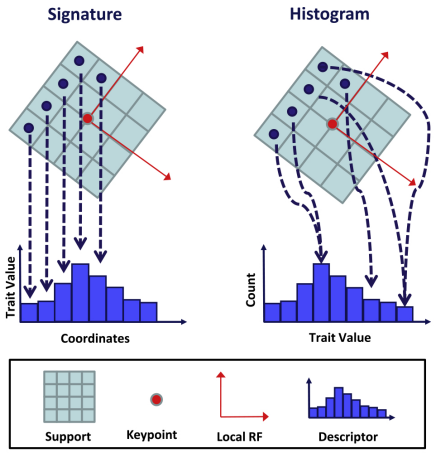


Fig. 7 Signatures and histograms. For the sake of clarity, the figure reports an example in a 2D domain.

Based on this taxonomy, SHOT was proposed to combine the advantages of signature-based methods with those of histogram-based methods. Hence, the name Signature of Histograms of OrienTations (SHOT). This design choice makes SHOT representation operate at a good trade-off point between descriptiveness and robustness to noise.

4.1.1 Local Reference Frame

According to [67], the local reference frame proposed in SHOT belongs to *covariance analysis* methods. The computation of $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ axes rely on the Singular Value Decomposition (SVD) or Eigenvalue Decomposition (EVD) of a covariance matrix of the points, p_i, lying within a spherical support of radius R centered at the feature point $\mathbf{p}$. We define k as the total number of points in the support. The covariance matrix can be obtained using the following equation:

$$\mathbf{C}(\mathbf{p}) = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \hat{\mathbf{p}})(\mathbf{p}_i - \hat{\mathbf{p}})^T \tag{7}$$

where $\hat{\mathbf{p}}$ is the centroid of the points lying within the support:

$$\hat{\mathbf{p}} = \frac{1}{k}\sum_{i=0}^{k}\mathbf{p}_i \tag{8}$$

One of the main disadvantages of this approach is the lack of uniqueness of the signs of the axes of the estimated local reference system. Indeed, as clearly discussed

in [7], although the eigenvectors of Equation 7 define the principal directions of the data, their sign is not defined unambiguously.

The SHOT descriptor proposes a local reference frame estimation that employs a slightly modified covariance matrix. The contributions of the points within support are weighted by their distance from $\mathbf{p}$:

$$\mathbf{C_w}(\mathbf{p}) = \frac{1}{\sum\limits_{i:d_i \leq R} (R - d_i)} \sum_{i:d_i \leq R} (R - d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T \tag{9}$$

with $d_i = \|\mathbf{p}_i - \mathbf{p}\|_2$. By using a weighted covariance matrix the repeatability in cluttered scenes in object recognition scenario is improved. Furthermore, to reduce computational complexity the centroid $\hat{\mathbf{p}}$ in (7) is replaced by the feature point $\mathbf{p}$. Then, similarly to [7], sign ambiguity is addressed by reorienting the sign of each eigenvector of $\mathbf{C_w}(\mathbf{p})$ so that its sign is coherent with the majority of the vectors it is representing. This procedure is applied to both $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$. So, if we refer to the eigenvector corresponding to the largest eigenvalue as the $\mathbf{x}^+$ axis and we denote as $\mathbf{x}^-$ the opposite vector, the sign ambiguity is removed according to:

$$S_x^+ \doteq \left\{ i : d_i \leq R \ \wedge \ (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^+ \geq 0 \right\} \tag{10}$$

$$S_x^- \doteq \left\{ i : d_i \leq R \ \wedge \ (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^- > 0 \right\} \tag{11}$$

$$\mathbf{x} = \begin{cases} \mathbf{x}^+, & |S_x^+| \geq |S_x^-| \\ \mathbf{x}^-, & \text{otherwise} \end{cases} \tag{12}$$

This results in the $\hat{\mathbf{x}}$ axis pointing in the direction of greater sample density. The same procedure is used to disambiguate the $\hat{\mathbf{z}}$ axis, while the third unit vector is computed via the cross-product $\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}}$.

4.1.2 Histogram Computation

SHOT descriptor encodes the histograms of the surface normals at different spatial locations. This choice stems from the related field of 2D descriptors. According to the authors, there are two major reasons behind **SIFT** effectiveness [35], the most successful proposal among 2D descriptors. Firstly, SIFT computes a set of local histograms on specific subsets of pixels defined by a regular grid superimposed on the patch. Secondly, the elements of these local histograms are based on first order derivatives describing the signal of interest, i.e. intensity gradients. Following these considerations, SHOT computes a set of local histograms over the 3D volumes defined by a 3D spherical grid superimposed on the support. As for the signature structure, the spherical grid is partitioned uniformly along the radial, azimuth and elevation axes. The grid is aligned with the axes given by the estimated local reference frame. Each local histogram counts the number of points falling into each

bin according to the cosine of the angle, θ_i, between the normal at each point $\mathbf{n}_i$ and the local $\hat{\mathbf{z}}_k$ axis. The hybrid structure of SHOT is sketched in 8.

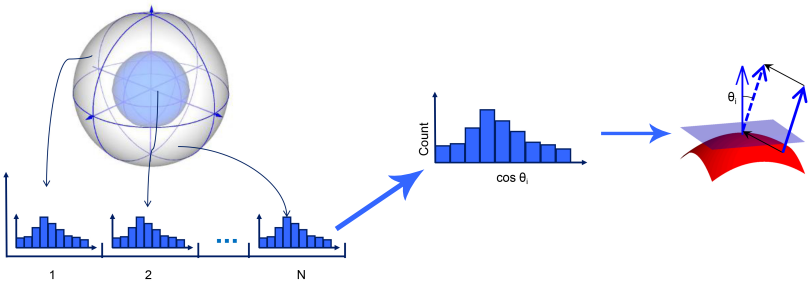


Fig. 8 The hybrid structure of shot. On the left side, the spherical support that acts as a signature, while on the right side the local histograms accumulate bin counters according to $cos\theta_i$.

The choice of cosine is mainly motivated by two aspects. The first is computational efficiency, as it can be computed as $cos\theta_i = \hat{\mathbf{z}}_k \cdot \mathbf{n}_i$. The second one is related to the descriptiveness of the algorithm. With an equally spaced binning on $cos\theta_i$, small differences in orthogonal directions to the normal, i.e. presumably the most informative ones, cause points to be accumulated in different bins leading to different histograms. Moreover, in the presence of quasi-planar regions (i.e. not very descriptive ones), this choice limits histogram differences due to noise by concentrating counts in a smaller number of bins.

Since the SHOT descriptor is generated by appending all the local histograms, the cardinality of the descriptor is related to the number of partitions. The authors indicate that 32 is a proper number of volumes, resulting from 8 azimuth divisions, 2 elevation divisions and 2 radial divisions. While the number of bins for the internal histograms is 11, leading to total descriptor length of 352.

The use of histograms could render the description very sensitive to boundary effects. Furthermore, due to the spatial subdivision of the support, boundary effects may also occur due to perturbations of the local reference frame. A commonly adopted solution is to perform linear interpolation between the point being accumulated into a specific local histogram bin and its neighbors, i.e. the neighboring bin in the local histogram and the bins having the same index in the local histograms corresponding to the neighboring subdivisions of the grid. In SHOT, this results in a quadri-linear interpolation, where each bin is incremented by a weight of $1 - d$ for each dimension. As for the local histogram, d is the distance of the current entry from the central value of the bin. As for elevation and azimuth, d is the angular distance of the entry from the central value of the volume. Along the radial dimension, d is the Euclidean distance of the entry from the central value of the volume. Along each dimension, d is normalized by the distance between two neighbor bins or volumes. Figure 9 illustrates a graphic description of the quadrilinear interpolation process. Finally, the whole descriptor is normalized to have Euclidean norm equal to 1, to increase robustness to point density variations.

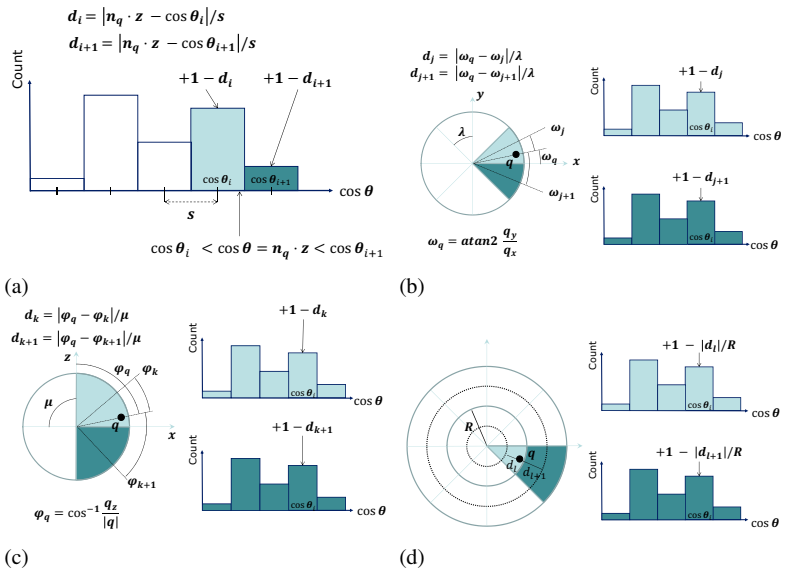


Fig. 9 SHOT quadrilinear interpolation. (a) Interpolation on normal cosines. (b) Interpolation on azimuth. (c) Interpolation on elevation. (d) Interpolation on distance.

4.2 Spin Images

The *Spin Images* is a surface representation technique that was initially introduced in [28]. The name gives an intuitive description about how the algorithm works. The term *image* means that a point is described using a 2D array, while *spin* mimics the process of constructing the image that can be thought as a 2D plane spinning around the normal at the keypoint and collecting counts of points of the support in the entries of the array. The spin image generation process is visualized in Figure 6 in [27].

4.2.1 Oriented Basis

Differently from descriptors based on a local reference frame, Spin Images achieves rotation invariance using a reference axis. The surface normal at each point can be used to compute a 2D oriented basis as shown in Figure 2 in [27]. We define an oriented point O on the surface of an object using the 3D position of the point $\mathbf{p}$ and the surface normal $\mathbf{n}$ at the point. An oriented point allows us to define a partial system of cylindrical coordinates centered at the point. Only two coordinates are used: α is the radial coordinate defined as the perpendicular distance to the line through the surface normal, and β represents the elevation coordinate defined as the signed perpendicular distance to the tangent plane defined by $\mathbf{n}$ at $\mathbf{p}$. The polar angle co-

ordinate is omitted because it cannot be defined robustly and unambiguously using just surface position and normal.

4.2.2 Histogram Computation

Using an oriented point basis, we can define a function called *spin-map* $S_o : R^3 \rightarrow R^2$ that projects a 3D point x to the 2D cylindrical coordinates of a particular basis $(\mathbf{p}, \mathbf{n})$ corresponding to the oriented point O:

$$S_o(x) \rightarrow (\alpha, \beta) = (\sqrt{\|(x-p)\|^2 - (n \cdot (x-p))^2}, n \cdot (x-p)) \tag{13}$$

In order to create a spin image for an oriented point p, the space α-β is then discretized into a 2D array. Then, for each point x_i in the support region the coordinates α and β are computed as in (13) and the bin indexed by (α,β) is incremented. Given a fixed *bin size (b)*, the size of the resultant spin-image (i_{max}, j_{max}) can be calculated as:

$$i_{max} = \frac{2\beta_{max}}{b} + 1 \qquad j_{max} = \frac{\alpha_{max}}{b} + 1 \tag{14}$$

where α_{max} and β_{max} are the maximum α and $|\beta|$ values for all the oriented points within support region. The elevation coordinate β can be both positive and negative, this is the reason why the size of the spin-image in the β direction is twice β_{max}. Finally, the mapping between cylindrical coordinates α, β and the spin image is computed as:

$$i = \lfloor \frac{2\beta_{max} - \beta}{b} \rfloor \qquad j = \lfloor \frac{\alpha}{b} \rfloor \tag{15}$$

The discretization of the α-β domain makes the result of the spin image very sensitive to noise. Therefore, the contribution of the point is bi-linearly interpolated to the four surrounding bins in the 2D array. The creation of a 2D array representation of a spin image can be seen in Figure 4 from [27].

The bilinear weights used to increment the bins in the spin image can be obtained as:

$$a = \alpha - ib \qquad b = \beta - jb \tag{16}$$

Spin images that are created using three different oriented points can be seen in Figure 3 in [27]. Dark pixels correspond to bins with a high number of points.

4.2.3 Parameters

The generation of a spin image is regulated by three parameters: *bin size*, *image width* and *support angle*.

The *bin size* is the width of each of the bin in the spin image. It is an important parameter because it defines the size in memory of the descriptor. As illustrated in Figure 3 in [28], the size in memory of a spin image relates also to the descriptiveness of the spin image: having small sized bins creates descriptive histograms. A common practice is to set the bin size approximately equal to the mesh or cloud resolution, *i.e.*the median distance between vertices/points in the mesh/cloud.

The *image width* defines the number of rows and columns in a square spin image. The product between bin size and image width constitutes a new parameter called *support distance (D_s)*. As show in Figure 4 in [28] this parameter acts on the amount of local versus global information embedded in the descriptor.

The *support angle (A_s)* allows to filter the points that contribute to the calculation of the spin-image and make the spin image less sensitive to the effect of self occlusion and clutter. We can define A_s as the maximum angle between the normal at the point for which we are computing the spin-image and the normal of points that are allowed to contribute to the spin image. Suppose $\mathbf{p}$ is the oriented point for which we are computing the spin-image and p_i is one of its oriented points within the support, the contribution of p_i will be accumulated in the histogram of $\mathbf{p}$ if:

$$acos(n_p \cdot n_{p_i}) < A_s \tag{17}$$

The most appropriate value for this parameter represents the right tradeoff between shape descriptiveness and matching robustness. Decreasing support angle decreases the descriptiveness, while increasing it increases the sensitivity to occlusion and clutter. Typically the support angle is kept small in object recognition and kept large in surface registration.

4.3 CGF: Compact Geometric Features

Compact Geometric Features (CGF) learns a mapping $f : \mathbb{R}^N \rightarrow \mathbb{R}^n$ from high-dimensional handcrafted representations to a very low-dimensional feature space. It was introduced by Khoury *et al*. [29] in **ICCV 2017**.

The authors try to overcome the point cloud representation problems by using an hand-crafted approach to represent the raw local geometry around points. The neural networks performs a dimensionality reduction in order to compute compact embedding. Thanks to its low dimensionality, CGF enables faster nearest-neighbor queries during the matching stage.

The network is trained using the triplet embedding loss [24]. In this regard, CGF is coupled with an effective negative sampling strategy which produces a highly discriminative embedding, described in Sec. 4.3.3.

4.3.1 Point cloud parametrization

Before CGF, the standard way to adapt point clouds to deep learning algorithms was to discretize the input into a uniform voxel grid. However, such representation is not efficient due to the high number of empty cells [26].

CGF captures the local geometry around each point into a handcrafted feature called spherical histogram. Inspired by 3DSC [16], a spherical histogram encodes the distribution of points in a local neighborhood with a non uniform binned radial grid. A simplified view is shown in Figure 2 from [29].

Given a feature point $\mathbf{p}$ a sphere $\mathbf{S}$ of radius r is centered at $\mathbf{p}$. To obtain rotational invariance, the local neighborhood is aligned to axes of a local reference frame computed as in 4.1.1. Considering the $\hat{\mathbf{z}}$ axis of the estimated lrf and the normal $\mathbf{n}$ at point $\mathbf{p}$, if the dot product $\langle \mathbf{n}, \hat{\mathbf{z}} \rangle < 0$, the signs of all three vectors in the local reference frame is flipped.

The volume bounded by $\mathbf{S}$ is divided into bins along the radial, elevation, and azimuth directions. The azimuth direction is split into $A = 12$ bins, each of extent $2\pi/A$. The elevation direction is subdivided into $E = 11$ bins, each of extent π/E. The radial direction, which has total span r, is logarithmically subdivided into 17 bins using the following thresholds:

$$r_i = \exp\left(\ln r_{\min} + \frac{i}{R} \ln\left(\frac{r}{r_{\min}}\right)\right). \tag{18}$$

Subdividing the radial direction in this fashion makes the histogram more robust to changes in shape near the center $\mathbf{p}$. The resulting spherical histogram has a size of 2,244 bins.

The value inside each bin reflects the point density of the local neighborhood around $\mathbf{p}$. Let $\mathcal{N} \subset \mathcal{P}$ be the set of neighboring points that lie inside the sphere $\mathbf{S}$. Each point $\mathbf{q} \in N$, is converted from euclidean to spherical coordinates and the corresponding bin that contains $\mathbf{q}$ is incremented. The final histogram is normalized by dividing each bin by $|\mathcal{N}|$.

4.3.2 Network architecture

CGF deep network maps supports from the high-dimensional space of spherical histograms to a very low-dimensional Euclidean space. The architecture is a fully-connected network with 5 hidden layers. Each hidden layer contains 512 nodes and is followed by a ReLU non-linearity. Weights are initialized from a normal distribution with mean 0 and standard deviation 0.1. The mini-batch size is 512 and Adam [30] is used as optimizer. The dimension of the learned embedding which is then used as descriptor is 32.

As far as the loss function is concerned, a standard triplet loss [24] tries to keep similar features together while pushing dissimilar features apart. Starting from a mini-batch of triplets of input histograms $\mathcal{T} = \{(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)\}_i$. Vector $\mathbf{x}_i^a$ is re-

ferred to as the anchor of triplet i, vector $\mathbf{x}_i^p$ is a positive example and vector $\mathbf{x}_i^n$ is a negative example. Given such a set of triplets, triplet loss is written as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \big[\|f(\mathbf{x}_i^a;\boldsymbol{\theta}) - f(\mathbf{x}_i^p;\boldsymbol{\theta})\|^2 - \|f(\mathbf{x}_i^a;\boldsymbol{\theta}) - f(\mathbf{x}_i^n;\boldsymbol{\theta})\|^2 + 1 \big]_+, \quad (19)$$

where $\boldsymbol{\theta}$ are the learned parameters and $[\cdot]_+$ denotes $\max(\cdot, 0)$.

4.3.3 Train data

Triplet loss is often used to learn discriminative features. During training, a triplet of input histograms, $\mathcal{T} = \{(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)\}_i$ is fed into the model as a single sample. The idea behind this is that distance between the learned embedding of anchor and positive should be smaller than that between anchor and negative embedding. In order to achieve this, it is essential to samples negatives accurately. A common choice is to draw as negative, random points that are pretty far from the anchor.

Khoury *et al*. use a smarter strategy. Given a set of point clouds $\{\mathcal{P}_i\}_i$ that describe overlapping scans of a 3D object or scene, let $\{\mathcal{T}_i\}_i$ be a set of rigid transformations that align the point clouds $\{\mathcal{P}_i\}_i$ in a common coordinate frame. Consider the set $\mathcal{O}$ of overlapping pairs of point clouds from $\{\mathcal{P}_i\}_i$. Each point $\mathbf{p} \in \mathcal{P}_i$ in each pair $(\mathcal{P}_i, \mathcal{P}_j) \in \mathcal{O}$ generates 40 triplets. These 40 triplets use as anchor point $\mathbf{p} \in \mathcal{P}_i$, while positives are grabbed from the local neighborhood of $\mathbf{p}$ in $\mathbf{T}_j\mathcal{P}_j$. With that in mind, we can denote as $\mathcal{N}^j_{\mathbf{p},\tau}$ in $\mathcal{P}_j$ the set of neighbors points that are at distance at most τ from $\mathbf{p}$. In addition consider $\mathcal{N}^j_{\mathbf{p},2\tau}$, the set of points in $\mathcal{P}_j$ that are at distance at most 2τ from $\mathbf{p}$. While the points in $\mathcal{N}^j_{\mathbf{p},\tau}$ are good correspondences for $\mathbf{p}$ in $\mathcal{P}_j$, the set $\mathcal{N}^j_{\mathbf{p},2\tau} \setminus \mathcal{N}^j_{\mathbf{p},\tau}$ contains difficult negative examples for $\mathbf{p}$. This is visualized in Figure 3 in [29]. They have similar local geometries of $\mathbf{p}$, but are far enough from it. Hence, 15 triplets are constructed by sampling negatives from $\mathcal{N}_{\mathbf{p},2\tau}$ and 25 are constructed by randomly sampling negatives from other scans.

4.4 PPF-FoldNet

PPF-FoldNet [13] is an extension of another descriptor [14], and it has been presented by Deng *et al*. at **CVPR 2018**. The main limitations of learned 3D descriptors are the big amount of labeled data required, sensitivity to rotations and the use of hand crafted input preparation. PPF-FoldNet improved upon these limitation by proposing an unsupervised rotation invariant local descriptor. *FoldingNet* [69] introduced the idea of deforming a 2D grid to decode a 3D surface as a point set, given a latent codeword encoding the 3D surface, and offered an interesting paradigm for

unsupervised learning with point clouds. Similarly, PPF-FoldNet uses folding operation to reconstruct the point pair feature [15, 4] of a support.

4.4.1 Point cloud parametrization

The local patch of a feature point $\mathbf{p}$ is represented by a collection of pair features, computed between $\mathbf{p}$ and the neighboring points:

$$\mathbf{F_\Omega} = \{\, \mathbf{f}(\mathbf{p}, \mathbf{p}_1) \cdots \mathbf{f}(\mathbf{p}, \mathbf{p}_i) \cdots \mathbf{f}(\mathbf{p}, \mathbf{p}_N) \,\} \in \mathbb{R}^{4 \times N-1} \tag{20}$$

As far as Point Pair Features (PPFs) are concerned, for two points $\mathbf{x_1}$ and $\mathbf{x_2}$ the features are defined as:

$$\boldsymbol{\psi}_{12} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)) \tag{21}$$

where $\mathbf{d}$ denotes the difference vector between points, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the surface normals at $\mathbf{x}_1$ and $\mathbf{x}_2$, $\|\cdot\|$ is the Euclidean distance and $\angle$ is the angle operator computed in a numerically robust manner as in [4]:

$$\angle(\mathbf{v}_1, \mathbf{v}_2) = \operatorname{atan2}\left(\|\mathbf{v}_1 \times \mathbf{v}_2\|\,,\, \mathbf{v}_1 \cdot \mathbf{v}_2\right) \tag{22}$$

$\angle(\mathbf{v}_1, \mathbf{v}_2)$ is guaranteed to lie in the range $[0, \pi)$.

The main motivation behind the use of PPFs is their invariance to rotations. Rotation invariance is an essential trait for every local descriptor and the PPFs are invariants under Euclidean isometry as distances and angles are preserved between every pair of points. A visualization of some local patches and their correspondent PPF signatures is shown in Figure 2 in [13].

4.4.2 Network architecture

PFF-FoldNet leverages on an encoder-decoder architecture, shown in Figure 1 in [13], to reconstructs a set of PPFs computed on a local patch around a given feature point $\mathbf{p}$. The encoder is composed by a three-layer, point-wise Multi Layer Perceptron (MLP) followed by a max-pooling layer that aggregates individual point features into a global one similar to [43, 69]. After the concatenation with skip-links, a two-layer MLP compress these features to the encoded codeword.

The decoder takes inspiration from FoldingNet [69], and try to deform a low-dimensional grid structure using the information encoded in the codeword learned by the encoder. Differently from [69], the decoder uses a deeper architecture, each *folding* operation rely on a 5-layer MLP. This choice is mainly due to higher dimensional of the reconstructed set, 4D vs 3D. The loss involved is the *Chamfer* distance between PPFs:

$$d(\mathbf{F}, \hat{\mathbf{F}}) = \max\left\{ \frac{1}{|\mathbf{F}|} \sum_{\mathbf{f} \in \mathbf{F}} \min_{\hat{\mathbf{f}} \in \hat{\mathbf{F}}} \|\mathbf{f} - \hat{\mathbf{f}}\|_2, \ \frac{1}{|\hat{\mathbf{F}}|} \sum_{\mathbf{f} \in \hat{\mathbf{F}}} \min_{\mathbf{f} \in \mathbf{F}} \|\mathbf{f} - \hat{\mathbf{f}}\|_2 \right\} \tag{23}$$

where F is computed as in 20 and the $\hat{}$ operator refers to the reconstructed set. The learned latent dimensional vector is called *codeword* and used as local descriptor of the underlying geometry around which the patch is extracted. The weights of the network are initialized using Xavier initialization [17]. The loss is minimized by ADAM Optimizer [30]. Batch size is 32, while the size of the final descriptor is 512.

4.4.3 Train data

Thanks to the unsupervised approach, the network is trained by random sampling points from the fragments of the 3DMatch dataset. For each keypoint the PPFs are computed using its neighboring points in a 30 cm vicinity. Since the number of points in a local patch is not fixed, each local patch is downsampled to an arbitrary number of points, thereby facilitating the train by organizing the data into regular batches, and increasing the robustness to noises and different point densities.

5 Dataset and Evaluation

Within this section we will validate some of the proposed algorithms in sections 3 and 4 using the popular 3D Match Benchmark Dataset [72]. This dataset is a collection of different real-world scenes retrieved from a pool of previous published ones such as Analysis-by-Synthesis [62], 7-Scenes [55], SUN3D [66], RGB-D Scenes v.2 [34] and Halber and Funkhouser [23]. The dataset includes indoor scenes like living rooms, offices, bedrooms, tabletops, and restrooms. Each scene is equipped by one or more RGB-D video sequences captured by real sensors, like Microsoft Kinect and Intel RealSense, together with camera intrinsic parameters. In order to be robust to noise point clouds for the fragments belonging to a single scene are generated by fusing 50 consecutive depth frames. The overall number of scenes in the dataset is 62, splitted in 54 for train and validation and 8 for benchmarking.

As for the selected descriptors, we have chosen: FPFH [47], Spin Image [28], SHOT [51], USC [60], 3DMatch [72], CGF [29], PPFNet [14], PPFFoldNet [13]. The main reasons behind the choice of hand-crafted descriptors is to compare features based on point density, such as Spin Images and USC, and algorithms that operate on geometric properties such as FPFH and SHOT. On the other hand, for the learned ones we rely on the most recently proposals.

To test the rotation invariance of 3D feature descriptors, in [13] a rotated version of the 3D Match Benchmark Dataset is proposed. The fragments in the test split are rotated around random sampled axis with a random chosen angle. We refer to it as *Rotated 3D Match Benchmark*.

A good way to check the performance of a given descriptor is to use it within a registration pipeline. Given a set of K fragments available for each scene in the dataset, the aim is to find the rigid transformation matrix, $\mathbf{G}$, that aligns every pair of overlapping fragments. Each scene in the dataset comes with a list of overlapping fragments and the rigid transformations $\mathbf{T}_{ij} \in SE(3)$ between them. In order to check the performance of a given 3D local feature descriptor, the following steps are performed. For all the fragment pairs (V_i, V_j) with overlap at least 30 %, a list of correspondences between all the keypoints detected in V_i and all the keypoints detected in V_j is formed by finding all the pairs that lie mutually close in the feature space by applying nearest neighbor search [13]. The set containing all the correspondences is called $\mathbf{C}_{i,j}$. Then, given a pair of matched keypoints (k_i, k_j), $k_i \in V_i, k_j \in V_j$, the set of correct correspondences, $\mathbf{C}_{gnd_{i,j}}$, can be identified based on the available ground truth transformations by checking whether the matched keypoints lay within a certain distance τ_1 in the canonical reference frame:

$$\mathbf{C}_{gnd_{i,j}} = \{(k_i, k_j) : \|k_i - \mathbf{T}_{ij} k_j\| \leq \tau_1\} \tag{24}$$

We can define the inlier ratio for $\mathbf{C}_{i,j}$ as the percentage of true matches in $\mathbf{C}_{i,j}$, $r_{in} = |\mathbf{C}_{gnd_{i,j}}|/|\mathbf{C}_{i,j}|$. Instead of using the estimated correspondences within a RANSAC pipeline and check the resulting rigid transformation, a fragment pair is considered correctly aligned if $r_{in} > \tau_2$.

Given a set of fragment pairs $\mathbf{S} = \{(\mathbf{P}, \mathbf{Q})\}$ that are used in the evaluation, the quality of a given feature descriptor is measured by the recall R of fragment pairs matched in $\mathbf{S}$:

$$R = \frac{1}{|\mathbf{S}|} \sum_{i=1}^{|\mathbf{S}|} \mathbb{1} \left(r_{in}\big(\mathbf{S}_i = (\mathbf{P}_i, \mathbf{Q}_i)\big) > \tau_2 \right) \tag{25}$$

On both 3D Match Benchmark and Rotated 3D Match Benchmark, τ_1 and τ_2 are set to 10 cm and 5%, respectively. For each fragment, descriptors are computed on 5000 randomly sampled feature points, provided by [72]. The radius support used to compute descriptor is 18 cm and for normal estimation is 9 cm. For the handcrafted descriptors we used the public implementation in **PCL**[50], while for the learned descriptors since the implementations are not public, the results were taken from [13].

6 Results

The experimental results on the standard 3D Match Benchmark are shown in table 1. From the results it is clear how the descriptors based on geometric attribute histograms tend to outperform those based on spatial distribution histograms. Indeed, SHOT, PPFFoldNet build up their representations leveraging properties of the surface like normals. Except for SHOT and USC, PPFNet and PPFFoldNet man-

Table 1 Results on the 3DMatch benchmark. Test data comes from SUN3D [66], except for *Red Kitchen* which is taken from 7-scenes [55].

	FPFH [47]	Spin Image [28]	SHOT [51]	USC [60]	3DMatch [72]	CGF [29]	PPFNet [14]	PPFFoldNet [13]
Kitchen	0.4802	0.5079	0.7470	0.7016	0.5830	0.6030	0.8972	0.7866
Home 1	0.7115	0.6987	0.8141	0.7756	0.7240	0.7110	0.5577	0.7628
Home 2	0.5769	0.6010	0.7548	0.6683	0.6150	0.5670	0.5913	0.6154
Hotel 1	0.6770	0.5973	0.7920	0.7788	0.5490	0.5710	0.5796	0.6814
Hotel 2	0.5673	0.5769	0.7692	0.6635	0.4810	0.5380	0.5796	0.7115
Hotel 3	0.7407	0.7222	0.8704	0.8148	0.6110	0.8330	0.6111	0.9444
Study	0.4486	0.4247	0.6918	0.5856	0.5170	0.3770	0.5342	0.6199
MIT Lab	0.3896	0.3506	0.5974	0.4675	0.5070	0.4550	0.6364	0.6234
Average	0.5740	0.5599	0.7546	0.6820	0.5730	0.5820	0.6231	0.7182

Table 2 Results on the rotated 3DMatch benchmark. Test data comes from SUN3D [66], except for *Red Kitchen* which is taken from 7-scenes [55].

	FPFH [47]	Spin Image [28]	SHOT [51]	USC [60]	3DMatch [72]	CGF [29]	PPFNet [14]	PPFFoldNet [13]
Kitchen	0.4921	0.5375	0.7470	0.7095	0.0240	0.6050	0.002	0.7885
Home 1	0.6987	0.7308	0.8141	0.8269	0.0380	0.7120	0.0000	0.7821
Home 2	0.5913	0.6250	0.7404	0.6490	0.0530	0.5720	0.0144	0.6442
Hotel 1	0.6903	0.6416	0.8053	0.7788	0.0180	0.5720	0.0044	0.6770
Hotel 2	0.5673	0.5673	0.7788	0.7115	0.0670	0.5380	0.0000	0.6923
Hotel 3	0.7407	0.7963	0.8519	0.8519	0.0190	0.8330	0.0000	0.963
Study	0.4555	0.4589	0.6952	0.6404	0.0270	0.3870	0.0000	0.6267
MIT Lab	0.4286	0.4675	0.6364	0.5714	0.0390	0.4550	0.0000	0.6753
Average	0.5831	0.6031	0.7586	0.7174	0.0360	0.5850	0.0026	0.7311

aged to match a higher number of fragment pairs compared to the other methods. This is a clear demonstration of how learning the descriptor using a stable and rich representation rather than using hand-crafted techniques allows achieving higher performances.

Results on the rotated benchmark are illustrated in Table 2. They show how a robust local reference frame is an essential aspect for any descriptor. Comparing the results from 1 and 2, FPFH, Spin Images, SHOT, USC and CGF demonstrate robustness to rotations. Their performance remains stable despite the introduction of random rotations in the data. The significant drop in performance achieved by 3DMatch and PPFNet also demonstrates how these descriptors hardly learn pose invariance directly from data. The best performance is obtained by PPFFoldNet and SHOT: this proves how the use of rotation-invariant features, such as Point Pair Features, or a stable local reference frame, can lead to a robust and descriptive representation.

7 Open Challenges

Within this chapter we have outlined how over the past few years there has been an intensive research activity in learning features for 3D data. Nevertheless, many challenges are still open. Although the use of deep learning algorithms has boosted the performance of local 3D descriptors, learning a rotation-invariant descriptor directly from 3D data such as point clouds remains an open problem. Indeed, most state-of-

the-art methods achieve rotation invariance by relying on hand-crafted features that are by design invariant to rotation. Moreover, many deep learning approaches still use handcrafted methods to operate on 3D data.

Additional open problems that 3D local descriptors will need to face in forthcoming years are the robustness to geometric transformation and noise. This is particularly relevant with learned-based approaches which suffer from the well-known domain shift problem when tested under unseen working conditions and data domains. Scalability towards large datasets (e.g. large scenes, many objects) and in general computational efficiency is also an important issue, especially considering the higher and higher relevance that mobile applications (such as, e.g., robotics, autonomous driving, augmented reality) are assuming in the context of 3D data processing.

8 Further Reading

We have presented an overview of methods proposed over the last two decades to describe the local properties of a 3D surface and successfully match it under rigid transformations and in the presence of noise, clutter, and occlusions.

Interestingly, the successful research avenue aiming at adapting neural network architectures such as CNNs and auto-encoders to process 3D data - and in particular point clouds - has blurred the line between the standard distinction between global and local 3D descriptors, since such architectures could be employed indifferently under both fashions and for most 3D processing tasks. To name a few methods, we refer the reader to the recent point cloud-based auto-encoders proposed in AtlasNet [18] and FoldingNet [69], or recent multi-resolution 3D descriptors for point clouds such as PointNet++ [45] and Fully-convolutional Point Networks [46]. All these methods have been, or could be, used for both global and local description tasks.

Although this chapter focuses on local features for 3D data, it is interesting to analyze how also in the field of images there has been a parallel line of research. One of the most remarkable works is represented by Lift [70]. For a more detailed discussion of learned 2D features we refer the reader to the following survey [53].

9 Hands-on 3D Descriptors

As a supplement to this chapter we provide some code snippets that show how to calculate some of the 3D descriptors presented in this chapter. The following examples use the open source implementations available in the Point Cloud Library (PCL) [50]. In particular, the descriptors currently available are: SHOT, FPFH, Spin Images and USC.

In the following snippets 1 2, we provide the code to compute two of the descriptors seen in this chapter, SHOT and USC respectively. As an exercise, we leave

to the reader the extension of the presented snippets to the other aforementioned descriptors available in PCL.

```
#include <pcl/io/pcd_io.h>
#include <pcl/features/normal_3d.h>
#include <pcl/features/shot.h>

using namespace pcl;
using namespace std;

typedef PointXYZ PointT;
typedef Normal NormalT;
typedef SHOT352 ShotT;

int main(int argc, char** argv)
{
    /*Create point cloud for coordinates*/
    PointCloud<PointT>::Ptr cloud(new PointCloud<PointT>);
    /*Create point cloud for normals*/
    PointCloud<NormalT>::Ptr normals(new PointCloud<NormalT>);
    /*Create point cloud for descriptors */
    PointCloud<ShotT>::Ptr descriptors(new PointCloud<ShotT>());

    /*Read a point cloud from the disk, exit in case of failure*/
    if (io::loadPCDFile<PointT>(argv[1], *cloud) != 0)
        return -1;

    /*Estimate the normals*/
    const int number_of_neighbours_for_normals = 17;

    NormalEstimation<PointT, NormalT> estimator_normals;

    estimator_normals.setInputCloud(cloud);
    estimator_normals.setKSearch(number_of_neighbours_for_normals
    );

    estimator_normals.compute(*normals);

    /*Estimate shot descriptor*/
    SHOTEstimation<PointT, NormalT, ShotT> shot;
    shot.setInputCloud(cloud);
    shot.setInputNormals(normals);

    const double radius_shot = 0.18;
    shot.setRadiusSearch(radius_shot);

    /*Set the radius to compute the Local Reference Frame*/
    shot.setLRFRadius(radius_shot);

    shot.compute(*descriptors);

    /*Print descriptors*/
    for(ShotT shot : (*descriptors))
    {
```

```
        for (int i = 0; i < shot.descriptorSize(); i++)
            cout << shot.descriptor[i] << endl;
    }
}
```

Listing 1 Compute SHOT descriptor

```
#include <pcl/io/pcd_io.h>
#include <pcl/features/usc.h>

using namespace pcl;
using namespace std;

typedef PointXYZ PointT;
typedef Normal NormalT;
typedef UniqueShapeContext1960 UscT;
typedef ReferenceFrame FrameT;

int main(int argc, char** argv)
{
    /*Create point cloud for coordinates*/
    PointCloud<PointT>::Ptr cloud(new PointCloud<PointT>);
    /*Create point cloud for descriptors */
    PointCloud<UscT>::Ptr descriptors(new PointCloud<UscT>());

    /*Read a point cloud from the disk, exit in case of failure*/
    if (io::loadPCDFile<PointT>(argv[1], *cloud) != 0)
        return -1;

     /*Estimate USC descriptor*/
    UniqueShapeContext<PointT, UscT, FrameT> usc;
    usc.setInputCloud(cloud);

    const double radius_usc = 0.18;
    usc.setRadiusSearch(radius_usc);

    /*Set the radius to compute the Local Reference Frame*/
    usc.setLocalRadius(radius_usc);

    /*The minimal radius value for the search sphere*/
    const double radius_minimal = radius_usc / 10.0;
    usc.setMinimalRadius(radius_minimal);

    /*Radius used to compute the local point density for the
    neighbors*/
    const double radius_density = radius_usc / 5.0;
    usc.setPointDensityRadius(radius_density);

    usc.compute(*descriptors);

    /*Print descriptors*/
    for(UscT usc : (*descriptors))
    {
        for (int i = 0; i < usc.descriptorSize(); i++)
```

```
47                cout << usc.descriptor[i] << endl;
48        }
49    }
```

Listing 2 Compute USC descriptor

References

1. Aitor Aldoma, Federico Tombari, Radu Bogdan Rusu, and Markus Vincze. Our-cvfh–oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, pages 113–122. Springer, 2012.
2. Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in neural information processing systems*, pages 831–837, 2001.
3. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
4. Tolga Birdal and Slobodan Ilic. Point pair features based object detection and pose estimation revisited. In *3D Vision (3DV), 2015 International Conference on*, pages 527–535. IEEE, 2015.
5. Davide Boscaini, Jonathan Masci, Simone Melzi, Michael M Bronstein, Umberto Castellani, and Pierre Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, pages 13–23. Wiley Online Library, 2015.
6. Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016.
7. Rasmus Bro, Evrim Acar, and Tamara G Kolda. Resolving the sign ambiguity in the singular value decomposition. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 22(2):135–140, 2008.
8. Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
9. Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28(10):1252–1262, 2007.
10. Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.
11. Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 1, 2017.
12. Marcello Demi, Marco Paterni, and Antonio Benassi. The first absolute central moment in low-level image processing. *Computer Vision and Image Understanding*, 80(1):57–87, 2000.
13. Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. *arXiv preprint arXiv:1808.10322*, 2, 2018.
14. Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. *Computer Vision and Pattern Recognition (CVPR). IEEE*, 1, 2018.
15. Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005. Ieee, 2010.
16. Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *European conference on computer vision*, pages 224–237. Springer, 2004.

17. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.
18. Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018.
19. Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, 116(1):66–89, 2016.
20. Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105(1):63–86, 2013.
21. Yulan Guo, Ferdous Ahmed Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. In *GRAPP/IVAPP*, pages 86–93, 2013.
22. Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006.
23. Maciej Halber and Thomas Funkhouser. Structured global registration of rgb-d scans in indoor environments. arxiv preprint. *arXiv preprint arXiv:1607.08539*, 2(3):9, 2016.
24. Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.
25. Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.
26. Takayuki Itoh and Koji Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, 1995.
27. A Johnson and M Hebert. Surface matching for object recognition in complex 3-d scenes. *Image and Vision Computing*, 1998.
28. AE Johnson and M Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, 1999.
29. Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 153–161. IEEE, 2017.
30. D Kinga and J Ba Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, volume 5, 2015.
31. Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 863–872. IEEE, 2017.
32. Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *European Conference on Computer Vision*, pages 589–602. Springer, 2010.
33. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
34. Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014.
35. David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.
36. Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

37. Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136. IEEE, 2011.
38. John Novatnack and Ko Nishino. Scale-dependent/invariant local 3d shape descriptors for fully automatic registration of multiple sets of range images. In *European conference on computer vision*, pages 440–453. Springer, 2008.
39. Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.
40. PCL, the point cloud library. Fast point feature histograms (fpfh) descriptors, 2013. [Online; accessed March 16, 2020].
41. PCL, the point cloud library. Point feature histograms (pfh) descriptors, 2013. [Online; accessed March 16, 2020].
42. Alioscia Petrelli and Luigi Di Stefano. On the repeatability of the local reference frame for partial shape matching. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2244–2251. IEEE, 2011.
43. Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.
44. Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.
45. Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.
46. Dario Rethage, Johanna Wald, Jürgen Sturm, Nassir Navab, and Federico Tombari. Fully-convolutional point networks for large-scale point clouds. In *European Conference on Computer Vision*, pages 625–640. Springer, 2018.
47. Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.
48. Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3384–3391. IEEE, 2008.
49. Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162. IEEE, 2010.
50. Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.
51. Samuele Salti, Federico Tombari, and Luigi Di Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125:251–264, 2014.
52. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.
53. Johannes L Schonberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1482–1491, 2017.
54. CE Shannon. A mathematical theory of communication, bell system technical journal, vol. *27(1948)*, 1948.
55. Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2930–2937, 2013.

56. Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 190–198. IEEE, 2017.
57. Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.
58. Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.
59. Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, volume 2, page 8, 2017.
60. Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.
61. Federico Tombari, Samuele Salti, and Luigi Di Stefano. A combined texture-shape descriptor for enhanced 3d feature matching. In *2011 18th IEEE International Conference on Image Processing*, pages 809–812. IEEE, 2011.
62. Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 323–332. IEEE, 2016.
63. Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.
64. Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1553, 2016.
65. Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.
66. Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.
67. Jiaqi Yang, Yang Xiao, and Zhiguo Cao. Toward the repeatability and robustness of the local reference frame for 3d shape matching: An evaluation. *IEEE Transactions on Image Processing*, 27(8):3766–3781, 2018.
68. Jiaqi Yang, Qian Zhang, Yang Xiao, and Zhiguo Cao. Toldi: An effective and robust approach for 3d local shape description. *Pattern Recognition*, 65:175–187, 2017.
69. Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018.
70. Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.
71. Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Keypoints and local descriptors of scalar functions on 2d manifolds. *International Journal of Computer Vision*, 100(1):78–98, 2012.
72. Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 199–208. IEEE, 2017.
73. Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. IEEE, 2009.