ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in

COMPUTER SCIENCE AND ENGINEERING

Ciclo 32°

Settore Concorsuale di afferenza: 09/H1

Settore Scientifico-Disciplinare: ING-INF/05

LEARNING TO UNDERSTAND THE WORLD IN 3D

Presentata da: Riccardo Spezialetti

Coordinatore dottorato

Chiar.mo Prof.
Davide Sangiorgi

Relatore

Chiar.mo Prof.
Luigi Di Stefano

Esame finale anno 2020

Riccardo Spezialetti: *Learning to understand the world in 3D,* Dottorato di ricerca in Computer Science and Engineering, © 2020

SUPERVISOR:
Luigi Di Stefano

LOCATION:
Bologna, Italy

To the CVLab family

ABSTRACT

3D Computer vision is a research topic gathering even increasing attention thanks to the more and more widespread availability of off-the-shelf depth sensors and large-scale 3D datasets. The main purpose of 3D computer vision is to understand the geometry of the objects in order to interact with them. Recently, the success of deep neural networks for processing images has fostered a data driven approach to solve 3D vision and graphics problems. Inspired by the potential of this field, in this thesis we will address two main problems: (a) how to leverage machine/deep learning techniques to build a robust and effective pipeline to establish correspondences between surfaces, and (b) how to obtain a reliable 3D reconstruction of an object using RGB images sparsely acquired from different point of views by means of deep neural networks.

At the heart of many 3D computer vision applications, such as *3D object recognition*, *surface registration*, and *SLAM*, lies surface matching, an effective paradigm aimed at finding correspondences between points belonging to different shapes. To this end, it is essential to first identify the characteristic points of an object and then create an adequate representation of them. We will refer to these two steps as *keypoint detection* and *keypoint description*, respectively. A variety of algorithms for keypoint detection and description exists in the scientific literature, and the majority of them are based on different ways to exploit and encode the geometric properties of a given surface. Despite the huge effort made, surface matching still represents an open problem because of challenging nuissances occurring in real world environments, such as sensor noise, change of view point, clutter and occlusions. As a first contribution (a) of this Ph.D thesis, we will propose data driven solutions to tackle the problems of keypoint detection and description. To validate our proposals, we will address two main tasks such as 3D object recognition and surface registration.

As a further interesting direction of research, we investigate the problem of 3D object reconstruction from RGB data only (b). If in the past this application has been addressed by SLAM and *Structure from motion* (SfM) techniques, this radically changed in recent years thanks to the dawn of deep learning. Indeed, learning-based solutions have shown promising results in

challenging condition, where SLAM and SfM generate bad reconstructions, *e.g.* when input images exhibit large baselines. Following this trend, we will introduce a novel approach that combines traditional computer vision techniques, like visual hull and voxel carving, with deep learning to perform a view point variant 3D object reconstruction from non-overlapping RGB views.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Luigi Di Stefano, for all the trust, the precious advice, and for always believing in me, even in the most critical moments. Reaching this target would not have been possible without his baggage of enthusiasm and his guidance.

I would also like to thank Federico Tombari for always being there and for the endless opportunities for growth, one above all: offering me the possibility to spend six month as Visiting Researcher at Google Zurich, a beautiful period that I consider essential for my growth as a researcher. My thanks extend to Samuele Salti, his friendship and scientific advice in the years was invaluable.

A special thanks goes to the members of Computer Vision Laboratory in Bologna: Alessio Tonioni, Alioscia Petrelli, Matteo Poggi, Fabio Tosi, Filippo Aleotti, Stefano Mattoccia, Daniele De Gregorio, Pierluigi Zama Ramirez, Gianluca Berardi, Tommaso Cavallari, Marlon Marcon, Paolo Galeone. Our insightful talks and discussions about scientific and non-scientific related topics have made the past years unforgettable. Another thanks goes to David Joseph Tan for the patience and the shouts while he mentored me.

Finally, I would like to thank my family and friends (it is impossible to list them all) for always supporting me during all my Ph.D. years.

Last but not least thanks goes to my girlfriend Elisa for bringing back color to my life.

CONTENTS

1

INTRODUCTION

We live in a three-dimensional world and a proper cognitive understanding of the 3D structures of the objects and the environments surrounding us is crucial to accomplish our daily tasks. Lifting this knowledge from *humans* to machines is a fundamental challenge to face when operating in many robotic and computer vision contexts such as grasping, navigation, augmented reality, 3D object recognition, surface registration and shape classification, to mention a few. In this thesis, we will focus on two particular tasks, 3D object recognition and surface registration. The goal of the former is to identify an object from a gallery of models within a scene in order to estimate its 6D pose. Differently, surface registration aims at recreating the shape and appearance of objects by registering a set of partial views. We will address these two applications considering as representation for 3D data point clouds or meshes. As a further contribution, we will also explore the feasibility of reconstructing 3D objects from RGB images.

The core component of many 3D computer vision applications is the estimation of similarity between surfaces, typically tackled by establishing correspondences. Surface matching methods analyze the geometry surrounding each point in the surface to extrapolate the most distinctive properties. These properties are called *features* and can be seen as a compact but rich representation of 3D data designed to be robust to the set of nuisances that can affect 3D data like clutter, view-point variations, self-occlusions, *etc*. Once features are computed, they are matched across surfaces to build point-to-point correspondences, this paradigm being referred to as *feature-based matching*. An illustration is shown in Fig. 1.1.

A *feature-matching pipeline* consists of three stages:

- **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. The aim is to look for geometric properties that characterize the given object. These points are called *keypoints*.

- **feature-description:** a *descriptor* is a compact representation of geometric properties of a point. Descriptors are usually 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 verifies that similar descriptors belong to the same physical 3D point. Several metrics can be used, the most popular being the Euclidean distance in the descriptors space.

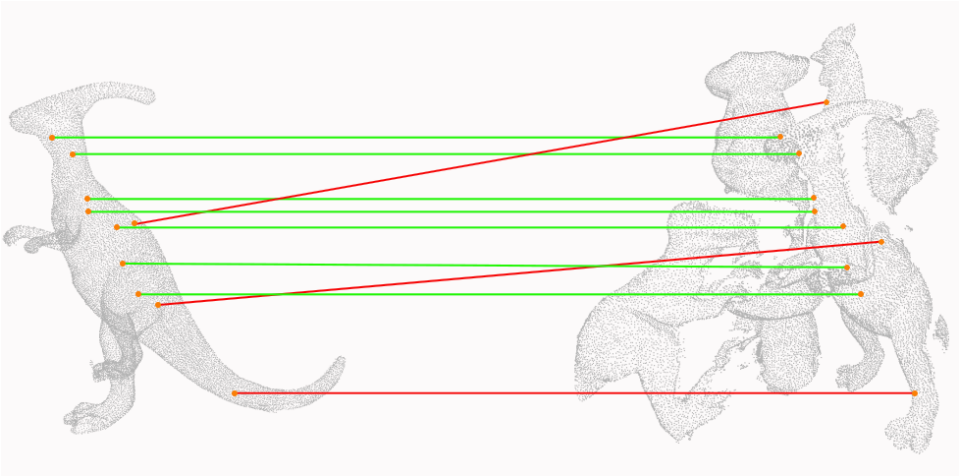


Figure 1.1: Feature-based matching paradigm in the context of 3D object recognition. Green lines correspond to correct correspondences, while red lines correspond to mismatches, *e.g.* different 3D points with similar descriptors.

The centrality of this task has boosted an intensive research in the field in the past years, as a consequence a plethora of detection and description solutions has been proposed in literature. A key trait of most keypoint detection algorithms is the definition of a saliency function adopted as a criterion to select a point as a keypoint or not. Usually these functions study the geometric properties of the surface aiming at discovering points with distinctive traits. However, their *handcrafted* nature makes them less flexible to operate in different application domains with heterogeneous datasets. As mentioned in [99], that happens for two main reasons: on the one hand different tasks may require different keypoints, in some cases it is required that these are highly precise, in others, easily identifiable. On the other hand, geometric methods start from the assumption that points showing high signal variation are keypoints but, in reality, they could be due to noise or local minimuma / maximuma. As an effective solution

to the above mentioned issues, in Chap. 3, we will propose a machine learning framework to learn a descriptor-specific keypoint detector aimed at optimizing the end-to-end performance of the feature matching pipeline. Accordingly, we cast 3D keypoint detection as a classification problem between surface patches that can or cannot be matched correctly by a given 3D descriptor, *i.e.* those either good or not in respect to that descriptor. Our proposal will be validated mainly in the context of 3D object recognition. In spite of the promising results achieved in Chap. 3, there is of course the open question of the choice of the most suited descriptor to operate in a specific scenario. As a possible answer to this question, in Chap. 4 we will report a performance evaluation of the detector-descriptor pairs obtained by learning a paired 3D detector, following the methodology proposed in Chap. 3, for the most popular 3D descriptors. In addition to 3D object recognition, we will also address experimental settings dealing with surface registration.

After having discussed how keypoint detectors can benefit from data-driven algorithms, in Chap. 6 we will focus on the description of 3D keypoints. Problems similar to those highlighted for detection algorithms arise also when working with descriptors. Indeed, it is difficult to identify the geometric characteristics that are most suitable to describe the geometry surrounding a point in order to create a representation that is compact and distinctive. The advances in deep learning have lead to data driven methods for the task that have shown promising results. Yet, the only explored way to learn rotation-invariant descriptors has been to feed neural networks with highly engineered and invariant representations provided by existing handcrafted descriptors, a path that goes in the opposite direction of end-to-end learning from raw data so successfully deployed for 2D images. In Chap. 7, we will explore the benefits of taking a stepback in the direction of end-to-end learning of 3D descriptors by disentangling the creation of a robust and distinctive rotation equivariant representation, which can be learned from unoriented input data, and the definition of a good canonical orientation, required only at test time to obtain an invariant descriptor. To this end, we will leverage two recent innovations: Spherical Convolutional Neural Networks (Spherical CNNs) to learn an equivariant descriptor and plane folding decoders to learn without supervision. The effectiveness of the proposed approach will be experimentally validated on a standard benchmark, where our method outperforms both handcrafted and learned descriptors.

Achieving invariance to rotation is crucial to the performance of a 3D descriptor as highlighted in [63]. A common adopted solution is to rely on the definition of a stable and robust *local reference frame* (LRF). This task is commonly addressed by handcrafted algorithms exploiting geometric cues deemed as distinctive and robust by the designer. However, the nuisances present in 3D scenarios, make the state-of-the-art methods hardly repeatable, especially when matching partial views. In Part III, we will investigate on the definition of a stable and repeatable local reference frame, by formulating two different proposals. In Chap. 10, we will present an handcrafted approach based on the computation of the intrinsic gradient of a scalar field defined on top of the input shape to define a repeatable tangent direction of the local frame. In the experimental results, we will showcase how existing local descriptors can directly benefit from our repeatable frames by increasing the descriptor matching performance with data affected by rigid as well as non rigid transformations. Differently, in Chap. 11 we will show the feasibility of learning a robust canonical orientation exploiting the *equivariance* property of Spherical CNNs [160]. In the experimental session, we will prove the effectiveness of our proposal on several public datasets by orienting local surface patches as well as whole objects.

In the last part of this thesis, we will tackle the problem of 3D object reconstruction from RGB images. In spite the diffusion of low cost depth sensors, cameras still remain a cheaper and more widespread acquisition source in many computer vision contexts where the information about the 3D structures of the object is mandatory to accomplish the desired goals. The advances in deep learning have dramatically boosted the process of estimating the 3D shape of an object from a single or multiple images. These leverage on learned models to regress the full object shape in a canonical pose, extrapolating the occluded parts based on learned priors. However, their viewpoint invariant technique discards the detailed structures visible from different input images, often resulting in oversimplified shapes. In Chap. 14, we will introduce a two steps pipeline to perform viewpoint variant object reconstructions by merging the details from non-overlapping images of the object. Our approach combines deep learning techniques with traditional 3D computer vision wisdom. To validate the proposed method, we will perform a comprehensive experimental evaluation on the synthetic reference benchmark for shape generation.

To summarize, the research work carried out during the development of this thesis focused on the adoption of machine/deep learning techniques to address the main building blocks of feature matching pipeline as well as 3D object reconstruction from sparse RGB images. Specifically, in Chap. 3 we will begin with the presentation of a 3D keypoint detector based on machine learning aimed at maximizing the feature matching performance when coupled with a specific 3D keypoint descriptor. Then in Chap. 4, we will combine the just introduced detector with most of the state-of-the-art 3D keypoint descriptors presenting a performance evaluation on both 3D object recognition and surface registration. In Chap. 7, we will go deeper in the learning process and exploit deep neural networks to cover the second stage of the feature matching pipeline and learn a local feature descriptor. In Chap. 10, we will show how to build a robust and repeatable local reference frame relying on handcrafted geometric function, whilst in Chap. 11, we will deploy a deep learning model to canonically orient both local as well as global point clouds. As last contribution of this dissertation, in Chap. 14, we will focus on the problem of reconstructing a 3D object from sparsely captured images combing deep learning knowledge with classical computer vision techniques. Finally, in Chap. 16, we will wrap up the contributions of this thesis and draw concluding remarks.

Before we go into this thesis, we provide some background definitions for a better understanding. We will start talking about the data structures most commonly used to represent 3D data, then 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 of 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 points clouds, 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 or the local reference frame 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 [5] is utilized.

- **Clutter and cluttered scene**: The concept of clutter is relate to 3D object recognition. 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.

Part I

LEARNING TO DETECT 3D KEYPOINTS

2

INITIAL REMARKS

3D Keypoint detection represents the first stage in the majority of modern 3D computer vision applications that need to establish automatic correspondences between surfaces. One of the main reason to focus just on keypoints, and not consider all the points within the object, is computational efficency. Unfortunately, provide a general and meaningful definition of a keypoint is not straightforward. Indeed, the criterion for determining whether a point is a keypoint or not, is strongly related to the application domain. From a very broad perspective, a keypoint can be defined as a point with a high degree of *saliency* with respect to a saliency function. Therefore, the standard paradigm for extracting 3D keypoints from point clouds or meshes relies on maximizing a handcrafted saliency function computed within a local neighborhood of each data point and then use that score as a discriminant. The algorithms that deal with keypoint extraction are called *detectors*. When designing a detector, two critical properties should be taken into account:

- *repeatability*: a keypoint should be easy to identify also when the input data are affected by different sources of noise such as: occlusions, missing parts, sensor noise and change in point density.
- *distinctiveness*: the area surrounding each keypoint should be highly descriptive and easy to discriminate. Satisfying this property is critical to the performance of descriptor matching.

In the past years, several 3D keypoint detection algorithms have been developped and, all the proposals differ substantially for the saliency function adopted. The purpose of these functions is to identify points where the surface has significant and easy to detect topological and geometric characteristics. Some of them use the *Gaussian* or the *mean Curvature* to identify areas in the surface with maximum concavity or convexity, as well as areas where the variation of the position of the points, within the size of the support, is significant in the three main dimensions through the *Eigen Value Decomposition*, up to less elegant but effective techniques, which discriminate saliency on the basis of empirical parameters. One of the main problems of

the aforementioned proposals, is the lack of generalization across heterogeneous datasets. Handcrafted saliency functions based on geometric cues often ensure good performance only on data that exhibit certain characteristics. To overcome these shortcomings, in Chap. 3 we are going to propose a general framework to learn a keypoint detector so as to optimize the end-to-end performance of the feature matching pipeline, when the detector is coupled with a predefined descriptor. The promising results obtained, encouraged us to learn an optimal detector for most of the state-of-the-art descriptors. Hence, in Chap. 4 we are going to investigate the effectiveness of the many possible combinations between state-of-the-art descriptors and the detector proposed in Chap. 3, so as to identify optimal pairs taking into account two applications: 3D object recognition and surface registration.

Before we go in there, in Sec. 2.1 we are going to revise some of the related works in 3D keypoint detection field.

2.1 RELATED WORK

The purpose of this section is to present the latest proposals at the state of the art for keypoint detection, highlighting how the different proposals differ in terms of saliency function.

2.1.1 *ISS: Intrinsic Shape Signature*

Intrinsic Shape Signature (ISS) was developed to work on unstructured 3D data and proposed in [48]. As far as the computation of the saliency function is concerned, ISS identifies points of the surface characterized by significant variations based on the *Eigenvalue Decomposition* of the *scatter-matrix* $M(p)$ computed on the points belonging to its support $B_r(\mathbf{p})$:

$$M(p) = \frac{1}{N} \sum_{q \in N(p)} (q - \mu_p)(q - \mu_p)^T \qquad \mu_p = \frac{1}{N} \sum_{q \in N(p)} q \tag{2.1}$$

In order to eliminate keypoints in areas of the surface where the distribution of points is similar along the main directions, ISS uses two thresholds,

$Th_{1,2}$ $Th_{2,3}$, together with the eigenvalues of the *scatter-matrix* in descending order, $\lambda_1, \lambda_2, \lambda_3$:

$$\frac{\lambda_2(P)}{\lambda_1(P)} < Th_{1,2} \wedge \frac{\lambda_3(P)}{\lambda_2(P)} < Th_{2,3} \tag{2.2}$$

Points which pass this first stage of selection are assigned as a salience value λ_3. The last phase of the algorithm foresees a *Non Maxima Suppression* (NMS).

2.1.2 *Harris3D*

Sipirian *et al*. in [65] extended to 3D data the Harris operator proposed in [9] by Harris and Stephen for 2D interest point detection. In this regard, for each point p on the surface a plane is fitted using the points in its support. Subsequently, the points within the support are rotated aligning the normal of the plane to the $\hat{\mathbf{z}}$ axis of a new reference system. The purpose of this transformation is to have points with maximum degree of variation along the $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ axes, hence the derivatives will be calculated along these directions. Once the Harris operator response has been computed for each point, the following two criteria can be applied to obtain the final set of keypoints:

- Select a constant number of points corresponding to the points with maximum response.
- Cluster the points and select the cluster representative.

In order to have points evenly distributed along the surface it is recommended to use the second approach.

2.1.3 *NARF: Normal Aligned Radial Feature*

NARF is a method for interest point detection on range images proposed in [57]. It has been designed taking into account two specific goals: select points in positions where the surface is stable in order to compute a robust normal and there are sufficient changes in the immediate neighbourhood, and make use of the object borders. The detection phase of NARF involves the search of the borders in the range image, where a border means a non-continuous traversals from foreground to background, by looking for

substantial increases in the 3D distances between neighboring image points. For each point, NARF looks at the local neighborhood and determines a score that represents how much the surface change at this position and a dominiant direction for this change, adding the information about borders. The saliency function is computed looking at the dominant directions and calculates a score that represents how much these directions differ from each other and how much the surface in that point is stable. Once the salience score has been calculated, a NMS is performed to find the final interest points.

3

LEARNING TO DETECT GOOD 3D KEYPOINTS

Keypoint detection is an important computer vision task pursuing extraction of repeatable and distinctive local structures from visual data. Matching keypoints across images has come forth as a well-established and successful procedure throughout the last 20 years, with most best known approaches, such as e.g. SIFT [28] and SURF [32], consisting of methods to both detect interest points as well as describe their local neighbourhood. Similarly to image matching, detection of keypoints from 3D data represented as point clouds or meshes is conducive to establishing correspondences between 3D surfaces, which has proven effective in pairwise and multi-view 3D registration, 3D object recognition and pose estimation, 3D shape retrieval and categorization. However, 3D keypoint detection is a far more recent and open research topic [89], most algorithms dating back to the last 5-7 years. Besides, unlike image features, many relevant proposals in the field of 3D features are focused on description of the local 3D neighbourhood and do not address keypoint detection. By the way of example, the reader might wish to consider Spin Images [18], arguably the most influential paper in the field, as well as later popular methods such as SHOT [98] and FPFH [45].

According to the taxonomy in [89], 3D keypoint detectors can be categorized into fixed-scale and adaptive-scale, depending on whether the size of the local support is fixed and provided as input parameter to the detector (fixed-scale), or it is automatically determined by the algorithm at each keypoint by means of a scale-space analysis (adaptive-scale). Among fixed-scale approaches, Intrinsic Shape Signatures (ISS) [48] is a widely-used, fast and effective proposal; the fixed-scale detector introduced by Mian et al. in [55] is a slower alternative, particularly robust to point density variations; NARF[57] is a method specifically conceived for 2.5D data such as range images. Among adaptive-scale detectors, MeshDoG [80] is an extension of the popular Difference of Gaussian detector [28] to scalar functions defined over a manifold approximated by a mesh; the adaptive-scale variant proposed in [55] maximizes the saliency function across scales to adaptively define the neighborhood size; [39] is another proposal aimed at extending the

DoG operator to meshes. Performance of 3D keypoint detectors is usually measured in terms of repeatability [89].

The ultimate goal of the feature detection/description/matching pipeline is to identify correct correspondences across visual data, which requires distinctiveness and robustness to nuisances of the description computed at the detected local regions. State-of-the-art 3D detectors, however, rely on hand-crafted saliency functions designed to maximize repeatability rather than "end-to-end" feature matching performance. In other words, 3D detectors are conceived to find repeatedly the same regions although these may not yield the best performance when encoded and matched according to the given descriptor. It is worthwhile highlighting how, in the context of visual tracking, the importance of pursuing the feature matching goal directly within the detection stage was pointed out by the popular work of Shi and Tomasi [13], where "good" features to be detected are those likely to yield correct matches between consecutive frames.

Based on the above considerations, we propose to cast 3D keypoint detection as a classification problem between the classes of points that can or cannot be effectively encoded and matched by a pre-defined 3D descriptor. In its simplest formulation, our approach is modeled as a binary classification problem, where a 3D point is classified as either "good" (i.e. a keypoint) or "bad" (i.e. not a keypoint) according to the likeliness of providing a correct match when represented by a given 3D descriptor computed at a fixed scale, this yielding a *learned* fixed-scale detector. Furthermore, we extend the methodology to *learn* an adaptive-scale detector: we cast the problem as a multi-class classification, where the different classes are associated with different characteristic scales; thereby, each 3D point may be classified either as a keypoint endowed with its characteristic scale or as not a keypoint. As discussed in [89], detecting 3D keypoints at multiple scales helps gathering important features that might be missed if saliency is measured at a fixed scale only.

Key to our fixed-scale and adaptive-scale detectors is a framework to define the training set required by our supervised learning approach. This is accomplished by sifting out from the training data those 3D points that can be matched successfully across multiple views based on the chosen descriptor, these good features providing the positive samples to learn the classification function. Automatic learning of such classification function enables to adaptively identify those points more likely to provide correct correspondences for a given dataset and descriptor, without being bound

to the specific geometric structures which would fire a chosen handcrafted detector. The ability to learn a descriptor-specific detector is particularly relevant to the domain of 3D features as many state-of-the-art descriptors lack a companion detector [18, 45, 83, 98]. In particular, throughout our experimental evaluation, we will consider descriptors as diverse as SHOT [98], FPFH [45] and Spin Images [18] in order to show how their matching pipelines can be enhanced significantly by deploying a companion detector learned by our method rather than state-of-the-art hand-crafted detectors.

Moreover, in 3D computer vision applications a variety of different sensors can be deployed, such as e.g. laser scanners and consumer depth-cameras, which allow for acquiring data quite diverse as regards resolution and/or noise. Such variability mandates careful tuning of a detector's parameters to extract meaningful keypoints across diverse datasets, possibly rendering a method useless when applied to data that differ from those it was originally conceived for. Hence, automatically learning the detector from representative training samples holds the potential to provide higher adaptiveness to the diverse kinds of data delivered by 3D sensors than handcrafted detectors. To this end, we show experiments dealing with learning our detector from a vast number of 3D models to attain a higher degree of generalization to unseen shapes and avoid the necessity of retraining on each new dataset.

3.1 RELATED WORK

A few researchers investigated the use of machine learning techniques for keypoint detection. As far as images are concerned, the most important contribution is probably FAST [41], which is based on the Accelerated Segment Tests to detect corner-like features: the order of the tests is learned in a tree from a training set to speed-up detection time. This approach lays at the core of several recent and successful keypoint detectors, such those deployed in BRISK [60] and ORB [64]. Another research line deals with refining the set of keypoints extracted by a standard detector to improve the overall performance in a particular task. In [95], Hartmann et al. apply machine learning algorithms to learn which keypoints are likely to be discarded in the descriptor matching stage among those extracted by a standard keypoint detector (i.e. DoG). By using a Random Forest [23] to learn such "matchability", they show that their approach can improve and speed-up considerably the feature matching stage of a Structure-from-

Motion pipeline. Similarly, in [46], the authors show how higher repeatability can be achieved by instructing a WaldBoost classifier to sift out only the keypoints known to be useful in a given scenario among those extracted by a standard detector. As an exemplar application, they demonstrate improved image matching in an urban environment, the classifier learning to focus on stable man-made structures while ignoring objects that undergo natural changes such as vegetation and clouds. A different approach is proposed in [115], where the most repeatable DoG keypoints across a set of training images are used to define the positive samples to train a saliency function able to highlight the same image points under drastic illumination changes caused by weather, season and time of day.

In contrast with the machine learning approaches proposed so far to refine or robustify 2D detectors [46, 95, 115], we directly use our classifier as a keypoint detector, thus avoiding the need to select a specific hand-crafted detector as a pre-processor. This holds the potential to yield higher adaptiveness to diverse input data, which otherwise may be limited by the suitability to the specific dataset of the geometric structures highlighted by the selected detector. Moreover, the choice of such preliminary 3D detector would turn out problematic as there exists not yet an established and generally applicable algorithm for 3D data as it may be considered DoG in the case of images.

Leveraging on datasets that already include the definition of the points of interest, a few papers have proposed to pursue 3D keypoint detection within a machine learning framework [81, 99, 127]. In particular, [81] investigates on the use of linear (LDA) and non-linear (AdaBoost) classifiers to detect facial landmarks in 3D meshes. Similarly, the authors of [99] propose to learn a classification forest to better cope with the high variability of the structures annotated as salient within the dataset proposed in [72], the evaluation task concerning detection of the points indicated as salient by human users rather than extraction of generic repeatable keypoints. Within the same settings as in [99], the authors of [127] advocate the use of a deep neural network consisting of three stacked auto-encoders to regress a saliency function suitable to detect the manually annotated landmarks defined in [72]. Differently, Holzer et al. [73] propose to speed up and improve the repeatability of curvature-based detectors by learning the saliency function by a regression forest that deploys binary depth comparisons as features.

Differently, we propose to learn a classification function that acts as a 3D keypoint detector aimed at identifying points likely to generate correct

matches when encoded by a given descriptor and, accordingly, we define a method to generate automatically the data corpus required to train the classifier. Therefore, with our approach, the definition of the interest points needs neither to be available together with the dataset, as in [81, 99, 127], nor approximate a hand-crafted criterion, as in [73]. Instead, it is peculiarly data-driven and attained automatically based on the ability to match specific descriptors, thereby generating a descriptor-specific keypoint detector.

3.2 TRAINING SET TO LEARN GOOD 3D KEYPOINTS

The goal of our approach is to learn to detect 3D keypoints that can yield good correspondences along with a given 3D descriptor. The definition of the training set, and, in particular, of positive examples, is therefore crucial. We exploit a set of partially overlapping 2.5D views of the 3D objects that are of interest in a particular dataset and, for each such object, select those points that can be matched correctly across different views in the chosen descriptor space. In this Section, we delineate a framework to learn a fixed-scale 3D keypoint detector given a generic 3D descriptor. As such, we fix the support of the descriptor so that the positive samples to train the detector are identified at a specific scale only. In Section 3.4, then, we extend the methodology to learn adaptive-scale detectors.

3.2.1 *Definition of the training set*

Let $\{V^i\}, i = 1, \ldots, N$, be a set of calibrated 2.5D views of an object belonging to a 3D dataset (Fig. 3.1).

The term calibrated views is used here to signify knowledge of the ground-truth roto-translations to bring each pair of views into a common reference frame. Should the object be provided within the dataset as a 3D model, the 2.5D views would be gathered by performing synthetic renderings, as also done, e.g., in [50, 67]. For instance, we deploy the method in [67], where 2.5D views are rendered from the nodes of an icosahedron centered at the centroid of the 3D model. For each point p belonging to each view V^i, we compute the given 3D descriptor, denoted as D^i_p. For each V^i, we select the subset:

$$\mathcal{V}^i = \{V^j | V^i \cap V^j \geqslant \tau, j = 1, \ldots, N, j \neq i\} \tag{3.1}$$

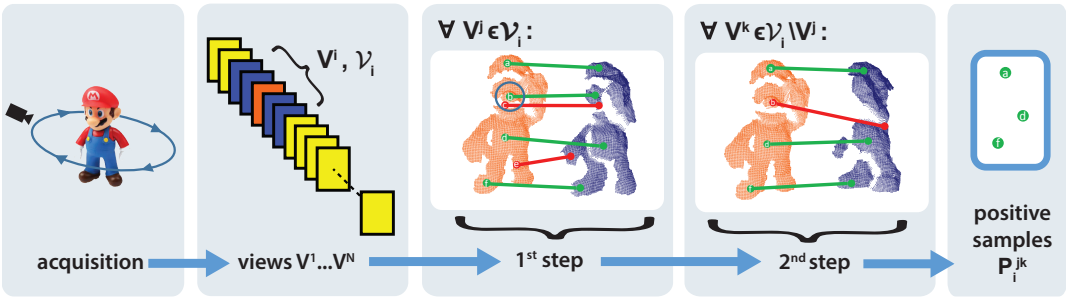


Figure 3.1: From left to right: if not provided within the dataset, a set of calibrated 2.5D views is attained by simulating a 3D sensor in N uniformly distributed vantage points around the object; for each view V^i, those other views exhibiting a sufficient overlap are selected; correspondences are established between V^i and each overlapping view V^j, such that points yielding correct matches are kept as candidate positive samples (e.g., *a*,*b*,*c*,*d*) while those either wrongly matched (e.g. *e*) or laying close to another one yielding a better correct match (e.g. *c*) are discarded; by matching every other overlapping view V^k, the set of candidate positive samples is refined by dismissing points yielding wrong matches (e.g. *b*).

i.e., the views partially overlapping with V^i according to a chosen threshold, τ. Then, for each $V^j \in \mathcal{V}^i$, we carry out the two-step point selection procedure exemplified in Fig. 3.1.

In the first step, we match each point $p \in V^i$ to a point $q \in V^j$ by finding the nearest neighbour descriptor according to the Euclidean distance $\|D^i_p - D^j_q\|_2$ in the descriptor space. All pairs of matching points, (p, q), are then sorted ascendantly based on the computed Euclidean distances between descriptors. Starting from the first element of the list of matching pairs, we check whether the match is indeed correct or not, i.e. if the points in the two views correspond to the same point of the 3D model. To make the learned detector robust to small shifts, we regard a match as correct if the Euclidean distance between p and q turns out smaller than a threshold ϵ once V^i and V^i are brought into the same reference frame by application of the known ground-truth roto-translation between the views. It is worth pointing out that this type of check also allows for discarding wrong matches dealing with points that do not belong to the region of overlap between the two views. If the match is found correct, we remove (p, q) from the sorted list of matching pairs and insert p into a set of candidate positive samples, which is referred to as P^j_i to denote that it includes the points in V^i that yield good matches when seen in V^j. To mimic the effect of the spatial non-maxima suppression usually performed by hand-crafted detectors to prevent multiple responses

around salient structures, upon insertion of p in P_i^j we also remove all the entries that include its neighbors from the list of matching pairs. More precisely, we remove all the pairs (p', q), with $p' \in V^i$ and $q \in V^j$, such that the Euclidean distance between p' and p is smaller than a threshold referred to as ϵ_{NMS}. Conversely, pair (p, q) is simply removed from the list in case the match turns out wrong. The procedure is then repeatedly executed until the sorted list becomes empty.

Hence, the first step provides a set of candidate positive samples, P_i^j, consisting in the points of V^i that yield good matches when seen from the vantage point associated with V^j. To robustify the selection by alleviating the dependence on the specific viewpoint change between V^i and V^j, in the second step we refine the list of positive samples by seeking for those points that may be matched correctly also in other views overlapping with V^i that we did not use for the definition of P_i^j. More precisely, for each $V_k \in \mathcal{V}^i \setminus \{V^j\}$, i.e., every view partially overlapping with V^i other than the already used V^j, we perform the second step as follows. First, we select the points in P_i^j that belong to the area of overlap between V^i and V^k. This is accomplished by transforming each point in P_i^j according to the ground-truth rotation and translation from V^i to V^k and checking if the transformed item has a neighbour in V^k within a distance ϵ. For every point in P_i^j that lies in the common area between the views, we find the nearest neighbor in the descriptor space among all descriptors computed on V^k so as to then check if the match is correct according to the same criterion as used in the first step to validate matches from V^i to V^j. Thus, given P_i^j, for each V^k we obtain a refined set of positive samples, referred to as P_i^{jk}, by retaining only those points in P_i^j that yield correct matches also towards V^k, that is those points in V^i that are good to be matched by the given descriptor when seen from the vantage points of both V^j and V^k. The final refined set of positive samples, $\tilde{P}_i^j$, is given by the union of sets P_i^{jk} across all views V^k:

$$\tilde{P}_i^j = \bigcup_{\substack{k=1 \\ k \neq i,j}}^{|\mathcal{V}^i|} P_i^{jk} \tag{3.2}$$

which implies that any point in P_i^j, i.e. a point in V^i that gets correctly matched in V^j, is kept within the final set $\tilde{P}_i^j$ in case it can be matched correctly in at least one of the other overlapping views V^k.

The final set of positive samples extracted from V^i, i.e. P_i, is the union of the refined sets of positive samples across all the overlapping views V^j:

$$P_i = \bigcup_{\substack{j=1 \\ j \neq i}}^{|\mathcal{V}^i|} \tilde{P}_i^j \tag{3.3}$$

Eventually, the set of positive samples extracted from the object, P, is given by the union of all the positive samples obtained by its N views :

$$P = \bigcup_{i=1}^{N} P_i \tag{3.4}$$

and the set of positive samples obtained from a dataset is the union of the sets extracted from all the objects belonging to the dataset.

To obtain the negative samples, for each object of the dataset we randomly sample from all the views a set of points which have not been included in P. To ensure getting points scattered across all the areas unfavorable to surface matching by the given descriptor, upon sampling a new negative sample from a view we also remove from the set of candidate negatives all its neighbours lying within a distance threshold ϵ_{neg}. Fig. 3.2 depicts exemplar positive and negative training samples extracted from two views dealing with two different objects. It is worthwhile pointing out how positive samples are unevenly distributed across surfaces and do not necessarily appear on the intuitively prominent structures, such as e.g. the knees and elbows of the right object or the tail of the left object, which would be extracted by most hand-crafted detectors engineered to capture geometric saliency. Indeed, although geometrically salient, such structures do not turn out amenable to matching by the given descriptor under viewpoint changes. Eventually we use the threshold parameter ϵ_{neg} as the main knob to create a globally balanced training set featuring negative samples gathered from all the object belonging to the dataset.

3.2.2 *Validation of the training set*

Before addressing the issue of how to realize a classifier trained to detect 3D keypoints by the previously defined training set, we investigate on the effectiveness of the proposed approach in sifting out good regions to

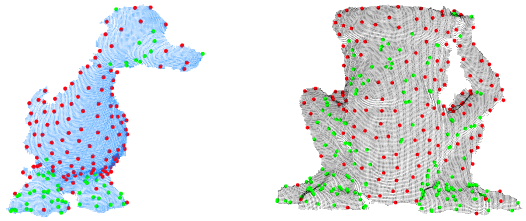


Figure 3.2: Exemplar positive (green) and negative (red) training samples obtained by the proposed method on two views dealing with different objects using SHOT [98] as the given 3D descriptor.

be encoded and matched with the given 3D descriptor. In other words, we aim at assessing whether, should a perfect classifier be available, the classification function defined by the training set would allow for detecting regions yielding more correct correspondences than those found by standard detectors designed to highlight geometric saliency. We rely on the matching performance measured by finding nearest neighbours in the descriptor space between the reference view V_i and the partially overlapping views V^k deployed in the second step of the procedure described in Section 3.2.1. More precisely, we compare the correct correspondences obtained when matching 3D descriptors computed at the points belonging to the set of candidate positive samples yielded by the first step of the procedure in Section 3.2.1, i.e. P_i^j, versus that achieved by matching descriptors computed at keypoints extracted by standard 3D detectors available in the Point Cloud Library (PCL) [1] such as, in particular, ISS [48] and Harris3D [67].

This experimental study is focused on comparing the quality of the regions highlighted by our novel proposal and the standard approaches concerned with maximizing geometric saliency, regardless of the repeatability of the actual keypoint detection process. Therefore, to determine which correspondences turn out correct, we mimic an ideal detector by transforming the keypoints extracted in V_i, i.e. those in P_i^j for our method as well as those extracted by ISS and Harris3D, according to the ground-truth rigid motion from V_i to V_k and, for each keypoint, check if the point in V_k associated with the nearest neighbor in the descriptor space lies closer than a distance ϵ from its transformed 3D coordinates. Accordingly, in Fig. 3.3 we rely on SHOT [98] to determine the keypoints extracted in a view by our method as well as to match descriptors computed at the regions found by all considered methods, and report both the average number and percentage of

[1] www.pointclouds.org

correct correspondences obtained on 35 views extracted from all the models belonging to the *Kinect* dataset [98].

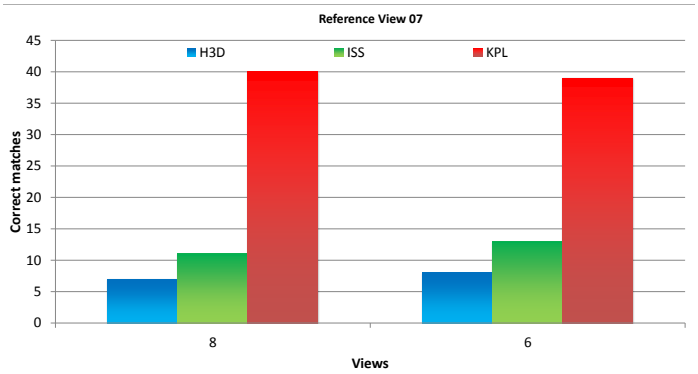


Figure 3.3: Number of correct matches obtained by computing the SHOT descriptor [98] on the regions selected by Harris3D, ISS and our proposed approach (referred to as KPL).

Results in Fig. 3.3 show how the local structures highlighted by our method hold the potential to improve the matching performance significantly with respect to the standard detectors built upon the notion of geometric saliency.

3.3 DESIGN OF THE CLASSIFIER

We rely on Random Forest [23] to learn the 3D keypoint detector from the sets of positive and negative training samples collected through the procedure described in Sec. 3.2.1. Indeed, Random Forests have been employed quite successfully to tackle a number of computer vision problems [71], including 3D keypoint detection [99]. Moreover, Random Forests are among the fastest classifiers as regards run-time prediction, even when dealing with complex classification functions, unlike, e.g., SVMs with non-linear kernels. This is relevant when using a classifier as a keypoint detector, as the prediction must be carried out at every single point of the input cloud. Finally, a Random Forest performs multi-class classification straightforwardly, which makes this classification approach amenable to generalize our framework in order to handle multiple support sizes, thereby attaining an adaptive-scale descriptor-specific detector, as illustrated in Section 3.4.

As far as features are concerned, we propose features inspired by SHOT [98] but able to achieve rotation invariance without computing the Local Reference Frame (LRF). In Fig. 3.4 provides a graphical overview of the feature computation process.

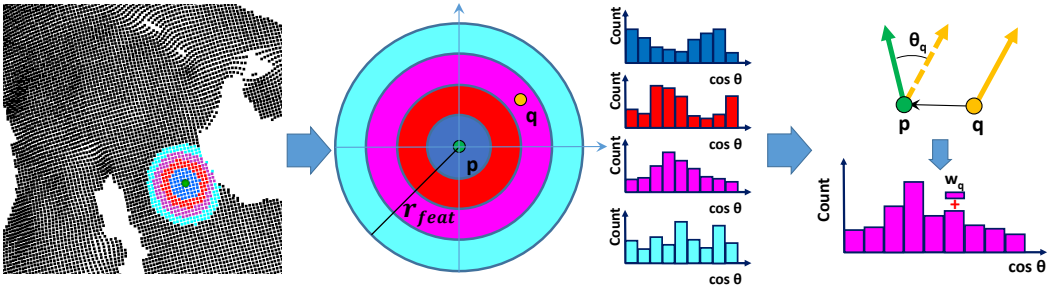


Figure 3.4: Overview of the feature computation process. In the example, $N_r = 4$ subdivisions along the radial coordinate are used to split the spherical support around p into equally many sectors (for ease of visualization, a 2D representation of the 3D spherical support is portrayed). For each sector (i.e. spherical shell), a histogram of orientations is obtained by accumulating the cosines of the angles between the normals at the points q falling within the sector and the normal at p, weighted according to a bilinear interpolation (w_q).

In particular, given the point under consideration, p, for every of its neighbours, q, within a spherical support of radius r_{feat}, we calculate the cosine of the angle between the normal at p and the normal at q and quantize such values into a set of histograms according to a subdivision into sectors of the support around p. In SHOT, the sectors are obtained by dividing the spherical support evenly along the radial, elevation, and azimuth polar coordinates of the LRF centered at p. To avoid computation of the LRF, we change here the shape of the sectors within the support so as to consider only N_r subdivisions along the radial dimension and compute a histogram with N_b bins for each spherical shell thus obtained. As the histograms are computed within spherical shells, the features turn out inherently rotation invariant so that calculation of a LRF is not needed. To avoid quantization artifacts, bilinear interpolation is performed upon casting a vote into a histogram. Finally, the histogram of every shell is normalized to have unitary Euclidean norm to improve robustness with respect to point density variations. We set r_{feat} to half the radius used to compute the descriptor.

Thus, given all the objects within a dataset, we obtain the positive and negative training samples as described in Sec. 3.2.1. Then, for all training samples, we extract the features described in this Section and train a Random Forest comprising T trees. To detect keypoints on an unseen point cloud, we apply the trained Random Forest classifier at each point, p, and count the number of trees, T_p, that predict p as a keypoint. The saliency, $s(p)$, associated with p, is given by the ratio T_p/T, the point being detected as a keypoint if it turns out a local maximum of the saliency function

within a neighborhood of radius r_{nms} and its saliency is sufficiently high, i.e. $s(p) \geqslant s_{min}$.

We would like to highlight that simple features derived from pairwise pixel comparisons have been usually deployed together with intensity and depth images so as to best leverage the inherent efficiency of Random Forests [71, 73]. However, taking repeatable pairwise comparisons within 3D neighborhoods subject to arbitrary 3D rotations would require the definition of an LRF, as done by most 3D descriptors, e.g. SHOT [98], MeshHoG [80], ROPS [83]. This can be a quite expensive step of the algorithm, sometimes accounting for the largest fraction of the overall time needed to compute the descriptor. Unlike 3D descriptors, though, which are typically calculated at a very small subset of salient points of the cloud, establishing a LRF for the sake of keypoint detection would require such costly computation at every point of the cloud, and that is the reason why our feature has been designed to avoid it.

3.4 ADAPTIVE-SCALE KEYPOINT DETECTION

In this Section, we describe how the proposed method can be extended to perform descriptor-specific adaptive-scale keypoint detection. The number of detectable scales S is finite and predefined. In this case, the detection problem becomes a multi-class classification problem, with $S + 1$ classes: the classifier has to assign each input point to either the "negative" class, if the point is not a keypoint, or to one of the remaining "positive" S classes, if the point is a keypoint, the class defining the scale of the keypoint. At training time, one of the scales is associated to each positive sample by probing all of them as described below. No scale has to be assigned to negative training samples. At test time, each keypoint found in a cloud is associated with one single scale only.

To define the training set, we first extract positive samples from the models at each predefined scale following the same procedure as described in Sec. 3.2.1. Two refinements are then performed, in order to associate a point only to one class and to obtain a balanced training set. In both refinements we use the mean Euclidean distance between the descriptor at the keypoint and those matched in the overlapping views, d_{avg}, so as to choose which keypoints to keep in the training set: such distance help us identifying those keypoints whose descriptors remain more similar under

viewpoint changes. In the first refinement, if a point is considered a positive training sample for different scales, we use it only for the scale at which d_{avg} is minimal. For example, let us consider again Fig. 3.1 and assume point a in view V^i (i.e. the top-most green dot in the orange cloud) to be a positive sample for two scales: it would then be kept in the training set only for the scale featuring the lowest mean Euclidean distance between the descriptor computed at a and those at the matching points in views V^j,V^k (corresponding green dots in the blue clouds). In the second refinement, to obtain a balanced training set, we remove from the training set for the larger classes the positive samples which have higher d_{avg} so as to reach approximately the same number of samples per class. Negative training examples are extracted randomly from the model point clouds according to the procedure already outlined in Sec. 3.2.1. We set ϵ_{neg} such that the overall distribution of samples per class in the training set is balanced, i.e. the number of negative samples is approximately the same as the number of samples in each positive class.

We use the same feature proposed for the fixed-scale detector (Sec. 3.3) and the same number of histogram bins N_b. r_{feat} and N_r are instead adapted to the new scenario. In particular, the radius of the spherical support r_{feat} to compute the feature is set to half the size of the largest predefined scale, coherently with the choice made for the fixed scale detector where r_{feat} was set to half the size of the support used to compute the 3D descriptor. The number of spherical subdivisions N_r for the adaptive-scale detector is instead defined so that the metric width of each spherical subdivision is approximately the same used in the fixed-scale algorithm. For instance, let assume in the fixed-scale case for a dataset the descriptor support is 40 and $N_r = 5$. We then get $r_{feat} = 20$ and the width of each subdivision in the fixed-scale case is $20/5 = 4$. If the range of detectable scales in the adaptive-scale case is then $S = [40, 50, 60]$, we get $r_{feat} = 30$ and $N_r = \lfloor 30/4 \rfloor = 7$ for the adaptive-scale detector.

When the feature is extracted at test time, the scale assigned to each point is unknown. Therefore, the size of the feature support r_{feat} used at training time is the same for all the points regardless of their scale, and is the same used at test time. As the random forest inherently performs feature selection at training time, we expect the forest to analyze different entries of the feature vector to detect different scales, i.e. to use the entries computed within closer subdivisions to reach the leaves assigned to smaller scales and those computed for more distant shells for larger scales.

When detecting keypoints on a cloud, each point p is classified by the T trees of the forest as belonging to a certain class, namely a keypoint and a characteristic scale or a negative. For each such class, c, we compute a prediction confidence as $P_c(p) = T_c/T$, with T_c the number of trees that classify p as belonging to c, the final class predicted for p given by that yielding the highest confidence. In case a positive class is predicted, a saliency score equal to the prediction confidence is also assigned to the point: $s(p) = P_c(p)$. Eventually, a point belonging to a positive class is detected as a keypoint if the saliency is above a threshold, $s(p) \geqslant s_{min}$, and it turns out a local maximum of the saliency while considering the points predicted to belong to the same class within a neighbourhood of radius r_{nms}.

The computational complexity of the proposed adaptive-scale detector is similar to that of the fixed-scale algorithm, both boiling down to traversing the trees of a random forest having the same structure (Sec. 3.5.1). In practice the run-time of the adaptive-scale detector turns out slightly, i.e. about 20%, higher due to the time spent in neighbor-search operations within the larger support deployed to compute the features.

As discussed in [89], the main benefit brought in by an adaptive-scale 3D detector concerns the ability to gather more features than a fixed-scale approach, as more scales are probed by the former and some surface patches might show up as salient at certain scales rather than others. Moreover, as 3D surfaces are most often matched under the assumption of Euclidean motion, i.e. rotation and translation without any size change, scale information in metric units may be deployed to reduce the search space within the feature matching process. Accordingly, while deploying an adaptive-scale detector, one would conveniently compare in the descriptor space only those keypoints that share the same scale in the considered views, as those detected at different scales are likely to correspond to patches having different sizes. This allows for speeding up the matching process and holds the potential to improve precision, as wrong matches caused by similar descriptors at different scales may be avoided.

3.5 EXPERIMENTAL RESULTS

Since we deal with improving feature matching performance by learning a descriptor-specific detector, rather than gathering descriptor-agnostic re-

peatability measurements (as, e.g., in [89]) we validate our proposal through descriptor matching experiments. In particular, we deploy both hand-crafted saliency-based detectors as well as our learned detector within a standard 3D feature matching pipeline in order to compare their performance.

Regarding hand-crafted fixed-scale detectors, we evaluate those available in PCL, namely ISS, Harris3D and NARF [57], alongside with the algorithm proposed by Mian et al. [55] and referred to as KPQ in [89], which, together with ISS, turned out particularly effective in the experiments carried out in [66]. We also consider uniform sampling of points, as this baseline 3D detector is often used within 3D feature matching pipelines. As for scale-adaptive detectors, we test the scale-adaptive version of KPQ (see again [55]) and MeshDoG [80]. To realize our proposal, hereinafter also referred to as KeyPoint Learning (KPL), we used the Random Forest implementation provided by OpenCV[2]. In the experiments related to the adaptive-scale formulation of our method, denoted as KPL-AS, we consider a set of three scales (i.e. radius sizes), which we found appropriate to demonstrate the effectiveness of the proposed multi-class classification approach without slowing down the training stage exceedingly.

We tested all detectors on four publicly available datasets, three of which had already been used to compare 3D detectors in [89]: the *Laser Scanner* dataset introduced by Mian et al. [55], the *Random Views* dataset, based on the Stanford 3D scanning repository[3], and the *Kinect* dataset proposed in [98]. The fourth, referred to here as *Venezia 3D* dataset[4], was introduced in [87]. Each dataset includes a list of models, which we used to train our detector, and several scenes, where we performed keypoint detection. When the models are full 3D, as it is the case of Laser Scanner and Random Views, to apply our learning algorithm we render 42 equally spaced 2.5D views from the nodes of an icosahedron using the implementation of the method proposed in [67] available in PCL. The scenes are 2.5D views acquired independently from the models and depicting a variety of arrangements concerning models and other objects (clutter). Therefore, the views extracted from the models to train the detector are unrelated to the views of the models appearing in the scenes. Moreover, this evaluation methodology mimics the likely use of the technique in a real application, with the models

[2] opencv.org
[3] http://graphics.stanford.edu/data/3Dscanrep/
[4] http://www.dsi.unive.it/~rodola/data.html

either known beforehand or acquired at initialization time and scenes being unseen data.

As for descriptors, we use SHOT [98], which has been reported to obtain good results in a number of comparative evaluations addressing object recognition [67, 69], 3D object classification and retrieval [79], semantic segmentation of point clouds [70], localization of medical landmarks [78]. Moreover, we learn to detect good 3D keypoints for Spin Images[18] and FPFH[45], two other prominent algorithms in the field of 3D descriptors, both relying on different design choices than SHOT. As the features used to train our Random Forest classifier are inspired by SHOT, successful application of the same methodology and features to three different descriptors such as SHOT, FPFH and Spin Images would vouch for the general validity of the novel concept advocated in this chapter. We rely on the implementation available in PCL for all descriptors.

Table 3.1 and Table 3.2 report the parameters used at test and training time on each dataset in the experiments pursuing fixed-scale and adaptive-scale detection, respectively. As it can be observed from the Tables, we use the same support size for all evaluated descriptors (SHOT, FPFH and Spin Images) across all datasets. Similarly, we rely on the same support size to compute both the features for our learned detector as well as the saliency function for the considered handcrafted methods.

Table 3.1: Parameters related to fixed-scale detection experiments. Some parameters concerning the training process of our method are not specified for Random Views as we did not learn a new forest on this dataset.

Dataset	r_{desc} [mm]	r_{feat} [mm]	τ	ϵ [mm]	ϵ_{nms} [mm]	ϵ_{neg} [mm]	r_{nms} [mm]	s_{min}	N_b	N_r
Laser Scanner	40	20	0.85	7	4	2	4	0.8	10	5
Random Views	40	20	-	7	-	-	4	0.8	10	5
Kinect	40	20	0.24	10	20	15	20	0.8	10	5
Venezia	40	20	0.50	7	4	5	4	0.95	10	5

Table 3.2: Parameters dealing with adaptive scale detection experiments. Some parameters concerning the training process of our method are not specified for Random Views as we did not learn a new forest on this dataset.

Dataset	r_{desc} [mm]	r_{feat} [mm]	τ	ϵ [mm]	ϵ_{nms} [mm]	ϵ_{neg} [mm]	r_{nms} [mm]	s_{min}	N_b	N_r
Laser Scanner	[40, 50, 60]	30	0.85	7	4	3.20	4	0.5	10	7
Random Views	[40, 50, 60]	30	-	7	-	-	4	0.5	10	7
Kinect	[40, 50, 60]	30	0.24	10	20	6	20	0.5	10	7

3.5.1 *Hyperparameter optimization*

Model selection with Random Forests deals with specifying a few self-explanatory hyperparameters. However, the impact on performance of parameters such as the tree depth and the minimum number of samples to stop splitting a node is somehow unclear: e.g., Breiman [23] suggests to grow a tree until just 1 sample is left in a node, whereas Criminisi et al. [71] rely on a higher number of samples so to estimate the posterior distribution at each node. Therefore, we chose the number of trees, the maximum tree depth, and the number of samples to stop node splitting by a cross-validation procedure. In particular, we carried out a three-fold cross validation where 2/3 of the N views of each model form the training set while the remaining one are deployed for testing.

As regards the considered hyperparameter ranges, we varied the number of trees from 10 to 100, the tree depth from 10 to 40 and the number of points to stop splitting from 1% of the training samples down to just 1 sample. Fig. 3.5 deals with the *Kinect* dataset and shows that performance of the classifier does not improve when relying on more than 50 trees.

Even more evidently, there is no advantage in letting the tree grow deeper than 25 levels and we get the best generalization results by stopping to split when 5 or less samples are left in a node. With the Laser Scanner dataset, which features a significantly larger quantity of training data (i.e. 100K positive samples compared to the 8000 positive samples of Kinect), we found that best performance are achieved by 100 trees, the same maximum depth as with Kinect and 1 node per sample to stop splitting. As for the Venezia dataset, where we trained a detector on almost 60 different models with a final number of samples per class of about 12,5M, the optimum number of trees and maximum tree depth remain unchanged, while the minimum number of samples to stop splitting a node should be as large as 7[5]. Eventually, as we found that the hyperparameters chosen to realize fixed scale detectors on the considered datasets do work well also when pursuing adaptive-scale detection, we did not carry out any further model selection procedure for the multi-class random forest classifiers.

[5]The increased minimum number of samples was motivated also by limitations concerning memory management by the OpenCV "io" module which we used to save and load the forest to and from disk. Indeed, the adopted implementation cannot handle correctly forests that are too large: increasing the minimum number of samples reduced the average depth of each tree in the forest and, thereby, the final file size of the forest.

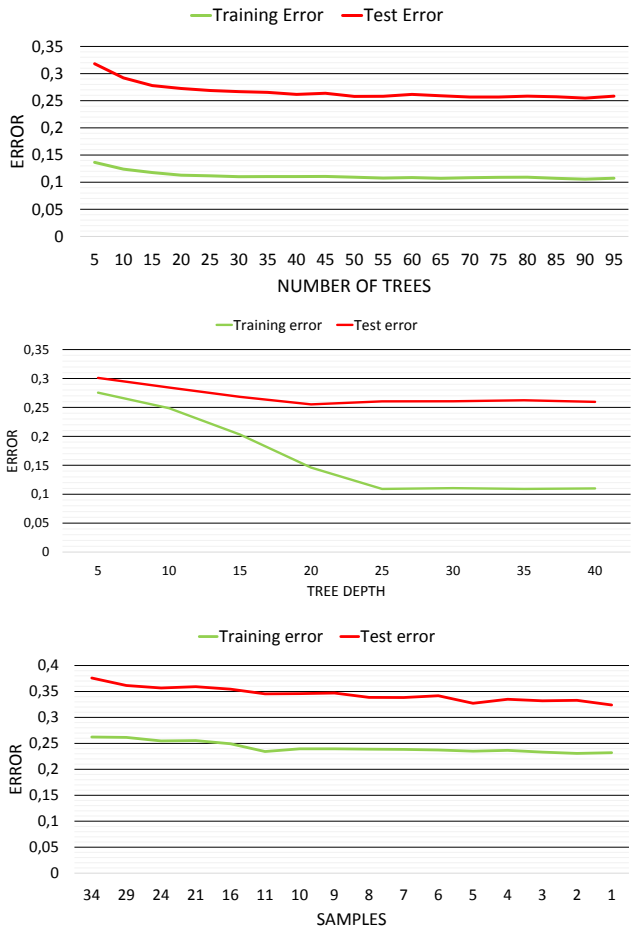
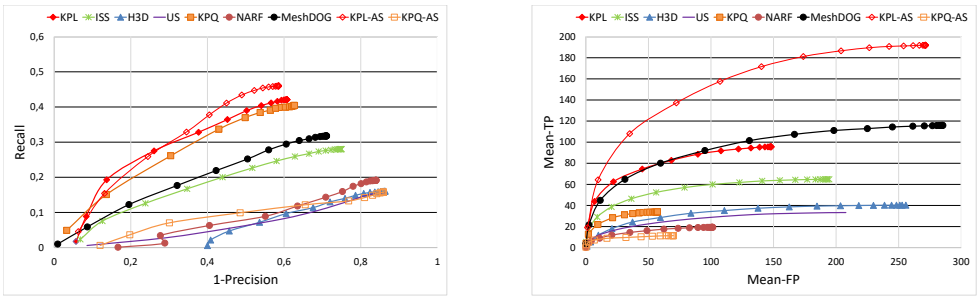


Figure 3.5: Hyperparameter optimization on the Kinect dataset.

3.5.2 *Results on the Laser Scanner dataset*

This dataset introduced by Mian *et al*. [55] consists of 4 full 3D models and 50 scenes wherein models significantly occlude each other. To create some clutter, scenes contain also an object which is not included in the model gallery. As scenes are scanned by a Minolta Vivid 910 scanner, they are corrupted by real sensor noise.

To carry out the descriptor matching experiment concerning fixed-scale detectors, firstly we detect keypoints on all views of all models, compute descriptors and create one kd-tree containing all model descriptors. We then run detectors on scenes, and for each scene keypoint compute the associated descriptor and establish a match with the Nearest Neighbor model descriptor via the kd-tree. To perform descriptor matching with adaptive-scale detectors, first we detect keypoints on all views of all models at all scales, compute descriptors according to the found characteristic scales



(a) (b)

Figure 3.6: Results provided on the Laser Scanner dataset by the keypoint detector learned for SHOT. Precision-Recall curves (a), Mean True Positives vs. Mean False Positives(b).

and create one kd-tree for each scale, so that each kd-tree contains all model descriptors computed at that scale. We then run detectors on scenes, and for each scene keypoint establish a match with the Nearest Neighbor model descriptor via the kd-tree associated with the same scale as that found for the scene keypoint.

In both cases, for each match we can check if it is correct based on the available ground-truth, and increment the true positives or false positives accordingly. By varying the threshold on the maximum distance between descriptors to accept a match, we obtain Precision-Recall curves, as shown, e.g., in Fig. 3.6 (a). According to the same methodology, we also compute Mean True Positive versus Mean False Positive curves (e.g., Fig. 3.6(b)), as proposed in [89] in order to compare descriptor matching performance for different detectors. These curves complement Precision-Recall curves because they highlight the differences in the number of keypoints extracted and matched correctly, which is an important trait when using detectors in practice as well as a possible source of bias when comparing Precision-Recall figures between algorithms yielding quite different quantities of extracted keypoints. Indeed, should a detector extract just one keypoint which then gets matched correctly, its Precision-Recall performance would be deemed as perfect though a single correct correspondence would hardly turn out useful in any practical setting.

Fig. 3.6 shows how, when using the SHOT descriptor together with hand-crafted detectors, the best performance according to Precision-Recall curves is provided by KPQ. This is somewhat not surprising, as this detector was originally proposed for the Laser Scanner dataset. ISS also yields reasonable

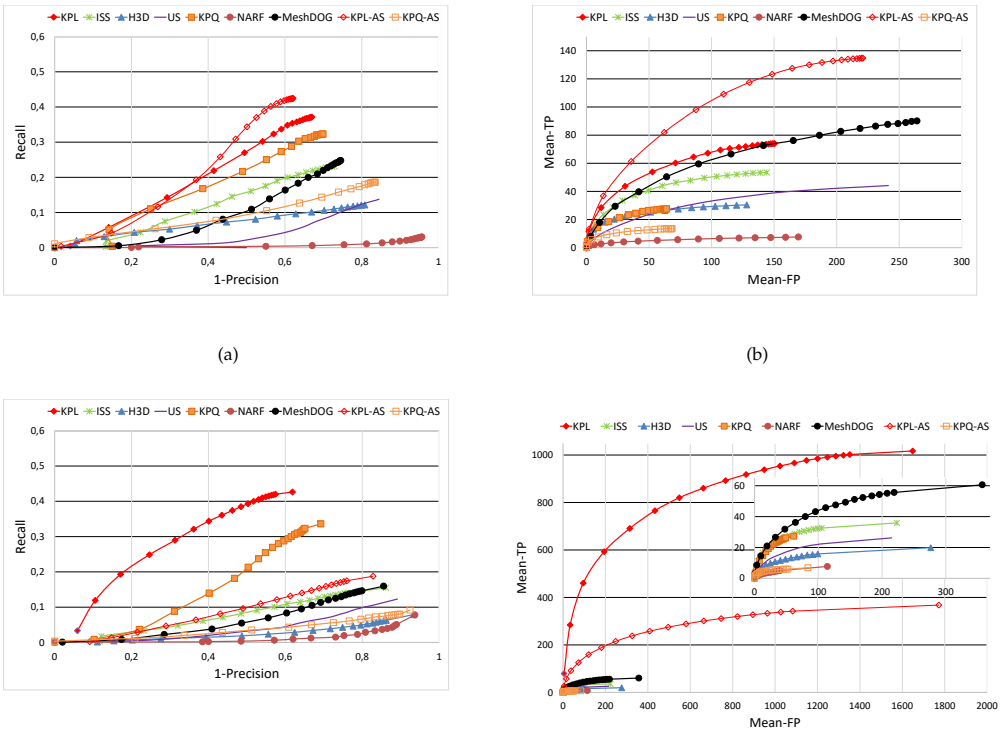


Figure 3.7: Results provided on the Laser Scanner dataset by the keypoint detector learned for Spin Images (a-b) and FPFH (c-d). Precision-Recall curves (a)-(c), Mean True Positives vs. Mean False Positives (b)-(d).

results, whereas NARF and Harris3D perform similarly to the baseline uniform sampling approach. However, our learned detectors (KPL, KPL-AS) are able to identify the best regions to be described and matched by SHOT, even more effectively than the detector specifically designed and tuned for this kind of data (i.e. KPQ), which substantiate our intuition that saliency-based detectors cannot select the best regions to optimize the performance of the overall detector-descriptor pipeline. Moreover, the use of the adaptive-scale variant of our proposal (KPL-AS) can provide higher Recall values than the fixed scale algorithm (KPL), especially in the lower precision region. Fig. 3.6 (b) highlights that KPL-AS can provide consistently a substantially higher number of correct correspondences than KPL given the same number of mismatches, and that the former detector provides the highest number of correct matches between all considered methods. This confirms the ability of the proposed methodology to learn to detect also the best scale at which the descriptor should be computed as well as the practical relevance of the adaptive-scale formulation.

Fig. 3.7 (a) and (b) shows that KPL and KPL-AS exhibit the best descriptor matching performance also when using the Spin Images descriptor, our adaptive-scale proposal delivering again the highest number of correct matches between all considered methods. Similarly, Fig. 3.7 (c) and (d) show that, when using FPFH, KPL and KPL-AS outperform, respectively, all fixed-scale and adaptive-scale detectors. Overall, the results in Fig. 3.7 vouch for the generality of the methodology proposed in this chapter. Indeed, the features deployed in our random forest classifier, although inspired by SHOT, do not force the use of learned detectors to this particular descriptor and, instead, turn out effective to learn to detect good surface patches for other descriptors alike. This can be observed also in the qualitative results reported in Fig. 3.8: though based on the same features, the saliency functions learned by our method for the three descriptors are different and, as such, fire at different regions, e.g. in the scene from the Laser Scanner dataset, i.e. (a) and (b), the chicken belly seems more amenable to establish good correspondence with Spin Images, whereas SHOT is apparently more effective across the neck. Moreover, in the scene from Random Views, i.e. (c) and (d), the saliency function learned for SHOT fires more on the body of the snake compared to that of FPFH which, in turn, seems much more effective on the body of the statuette.

3.5.3 *Transfer learning on the Random Views dataset*

The Random Views dataset is based on the Stanford 3D scanning repository [6] and originally proposed in [89]. This dataset comprises 6 full 3D models and 36 scenes obtained by synthetic renderings of random model arrangements. Scenes feature occlusions but no clutter. Moreover, scenes are corrupted by different levels of synthetic noise. In the experiments we consider scenes with Gaussian noise equal to $\sigma = 0.1$ mesh resolution units.

The performance assessment protocol is the same as described in Sec. 3.5.2. As Random Views presents a level of detail and noise comparable to that of the Laser Scanner dataset, in the experiments described in this section we do not train new classifiers in order to pursue keypoint detection but, rather, just apply those already learned on Laser Scanner. This allows us to investigate on the ability of our method to transfer the concepts learned from a set of representative objects to previously unseen shapes. In fact, a

[6] 3 http://graphics.stanford.edu/data/3Dscanrep/

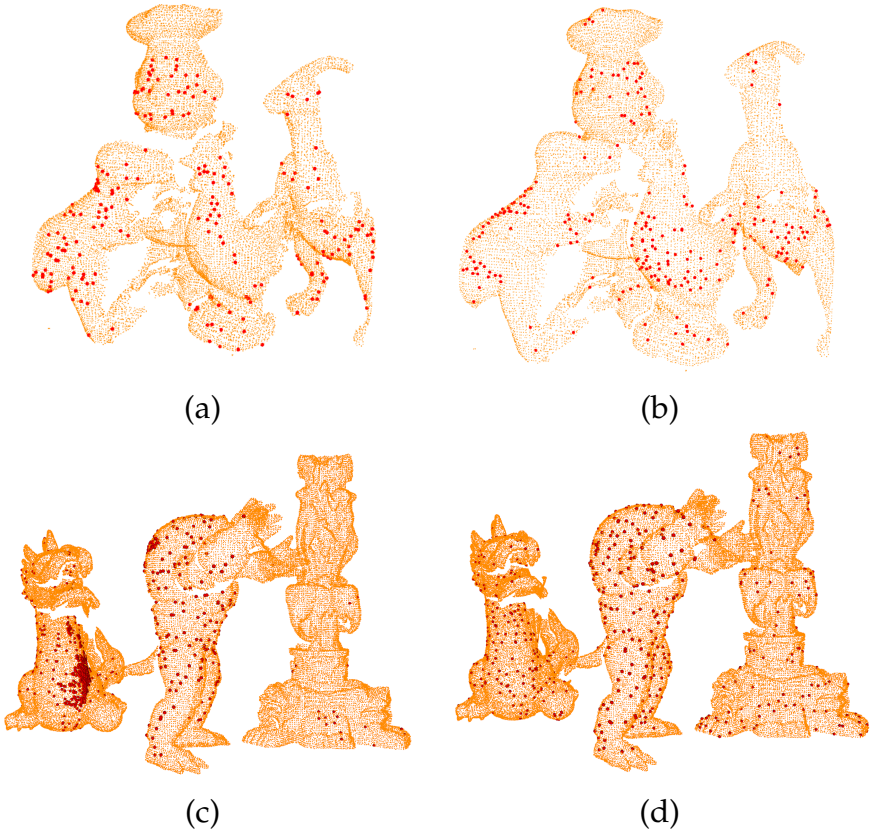
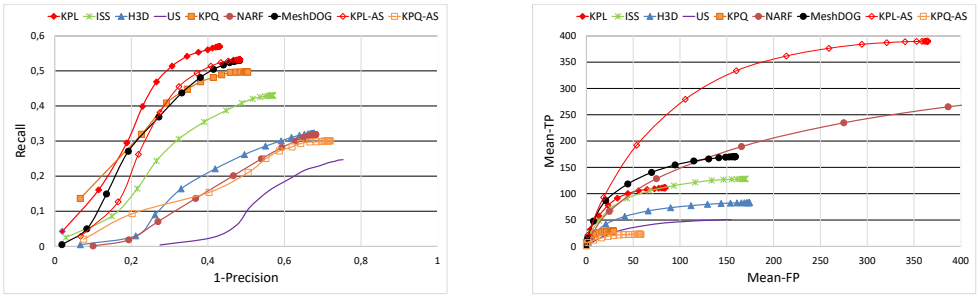


Figure 3.8: Exemplar keypoints extracted by the detectors learned for SHOT (a) (c), Spin Images (b) and FPFH (d) on scenes of the *Laser Scanner* dataset (a-b) and *Random Views* dataset (c-d).

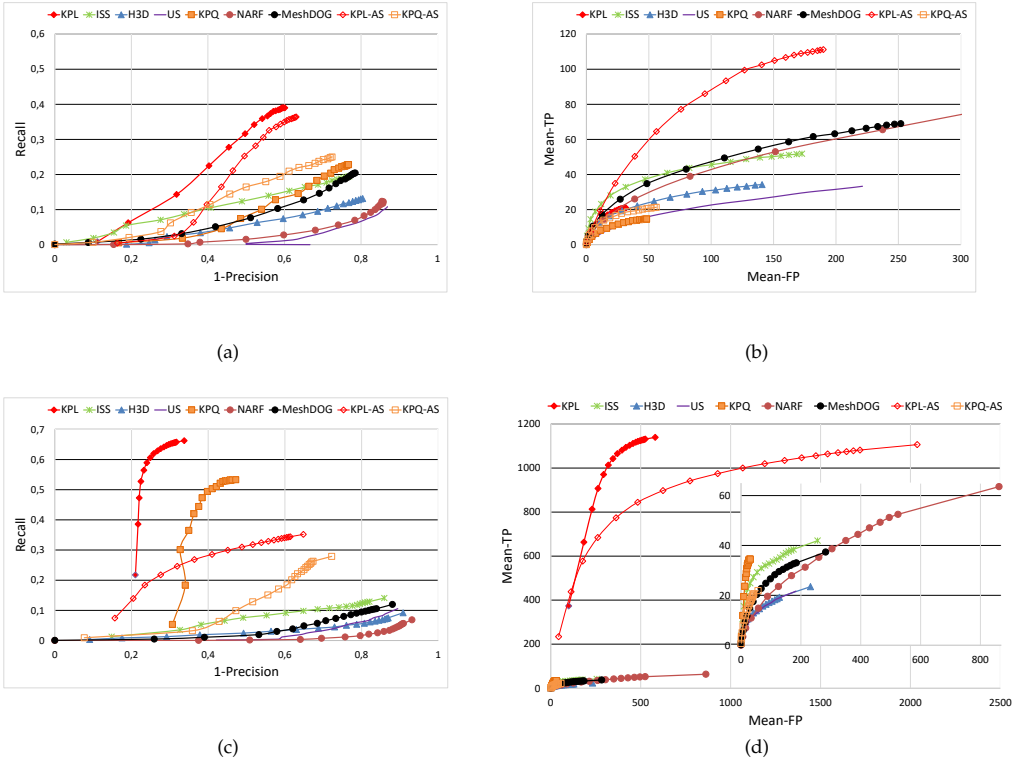
detector able to learn to detect good local features likely to appear across different datasets sharing similar traits may be conveniently trained on just one dataset and then successfully applied also to the others, thereby diminishing the overall training time vastly. Moreover, there exist relevant surface matching scenarios, such as, e.g., point cloud registration and 3D reconstruction, where a training set in the form required by our method is typically not available.

The results concerning SHOT are reported in Fig. 3.9 and show that, overall, KPL provides again the best performance. However, it is interesting to note that the gap between our proposal and KPQ widens and the ranking of saliency-based detector changes, with MeshDoG providing more distinctive regions than KPQ for this dataset: saliency-based detectors, in fact, exhibit more difficulties in maintaining a similar performance level across different datasets. Moreover, these results show that the way we create the training set, the features we propose, and the selected hyperparameters for the classifier are effective in learning a classification function with



(a) (b)

Figure 3.9: Results on Random Views by the keypoint detector learned for SHOT on Laser Scanner. Precision-Recall curves (a), Mean True Positives vs. Mean False Positives (b).



(a) (b)

(c) (d)

Figure 3.10: Results on Random Views by the keypoint detector learned for Spin Images (a-b) and FPFH (c-d) on Laser Scanner. Precision-Recall curves (a-c) , Mean True Positives vs. Mean False Positives (b-d).

(a) Laser Scanner (b) Random Views (c) Queen's 3D (d) Venezia 3D (model) (e) Venezia 3D (scene)

Figure 3.11: Universal detector: some models used to for training (a-d) and keypoints detected on a scene of the Venezia 3D dataset (e).

high generalization abilities, a very useful trait in practical deployments of learning-based algorithms which, in our settings, allows our proposal to learn to detect good local features across different datasets. On this dataset, KPL-AS is less effective than KPL: it is interesting to note that the same trend is shown by the two variants of KPQ, which may be interpreted as a symptom of an inherent ambiguity of descriptions at the selected scales for this dataset. Nonetheless, even in this more challenging experiment dealing with transferring the concepts learned from one dataset to another one, KPL-AS can yield a much larger quantity of correct correspondences than any other method at any given level of false positives (Fig. 3.9 (b)).

Similar considerations on the effectiveness of our proposal with respect to hand-crafted detectors may be drawn from the results dealing with Spin Images and FPFH (Fig. 3.10). Moreover, as for the results dealing with Spin Images, it is interesting to note how ISS turns out to be now as effective as KPQ while MeshDoG performance drops. This confirms that, in general, it is quite hard to know beforehand the best detector for a given descriptor and vouches in favor of the inherent adaptability offered by our learning-based approach. Finally, Fig. 3.9 and Fig. 3.10 provide additional support to our claim concerning the general validity of our proposal, due to the very same learning framework leading to superior performance with respect to hand-crafter detectors when deploying descriptors as diverse as SHOT, FPFH and Spin Images.

3.5.4 *Training and testing a* universal detector

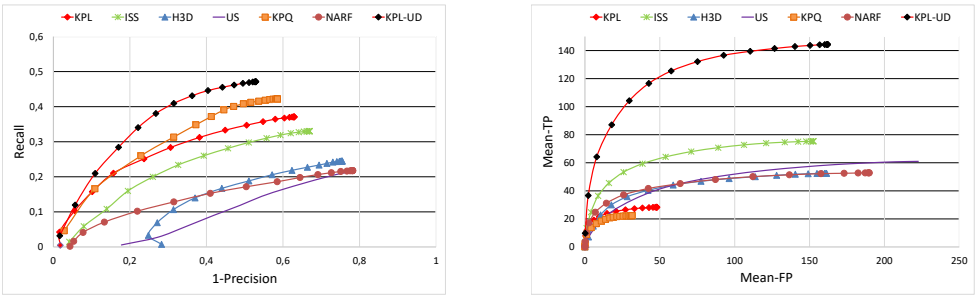
We present a different kind of experiment, which highlights another approach to leverage on our keypoint learning framework. In particular, rather than learning a specific detector for a given dataset (Sec. 3.5.2) or transferring a learned detector from one dataset to another (Sec. 3.5.3), we try to learn

the best possible saliency function for a given sensing modality and 3D descriptor based on all available datasets. We refer to the detector thus achieved as to *universal detector*.

Both the training procedure and the classifier implementation concerning the *universal detector* are the same as described in Sec. 3.2 and Sec. 3.3, the key difference consisting in the higher number and variety of models deployed to define the training set. Although we pursue here training of the fixed-scale instance of the *universal detector* only, its adaptive-scale counterpart may be obtained without additional conceptual effort by applying the methodology described in Sec. 3.4 to a similarly large and varied corpus of training data. To create the training set we deployed 59 different models coming from 4 lidar-based datasets (namely, Laser Scanner, Random Views, Queen's 3D [36] and Venezia 3D). Fig. 3.11 (a-d) shows four of the models used to train the *universal detector*, one for each dataset. About 1250000 positive and negative samples were used to create a mixed dataset that includes a large part of the 3D structures in the models.

We expect the function learned by the *universal detector* to be able to detect better keypoints than the variant trained only on one dataset, the main reason being that the models from other datasets allow to sample different types of surface patches that may be present in the unseen scenes of the test dataset. It is interesting to note that, with this kind of training, a generalization of the samples is implicitly performed by the random forest. For example, points belonging to different models but exhibiting similar neighborhoods are grouped together at the level of the leaves of each tree - in other words, each leaf can be seen as a cluster of visually similar 3D points. Also worth pointing out, the use of a high number of samples and different models forces the detector to create positive leaves that are less specific of a given object, hence producing a more general description of what is a good keypoint. Moreover, the use of a large training corpus pushes the detector to learn how to recognize more 3D structures than those trained on specific models. For this reason, during the detection stage, the *universal detector* would tend to classify a larger number of good surface patches as possible keypoints, thereby increasing the potential number of good matches.

The results achieved by testing the *universal detector* learned for SHOT on the scenes of Venezia 3D are reported in Fig. 3.12, in addition a qualitative example is also shown in Fig. 3.11(e). To verify our intuition that a large and varied training set leads to a better detection function than using the models from a dataset alone, in Fig. 3.12 we compare the universal detector (denoted



(a) (b)

Figure 3.12: Results on Venezia 3D for the universal detector trained for SHOT. Precision-Recall curves (a), Mean True Positives vs. Mean False Positives (b).

as KPL-UD) to the detector trained only on the models belonging to the Venezia 3D dataset (denoted as KPL, coherently with previous experiments). While KPL obtains performance similar to KPQ, the use of a larger and more varied training set allows the Random Forest to learn a far better saliency function, which enables KPL-UD to outperform all other detectors by a large margin, both in terms of Precision-Recall, as well as, even more evidently, in terms of absolute number of correct correspondences.

3.5.5 *Kinect dataset*

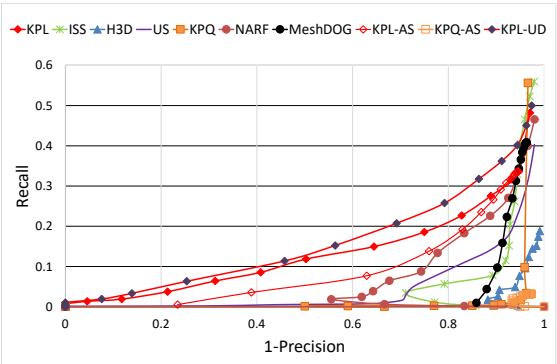


Figure 3.13: Results on the Kinect dataset by the keypoint detector learned for SHOT.

The Kinect dataset comprises 7 models provided in the form of 2.5D views together with 17 scenes where the models are acquired under heavy clutter

and occlusions. Due to acquisition by means of a low-cost consumer depth camera, the data is also quite noisy.

To compare detectors on Kinect we follow the evaluation protocol described in the paper that introduced this dataset [98]. Accordingly, first keypoints are extracted and described from all model views. Then, descriptors are extracted from scenes, both at the locations of model keypoints as well as from clutter. Every descriptor computed from the scene is then matched against the set of model descriptors by the ratio criterion [28] and checked for geometric correctness. Varying the threshold of the ratio test, allows for drawing Precision-Recall curves as in Fig. 3.13.

The Kinect dataset is particularly challenging, such that, as also found in a recent experimental evaluation [124], the performance of the feature matching pipelines decreases dramatically with respect to previous datasets (see also Fig. 5 in [124]). It is worth noticing how, in the results dealing with matching SHOT descriptors reported in Fig. 3.13, the relative ordering of hand-crafted detectors does change: on the one hand, the baseline uniform sampling approach performs better than KPQ, ISS, and Harris3D; on the other hand, NARF, whose performance was quite unsatisfactory on previous datasets, turns out the best hand-crafted detector on Kinect. Again, our learned detectors (KPL, KPL-AS) clearly surpass existing proposals, which vouches for the better ability to adapt to different sensing modalities of the machine learning approach advocated in this chapter.

Given that the training set for the Kinect dataset consists of fewer points compared to the other datasets, e.g. $\sim$ 8000 positive samples versus $\sim$ 100000 in Laser Scanner, and motivated by the results obtained with the lidar-based *universal detector* (Sec. 3.5.4), we have carried out another experiment aimed at training our detector by a larger number of models. In particular, we have created a fixed-scale training set using the models from the Kinect and the *TUW*[91] datasets, both including real 2.5D calibrated views acquired by low-cost consumer depth camera, so as to obtain $\sim$ 70000 positive training samples and to train the Random Forest according to the standard methodology and parameters (Table 3.5). Fig. 3.13 shows that this new classifier, referred to as KPL-UD, provides slightly superior performance with respect to KPL on the scenes of the Kinect dataset. This is consistent with the finding of Sec. 3.5.4 about training with more models turning out beneficial to learning the keypoint detection function, even though these additional models are not going to appear within the actual scenes used for testing,

which, in essence, vouches for a machine learning framework amenable to generalize well rather than overfit.

4

PERFORMANCE EVALUATION OF LEARNED 3D FEATURES

In Chap. 3, we have proposed a novel machine-learning framework to address the problem of 3D keypoint detection. In this chapter, instead, we want to go one step further and investigate how the state-of-the-art proposals for 3D detectors and descriptors can be coupled to create an effective pipeline. Indeed, unlike the related field of *local image features*, methods to either detect or describe 3D features have been designed and proposed separately, alongside with specific application settings and related datasets. Hence, so far, it is yet unclear how to effectively combine a 3D keypoint detector with a 3D keypoint descriptor. This is also vouched by the main performance evaluation papers in the field, which address either repeatability of 3D detectors designed to highlight geometrically salient surface patches [89] or distinctiveness and robustness of popular 3D descriptors [124]. In the object recognition experiment in Sec. 3.5.2 we have shown that, with the considered descriptors (SHOT [98], Spin Image (SI) [18], FPFH [45]), learning to detect specific keypoints leads to better performance than relying on existing general-purpose handcrafted detectors (ISS [48], Harris3D [65], NARF [57]). By enabling an optimal detector to be learned for any descriptor, we sets forth a novel paradigm to maximize affinity between 3D detectors and descriptors. This opens up the question of which learned detector-descriptor pair may turn out most effective in the main application areas. This chapter tries to answer this question by proposing an experimental evaluation of learned 3D pipelines. In particular, we address 3D object recognition and surface registration and compare the performance attained by learning a paired feature detector for the most popular handcrafted 3D descriptors (SHOT [98], SI [18], FPFH [45], USC[58], RoPS [83]) as well as for a recently proposed descriptor based on deep learning (*CGF-32* [143]).

Within this section we briefly review state-of-the-art handcrafted and learned methods for description of 3D local features. For a more exhaustive discussion please refer to Sec. 6.1.

Many hand-crafted feature descriptors represent the local surface by computing geometric measurements within the supporting patch and then accumulating values into histograms. *Spin Images (SI)* [18] relies on two coordinates to represent each point in the support: the radial coordinate, defined as the perpendicular distance to the line trough the surface normal at the keypoint, and the elevation coordinate, defined as the signed distance to the tangent plane at the keypoint. The space formed by this two values is then discretized into a 2D histogram.

In *3D Shape Context (3DSC)* [27] the support is partitioned by a 3D spherical grid centered at the keypoint with the north pole aligned to the surface normal. A 3D histogram is built by counting up the weighted number of points falling into each spatial bin along the radial, azimuth and elevation dimensions. *Unique Shape Context (USC)* [58] extends *3DSC* with the introduction of a unique and repeatable canonical reference frame borrowed from [98].

SHOT [98], alike, deploys both a unique and repeatable canonical reference frame as well as a 3D spherical grid to discretize the supporting patch into bins along the radial, azimuth and elevation axes. Then, the angles between the normal at the keypoint and those at the neighboring points within each bins are accumulated into local histograms. *Rotational Projection Statistics (RoPS)* [83] uses a canonical reference frame to rotate the neighboring points on the local surface. The descriptor is then constructed by rotationally projecting the 3D points onto 2D planes to generate three distribution matrices. Finally, a histogram encoding five statistics of distribution matrices is calculated. *Fast Point Feature Histograms (FPFH)* [45] operates in two steps. In the first, akin to PFH [42], four features, refereed to as SPFH, are calculated using the Darboux frame and the surface normals between the keypoint and its neighbors. In the second step, the descriptor is obtained as the weighted sum between the SPFH of the keypoint and the SPFHs of the neighboring points.

The success of deep neural networks in so many challenging image recognition tasks has motivated research on learning representations from 3D data. One of the pioneering works is *3D Match* [159], where the authors

deploy a siamese network trained on local volumetric patches to learn a local 3D descriptor. The input to the network consists of a Truncated Signed Distance Function (TSDF) defined on a voxel grid. In [143], the authors deploy a fully-connected deep neural network together with a feature learning approach based on the *triplet ranking loss* in order to learn a very compact 3D descriptor, referred to as *CGF-32*. Their approach does not rely on raw data but on an hand-crafted input representation similar to [27], canonicalized by the local reference frame presented in [98].

4.2 KEYPOINT LEARNING

In order to carry out the performance evaluation proposed in this chapter, we extend the framework proposed in Chap. 3 for most local descriptors reviewed in Sec. 4.1. We consider only the case of fixed-scale detector. Hence, before we go any further, we briefly recall the methodology and refer the reader to [113, 181] and Sec. 3.2 for a detailed description.

The idea behind keypoint learning is to learn to detect keypoints that can yield good correspondences when coupled with a given descriptor. To this end, keypoint detection is cast as binary classification, *i.e.* a point can either be a good candidate or not when used to create matches by means of the given descriptor, and a Random Forest is used as classifier. Training of the classifier requires to define the training set, *i.e.* both positive (good) and negative (not good) points, as well as the feature representation.

As for positive samples, the method tries to sift out those points that, when described by a chosen descriptor, can be matched correctly across different 2.5D views of a 3D object. Thus, starting from a set of 2.5D views $\{V_i\}, i = 1, \dots, N$ of an object from a 3D dataset, each point $p \in V_i$ in each view V_i is embedded by the chosen descriptor. Then, for each view V_i, a subset of overlapping views is selected based on an overlap threshold τ. A two-step positive samples selection is performed on V_i and each overlapping view V_j. In the first step, a list of correspondences between descriptors is created by searching for all descriptors $d \in V_i$ the nearest neighbor in the descriptor space between all descriptors $g \in V_j$. A preliminary list of positive samples P_i^j for view V_i is created by taking only those points that have been correctly matched in V_j, *i.e.* the points belonging to the matched descriptors in the two views correspond to the same 3D point of the object according to threshold ϵ. The list is then filtered removing *non-maxima* local

extrema within ϵ_{nms} using the descriptor distance as saliency. In the second step, the list of positive samples is refined by keeping only the points in V_i that can be matched correctly also in those others overlapping views that have not been used in the first step. Negative samples are then extracted on each view, sampling random points among those points which are not included in the positive set. A distance threshold ϵ_{neg} is used to avoid a negative being too close to a positive and to other negative samples, and also to balance the size of the positive and negative sets.

As far as the representation input to the classifier is concerned, the method relies on histograms of normal orientations inspired by SHOT [98]. However, to avoid computation of the local Reference Frame while still achieving rotation invariance, the spherical support is divided only along the radial dimension so as to compute a histogram for each spherical shell thus obtained.

4.3 EVALUATION METHODOLOGY

The performance evaluation here proposed aims to compare different learned detector-descriptor pairs while addressing two main application settings, namely 3D object recognition and surface registration. In this section, we highlight the key traits and nuisances which characterize the two tasks, present the datasets and performance evaluation metrics used in the experiments and, finally, provide the relevant implementation details.

4.3.1 *3D object recognition*

An introduction about 3D object recognition settings has already been provided in Sec. 3.5. However, to make the chapter self-contained we discuss in details the methodology adopted. In typical 3D object recognition settings, one wishes to recognize a set of given 3D models into scenes acquired from an unknown vantage point and featuring an unknown arrangement of such models. Peculiar nuisances in this scenario are occlusions and, possibly, clutter, as objects not belonging to the model gallery may be present in the scenes. In our experiments we rely on the two popular object recognition datasets: Laser Scanner and Random Views introduced in Sec. 3.5. Two examples of scenes are showed in Fig. 4.1.

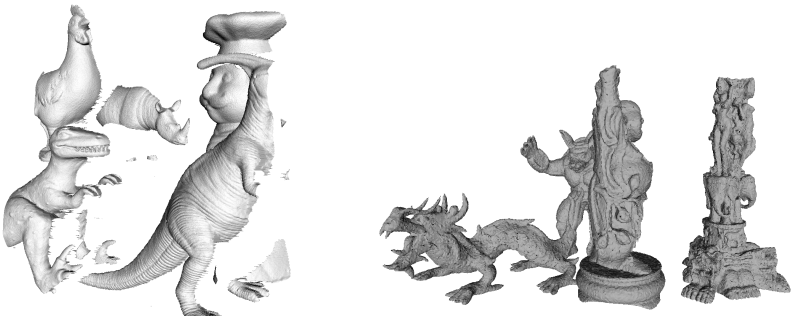


Figure 4.1: Scene from the Laser Scanner (left) and Random Views (right) datasets.

To evaluate the effectiveness of the considered learned detector-descriptor pairs we rely on descriptor matching experiments. Specifically, for both datasets, we run keypoint detection on synthetically rendered views of all models. Then, we compute and store into a single kd-tree all the corresponding descriptors. Keypoints are detected and described also in the set of scenes provided with the dataset, $\{S_j\}, j = 1, \ldots, N_S$. Eventually, a correspondence is established for each scene descriptor by finding the nearest neighbor descriptor within the models kd-tree and thresholding the distance between descriptors to accept a match as valid. Correct correspondences can be identified based on knowledge of the ground-truth transformations which bring views and scenes into a common reference frame and checking whether the matched keypoints lay within a 3D distance ϵ. Indeed, denoting as $(k_j, k_{n,m})$ a correspondence between a keypoint k_j detected in scene S_j and a keypoint $k_{n,m}$ detected in the n-th view of model m, as $T_{j,m}$ the transformation from S_j to model m, as $T_{n,m}$ the transformation from the n-th view and the canonical reference frame of model m, the set of correct correspondences for scene S_j is given by:

$$\mathcal{C}_j = \{(k_j, k_{n,m}) : \|T_{j,m}k_j - T_{n,m}k_{n,m}\| \leqslant \epsilon\} \tag{4.1}$$

From $\mathcal{C}_j$, we can compute True Positive and False Positive matches for each scene and, by averaging them across scenes, for each of the considered datasets. The final results for each dataset are provided as *Recall vs. 1-Precision* curves, with curves obtained by varying the threshold on the distance between descriptors.

4.3.2 *Surface Registration*

The goal of surface registration is to align into a common 3D reference frame several partial views (usually referred to as scans) of a 3D object obtained by a certain optical sensor. This is achieved through rather complex procedures that, however, typically rely on a key initial step, referred to as *Pairwise Registration*, aimed at estimating the rigid motion between any two views by a *feature-matching pipeline*. Thus, in surface registration, 3D feature detection, description and matching are instrumental to attain an as good as possible set of pairwise alignments between the views which then undergoes further processing to get the final global alignment. Differently from object recognition scenarios, the main nuisances deal with missing regions, self-occlusions, limited overlap area between views and point density variations. In our experiments we rely on the following surface registration dataset:

- *CGF-Laser Scans* dataset, recently proposed in [143]. This dataset includes 8 public-domain 3D models, *i.e.* 3 taken from the AIM@SHAPE repository (*Bimba*, *Dancing Children* and *Chinese Dragon*), 4 from the Stanford 3D Scanning Repository (*Armadillo*, *Buddha*, *Bunny*, *Stanford Dragon*) and *Berkeley Angel* According to the protocol described in [143], training should be carried out based on synthetic views generated from *Berkeley Angel*, *Bimba*, *Bunny* and *Chinese Dragon*, while the test data consists of the the real scans available for the remaining 3 models (*Armadillo*, *Buddha* and *Stanford Dragon*).

Thus, given a set of M real scans available for a test model, we compute all the possible $N = \frac{M(M-1)}{2}$ view pairs $\{V_i, V_j\}$. For each pair, we run keypoint detection on both views. Due to partial overlap between the views, a keypoint belonging to V_i may have no correspondence in V_j. Hence, denoted as T_i and T_j the ground-truth transformations that, respectively, bring V_i and V_j into a canonical reference frame, we can compute the set $\mathcal{O}_{i,j}$ that contains the keypoints in V_i that have a corresponding point in V_j. In particular, given a keypoint $k_i \in V_i$:

$$\mathcal{O}_{i,j} = \{k_i : \|T_i k_i - \mathcal{NN}(T_i k_i, T_j V_j)\| \leqslant \epsilon_{ovr}\} \tag{4.2}$$

where $\mathcal{NN}(T_i k_i, T_j V_j)$ denotes the nearest neighbor of $T_i k_i$ in the transformed view $T_j V_j$. If the number of points in $\mathcal{O}_{i,j}$ is less than 20% of the keypoints in V_i, the pair (V_i, V_j) is not considered in the evaluation experi-

ments due to insufficient overlap. Conversely, for all the view pairs (V_i, V_j) exhibiting sufficient overlap, a list of correspondences between all the keypoints detected in V_i and all the keypoints extracted from V_j is established by finding the nearest neighbor in the descriptor space via kd-tree matching. 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, $\mathcal{C}_{i,j}$, can be identified based on the available ground-truth transformations by checking whether the matched keypoints lay within a certain distance ϵ in the canonical reference frame:

$$\mathcal{C}_{i,j} = \{(k_i, k_j) : \|T_i k_i - T_j k_j\| \leqslant \epsilon\} \tag{4.3}$$

Then, the *precision* of the matching process can be computed as a function of the distance threshold ϵ [143]:

$$\text{precision}_{i,j}(\epsilon) = \frac{|\mathcal{C}_{i,j}|}{|\mathcal{O}_{i,j}|} \tag{4.4}$$

The *precision* score associated with any given model is obtained by averaging across all view pairs. We also average across all test models so as to get the final score associated to the Laser Scanner dataset.

Table 4.1: Parameters for object recognition datasets.

Dataset	$r_{desc}(mm)$	$r_{det}(mm)$	τ	$\epsilon(mm)$	$\epsilon_{nms}(mm)$	$\epsilon_{neg}(mm)$	$r_{nms}(mm)$	$s_{min}(mm)$
Laser Scanner	40	20	0.85	7	4	2	4	0.8
Random Views	40	20	-	7	-	-	4	0.8

Table 4.2: Parameters for surface registration dataset.

Model Name	$r_{desc}(mm)$	$r_{det}(mm)$	τ	$\epsilon(mm)$	$\epsilon_{nms}(mm)$	$\epsilon_{neg}(mm)$	ϵ_{ovr}	$r_{nms}(mm)$	$s_{min}(mm)$
Angel	40	20	0.85	7	4	2	-	-	-
Bimba	40	20	0.85	7	4	2	-	-	-
Bunny	40	20	0.65	7	4	2	-	-	-
Chinese Dragon	40	20	0.65	7	4	2	-	-	-
Armadillo	40	20	-	7	-	-	2	4	0.5
Buddha	40	20	-	7	-	-	2	4	0.5
Stanford Dragon	40	20	-	7	-	-	2	4	0.5

4.3.3 *Implementation*

For all handcrafted descriptors considered in our evaluation, we use the implementation available in the PCL library. For *CGF-32*, we adopt the

public implementation made available by the authors [143]. As for the KPL framework we keep all the hyperparameters to the same values used in Sec. 3.5.1. Accordingly, each forest consists of 100 trees of maximum depth equal to 25 while the minimum number of samples to stop splitting a node is 1. During the detection phase, each point of a point cloud is passed through the Random Forest classifier which produces a score. A point is identified as a keypoint if it exhibits a local maximum of the scores in a neighborhood of radius r_{nms} and the score is higher than a threshold s_{min}. For each descriptor considered in our evaluation, we train its paired detector according to the KPL framework. As a result, we obtain six detector-descriptor pairs, referred to from now on as *KPL-CGF32*, *KPL-FPFH*, *KPL-ROPS*, *KPL-SHOT*, *KPL-SI*, *KPL-USC*.

In object recognition experiments, the training data for all detectors are generated from the 4 full 3D models present in the Laser Scanner dataset. According to the KPL methodology [113], for each model we render views from the nodes of an icosahedron centered at the centroid. Some of the generated views are presented in Fig. 4.2.

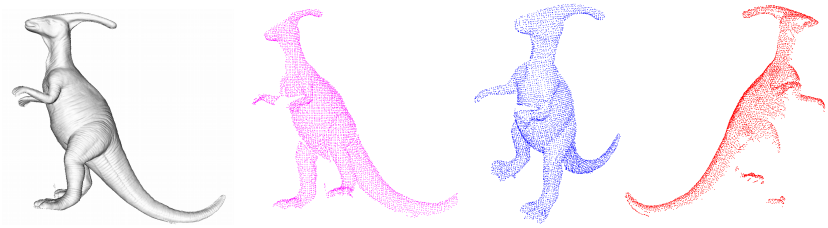


Figure 4.2: A 3D model and some rendered views from Laser Scanner.

Then, the detectors are used in the scenes of the Laser Scanner dataset as well as in those of the Random Views dataset. Thus, similarly to Sec. 3.5.3 we do not retrain the detectors on Random Views in order to test the ability of the considered detector-descriptor pairs to generalize well to unseen models in object recognition settings. A coherent approach was pursued for the CGF-32 descriptor. As the authors do not provide a model trained on the Laser Scanner dataset, we trained the descriptor on the synthetically rendered views of the 4 Laser Scanner models using the code provided by the authors and following the protocol described in the paper in order to generate the data needed by their learning framework based on the *triplet ranking loss*. Thus, KPL-CGF32 was trained on Laser Scanner models and,

like all other detector-descriptor pairs, tested on both Laser Scanner and Random Views scenes.

In surface registration experiments we proceed according to the protocol proposed in [143]. Hence, detectors are trained with rendered views of the train models provided within the CGF-Laser scans dataset (*Angel*, *Bimba*, *Bunny*, *Chinese Dragon*) and tested on the real scans of the test models (*Armadillo*, *Buddha*, *Stanford Dragon*). As CGF-32 was trained exactly on the abovementioned train models [143], to carry out surface registration experiments we did not retrain the descriptor but used the trained network published by the authors[1].

The values of the main parameters of the detector-descriptor pairs used in the experiments are summarized in Tab. 4.1 and Tab. 4.2. As it can be observed from Tab. 4.1, train parameters for Random Views dataset are not specified as we did not train KPL detectors on this dataset. For surface registration, since models belong to different repositories, we report parameters grouped by model. Test parameters for Angel, Bimba, Bunny and Chinese Dragon are not reported as they are only used in train. Similarly, we omit train parameters for Armadillo, Buddha and Stanford Dragon. Due to the different shapes of the models in the dataset, τ is tuned during the train stage so that the number of overlapping views remains constant across all models.

4.4 EXPERIMENTAL RESULTS

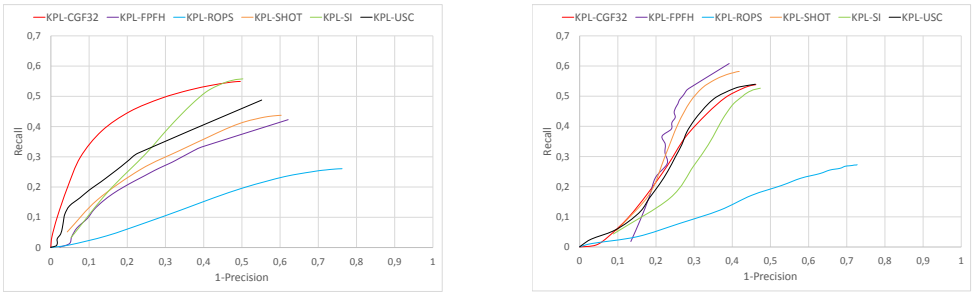
4.4.1 *3D object recognition*

Results on the Laser Scanner dataset are shown in Fig. 4.3. First, we wish to highlight how the features based on descriptors which encode just the spatial densities of points around a keypoint outperform those relying on higher order geometrical attributes (such as, *e.g.*, normals). Indeed, KPL-CGF32, KPL-USC and KPL-SI yield significantly better results than KPL-SHOT and KPL-FPFH. These results are coherent with the findings and analysis reported in the performance evaluation by Guo *et al.* [124], which pointed out the former feature category being more robust to clutter and sensor noise. It is also worth observing how the representation based on the spatial tessellation and point density measurements proposed in [27] together with

[1] https://github.com/marckhoury/CGF

the local reference frame proposed in [98] turn out particularly amenable to object recognition, as it is actually deployed by both features yielding neatly the best performance, namely KPL-CGF32 and KPL-USC. Yet, learning a dataset-specific non-linear mapping by a deep neural network on top of this good representation does improve performance quite a lot, as vouched by KPL-CGF32 outperforming KPL-USC by a large margin. Indeed, the results obtained in this chapter by learning both a dataset-specific descriptor as well as its paired optional detector, *i.e.* the features referred to as KPL-CGF32, turn out significantly superior to those previously obtained on Laser Scanner object recognition dataset in Sec. 3.5.2.

In Sec. 3.5.3, the results achieved on Random Views by the detectors trained on Laser Scanner prove the ability of the KPL methodology to learn to detect general rather than dataset-specific local shapes amenable to provide good matches alongside with the paired descriptor, and even more effectively, in fact, than the shapes found by handcrafted detectors. Thus, when comparing the different features, we can assume here that descriptors are feed by detectors with optimal patches and focus on the ability of the former to handle the specific nuisances of the Random Views dataset. As shown in Fig. 4.3, *KPL-FPFH* and *KPL-SHOT* perform slightly better than KPL-USC, KPL-CGF32 and KPL-SI. Again, this is coherent with previous findings reported in literature (see [124] and [181]), which show how descriptors based on higher order geometrical attributes turn out more effective on Random Views due to the lack of clutter and real sensor noise. As for KPL-CGF32, although it performs still overall better than the other descriptors based on point densities, we observe quite a remarkable performance drop compared to the results on the Laser Scanner dataset, much larger, indeed, than that observed for KPL-USC, which shares with KPL-CGF32 a very similar input representation. This suggests that the non-linear mapping learned by KPL-CGF32 is highly optimized to tell apart the features belonging to the objects present in the training dataset (*i.e.* Laser Scanner) but turns out quite less effective when applied to unseen features, like those found on the objects belonging to Random Views. This *domain shift* issue is a peculiar wick trait of learned features, which may cause them to yield less stable performance across diverse datasets than handcrafted representations.



(a) (b)

Figure 4.3: Quantitative results on 3D object recognition. Column a: Laser Scanner dataset. Column b: Random Views dataset.

4.4.2 *Surface Registration*

First, it is worth pointing out how, unlike in object recognition settings, in surface registration it is never possible to train any supervised machine learning operator, either detector or descriptor, on the very same objects that would then be processed at test time. Indeed, should one be given either a full 3D model or a set of scans where ground-truth transformations are known, as required to train 3D feature detectors (*i.e.* KPL) or descriptors (e.g. CGF-32), there would be no need to carry out any registration for that object. Surface registration is about stitching together several scans of a new object than one wishes to acquire as a full 3D model. As such, any learning machinery is inherently prone to the domain shift issue.

As mentioned in Sec. 4.3.2, our experiments rely on the CGF-Laser Scans dataset [143] and follow the split into train and test objects proposed by the authors. As shown in Fig. 4.4, when averaging across all test objects, the detector-descriptor pair based on the learned descriptor CGF-32 provides the best performance. This validates the findings reported in [143], where the authors introduce CGF-32 and prove its good registration performance on CGF-Laser scans, also in our experimental setting where an optimal detector is learned for every descriptor. This last experiment is particularly interesting due to CGF-32 consisting in a non-linear mapping learnt on top of the input representation deployed by USC. Unlike in object recognition, though, this mapping can only be learnt on different objects than those seen at test time and, hence, because of the domain shift, may not always turn out beneficial.

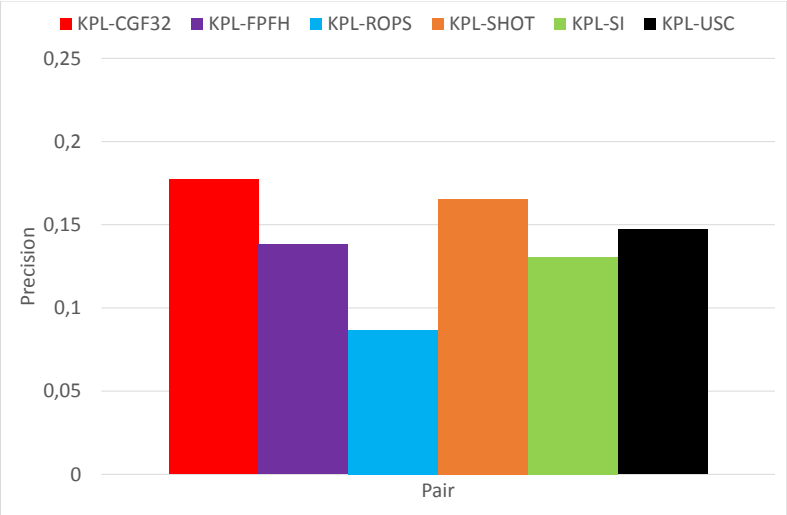


Figure 4.4: Surface registration results on the CGF-Laser scans dataset.

5

CONCLUSIONS

In this part, we have extensively addressed the problem of 3D keypoint detection, first by presenting a general methodology that extends [113] (Chap. 3), and then by coupling the learned detector with some of the state-of-the-art proposal for 3D keypoint description in two challenging applications, 3D object recognition and surface registration. In particular in Chap. 3, we have shown how a Random Forest can learn to detect interest regions on which a given 3D descriptor yields a higher number of good matches compared to keypoints extracted by maximizing a geometric and descriptor-agnostic saliency function. Key to the approach is the procedure to define the positive training samples, which incorporates knowledge of the chosen descriptor to allow the classifier to learn a descriptor-specific saliency robust to both viewpoint variations as well as nuisances peculiar to the 3D sensor. The proposed approach is conducive to both fixed-scale and adaptive scale keypoint detection. The latter is realized by leveraging on the ability of Random Forests to handle multi-class classification seamlessly, thereby empowering the learned detector with the capacity of selecting the optimal support size to compute the chosen descriptor and increasing the quantity of correct correspondences provided by the 3D feature matching pipeline. Although the features deployed by the Random Forest classifiers resemble the 3D representation introduced by SHOT [98], the proposed keypoint learning methodology is by no means bound to work with this specific descriptor only. Indeed, in the evaluation proposed in Chap. 4 we show how it may be leveraged successfully with any 3D descriptor. As a promising direction for future work concerns deformable shape matching [47, 62]. Although in this domain dense point-to-point correspondences between surfaces are typically sought for, keypoint detection has proven effective to determine global similarities between non-rigid shapes. This is the case, for example, of [47], that proposes a method to robustly extract keypoints in order to detect persistent structures across a large set of deformable models, as well as to recognize the symmetric structure of a certain model. Also interestingly, in [62] keypoint detection is deployed to drive the dense matching process, so to match first distinctive keypoints to reduce the

number of potential candidates for the remaining points. This relies on the idea that a few reliable keypoint correspondences can constrain the space of possible solutions and, thus, vastly diminish matching ambiguity. Our keypoint learning methodology may be applied in deformable shape matching scenarios by leveraging existing datasets endowed with ground-truth point-to-point correspondences in order to sift out as positive samples those points that may be matched successfully by the chosen descriptor despite isometric shape deformations, topology changes and varying point density.

As another fundamental contribution of this chapter, the performance evaluation in Chap. 4 offers an interesting discussion about the adoption of learning-based methods in computer vision applications. Therefore, we found out how 3D object recognition settings turn out quite amenable to deploy learned 3D features. Indeed, one can train upon a set of 3D objects available beforehand, *e.g.* due to scanning by some sensor or as CAD models, and then seek to recognize them into scenes featuring occlusions and clutter. These settings allow for learning an highly specialized descriptor alongside its optimal paired detector so to achieve excellent performance. In particular, the learned pair referred to in this chapter as KPL-CGF32 sets the new state of the art in descriptor matching on the Laser Scanner benchmark dataset. Although the learned representation may not exhibit comparable performance when transferred to unseen objects, in object recognition it is always possible to retrain on the objects at hand to improve performance. An open question left to future work concerns whether the input parametrization deployed by CGF-32 may enable to learn an highly effective non-linear mapping also in datasets characterized by different nuisances (*e.g.* CGF-Laser Scans) or one should better try to learn 3D representations directly from raw data, as vouched by the success of deep learning from image recognition. Features based on learned representations, such as KPL-CGF32, are quite effective also in surface registration, although this scenario is inherently more prone to the domain shift issue and, indeed, features based on handcrafted descriptors, like in particular KPL-SHOT and KPL-USC, turn out very competitive. We believe that these findings may pave the way for further research on the recent field of learned 3D representations, in particular in order to foster addressing domain adaptation issues, a topic investigated more and more intensively in nowadays deep learning literature concerned with image recognition. Indeed, 3D data are remarkably diverse in nature due to the variety of sensing principles and related technologies and we witness a lack

of large training datasets, *e.g.* at a scale somehow comparable to ImageNet, that may allow learning representations from a rich and varied corpus of 3D models. Therefore, how to effectively transfer learned representations to new scenarios seems a key issue to the success of machine/deep learning in the most challenging 3D computer vision tasks. Finally, KPL has established a new framework whereby one can learn an optimal detector for any given descriptor. In Chap. 4 we have shown how applying KPL to a learned representation (CGF-32) leads to particularly effective features (KPL-CGF32), in particular when pursuing object recognition. Yet, according to the KPL methodology, the descriptor (*e.g.* CGF-32) has to be learned before its paired detector: one might be willing to investigate on whether and how a single end-to-end paradigm may allow learning both component jointly so as to further improve performance.

Part II

LEARNING TO DESCRIBE 3D KEYPOINTS

6

INITIAL REMARKS

In Part I we widely discussed about 3D keypoint detection, the first stage in the feature matching pipeline. In this chapter, instead, we are going to focus on the second step: 3D keypoint description. This stage consists in creating a compact but distinctive representation of the geometry of the surface involved, referred to as *descriptor*. In order to correctly match features across shapes, descriptors are designed to be *invariant* or *robust* to a large set of nuisances that are often encountered in 3D vision applications, like rotation, point density variations, sensor noise, view point changes, occlusions and clutter. These challenging working conditions make offering an effective solution to the problem of 3D keypoint description far from being an easy task. Moreover, it is not obvious which are the most suited geometric attributes of the surface to encode in order to produce a *robust* and *descriptive* descriptor, as highlighted in the most recent evaluation in the field [124]. According to [124], two key aspects for a 3D local descriptor are:

- *descriptiveness*: referred to as the capacity to capture predominant traits of the surface.
- *robustness*: referred to as the ability 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 in the given application [124]. Depending on the surface area deployed as support, we can classify descriptors as either *global* or *local*. *Global* methods encode the entire 3D model in the descriptor [24, 56, 68] and are usually applied after a segmentation step to limit the influence of clutter. *Local* descriptors, instead, encode a small part of the surface, *i.e.* a patch around the feature point [18, 45, 82, 94, 98]. 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 and, as a result, a variety of methods have been proposed in literature. Before the advent of

the deep learning revolution, scholars have handcrafted their descriptors through the extraction of features computed over the points of the surfaces. Once extracted, the distribution of the feature were discretized according to a domain of quantization and approximated by means of *histograms*. The resulting descriptor is a collection of scalar values arranged into a multidimensional vector. Handcrafted local 3D descriptors can be grouped in *spatial distribution histogram* and *geometric attribute histogram* [124]. 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. Recently, data-driven approach to learn descriptive features from 3D data by means of deep neural network have been proposed [159, 161, 162, 191]. Hence, a more appropriate classification deals with: those encoding handcrafted 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. However, 3D data representations sharply differ from the images. In particular, the unorganized structure of point clouds makes the adaptation of popular deep learning techniques, *e.g.* CNNs, to the 3D domain not straightforward. Another critical issue for robustness of learned 3D descriptors is achieving rotation invariance by only means of data augmentation. While the former may be solved by parameterizing the input data by voxel grid or high-dimensional representation, for the latter two possible solutions have been explored: the points within the support of a feature point can be either parametrized with rotation invariant features or canonically oriented w.r.t. a local reference frame, before being sent as input to a deep neural network. To address these problems, in Chap. 7, we are going to present a 3D local descriptor learned in an unsupervised way from raw point clouds data. We employ a Spherical CNNs encoder to learn an equivariant embedding that can be turned into an invariant descriptor a test time with the help of an off the shelf local reference frame. But first in Sec. 6.1, we are going to provide an extensive review of the main proposals in the literature of 3D local descriptors and the way they have evolved from handcrafted proposals to those based on deep learning.

In Fig. 6.1 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* [17], *Spin Images* [18], *3D Shape Context* [27], *Local Surface Patch* [35], *Point Feature Histograms* [42], *Scale-Dependent Local Shape Descriptor* [40], *Heat Kernel Signature* [47], *Fast Point Feature Histograms* [45], *Intrinsic Shape Signature* [48], *MeshHog* [80], *3D Surf* [54], *SHOT* [98], *Unique Shape Context* [58], *Rops* [94], *Trisi* [82], *Toldi* [157], *3D Match* [159], *CGF* [143], *PPF-Net* [162] and *PPF-FoldNet* [161].

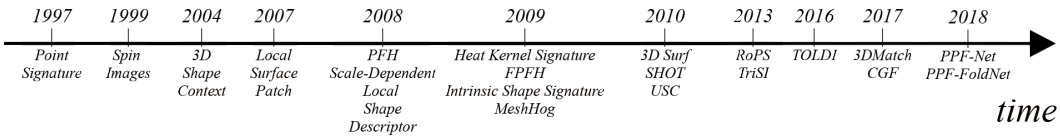


Figure 6.1: Temporal evolution of 3D feature descriptors.

6.1.1 *3D Shape Context*

3D shape context (3DSC) [27] directly extends the 2D shape context descriptor [22] to three dimensions. 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. 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.

6.1.2 *Unique Shape Context*

Unique Shape Context (USC) [58] improves 3DSC by leveraging a definition of repeatable and unambiguous Local Reference Frame. 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 Sec. 9.1.2 and proposed in [58], then construct the descriptor likewise 3DSC. 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.

6.1.3 *Rotational Projection Statistics*

Rotational Projection Statistics (RoPS) [94] is a local 3D descriptor designed to operates on meshes. 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, 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 [2, 20] and Shannon entropy [1] 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.

6.1.4 *Point Feature Histogram*

Point Feature Histogram (PFH) [42] 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. 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 $\mathbf{p}_t$:

$$\mathbf{u} = \mathbf{n}_s, \mathbf{v} = \mathbf{u} \times \frac{\mathbf{p}_t - \mathbf{p}_s}{\|\mathbf{p}_t - \mathbf{p}_s\|}, \mathbf{w} = \mathbf{u} \times \mathbf{v} \tag{6.1}$$

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:

$$\alpha = \langle \mathbf{v}, \mathbf{n}_t \rangle, \phi = \langle \mathbf{u}, \mathbf{p}_t - \mathbf{p}_s \rangle / d, \theta = \arctan(\langle \mathbf{w}, \mathbf{n}_t \rangle, \langle \mathbf{u}, \mathbf{n}_t \rangle), d = \|\mathbf{p}_t - \mathbf{p}_s\| \tag{6.2}$$

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.

6.1.5 *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 in [45]. 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. The FPFH descriptor can reduce the computational complexity from $O(nk^2)$ to $O(nk)$, where k is the number of neighboring points.

6.1.6 *SHOT: Unique Signatures of Histograms for Local Surface Description*

The SHOT descriptor [98] can be used both for surface description, as originally presented in [58], and for combined shape and texture description [66], if RGB information are available. The approach is based on the observation that local 3D descriptors can be categorized into *signatures* and *histograms*:

- **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.

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. Shot is equipped with a local reference, for which we will provide an exhaustive description in Sec. 9.1. 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 [28], 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 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.

6.1.7 *Spin Images*

The Spin Images is a surface representation technique that was initially introduced in [18]. 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. 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. 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 coordinate is omitted because it cannot be defined robustly and unambiguously using just surface position and normal.

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:

$$S_o(x) \rightarrow (\alpha, \beta) = (\sqrt{\|(x-p)\|^2 - (n \cdot (x-p))^2}, n \cdot (x-p)) \tag{6.3}$$

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 (Eq. 6.3) 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:

$$i_{max} = \frac{2\beta_{max}}{b} + 1 \qquad j_{max} = \frac{\alpha_{max}}{b} + 1 \tag{6.4}$$

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:

$$i = \lfloor \frac{2\beta_{max} - \beta}{b} \rfloor \qquad j = \lfloor \frac{\alpha}{b} \rfloor \tag{6.5}$$

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.

6.1.8 *3DMatch*

One of the first methods that learns a 3D local feature descriptor by means of deep learning algorithms is 3DMatch [159].

Early work [112, 128] 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) [61, 133]. In 3DMatch a standard Siamese 3D ConvNet is trained with a contrastive loss function [33] 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 [74]. The train set consists of RGB-D images of indoor scenes collected from existing popular datasets [88, 90, 97, 125, 133]. 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.

6.1.9 *PPFNet: Point Pair Feature NETwork*

Point Pair Feature NETwork [162] aims to learn a local descriptor by working directly on point coordinates augmented with handcrafted features. PPFNet combines the permutation invariant characteristic of PointNet [148] together with the high descriptive capacity of *Point Pair Features (PPF)* [52]. The network is fed by geometries containing information about the local neighborhood of a 3D point such as raw point coordinates, normals and PPFs. 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 T between scans is known. The distance between learned features of corresponding patches is minimized

acting on two distance matrices: a correspondence matrix $M \in \mathbb{R}^{N \times N}$, built on the points of the aligned scans, $M = (m_{ij})$ with

$$m_{ij} = \mathbb{1}(\|x_i - Ty_j\|_2 < \tau) \tag{6.6}$$

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

$$d_{ij} = \|f(x_i) - f(y_j)\|_2. \tag{6.7}$$

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

$$L = \sum^* \left(\frac{M \circ D}{\|M\|_2^2} + \alpha \frac{max(\theta - (1 - M) \circ D, 0)}{N^2 - \|M\|_2^2} \right) \tag{6.8}$$

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 [159]. 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.

6.1.10 *CGF: Compact Geometric Features*

Compact Geometric Features [143] learns a mapping $f : \mathbb{R}^N \to \mathbb{R}^n$ from high-dimensional handcrafted representations to a very low-dimensional feature space. 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 [106]. In this regard, CGF is coupled with an effective negative sampling strategy which produces a highly discriminative embedding. 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 [14]. CGF captures the local geometry around each point into a handcrafted feature called spherical histogram. Inspired by 3DSC [27], a spherical histogram encodes the distribution of points in a local neighborhood with a non uniform binned radial grid. To obtain rotational invariance, the local neighborhood is aligned to the axes of a local reference frame computed as in Sec. 9.1.2. 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 the spherical support 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:

$$r_i = \exp\left(\ln r_{min} + \frac{i}{R} \ln\left(\frac{r}{r_{min}}\right)\right) \tag{6.9}$$

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 $|\mathcal{N}|$. 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 [96] 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 [106] 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 referred 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:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{T}|} \sum_{i=1}^{|\mathcal{T}|} \left[\|f(\mathbf{x}_i^a; \theta) - f(\mathbf{x}_i^p; \theta)\|^2 - \|f(\mathbf{x}_i^a; \theta) - f(\mathbf{x}_i^n; \theta)\|^2 + 1 \right]_+, \quad (6.10)$$

where θ are the learned parameters and $[\cdot]_+$ denotes $\max(\cdot, 0)$. 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 $\mathcal{T}_j\mathcal{P}_j$. 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 $\mathbf{p}$. 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.

6.1.11 *PPF-FoldNet*

PPF-FoldNet [161] is an extension of Sec. 6.1.9. The main limitations of learned 3D descriptors are the big amount of labeled data required, sensitiv-

ity 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* [186] 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 [52, 102] of a support. 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:

$$\mathbf{F}\cdot = \{\, \mathbf{f}(\mathbf{p}, \mathbf{p}_1) \cdots \mathbf{f}(\mathbf{p}, \mathbf{p}_i) \cdots \mathbf{f}(\mathbf{p}, \mathbf{p}_N) \,\} \in \mathbb{R}^{4 \times N-1} \tag{6.11}$$

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

$$\boldsymbol{\psi}_{12} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)) \tag{6.12}$$

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 [102]:

$$\angle(\mathbf{v}_1, \mathbf{v}_2) = \text{atan2}\left(\|\mathbf{v}_1 \times \mathbf{v}_2\|\,,\, \mathbf{v}_1 \cdot \mathbf{v}_2\right) \tag{6.13}$$

$\angle(\mathbf{v}_1, \mathbf{v}_2)$ is guaranteed to lie in the range $[0, \pi)$. The main motivation behind the use of PPFs is their invariance to rotations. Indded, PPFs are invariants under Euclidean isometry as distances and angles are preserved between every pair of points. PFF-FoldNet leverages on an encoder-decoder architecture, 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 [148, 186]. After the concatenation with skip-links, a two-layer MLP compress these features to the encoded codeword.

The decoder takes inspiration from FoldingNet [186], and try to deform a low-dimensional grid structure using the information encoded in the codeword learned by the encoder. Differently from [186], 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:

$$d(\mathbf{F}, \hat{\mathbf{F}}) = \max\left\{ \frac{1}{|\mathbf{F}|} \sum_{\mathbf{f} \in \mathbf{F}} \min_{\hat{\mathbf{f}} \in \hat{\mathbf{F}}} \|\mathbf{f} - \hat{\mathbf{f}}\|_2, \ \frac{1}{|\hat{\mathbf{F}}|} \sum_{\mathbf{f} \in \hat{\mathbf{F}}} \min_{\mathbf{f} \in \mathbf{F}} \|\mathbf{f} - \hat{\mathbf{f}}\|_2 \right\} \tag{6.14}$$

where F is computed as in 6.11 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 [53]. The loss is minimized by ADAM Optimizer [96]. Batch size is 32, while the size of the final descriptor is 512. 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.

6.1.12 *3D-SmoothNet*

3D-SmoothNet (3DSN) [191] is a compact 3D descriptor learned by training a supervised deep learning network in a siamese fashion-way [30]. This work proposes as main novelty a *smoothed density value voxelization* (SDV), to employ as point cloud representation prone to be adopted in deep learning convolutional architectures. As another key contribution, to achieve invariance to rotation, the authors present an improved version of the LRF proposed by TOLDI *et al*. in [157]. Differently, from [157], where only $\frac{1}{3}$ of the points within the support contribute to the computation of the covariance matrix, the authors of 3DSN rely on all the points within the support. Moreover, during the computation of the covariance matrix, the centroid is replaced with the feature point $\mathbf{p}$. A detailed explanation about TOLDI LRF will be provided in Sec. 9.1.7. Once canonically oriented, the coordinates of the points in the local 3D patches are converted into a voxel grid represented as a three dimensional matrix $\mathbf{X}^{SDV} \in \mathbb{R}^{W \times H \times D}$, where each

cell $(\mathbf{X}^{SDV})_{jkl} =: x_{jkl}$ stores a scalar value computed using the Gaussian smoothing kernel with bandwidth h:

$$x_{jkl} = \frac{1}{n_{jkl}} \sum_{i=1}^{n_{jkl}} \frac{1}{\sqrt{2\pi}h} \exp \frac{-\|c_{jkl} - p_i\|_2^2}{2h^2} \quad \text{s.t. } \|c_{jkl} - p_i\|_2 < 3h \tag{6.15}$$

where n_{jkl} denotes the number of points p_i within the support that lie within the distance $3h$ from the voxel centroid c_{jkl}. To robustly managed the point cloud densities variations, all the values of $\mathbf{X}^{SDV}$ are normalized such that they sum up to 1. Similarly to CGF, 3SDN minimizes a triplet loss during learning, but with a more effective negative sampling strategy. Instead of pre-sampling negative examples before training, the negative samples are picked on the fly according to the hardest-in-batch method [141]. A feature descriptor is considered a negative sample for a triplet, if is the hardest non-corresponding positive sample in the mini-batch, *i.e.* a positive sample, across all the positives in the mini-batch, with the minimum distance in the space of the learned features from the positive of the triplet. Thanks to this strategy, 3DSN is able to create a distinctive and robust representation for 3D keypoints, using only 32 values for each descriptor.

7

LEARNING AN EFFECTIVE EQUIVARIANT 3D DESCRIPTOR WITHOUT SUPERVISION

We started this part by explaining the primary issues to overcome when designing a local or global feature descriptor for 3D data. In particular we have seen how, the adoption of deep learning techniques in this field requires interesting challenges to face, such as the creation of a parameterization for input data and a robust strategy to achieve invariance to rotation, above all. Moreover, the extensive and detailed section of related work, Sec. 6.1, has brought to light how state-of-the art proposals do not actually learn new local 3D descriptors from the input data but from existing handcrafted 3D descriptors, which are already rotation-invariant by design: e.g., CGF [143] starts from a high-dimensional input parameterization which closely resembles the Unique Shape Context (USC) descriptor [58], 3DSmoothNet [191] computes a smoothed voxel grid, while PPF-FoldNet [161] relies on the well-known Point Pair Features (PPF) [52]. In other words, due to the difficulty of feeding neural networks with unorganized input data [143], these approaches create new descriptors by actually learning how to robustly *compress* a specific invariant handcrafted descriptor. We argue that the drawbacks of relying on invariant handcrafted descriptors as input data to feed neural networks are twofold. On one hand, there not exist an optimal handcrafted descriptor across applications and datasets, as vouched by recent evaluations [124]. Therefore, for instance, performance of PPF-FoldNet are limited on some scenarios and datasets by the handcrafted design decision of using PPF as input representation. On the other hand, to achieve rotation invariance, existing handcrafted descriptors used in deep learning pipelines rely either on the normal at the point as a reference axis [161] or on a LRF [143] to express point coordinates and angles with respect to a canonically oriented reference frame. As we will show in Part III, repeatability of the axes or the LRF directly affects the invariance and robustness of the input descriptor [63, 75] and, in turn, of the descriptor learned from such representations. However, parameters used to obtain such canonical orientation (e.g. the number of neighbours to estimate the normals, how to establish a reference direction on the tangent plane in a LRF,

etc.) are again handcrafted design decisions and are not optimized during training.

Reliance on rotation-invariant handcrafted descriptors as input representations deviates significantly from the end-to-end learning paradigm so successfully applied to images. Therefore, in this chapter we investigate on whether leaving the model free to learn an optimal descriptor from a non-canonically oriented input representation may unleash the untapped potential of deep learning also in this scenario. To this end, we exploit the paradigm recently proposed in FoldingNet [186] and AtlasNet [165] to realize unsupervised learning of an embedding space from 3D data, which learns to deform, according to the latent representation, points sampled from a plane so as to reconstruct the input surface. This concept has already been deployed to obtain an invariant 3D descriptor by reconstructing the Point Pair Features of the input data [161]. In the proposal we are going to describe in this chapter, however, the learned latent space has to encode pose information in order to be able to reconstruct the input under arbitrary poses, as it will be shown later (Sec. 7.2.3). We argue that the ability to learn an embedding *equivariant* with respect to rotations of the input is the most sound approach to include pose information in the latent space. To this end, we leverage recent work on Spherical CNNs [160, 163], which have enhanced the deep learning machinery by enabling it to learn also rotation-equivariant representations from 3D spherical signals by means of correlations defined for the $SO(3)$ group of rotations. Hence, in our architecture, a Spherical CNN encoder learns to summarize the geometry around a feature point into a rotation-equivariant embedding and a decoder warps a 2D grid in order to reconstruct the raw input data. This enables learning of an equivariant embedding without using noisy and arbitrary canonical orientations at training time.

To perform pose invariant descriptor matching at test time, we have investigated two alternative ways to orient our equivariant descriptor: we can again exploit the peculiar nature of the Spherical CNN output, which is a signal living in $SO(3)$, to define a canonical orientation directly from the computed embedding; or we can orient the descriptor according to a canonical orientation provided by an external local reference frame computed on the input data. While the first approach enables end-to-end learning of the descriptor and the LRF, we have so far obtained better results with the second one. In particular, we have validated our claim on the superiority of learning a local descriptor from raw unoriented input data by comparing

the two variants against handcrafted and learned methods, presentend in Sec. 6.1, on the popular 3DMatch benchmark data set [159]. Our proposal improves the state-of-the art by a remarkable margin, outperforming the method based on the same unsupervised learning framework, but applied to an invariant descriptor, by more than 0.23 points of fragments registation recall (31% increase).

Before starting, we give an overview about the current approaches used to learn features from 3D data as well as a briefly recap of the concepts exposed in Sec. 6.1.

7.1 RELATED WORK

Hand-crafted 3D local descriptors collects geometric or topological measurements into histograms. Approaches such as Spin Images [18], Unique Shape Context [58] and RoPs [83] rely on the spatial distribution of the points on the surface, while others like FPFH [45] and SHOT [98] exploit geometric properties of the surface such as normals or curvatures. Rotation invariance is achieved using either a LRF or a Reference Axis. On the other hand, learned 3D local descriptors learn a representation of the geometry from 3D data. As a consequence of the unorganized nature of point clouds, several parallel tracks regarding the representation of the input data have emerged. Early works represent a 3D object as a collection of 2D views [114, 134]. Another approach concerns dense 3D voxel grids, with voxels containing either a binary occupancy grid [112, 155] or an alternative representation of the surface [159, 191]. To limit the memory occupancy of voxel grids, researchers either rely on coarse spatial resolutions, which, however, introduce artifacts and hinder the ability to learn fine geometric structures, or on space partition methods like k-d trees or octrees [144, 151]. Other methods, differently, deploy high-dimensional hand-crafted features to parameterize the input point cloud and then use deep learning to project it into lower dimensional spaces [143, 161].

Several approaches, not directly related to the 3D local descriptor field, exist to do learning from raw 3D data. PointNet [148] and PointNet++ [149] are pioneering works presenting a general framework to learn features directly from raw point clouds data. Although yielding excellent performance in point cloud segmentation and classification tasks, these architectures have not been used yet to perform local surface description, likely due to

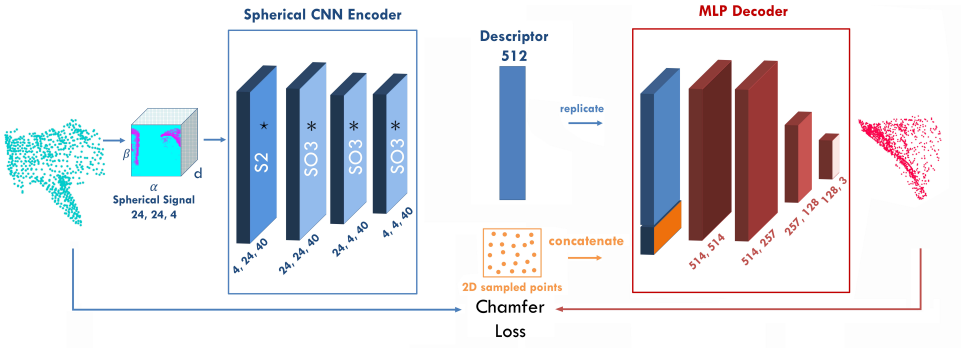


Figure 7.1: Architecture of the proposed method. The points within the local support of a given feature point $\mathbf{p}$ are converted into a spherical signal representation, and then sent through the spherical encoder to get an equivariant descriptor. The numbers below the spherical signal indicate the number of cells along α, β and d. The decoder reconstructs the original point cloud deforming sampled 2D points according to the descriptor. Operations in the encoder are implemented through the Generalized Fourier Transform with signals discretized according to a bandwidth parameter [160]. The triplets below the encoder layers indicate input bandwidth, output bandwidth and number of channels. As for the decoder, the pairs indicate the number of input and output channels, respectively.

the inability of providing rotation invariance. Nonetheless, PointNet is the core building block of PPFNet [162], which relies on raw point coordinates, normals and Point-Pair Features in order to learn a local feature descriptor. Indeed, due to the reliance on the PointNet architecture, PPFNet is not rotation invariant.

7.2 Learning an equivariant 3D descriptor from spherical signals

In this section we present the whole pipeline of our method, graphically illustrated in Fig. 7.1. Please note that our encoder only contains correlation layers, *i.e.* it does not include a max pooling layer at the end to learn a pose-invariant descriptor, which is instead present in the architectures proposed in [160].

7.2.1 *Background on Spherical CNNs*

As we rely on Spherical CNNs, we provide a brief overview of the mathematical model behind it. For more details, please refer to [160].

The basic intuition behind Spherical CNNs can be grasped by analogy with the classical planar correlation used by traditional CNNs. As explained in [160], the value of the output feature map at $x \in \mathbb{Z}^2$ in a planar correlation can be understood as the inner product between the input feature map and the learned filter shifted by x. By analogy, the value of the output feature map at $R \in SO(3)$ in a spherical correlation can be understood as the inner product between the input feature map and the learned filter, *rotated* by R.

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

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

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

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

Rotations A rotation in three dimensions lives in a three-dimensional manifold called $SO(3)$, the "special orthogonal group". As in [160] the rotation group $SO(3)$ can be parameterized by ZYZ-Euler angles $\alpha \in [0, 2\pi]$, $\beta \in [0, \pi]$, and $\gamma \in [0, 2\pi]$. Rotations can be represented by 3×3 matrices that preserve distance (i.e. $\|Rx\| = \|x\|$) and orientation ($det(R) = +1$). If we represent points on the sphere as 3D unit vectors x, a rotation can be performed by using the matrix-vector product Rx.

Rotations of Spherical Signals The spherical correlation operator needs to rotate the filters on the sphere. For this purpose, [160] introduces the operator L_R that takes a function f and produces a rotated function $L_R f$ by composing f with the rotation R^{-1}:

$$[L_R f](x) = f(R^{-1}x) \tag{7.1}$$

Spherical Correlation Denoting with $\langle \psi, f \rangle$ the inner product on the vector space of spherical signals defined as in [160], the correlation between a K-valued spherical signal f and a filter ψ, $f, \psi : S^2 \to \mathbb{R}^K$ can be formalized as:

$$[\psi \star f](R) = \langle L_R \psi, f \rangle = \int_{S^2} \sum_{k=1}^{K} \psi_k(R^{-1}x) f_k(x) dx. \tag{7.2}$$

This is the operation performed by the first layer of our encoder (Fig. 7.1). Unlike the standard definition of spherical convolution [11], which gives as output a function on the sphere S^2, the spherical correlation yield a signal on $SO(3)$. The use of a conventional convolution definition would limit the expressive capacity of the network due to the symmetry along the Z axis of the learned filters.

Rotation of $SO(3)$ Signals Similarly to what has been defined for spherical correlation in Eq. 7.2, to define a correlation in $SO(3)$ the operator in Eq. 7.1 must be generalized so that it can act on $SO(3)$. For a signal $h : SO(3) \to \mathbb{R}^K$, and $R, Q \in SO(3)$:

$$[L_R h](Q) = h(R^{-1}Q). \tag{7.3}$$

The term $R^{-1}Q$ in Eq. 7.3 denotes the composition of rotations.

Rotation Group Correlation Likewise in Eq. 7.2, we can define the correlation between a signal and a filter on the rotation group, $h, \psi : SO(3) \to \mathbb{R}^K$, as follows:

$$[\psi * h](R) = \langle L_R \psi, f \rangle = \int_{SO(3)} \sum_{k=1}^{K} \psi_k(R^{-1}Q) h_k(Q) dQ. \tag{7.4}$$

This is the operation performed by the all the layers of our encoder but the first one (Fig. 7.1). The integration measure dQ is the invariant measure on $SO(3)$, which may be expressed in ZYZ-Euler angles as $d\alpha \sin(\beta) d\beta d\gamma / (8\pi^2)$. Please note that unlike in [160] for better clarity we denote as $\star$ the spherical correlation Eq. 7.2 while with $*$ the rotation group correlation Eq. 7.4.

7.2.2 *Learning from Spherical Signals*

Our feature encoder operates on signals defined in a spherical domain. Hence, the local geometry surrounding a feature point **p** needs to be con-

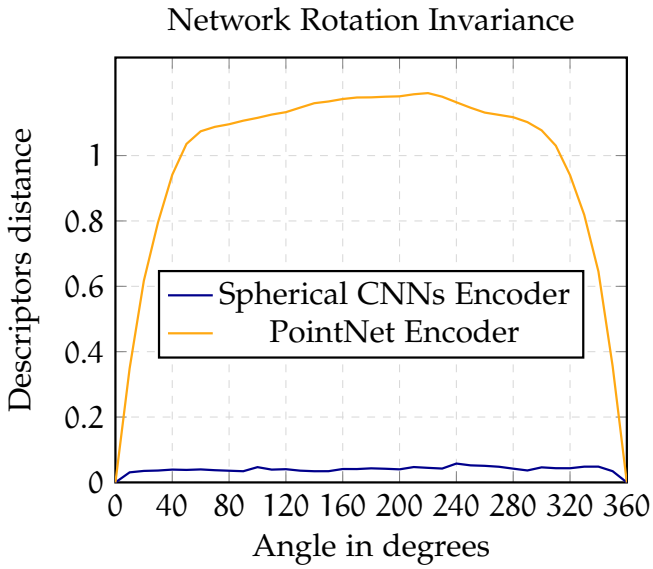


Figure 7.2: Comparison between PointNet and Spherical CNN used as encoders in our framework.

verted into a spherical representation. A common strategy adopted by [160, 163] is to project a 3D mesh onto an enclosing discretized sphere using a raycasting scheme. Since our input data is not a regular watertight mesh, but a point cloud corresponding to the neighborhood of the point we wish to describe, we first convert 3D points into a spherical coordinate system and then construct a quantization grid in this new coordinate system, similarly to [187]. The i-th cell in the quantization is identified with three spherical coordinates $(\alpha[i], \beta[i], d[i]) \in S^2 \times D$ where $\alpha[i]$ and $\beta[i]$ represent the azimuth and inclination angles of its center and $d[i]$ is the distance from the sphere center. The K-valued spherical signal $f : S^2 \to \mathbb{R}^K$ is then composed by K concentric spheres corresponding to the number of subdivisions along the distance axis, each sphere encoding the density of the points within each cell $(\alpha[i], \beta[i])$ at a given distance $d[k]$. To take into account the non-uniform spacing in the spherical space, cells near the south or north pole are wider in spherical coordinates, as discussed in [187].

A spherical signal is computed on the support of every keypoint. The signal then goes through our architecture to learn an equivariant bottleneck layer, which can then be used as a descriptor of the local geometry around the keypoint.

7.2.3 *Rotation-Equivariant Descriptor*

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

Learning an equivariant bottleneck removes the requirement to have invariant representations as input to the network at training time as the only way to achieve rotation invariance, the standard approach in existing proposals [143, 161]. In our framework, instead, we can delay the choice on how to canonically orient the descriptor at test time, which brings in two important benefits. On one hand, we do not have to choose a specific way to orient the input, e.g. a specific LRF, at training time, which means that we can train the network to learn the descriptor from less pre-processed input data than existing proposals, moving a step closer toward end-to-end descriptor learning. On the other hand, not using a LRF at training time frees our method from unavoidable errors of the LRF itself, which in turn inject noise in the training process. We expect both benefits to concur to increase the effectiveness of the learned descriptor.

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

Please note that a truly rotation-equivariant CNN like Spherical CNNs is mandatory in our framework, as anticipated in the introduction. Indeed, only a descriptor that lives in $SO(3)$ can be rotated after having been computed, i.e. only the output of a Spherical CNN to date. All the other standard representations, e.g. the output of a Multi Layer Perceptron (MLP) as used in PointNet, cannot be rotated after having been computed. Therefore, if we want to use them in our framework where the input is not canonically oriented for the reasons discussed above, we can only hope the network learns to obtain directly a rotation-invariant descriptor by observing rotated versions of the same neighborhood during training without explicit supervision, which is however a harder task in our setup than learning an equivariant descriptor.

We have validated how harder this is experimentally, by using a standard PointNet encoder instead of the spherical one to learn an invariant descriptor. Results of the comparison are shown in Fig. 7.2. Please note that equivariance

is a theoretical property of a Spherical CNN, regardless of whether it is trained or not. Indeed, in the results in Fig. 7.2, the Spherical encoder has **not** been trained, while the PointNet encoder has been trained on the 3DMatch Benchmark presented in Sec. 7.3.1. Given a neighborhood, we rotate it around a random axis by a growing angle, whose value is reported along the horizontal axis of the chart. For every rotation, we pass the rotated neighborhood through a Spherical CNN encoder and a PointNet encoder. The output of the Spherical CNN is then rotated by the inverse of the applied rotation (simulating the availability of a perfect LRF) and the distance between the descriptor obtained from the rotated neighborhood and the descriptor obtained from the un-rotated neighborhood is plotted. We can clearly see that PointNet fails to learn an invariant descriptor in our setup, while the equivariant representation provided by a Spherical CNN can achieve almost perfect invariance when properly rotated.

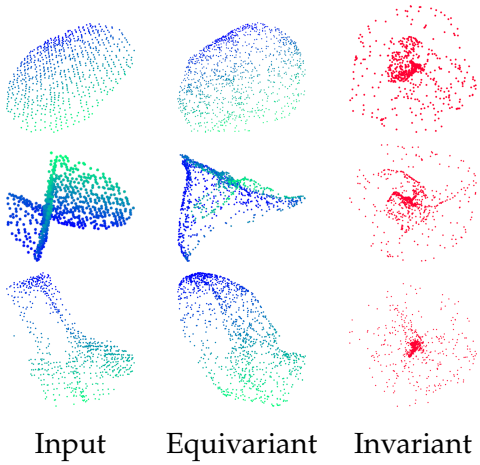


Figure 7.3: Comparison between the reconstructions obtained when using the Spherical CNN encoder to learn an equivariant versus an invariant bottleneck. Results after 10K training iterations.

Moreover, even if PointNet were able to learn a perfectly invariant bottleneck, we have found experimentally that this would result in low quality reconstructions. The reason is that it is not possible for frameworks like FoldingNet/AtlasNet to converge to sensible reconstructions if the learned bottleneck does not contain any pose information, i.e. it is almost perfectly invariant. This is shown in Fig. 7.3. where we compare the quality of the reconstructions produced by our framework when using an equivariant bottleneck layer versus an invariant one. The invariant one in this case is obtained by removing the last $SO(3)$ correlation layer from our encoder, which produces the equivariant descriptor in our architecture, and adding a

max pooling layer selecting the maximum of each one of the now top-level 40 feature maps, followed by a fully connected layer to expand the codeword dimensionality to 512. As shown in the figure, if the encoder produces an invariant descriptor, the decoder doesn't have enough information to know in which pose it should reconstruct the input so as to minimize the loss. The best it can do is to produce reconstructions trying to account for all possible rotations of the input, e.g. the atom-like structures depicted in the last column, almost ignoring the invariant bottleneck layer.

7.2.4 *Invariant Feature Descriptor*

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

The first is the most intellectually satisfying, and leverages again the peculiar properties of Spherical CNNs. Indeed, every bin of a feature map in a Spherical CNN represents an element of $SO(3)$, *i.e.* a potential LRF. This has been already exploited to align full shapes in [163], by finding the argmax of the correlation between two feature maps. Note that we cannot use the same approach in the context of invariant descriptor matching, as this would require a costly computation to compute the distance between every pair of source and target descriptors.

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

$$\begin{aligned}
[\psi * m](R_m) &= [\psi * [L_Q h]](R_m) \\
&= \langle L_{R_m}\psi, L_Q h\rangle \\
&= \langle L_{Q^{-1}R_m}\psi, h\rangle \\
&= [\psi * h](Q^{-1}R_m) \coloneqq [\psi * h](R_h)
\end{aligned} \tag{7.5}$$

In other words, chosen an entry in a descriptor $[\psi * h]$ obtained when processing f, e.g. R_h, if, when the input is rotated by Q, we are able to find

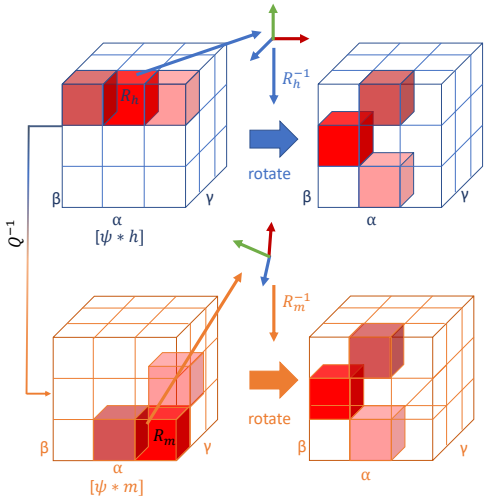


Figure 7.4: Self-orienting property of the learned equivariant descriptor. Every bin of our bottleneck layer corresponds to three Euler angles which define a rotation. If the descriptor is computed starting from a rotated input (second row), the values shifts in the feature maps. By finding two corresponding bins in the two descriptors and rotating them by the inverse of the corresponding rotations, the descriptors can be aligned, i.e. become pose invariant.

the same entry independently in the rotated descriptor $[\psi * m]$, we will find it at rotation $R_m = QR_h$.

Therefore, given the two descriptors, we can align them to a common pose by applying the inverse of such rotations

$$\begin{aligned}
[L_{R_m^{-1}}[\psi * m]](R) &= [\psi * m](R_m R) \\
&= [\psi * [L_Q h]](R_m R) \\
&= [\psi * h](Q^{-1} R_m R) \qquad (7.6) \\
[L_{R_h^{-1}}[\psi * h]](R) &= [\psi * h](R_h R) \\
&= [\psi * h](Q^{-1} R_m R) \qquad (7.7)
\end{aligned}$$

as shown by last terms of the transformations being equal (and graphically in Fig. 7.4). Please note that all transformations are applied to the descriptor (which is a feature map) obtained from the unrotated input and not to the input itself, i.e. we can rotate the descriptor computed from an unoriented input to achieve rotation invariance. The dimensionality of ours descriptor does not change under rotations, as it is rotated by remodulating the spherical harmonics functions resulting from its Fourier transform. A through treatment of this topic can be found in [16].

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

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

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

7.2.5 *Decoder and Loss*

Differently from [162], our goal is to reconstruct the whole set of points representing the local neighborhood of a given feature point $\mathbf{p}$. Inspired by [165] and [186], our decoder will try to deform points in $\mathbb{R}^2$ to surface points

in $\mathbb{R}^3$ according to the learned descriptor. Given a feature representation $\mathbf{d}$ for a 3D surface, let $\mathcal{A}$ be a set of points sampled in the unit square $[0,1]^2$, the descriptor $\mathbf{d}$ is concatenated with the sampled point coordinates $(a_x, a_y) \in \mathcal{A}$ and then forwarded through a stack of MLP layers as shown in Fig. 7.1. We then minimize the Chamfer loss between the set of generated 3D points and the input points.

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

$$\begin{aligned}\mathcal{L}(\mathcal{S},\mathcal{S}^\star)_\theta = &\frac{1}{|\mathcal{S}|}\sum_{\mathbf{x}\in\mathcal{S}}\min_{\mathbf{x}^\star\in\mathcal{S}^\star}\|\mathbf{x}-\mathbf{x}^\star\|_2 + \\ &\frac{1}{|\mathcal{S}^\star|}\sum_{\mathbf{x}^\star\in\mathcal{S}^\star}\min_{\mathbf{x}\in\mathcal{S}}\|\mathbf{x}^\star-\mathbf{x}\|_2.\end{aligned} \tag{7.8}$$

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

7.2.6 *Network and training parameters*

To learn our descriptor, we use one S^2 convolution layers and three $SO(3)$ convolution layers with constant number of channels, 40, while the bandwidths is set to 24 for the first three layer and 4 for the last one, which results in a descriptor with 512 entries. The architecture of our decoder is made of 4 fully-connected layers, with ReLU non-linearities on the first three layers and tanh on the final output layer. The network is trained with mini-bacthes of size 32 by using ADAM [96]. The starting learning rate is set to 0.001 and is decayed every 4000 iterations. We train the network for 14 epochs.

7.3 EXPERIMENTAL RESULTS

7.3.1 *Experimental setup*

To test our proposal, we use the standard benchmark for the evaluation of learned 3D descriptors, the 3DMatch benchmark [159]. This benchmark

Table 7.1: Results on the 3DMatch benchmark. Test data are from SUN3D [90], except for *Red Kitchen* data which is from 7-scenes [88]. Best result on each row is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	0.7391	0.7885	0.6442	0.8142	0.7115	0.8889	0.7432	0.7013	0.7539
Spin Images	0.6561	0.7564	0.6731	0.6770	0.6346	0.7407	0.4692	0.4545	0.6327
SHOT	0.8893	0.8974	0.8221	0.9336	0.8750	0.8889	0.8630	0.8312	0.8751
USC	0.9308	0.9103	0.7788	0.9204	0.8462	0.8889	0.8664	0.8052	0.8684
3DMatch	0.5810	0.7244	0.6154	0.5442	0.4808	0.6111	0.5171	0.5065	0.5726
CGF	0.4605	0.6154	0.5625	0.4469	0.3846	0.5926	0.4075	0.3506	0.4776
PPFNet	0.8972	0.5577	0.5913	0.5796	0.5769	0.6111	0.5342	0.6364	0.6231
PPFFoldNet	0.7866	0.7628	0.6154	0.6814	0.7115	0.9444	0.6199	0.6234	0.7182
3DSmoothNet	0.9700	0.9550	0.8940	0.9650	0.9330	**0.9820**	**0.9450**	**0.9350**	**0.9474**
Ours SO	0.8854	0.9487	0.8654	0.9204	0.8462	0.9630	0.8870	0.8182	0.8918
Ours LRF	**0.9763**	**0.9615**	**0.8942**	**0.9823**	**0.9519**	0.9815	0.9178	0.8701	0.9420

Table 7.2: Results on the rotated 3DMatch benchmark. Test data are from SUN3D [90], except for *Red Kitchen* data which is from 7-scenes [88]. Best result on each row is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	0.7451	0.7949	0.6587	0.8142	0.7212	0.9259	0.7260	0.7530	0.7674
Spin Images	0.6502	0.7628	0.6635	0.6903	0.6635	0.7222	0.4692	0.4935	0.6394
SHOT	0.8794	0.8910	0.8317	0.9425	0.8654	0.9074	0.8493	0.8312	0.8747
USC	0.9170	0.9103	0.7548	0.9292	0.8558	0.9074	0.8836	0.8571	0.8769
3DMatch	0.0040	0.0128	0.0337	0.0044	0.0000	0.0096	0.0000	0.0260	0.0113
CGF	0.4466	0.6667	0.5288	0.4425	0.4423	0.6296	0.4178	0.4156	0.4987
PPFNet	0.0020	0.0000	0.0144	0.0044	0.0000	0.0000	0.0000	0.0000	0.0026
PPFFoldNet	0.7885	0.7821	0.6442	0.6770	0.6923	0.9630	0.6267	0.6753	0.7311
3DSmoothNet	0.9720	0.9620	**0.9090**	0.9650	0.9230	**0.9820**	**0.9450**	**0.9350**	**0.9491**
Ours SO	0.8893	0.9423	0.8413	0.9204	0.8558	0.9074	0.8733	0.7922	0.8778
Ours LRF	**0.9763**	**0.9679**	0.8894	**0.9779**	**0.9615**	0.9815	0.9110	0.8442	0.9387

addresses registration of unordered 3D views and the dataset has been put together by merging a large part of the publicly available datasets such as Analysis-by-Synthesis [133], 7-Scenes [88], SUN3D [90], RGB-D Scenes v.2 [97] and Halberand Funkhouser [139]. It contains 62 scenes in total, and, following [161], we use 54 for training and validation, while 8 scenes are used only at test time to run comparisons. The dataset already provides so-called fragments, *i.e.* the point clouds resulting from the fusion of 50 consecutive depth frames, for the test scenes, and we obtained the training fragments generated by the same methodology as the authors of [161]. We also consider the rotated version of the 3D Match benchmark, generated by the same authors by rotating all the fragments in the 3DMatch benchmark with randomly sampled axes and angles over the whole rotation space.

We use the same setup as proposed in [161]: we downsample the fused fragments with a voxel grid filter of size 2 cm and compute surface normals

using [10] a 17-point neighborhood; we consider a radius of 30 cm to define the neighborhood of a keypoint.

7.3.2 *Evaluation methodology*

We test our proposal in the context of surface registration following the evaluation methodology proposed by [161]. This protocol build fragments pairs like the methodology exposed in Sec. 4.3.2. Only fragments with at least 30% overlap are picked during the evaluation. With regard to the metrics involved, here the *recall* refers to the fragment recall introduced in [161] that is the standard metric for this benchmark. A pair of fragments is considered correctly registered if the number of correctly matched keypoints is greater than the inlier ratio threshold τ_2, set to 5% of the extracted keypoints. Two keypoints are considered a matched couple if their l_2 distance is below a threshold $\tau_1 = 10$ cm. For each fragment, the descriptors are computed on 5000 uniformly sampled points, provided with the benchmark [159]. For handcrafted descriptors we used the implementation in PCL [67], while for learned descriptors results were taken from [161].

7.3.3 *Quantitative results*

Results of the tests on the 3D Match benchmark in terms of recall are reported in Tab. 7.1. With Ours SO we refer to the self orienting descriptor introduced in Sec. 7.2.4, while with Ours LRF we refer to the descriptor oriented with an external local reference frame. In particular for this experiments we have used the LRF algorithm proposed in [75], which we will denote as FLARE according to the acronym used in its PCL implementation [67].

The first outcome of our experiments is that the use of an external LRF outperforms the self-orienting variant of our algorithm. Note that the two rows describe exactly the same equivariant descriptor under two different ways to compute a canonical orientation. Hence, the highest one is indicative of the quality of the learned descriptor itself. Although the performance of the self-orienting variant of our method is inferior to our descriptor oriented by an external LRF, it is remarkable that it delivers the third best recall on the dataset, *i.e.* it would provide state-of-the-art performance, between unsupervised approach, if we were not to orient our equivariant descriptor also with an external LRF. Our self-orienting variant is closely followed

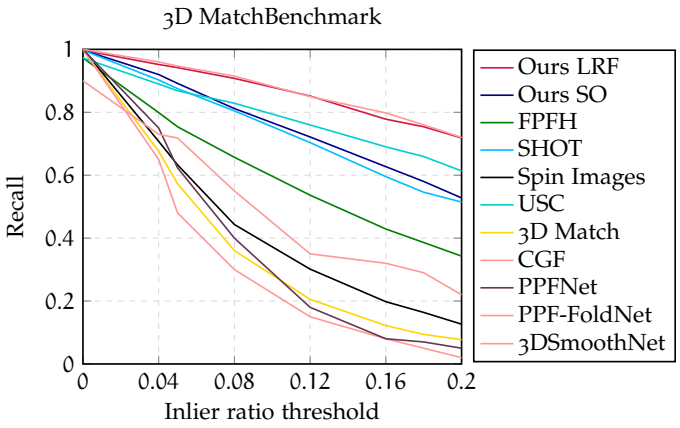


Figure 7.5: Results under varying inlier ratio threshold τ_2.

by SHOT and USC, *i.e.* two handcrafted descriptors, while the other tested methods, other than 3DSmoothNet, deliver significantly lower recalls. The best learned approach is 3DSmoothNet, which is performing similar to our proposal with external LRF, but being supervised. Hence, unlike ours, it requires labeled data to be trained. The better performance of SHOT and USC with respect to PPFFoldNet offers support to the inspiring ideas behind this work: deep learning alone, if constrained to learn from highly engineered representations, cannot guarantee superior performance. It is also interesting to analyze these results in light of our main claim: to learn an equivariant descriptor and then orient it to achieve invariance instead of learning directly an invariant one boosts its quality. If we compare the performance of our method when oriented with both tested variants against methods learning from invariant representations, like PPFFoldNet and CGF, we can interpret the large gap in performance (0.23 and 0.47 points of recall from the external LRF variant, respectively), as a validation of the drawbacks of existing learned descriptors discussed in the introduction. In Fig. 7.5, we report results when varying the threshold τ_2 on the percentage of correct matches to consider a pair as correctly registered, as done in [161]. Our proposal oriented with an external LRF is performing similar to 3DSmoothNet for all thresholds, whilst, our self-orienting variant attains again recall values similar to SHOT, and slightly inferior to USC at the largest thresholds.

Finally, results of the tests on the rotated 3D Match benchmark are reported in Tab. 7.2. The dataset was proposed in [161] to test robustness against large rotations, not present in the original benchmark. As expected, all rotation-invariant methods obtain performance similar to the results reported in

Tab. 7.1, and our equivariant descriptor oriented with the external LRF performs similar to 3DSmoothNet that exhibits the best results.

7.4 A MORE EFFECTIVE EQUIVARIANT EMBEDDING

As a further contribution of this chapter, we perform an ablation study on the architecture of our spherical encoder in order to improve the results presented in Sec. 7.3.3. For this study, we focus only on the descriptor oriented at test time with the external LRF, *i.e.* Ours LRF. As a design choice of Spherical CNNs [160] architecture, two distinct types of spherical grids are available, both for S2 and $SO(3)$ convolutions: the near identity and equatorial grids. The former define spatially localized kernels, initialized on the north pole and rotated over the sphere via the action of $SO(3)$, while the latter, define a ring-like kernels around the equator. Together with the grids, we vary the number of $SO(3)$ layers which follow the first S2 layer, the input bandwidth and the number of channels at each layer. To validate our design choices, we compare the architecture adopted so far, *i.e.* Sec. 7.2.6, in terms of recall and description time against the new configurations obtained changing the values of the parameters specified above. Thus, for each configuration we trained a new network and then, test the resulting descriptor on the test split of 3DMatch benchmark, following the protocol presented in Sec. 7.3.2, but with a reduced number of keypoints, 500 instead of 5000. The result of the ablation study is shown in Tab. 7.3, with configuration A referring to the architecture and parameters adopted so far and described in Sec. 7.2.6. To improve the selection criteria, we also consider the computational time for description, considering a mini-batch size equal to 25. The best trade off configuration is selected according to the Pareto analysis, presented in Fig. 7.6. It turns out that the configuration N presents the best recall performance on the Pareto frontier with a significant reduction in processing time. Hence, we train a new network with this configuration following the training protocol described in Sec. 7.2.

Network and training parameters

Based on the ablation study in Sec. 7.4, we train a new network according the the configuration N in Tab. 7.3. The spherical encoder has a S^2 convolution layer and four $SO(3)$ convolution layers with a constant number of channels, 40. The input bandwidths of these five layers are respectively, 24, 16, 12, 8

Table 7.3: Ablation study results on the 3DMatch benchmark. Networks on the Pareto frontier on the column Network, best values on recall and Normalized time in bold. Tests performed for a subset of 500 keypoints.

Network	Grid Eq.	NI	SO(3) layers	Input BW	Channels	Recall	Time
A	✓		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		✓	4	[16, 12, 8, 6]	40	**0.929**	0.632

and 6. After the layers we apply a *BatchNorm* step and ReLU non-linearities. The last layer of the encoder outputs the descriptor, with 512 entries. For the decoder we use the same configuration as in Sec. 7.2.6. Finally, we trained the network using ADAM [96] and a learning rate of 0.001.

7.4.1 *Experimental setup*

To stress the ability of our method to generalize to geometric varieties not seen during training, we add to the evaluation a new outdoor dataset: the ETH dataset [76]. Hence, we first train a network on the training split of 3DMatch dataset and then we test it on ETH, 3DMatch and 3DMatch rotated. The ETH dataset [76] is a challenging outdoor dataset, which presents 8 sequences with semistructured and unstructured environments, repetitive and dynamic elements. The dataset is acquired with a laser scanner sensor and contains partially overlapping scans of sparse and dense vegetation (e.g., trees and bushes). To test our approach, we used, as in [191], a subset of four scenes named: *Gazebo-Summer*, *Gazebo-Winter*, *Wood-Autumn* and *Wood-Summer*. We test our proposal on the 3DMatch dataset following the protocol and the same settings illustrated in Sec. 7.3.1, whilst for ETH, we

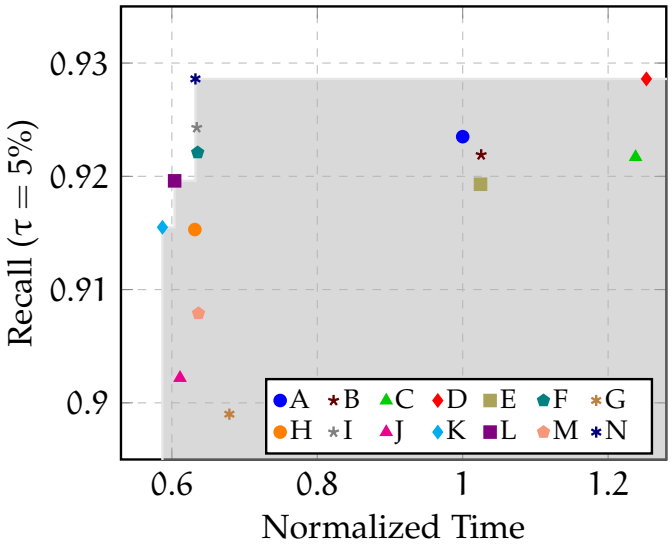


Figure 7.6: Ablation study comparing the different configurations in terms of registration recall and normalized description time. For more detail of each configuration refer to Tab. 7.3. The light gray area shows the Pareto frontier of the test. The computational time is expressed in term of percentage of increase or decrease compared to the time of our baseline (A), *e.g.* the configuration N is 40% faster than A.

adhere to the setup proposed in [191]. To take into account the different scales of the two dataset, in ETH we describe the support of a feature point $\mathbf{p}$ within a radius of 1.0m.

7.4.2 *Results on 3DMatch dataset*

We report results of the experimental evaluation on the 3D Match benchmark in terms of recall in Tab. 7.5. With Ours we refer to our equivariant embedding trained with the configuration explained in Fig. 7.4 and oriented at test time with FLARE LRF. Our descriptor achieves an average recall of 95.84% outperforming all state-of-the-art 3D local descriptors on the standard registration benchmark, surpassing the most challenging competitor, 3DSmoothNet [191]. Finally, we report the results for fragment registration recall on the rotated 3D Match benchmark in Tab. 7.5, which, again, confirms how a proper design choice of the architecture can significantly improve the quality of the learned descriptor.

As a further proof of the effectiveness of our approach, we compare the number of correct correspondences generated by our descriptor against the state-of-the-art methods. To establish the set of correspondences between descriptors of two fragments, we use a nearest neighbor query, then the

Table 7.4: Results on the 3DMatch benchmark. Test data are from SUN3D [90], except for *Kitchen* data which is from 7-scenes [88]. Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	0.7391	0.7885	0.6442	0.8142	0.7115	0.8889	0.7432	0.7013	0.7539
Spin Images	0.6561	0.7564	0.6731	0.6770	0.6346	0.7407	0.4692	0.4545	0.6327
SHOT	0.8893	0.8974	0.8221	0.9336	0.8750	0.8889	0.8630	0.8312	0.8751
USC	0.9308	0.9103	0.7788	0.9204	0.8462	0.8889	0.8664	0.8052	0.8684
3DMatch	0.5810	0.7244	0.6154	0.5442	0.4808	0.6111	0.5171	0.5065	0.5726
CGF	0.4605	0.6154	0.5625	0.4469	0.3846	0.5926	0.4075	0.3506	0.4776
PPFNet	0.8972	0.5577	0.5913	0.5796	0.5769	0.6111	0.5342	0.6364	0.6231
PPFFoldNet	0.7866	0.7628	0.6154	0.6814	0.7115	0.9444	0.6199	0.6234	0.7182
3DSmoothNet	0.9700	0.9550	0.8940	0.9650	0.9330	**0.9820**	**0.9450**	**0.9350**	0.9474
Ours	**0,9901**	**0.9808**	0.9135	**0.9956**	**0.9808**	0.9815	0.9418	0.8831	**0.9584**

Table 7.5: Results on the rotated 3DMatch benchmark. Test data are from SUN3D [90], except for *Kitchen* data which is from 7-scenes [88]. Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	0.7451	0.7949	0.6587	0.8142	0.7212	0.9259	0.7260	0.7530	0.7674
Spin Images	0.6502	0.7628	0.6635	0.6903	0.6635	0.7222	0.4692	0.4935	0.6394
SHOT	0.8794	0.8910	0.8317	0.9425	0.8654	0.9074	0.8493	0.8312	0.8747
USC	0.9170	0.9103	0.7548	0.9292	0.8558	0.9074	0.8836	0.8571	0.8769
3DMatch	0.0040	0.0128	0.0337	0.0044	0.0000	0.0096	0.0000	0.0260	0.0113
CGF	0.4466	0.6667	0.5288	0.4425	0.4423	0.6296	0.4178	0.4156	0.4987
PPFNet	0.0020	0.0000	0.0144	0.0044	0.0000	0.0000	0.0000	0.0000	0.0026
PPFFoldNet	0.7885	0.7821	0.6442	0.6770	0.6923	0.9630	0.6267	0.6753	0.7311
3DSmoothNet	0.9720	0.9620	**0.9090**	0.9650	0.9230	**0.9820**	0.9450	**0.9350**	0.9491
Ours	**0.9921**	**0.9744**	0.8990	**0.9956**	**0.9712**	0.9815	**0.9452**	0.9221	**0.9601**

matched keypoints are aligned using the ground-truth rigid motion matrix between the fragments. We consider a correspondence correct if the l_2 distance between two keypoints is smaller than $\tau_1 = 10cm$. As can be seen in Tab. 7.6, also using this direct measurement our descriptor compares favourably to the other methods on the scenes of the benchmark, as well as on the average.

7.4.3 *Transfer learning on ETH dataset*

To check if ours proposal generalizes to in outdoor environments, we tested the learned descriptor on the ETH dataset. We get our model trained on the 3DMatch training set and perform a transfer learning on this dataset. The results are show in Tab. 7.7. Our descriptor achieves by far the best performance on this very challenging dataset with 97.5% average recall, outperforming all the state-of-the-art techniques by almost 20%. These

Table 7.6: Average number of correct correspondences on the 3DMatch benchmark. Best result on each column is in bold.

	Kitchen	Home 1	Home 2	Hotel 1	Hotel 2	Hotel 3	Study	MIT Lab	Average
FPFH	89	142	125	86	94	119	56	74	98
SHOT	154	206	182	131	124	159	84	121	145
SI	120	145	152	102	91	111	51	71	105
USC	150	216	175	147	120	159	97	161	153
3DMatch	103	134	125	73	64	64	64	84	88
CGF	125	156	142	90	94	130	55	78	108
3DSmoothNet	**274**	324	**318**	272	238	276	171	246	264
Ours	273	**336**	314	**307**	**277**	**310**	**226**	290	**292**

Table 7.7: Results on the *ETH* data set.

Method	Gazebo		Wood		Average
	Summer	Winter	Summer	Autumn	
FPFH	0.3860	0.1420	0.1480	0.2080	0.2210
SHOT	0.7450	0.4530	0.6170	0.6320	0.6118
3DMatch	0.2280	0.0830	0.1390	0.2240	0.1685
CGF	0.3750	0.1380	0.1040	0.1920	0.2023
3DSmoothNet	0.9130	0.8410	0.6780	0.7280	0.7900
Ours	**0.9239**	**0.9862**	**1.0000**	**0.9913**	**0.9753**

results show that our unsupervised approach is very flexible and could present remarkable results on a very challenging dataset, such as ETH without being trained on it.

7.4.4 *Qualitative results for surface registration*

To verify the quality of registrations provided by the proposed method, we wrap up our descriptor in a surface registration pipeline. Hence, after the descriptor matching stage, we use RANSAC [8] to retrieve the aligning motion matrix $T \in SE(3)$ between a pair of fragments. We follow the experimental setup of [159] for RANSAC parameters. We present qualitative reconstructions in Fig. 7.7 for the 3DMatch dataset and in Fig. 7.8 for the ETH dataset. We compare the registration results of our descriptor against the 3DMatch descriptor [159], as the baseline method for this dataset, and with 3DSmoothNet [191], both being supervised approaches. The examples show that the rigid motion matrix produced by matching our descriptors can successfully align two fragments better than our competitors and, sometimes present result qualitatively better than those obtained by registration

transformations provided as ground-truth, as we can see on the *Home 2* and *Hotel 2* scenes.

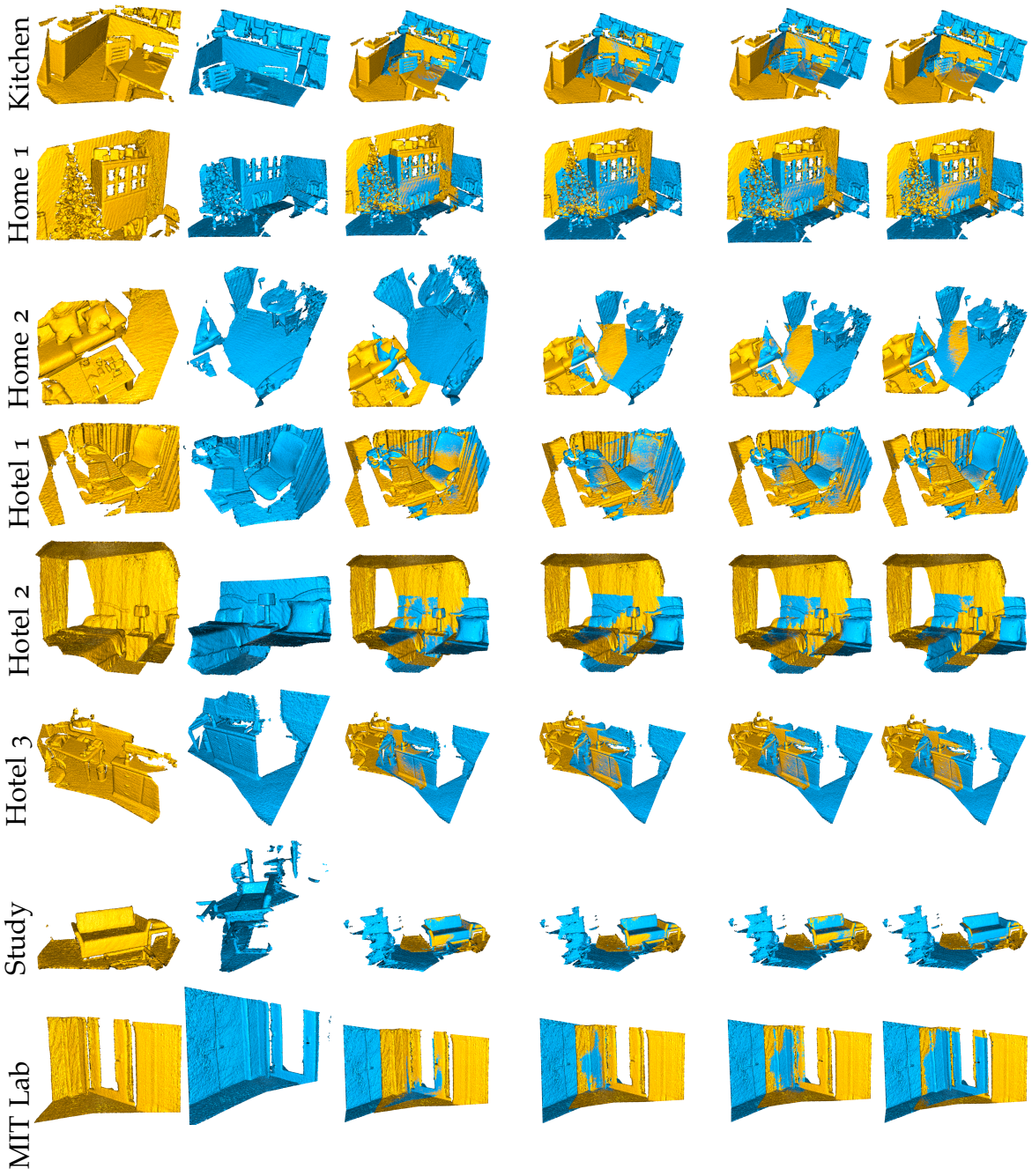


Figure 7.7: Registration results on the 3DMatch Benchmark after RANSAC.

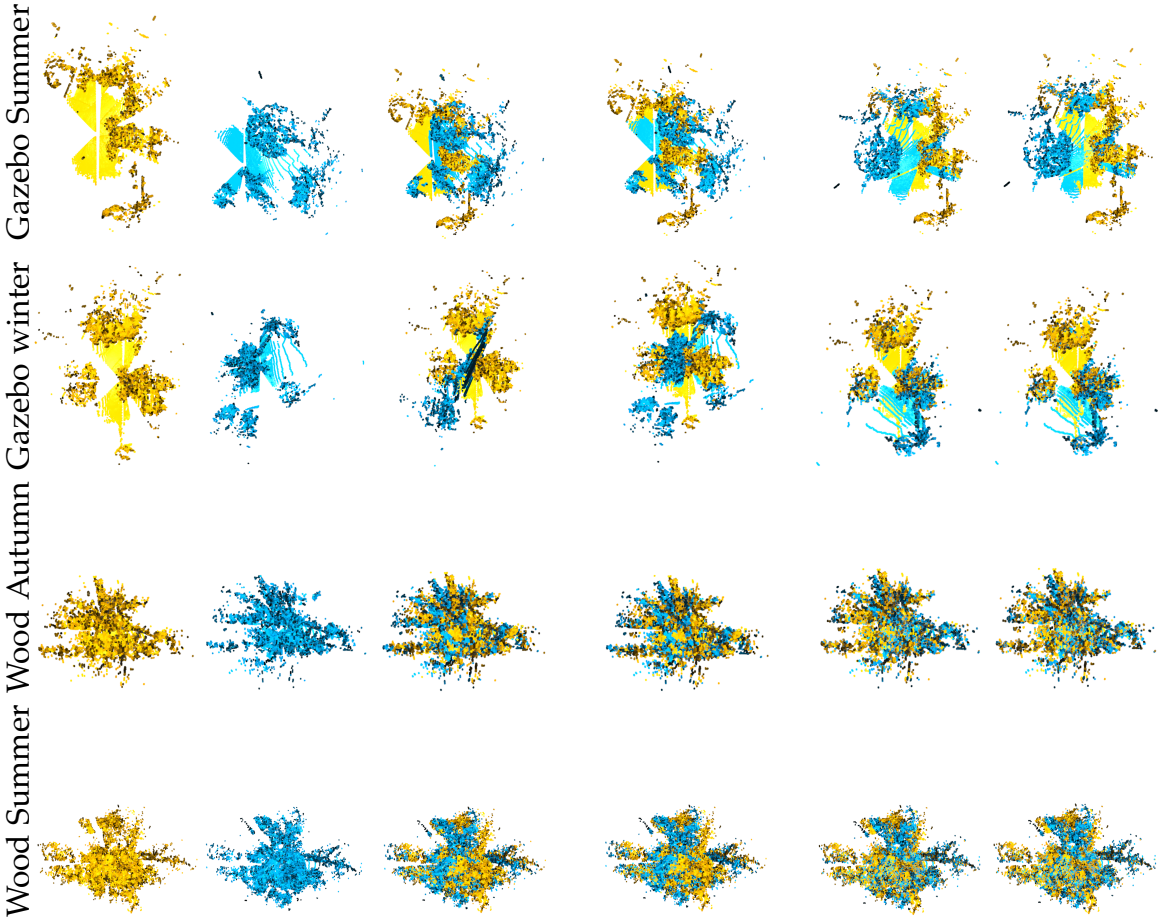


Figure 7.8: Registration results on the ETH Benchmark after RANSAC.

8

CONCLUSIONS

In Chap. 6, we have shown how the problem of learning an effective descriptor can be disentangled into the orthogonal problems of learning a robust equivariant representation and defining a good canonical orientation to make it invariant at test time. Our proposal to learn an equivariant representation in an unsupervised way leverages as encoder the recently proposed Spherical CNNs and turns out highly effective in tackling the first problem. When coupled with a robust algorithm to compute a local reference frame from the input cloud, it significantly advances the state of the art on a challenging benchmark.

We have also shown how the very same framework could be used to define a canonical orientation by exploiting the peculiar nature of the feature maps computed by the Spherical CNNs. Although this approach delivers performance on par with the state of the art, it is inferior to the use of an external LRF.

In the last part of this chapter, we have presented a detailed ablation study concerning the parameters of the Spherical CNNs encoder adopted. With a more effective architecture, we are able to set the new state-of-the-art performance for fragment registration recall on 3DMatch dataset and on ETH as well.

We hope that the proposed method fosters for further studies along this line of research, with the aim of defining an end-to-end learned solution to the problem of invariant 3D description.

Part III

ESTABLISHING AND LEARNING A ROBUST LOCAL REFERENCE FRAME

9

INITIAL REMARKS

The experimental results of the previous chapter, exposed in Sec. 7.3.3, have highlighted the strong link between the performance of the descriptor and the quality of the algorithm involved to achieve invariance to rotation. Indeed, as can be seen from Tab. 7.1, Ours SO and Ours LRF report the results for exactly the same descriptor, but with two different methods to compute a canonical orientation. We can thus point out that, the effectiveness of the descriptor directly depends on the reliability of its underlying LRF; in turn, the quality of the latter is determined by its invariance to transformations that can be observed in the data. However, this is a well-known problem in the literature of 3D local descriptors. In [63], Petrelli and Di Stefano have shown that the performance of a local descriptor is strongly tied to the repeatability of the LRF adopted.

Recently, there has been a significant increase in the interest around the topic, mainly because the new deep-learned local 3D descriptors rely either on a Reference Axis (RA) or on a LRF in order to be rotation-invariant [143, 161, 162, 191]. Objects under different rotations induce different features in the network as pointed out in [130], thus, the geometry of an object, or a local patch, has to be canonically oriented before being sent to the network. Based upon these considerations, we argue that the definition of a robust and repeatable LRF is still an open and challenging problem that underpins many existing approaches and is ubiquitous in many applications. For this reason, part of this dissertation will be concerned with the problem of the construction of a stable and repeatable local reference frame. A local reference frame, can be defined as a local system of Cartesian coordinates defined at each point, with respect to which the local geometric structure around that point is canonically oriented. For a given 3D shape $\mathcal{M}$, an LRF $\mathcal{L}(p)$ at point $p \in \mathcal{M}$ is an orthogonal set of unit vectors:

$$\mathcal{L}(p) = \{\hat{\mathbf{x}}(p), \hat{\mathbf{y}}(p), \hat{\mathbf{z}}(p)\} \tag{9.1}$$

satisfying the right-hand rule $\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}}$. LRFs algorithms are designed to be as repeatable as possible. To this end, they leverage geometric properties of

the surface to define cues that can be easily detected when the input data is affected by noise. In this sense, one of the main issues to be addressed is the robustness to self-occlusion or missing part, that may occur when an object or scene is captured from multiple points of view. According to [185], LRFs can be subdivided into two categories, LRFs based on covariance analysis and LRFs based on geometric attributes. The former family includes methods that define the axes of the LRF, for a given point $\mathbf{p}$, as the eigenvectors of the 3D covariance matrix computed on the local support of $\mathbf{p}$. Due to the ambiguity of the sign of the eigenvectors [37], is hard to define a repeatable directions, thus, efforts within these methods, have largely concentrated on the reliable disambiguation of the axes sign. As for the methods based on geometric attributes, they use as $\hat{\mathbf{z}}$ axes the normal at point $\mathbf{p}$ and then determine $\hat{\mathbf{x}}$ by identifying a reference point $\mathbf{q}$ in the support region. The difference vector $\mathbf{q} - \mathbf{p}$ is then projected on the tangent plane at $\mathbf{p}$. While the methods of the first category suffer from ambiguity of the sign, it is difficult for the latter to highlight geometric attributes that can be easily identified on heterogeneous dataset acquired with different type of sensors.

This brief introduction on the world of LRFs is mandatory to better understand the two fundamental contributions that we are going to expose in the next chapters. Indeed, the problems outlined above, have prompted us to formulate two different ways to establish a robust and repeatable LRF. In Chap. 10, we present an handcrafted method that exploits the intrinsic properties of the surface to define a repeatable orientation. Particularly, the $\hat{\mathbf{x}}$ axis of the proposed LRF is determined according to the computation of the intrinsic gradient of a scalar field defined on top of the input surface, while for the $\hat{\mathbf{z}}$ axis, we adopt the surface normal $\mathbf{n}$ at the feature point $\mathbf{p}$. In a different way, in Chap. 11, we introduce a self-supervised method that exploits the equivariance property of Spherical CNNs to learn a canonical orientation from raw point clouds data. The latter proposal extends the intuition behind the *self-orienting* descriptor described in Sec. 7.2.4, but differs significantly since we learn a canonical orientation instead of obtaining it as a by-product when learning a descriptor.

Before starting, in Sec. 9.1 we will cover most of the related work in the field of local reference frame.

A wide variety of different LRFs have been proposed in the literature, in this section we list the most important ones in terms of the influence they had on research and of the achieved performances.

9.1.1 *Mian*

The unit axes of the LRF proposed in [55] are given by the normalized eigenvectors of the covariance matrix of the 3D coordinates of the points, $\mathbf{p}_i$, in a spherical support of radius r centered at the feature point $\mathbf{p}$:

$$\mathbf{C}(\mathbf{p}) = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \hat{\mathbf{p}})(\mathbf{p}_i - \hat{\mathbf{p}})^T \tag{9.2}$$

where $\hat{p}$ indicates the barycenter of the points lying within the support:

$$\mathbf{C}(\mathbf{p}) = \frac{1}{k}\sum_{i=0}^{k}(\mathbf{p}_i - \hat{\mathbf{p}})(\mathbf{p}_i - \hat{\mathbf{p}})^T \tag{9.3}$$

As clearly discussed in [37], although the eigenvectors of Eq. 9.2 define the principal directions of the data, their sign is not defined unambiguously. Hence, this approach lacks of uniqueness of the signs of the axes of the estimated LRF.

9.1.2 *SHOT*

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

$$\mathbf{C}_w(\mathbf{p}) = \frac{1}{\sum_{i:d_i \leqslant r}(r-d_i)}\sum_{i:d_i \leqslant r}(r-d_i)(\mathbf{p}_i - \mathbf{p})(\mathbf{p}_i - \mathbf{p})^T \tag{9.4}$$

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 (Eq. 9.2) is replaced by the feature point $\mathbf{p}$. Then, similarly to [37], 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:

$$S_x^+ \doteq \{i : d_i \leqslant r \wedge (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^+ \geqslant 0\} \tag{9.5}$$

$$S_x^- \doteq \{i : d_i \leqslant r \wedge (\mathbf{p}_i - \mathbf{p}) \cdot \mathbf{x}^- > 0\} \tag{9.6}$$

$$\hat{\mathbf{x}} = \begin{cases} \mathbf{x}^+, & |S_x^+| \geqslant |S_x^-| \\ \mathbf{x}^-, & \text{otherwise} \end{cases} \tag{9.7}$$

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 $\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}}$.

9.1.3 *ROPS*

ROPS descriptor has been proposed in [83] and works on meshes. To compute the associated LRF, rather than calculating a single covariance matrix over the entire spherical support, covariance matrices are computed per-triangle and aggregated in a weighted sum. The weights are designed to counteract the variations in mesh resolution, and provide robustness to clutter and occlusions. The three axes are obtained via eigenvalue decomposition of the resulting covariance matrix. In order to eliminate the sign ambiguity, similarly to Sec. 9.1.2, the $\hat{\mathbf{x}}$ and axis is made to point in the direction of greater *mesh* density as follows:

$$\hat{\mathbf{x}} = \hat{\mathbf{x}} \cdot \mathrm{sign}(h) \tag{9.8}$$

where $(\cdot)$ returns the sign of a real number and h is defined as:

$$h = \sum_{T \in B_r(p)} w_1 w_2 \Big(\frac{1}{6} \sum_{i=0}^{3} (\mathbf{p_i}^T - \mathbf{p})\hat{\mathbf{x}}\Big). \tag{9.9}$$

Where $B_r(\mathbf{p})$ indicates the support of $\mathbf{p}$ and T the single triangle within it. The same procedure is applied to the $\hat{\mathbf{z}}$ axis. Since the $\hat{\mathbf{y}}$ axis is defined via $\hat{\mathbf{z}} \times \hat{\mathbf{x}}$, the LRF defined by ROPS is unique and unambiguous.

9.1.4 *EM*

In the method proposed by Novatnack and Nishino [40], the $\hat{\mathbf{z}}$ axis is given by the normal $\mathbf{n}$ at the feature point $\mathbf{p}$. As regards to the $\hat{\mathbf{x}}$ axis, the eigenvector of Eq. 9.2 associated with the largest eigenvalue is projected onto the tangent plane defined by $\mathbf{n}$. Then, the $\hat{\mathbf{y}}$ axis is given by $\hat{\mathbf{z}} \times \hat{\mathbf{x}}$.

9.1.5 *Board*

Board LRF [63] follows [40] in using the normal vector $\mathbf{n}$ at point $\mathbf{p}$ as the $\hat{\mathbf{z}}$ axis. To compute a robust normal vector, a tangent plane is fitted to the entire $B_r(\mathbf{p})$, then its normal vector $\hat{\mathbf{z}} = \pm\mathbf{n}$ is disambiguated by taking the sign yielding a positive inner product with the average normal of the points in $B_r(\mathbf{p})$. Since normals prove repeatable, the intuition for the estimation of the $\hat{\mathbf{x}}$ axis rely again on surface normals. Hence, the method searches for a reference point, $\mathbf{q}$, considering all the p_i points within the support, and selects the one with the largest angular deviation with respect to $\mathbf{n}$. Then the $\hat{\mathbf{x}}$ axis is pointed towards $\mathbf{q}$. For the sake of robusteness, instead of considering n_i as normal vector, a more robust normal is computed by averaging normals over a small neighborhood of p_i. Through experimental results, the authors have demonstrated that $\mathbf{q}$ lies close to the margin of the support. To speed up the computation, only the points having distance greater than $(0.85 \times r)$ from $\mathbf{p}$, are considered when searching for $\mathbf{q}$.

9.1.6 *FLARE*

The follow up work of Sec. 9.1.5 is referred to as FLARE [75]. The authors follow a similar approach whereas $\mathbf{q}$ is selected as the point within the periphery of the support exhibiting the largest signed distance (rather than angle) to the tangent plane. Thus, it computes the $\hat{\mathbf{z}}$ axis as in Sec. 9.1.5, but for the $\hat{\mathbf{x}}$ axis it uses the signed distance instead of the cosine with the normal. The signed distance of a point $\mathbf{p}_i$ is defined as:

$$d_{SD}(\mathbf{p}_i) = \mathbf{p}\mathbf{p}_i \cdot \hat{\mathbf{z}} \tag{9.10}$$

with $\mathbf{p}\mathbf{p}_i$ being the vector going from $\mathbf{p}$ to $\mathbf{p}_i$. The algorithm selects the point with the largest signed distance, which is the point whose component

parallel to the $\hat{\mathbf{z}}$ axis is the largest, meaning the most distant point from the tangent plane at point $\mathbf{p}$. As in Sec. 9.1.5, the projection of this point to the tangent plane is normalized and taken as the $\hat{\mathbf{x}}$ axis.

9.1.7 *TOLDI*

TOLDI [157] proposes a new local reference frame together with a method for local shape description. As regards the LRF, it computes the $\hat{\mathbf{z}}$ axis similar to covariance analysis methods, presented in the initial remarks of this part, as it computes the covariance matrix (using the barycenter) and considers as $\mathbf{z}^+$ the unit eigenvector with the smallest eigenvalue. The first novelty is that this covariance is computed using only $\frac{1}{3}$ of the points within the support instead of using all of them. The other novelty is in the disambiguation of the sign, that follows the rule:

$$\hat{\mathbf{z}} = \begin{cases} \mathbf{z}^+ & \text{if } \mathbf{z}^+ \cdot \sum_{\mathbf{p}_i \in \mathcal{B}_r(\mathbf{p})} \mathbf{p}_i\mathbf{p} \geqslant 0 \\ -\mathbf{z}^+ & \text{otherwise} \end{cases} \tag{9.11}$$

The $\hat{\mathbf{x}}$ axis will lie in the tangent plane of $\mathbf{p}$ with respect to the normal $\hat{\mathbf{z}}$, but finding an orientation for such plane is more difficult than finding $\hat{\mathbf{z}}$.
The first step is the computation of the projections of the support points on the plane. For each $\mathbf{p}_i \in B_r(\mathbf{p})$,

$$\mathbf{v}_i = \mathbf{p}\mathbf{p}_i - (\mathbf{p}\mathbf{p}_i \cdot \hat{\mathbf{z}}(p)) \cdot \hat{\mathbf{z}}(p) \tag{9.12}$$

The $\hat{\mathbf{x}}$ axis is then computed as a weighted sum:

$$\hat{\mathbf{x}} = \frac{1}{\left\| \sum_{i=0}^{k} w_{i1} w_{i2} \mathbf{v}_i \right\|_2} \sum_{i=0}^{k} w_{i1} w_{i2} \mathbf{v}_i \tag{9.13}$$

In this sum, w_{i1} is related to the distance of the point and is designed to improve robustness to clutter, occlusion and incomplete border regions, while w_{i2} is set to make the points with larger projection distance contribute more to the $\hat{\mathbf{x}}$ axis, since such distance feature is a distinctive cue and can provide high repeatability on flat regions. They are defined as follows:

$$w_{i1} = (r - \|\mathbf{p} - \mathbf{p}_i\|_2)^2 \tag{9.14}$$

$$w_{i2} = (\mathbf{pp}_i \cdot \hat{\mathbf{z}}(\mathbf{p}))^2 \tag{9.15}$$

As usual, the $\hat{\mathbf{y}}$ axis is computed by cross product.

10

GRADIENT-BASED LOCAL REFERENCE FRAME FOR 3D SHAPE MATCHING

In the initial remarks of this part, we have discussed about the main limitations of state-of-the-art algorithms for LRF estimation and how they are sensitive to different sources of noise. In this chapter, we propose a novel LRF that is demonstrably robust to severe sampling artifacts, vertex noise, and object deformation. Key to our method is the definition of the tangent component as the intrinsic gradient of a scalar function defined on the 3D object. Thus, we dub it *GFrames*. We will also consider dataset with non rigid-transformations, depicting, thus, a challenging experimental setup. The choice of the function directly determines the invariance classes of the resulting LRF; by doing so, we crucially shift the key difficulties of directly dealing with the object geometry to the simpler manipulation of a vector space of real-valued functions. The intrinsic construction further makes our LRF a natural choice in the more challenging non-rigid setting (see Fig. 10.1). GFrames can be used as-is to improve existing descriptors and provide a robust choice in applications requiring a repeatable LRF. An example on the dog shape is shown in Fig. 10.1, where the LRF obtained with our approach turns out to be more robust in comparison to a de-facto standard LRF (SHOT [98], Sec. 9.1.2). We introduce a novel, theoretically principled LRF for 3D shapes that is remarkably robust to *sampling*, and that can be made provably invariant to *non-rigid* near-isometric transformations. Our algorithm is simple and effective, as we are going to demonstrate benchmarking on datasets addressing deformable matching of meshes as well as rigid point cloud registration. In order to provide a more in-depth study on the link between the repeatability of the LRF and the performance of the descriptors, we make a special effort of comparing on benchmarks which include tasks such as deformable matching of complete meshes and registration of partial point clouds, demonstrating in both cases the effectiveness of our approach. The results obtained prove that our method can be used broadly across the board.

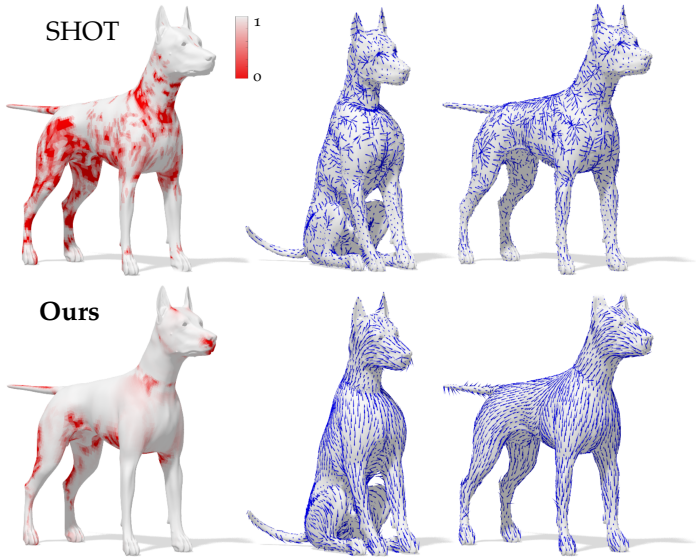


Figure 10.1: Comparison of LRF repeatability measured as mean cosine error on two non-rigid poses of the dog shape. We compare with the de-facto standard SHOT [98]. *Left*: The error is encoded as a heat map, growing from white (perfectly aligned LRFs) to red (gross misalignment). *Right*: The computed LRFs; we only show the $\hat{\mathbf{x}}$ axes for visualization purposes.

10.1 RELATED WORK

We briefly recap the proposals already explained in Sec. 9.1. A robust and repeatable LRF is a key component for most handcrafted 3D local descriptors, such as fast point feature histograms [45], exponential mapping [40], SHOT [98], ROPS [83], unique shape contexts [58], and point signatures [17], to name just a few. Furthermore, robust local frames have a crucial role in the recent geometric deep learning approaches [120], constructing non-Euclidean analogies of CNNs on meshes through local patch operators [111, 118, 146]. Local descriptors that *do not* exploit an LRF are either not distinctive enough [49], costly to compute [44], or suffer from poor performance in the presence of noise and missing parts [47]. Covariance analysis family includes methods that define the axes in $\mathcal{L}(\mathbf{p})$ as eigenvectors of the 3D covariance matrix between points lying within the spherical support $B_r(\mathbf{p})$. Inherent to such methods is the *sign ambiguity* of eigenvectors, making it hard to define repeatable directions; thus, efforts have largely concentrated on the reliable disambiguation of the axes sign. In [40], no disambiguation takes place, and the axis $\hat{\mathbf{x}}$ is simply defined as the principal eigenvector projected onto the tangent plane defined by the normal $\mathbf{n}$ (assumed to be given as input). In [55], all the three axes are given

directly by the eigenvectors; however, here $\pm\hat{\mathbf{z}}$ is disambiguated by evaluating the two inner products $\langle \mathbf{n}, \pm\hat{\mathbf{z}} \rangle$ and keeping the sign yielding a positive number. Axis $\hat{\mathbf{x}}$ nevertheless remains ambiguous. The LRF proposed with the SHOT descriptor [98] employs a different covariance matrix, where the contributions of the points in $B_r(\mathbf{p})$ are weighted by their distance to $\mathbf{p}$. Sign ambiguity is addressed by choosing the sign that makes the eigenvector consistent with the majority of the measurements [37]; in practice, this results in the $\hat{\mathbf{x}}$ axis pointing in the direction of greater sample density. Similarly, in the ROPS descriptor [83] the axis $\hat{\mathbf{x}}$ is made to point in the direction of greater mesh density. Methods based on geometric attributes determine $\hat{\mathbf{x}}$ by identifying a reference point $\mathbf{q} \in B_r(\mathbf{p})$ within the support region. As an early example, point signatures [17] first fit a plane to the boundary path $\gamma = \mathcal{M} \cap B_r(\mathbf{p})$; the reference is then selected as the point $\mathbf{q} \in \gamma$ with the largest positive distance to the fitted plane. In [63], a tangent plane is fitted to the entire $B_r(\mathbf{p})$; its normal vector $\hat{\mathbf{z}} = \pm\mathbf{n}$ is disambiguated by taking the sign yielding a positive inner product with the average normal of the points in $B_r(\mathbf{p})$. The reference is then taken as the point $\mathbf{q} \in B_{r'>r}(\mathbf{p})$ having the largest angular deviation with respect to $\mathbf{n}$. The method of [75] follows a similar approach, whereas $\mathbf{q}$ is selected as the point exhibiting the largest signed distance (rather than angle) to the tangent plane. Finally, several deep learning-based 3D descriptors have been proposed in recent years, with most of them relying upon fixed LRFs in order to achieve rotation invariance. This is the case of proposals like CGF-32 [143], PPFNet [162], and metric learned SHOT [122] which all deploy the LRF proposed in [98]. In ACNN [118, 119] and MoNet [146] architectures, the local patches are oriented using the principal curvature direction. Differently, the PointNet architecture [148] uses a spatial transformer network [107] to predict a rigid transformation to canonically align the input data, while PCPNet [166] applies the spatial transformer locally.

10.2 ESTABLISHING A GRADIENT-BASED LOCAL REFERENCE FRAME

In this section, we present our proposal, but first we start with a description of the mathematical background necessary to better understand how we formulate and compute the gradient. Since our method can work on 3D shapes represented as meshes or point clouds. We first open with a continuous mathematical model and then discuss the discretization.

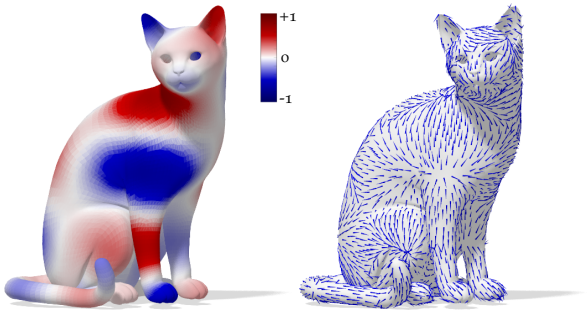


Figure 10.2: A scalar field on shape $\mathcal{M}$, and its intrinsic gradient ∇f.

10.2.1 *Background*

Manifolds: We assume that our shapes arise from the sampling of 2-dimensional Riemannian manifolds (surfaces) $\mathcal{M}$, possibly with boundary $\partial\mathcal{M}$, embedded into $\mathbb{R}^3$. Locally around each point $x \in \mathcal{M}$, the manifold is homeomorphic to the *tangent plane* $T_p\mathcal{M}$; the disjoint union of all such planes forms the *tangent bundle* $T\mathcal{M}$. We further equip the manifold with a *Riemannian metric*, defined as an inner product $\langle \cdot, \cdot \rangle_{T_p\mathcal{M}} : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ on the tangent plane depending smoothly on $\mathbf{p}$. Functions of the form $f : \mathcal{M} \to \mathbb{R}$ and $F : \mathcal{M} \to T\mathcal{M}$ are referred to as *scalar-* and *(tangent) vector fields*, respectively. Properties expressed solely in terms of the metric are called *intrinsic*. In particular, *isometric* deformations of the manifold (such as a change in pose) preserve all intrinsic structures.

Intrinsic gradient: In classical calculus, derivatives describe how a function f changes with an infinitesimal change of its argument x. Due to the lack of vector space structure on the manifold (meaning that we cannot add two points, *i.e.*, an expression like "$p + dp$" is meaningless), one needs to define the *differential* of f as an operator $df : T\mathcal{M} \to \mathbb{R}$ acting on tangent vector fields. At each point p, the differential is a linear functional $df(p) = \langle \nabla f(p), \cdot \rangle_{T_p\mathcal{M}}$ acting on tangent vectors $F(p) \in T_p\mathcal{M}$, which models a small displacement around p. The change of the function value as the result of this displacement is given by applying the differential to the tangent vector, $df(p)F(p) = \langle \nabla_{\mathcal{M}} f(p), F(p) \rangle_{T_p\mathcal{M}}$. This can be thought of as an extension of the notion of the classical directional derivative, where the linear operator $\nabla_{\mathcal{M}} f : L^2(\mathcal{M}) \to L^2(T\mathcal{M})$ is called the *intrinsic gradient*, and is similar to the classical notion of the gradient defining the direction of the steepest change of the function at a point, with the only difference that the direction is now a tangent vector; see Fig. 10.2 for an example.

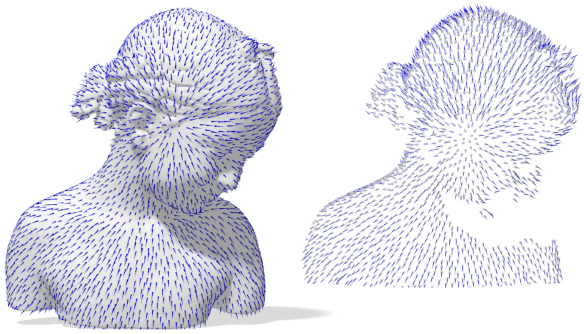


Figure 10.3: Gradient estimation on a triangle mesh (left) and on a sparse point cloud representing a partial scan of the object (right). Our approach only needs a notion of a tangent space to be applicable to any given representation.

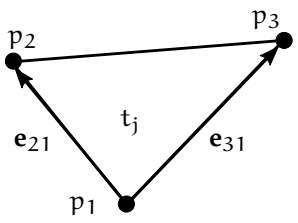
Discretization: Let us now assume the manifold is sampled at n points $p_1, \ldots, p_n$, being the most basic representation of the shape called a point cloud. Equipping it further with a simplicial structure with edges $\mathcal{E}$ and triangular faces $\mathcal{F}$ yields a triangular mesh, which we assume to be a (discrete) manifold. Scalar functions $f : \mathcal{M} \to \mathbb{R}$ are represented as vectors $\mathbf{f} = (f(p_1), \ldots, f(p_n))^\top$ encoding the value of f at each point. Following standard practice, functions are assumed to behave linearly between neighboring points (within each triangle in the case of meshes).

On meshes, the discrete intrinsic gradient ∇f yields tangent vector fields defined on the mesh *triangles*; on each triangle t_j, it is computed as a 3D vector

$$\nabla f(t_j) = \begin{pmatrix} \mathbf{e}_{21} & \mathbf{e}_{31} \end{pmatrix} \begin{pmatrix} E & F \\ F & G \end{pmatrix}^{-1} \begin{pmatrix} f(p_2) - f(p_1) \\ f(p_3) - f(p_1) \end{pmatrix} \tag{10.1}$$

where $E = \|\mathbf{e}_{21}\|^2$, $F = \langle \mathbf{e}_{21}, \mathbf{e}_{31} \rangle$, and $G = \|\mathbf{e}_{31}\|^2$ (see the inset for the notation).

On point clouds, intrinsic gradient is discretized as follows. For each point p, we first estimate its tangent space by locally fitting a plane to points within radius r around p. These points are projected onto the plane, where they are locally meshed into a triangle patch $\mathcal{P}$ using Delaunay triangulation. We then take the weighted average $\nabla f(p) = \frac{1}{\sum A(t_j)} \sum_{t_j \in \mathcal{P}} A(t_j) \nabla f(t_j)$, where $A(t_j)$ denotes the area of triangle t_j.



We remark that this procedure is *not* equivalent to applying Eq. 10.1 to a surface reconstruction of the point cloud: only a *local* reconstruction is

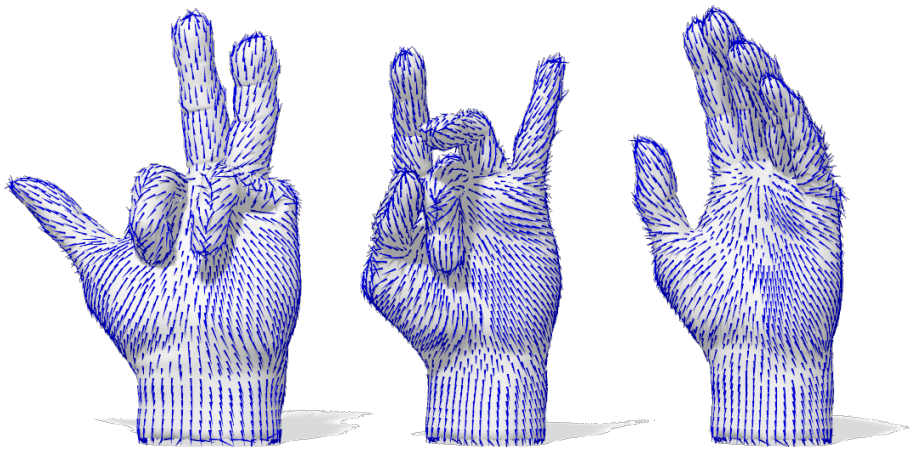


Figure 10.4: The $\hat{\mathbf{x}}$ axis of our LRF on different hand poses. In this example, repeatability is almost ideal due to the repeatability of the chosen scalar function $f(p_i) = \frac{1}{n}\sum_{j=1}^{n} d(p_i, p_j)$ equal to the average geodesic distance [31] from each point to all the others.

carried out at each point p, and then thrown away once $\nabla f(p)$ is estimated. This brings additional robustness and efficiency in the presence of clutter or large point clouds; see Fig. 10.3 for an example.

Finally, normals on point clouds are estimated via standard total least squares [26]; for triangle meshes, the normal $\mathbf{n}(p)$ at a vertex p is computed as the area-weighted average of the triangle normals of the triangles sharing the vertex p.

10.2.2 *GFrames*

Our technique is based upon the construction of tangent vector fields as *gradients* of scalar functions $f : \mathcal{M} \to \mathbb{R}$. We compute the *average gradient* of f around p as:

$$\mathbf{x}(p) := \frac{1}{\sum_{t_j \in \mathcal{N}_r(p)} A(t_j)} \sum_{t_j \in \mathcal{N}_r(p)} A(t_j) \nabla f(t_j) \tag{10.2}$$

where $\mathcal{N}_r(p)$ is the set of triangles within distance r from p.

While it brings resilience to noise, the averaging process does not guarantee orthogonality to the normal vector $\mathbf{n}(p)$. We thus project $\mathbf{x}(p)$ onto the

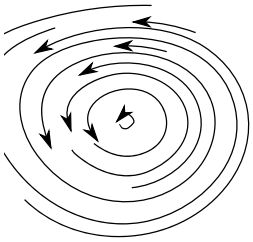
plane identified by $\mathbf{n}(p)$ and rescale the projection to unit norm, leading to the reference frame $\mathcal{L}_f(p) = \{\hat{\mathbf{x}}(p), \hat{\mathbf{y}}(p), \hat{\mathbf{z}}(p)\}$:

$$\hat{\mathbf{x}}(p) := (\mathbf{x}(p) - (\mathbf{x}(p)^\top \mathbf{n}(p))\mathbf{n}(p))_{\|\|} \tag{10.3}$$

$$\hat{\mathbf{y}}(p) := \hat{\mathbf{z}}(p) \times \hat{\mathbf{x}}(p) \tag{10.4}$$

$$\hat{\mathbf{z}}(p) := \mathbf{n}(p) \tag{10.5}$$

where $(\cdot)_{\|\|}$ denotes vector normalization and the notation $\mathcal{L}_f$ emphasizes that the definition of the LRF depends on the choice of the scalar function f.



Being defined on top of a gradient ∇f, our frames are guaranteed *curl-free* (*i.e.*, they never behave like local vortices, see inset). This is desirable, since vortex-like patterns lead to strong LRF inconsistency. Eq. 10.2 requires f to be differentiable; this is always true in our case, due to the assumption of piecewise-linearity. A separate question concerns the presence of singular points (where $\nabla f(p) = \mathbf{0}$). In the particular case of closed (genus zero) surfaces, these are unavoidable due to the Poincaré-Hopf ("Hairy Ball") Theorem stating that the only surface with nowhere vanishing tangent vector field is torus-like (genus 1); as we will show in our experiments, however, such points are rare and do not affect the overall quality of the LRF.

The choice of the function plays a role in determining the invariance class induced by the LRF. A specific choice thus depends on the task: for instance, in order to achieve invariance to 3D rotations, it is sufficient that the function does not depend on the position of the object in space (an example is mean curvature). We will provide several possible choices in the experimental Sec. 10.4.

From a different perspective, constructing local descriptors $d : \mathcal{M} \to \mathbb{R}^k$ on top of a smooth frame field $\mathcal{L}_f$ can be seen as "steering" the descriptor field d along a given flow. For shape matching and surface registration applications, this fact can be exploited by designing flows that make use of prior knowledge (in the form of sparse input correspondence). Specifically, given a single point-wise match $(x^*, y^*) \in \mathcal{M} \times \mathcal{N}$, the simple Euclidean distance from x^* (resp. y^*) to all other points in $\mathcal{M}$ (resp. $\mathcal{N}$) have compatible gradients (see Fig. 10.3), making our LRFs an ideal choice in correspondence pipelines.

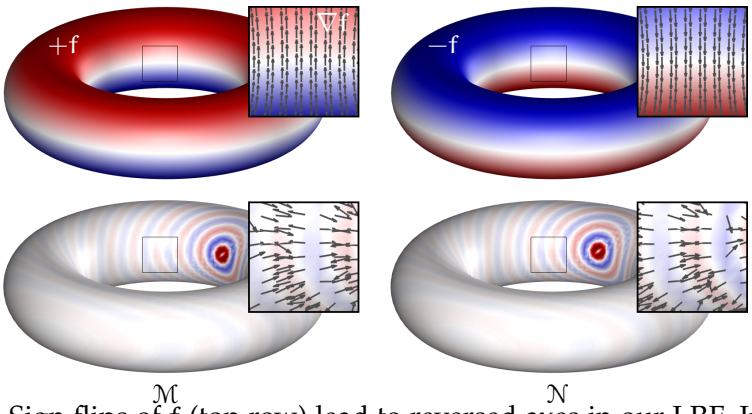


Figure 10.5: Sign flips of f (top row) lead to reversed axes in our LRF. In the bottom row, two high-frequency functions which are not exactly repeatable on $\mathcal{M}$ and $\mathcal{N}$ lead to local axis flips.

10.3 PROPERTIES OF GFRAMES

We list some of the key properties that make our proposed LRFs suitable for challenging settings. Additional properties depend on the choice of the underlying function, and will be explored in the experimental section.

Invariance properties depend on the choice of the scalar function f; for example, mean curvature endows the LRF with rotation invariance, and Gaussian curvature with invariance to isometric deformations. If available, it is also possible to use color texture as f. The chosen function must be repeatable only up to a global scale, since $\nabla \alpha f = \alpha \nabla f$ for any $\alpha \in \mathbb{R}$ and the normalization Eq. 10.3 make all options automatically scale-invariant; see Fig. 10.4 for an example.

Sign ambiguity is arguably the key issue of existing LRFs (see Sec. 10.1), with a direct impact on their reliability. Our frames do not suffer from sign ambiguity unless the sign of f is flipped (Fig. 10.5, top), or if f contains high frequency oscillations (Fig. 10.5, bottom).

Robustness to sampling is another central weakness of many state-of-the-art LRFs [75, 98]. To the best of our knowledge, none of the existing methods is able to deal with strong differences in sampling (arising, for example, when matching a CAD model to a 2.5D scan) or severe subsampling. We run a full quantitative comparison with such methods in Fig. 10.6, using the repeatability measure defined in Sec. 10.4.1; for these and the following tests, we average over a representative dataset of six different shape classes (cat, centaur, dog, hand, human, squirrel) of varying resolution (ranging from 6K to 28K vertices).

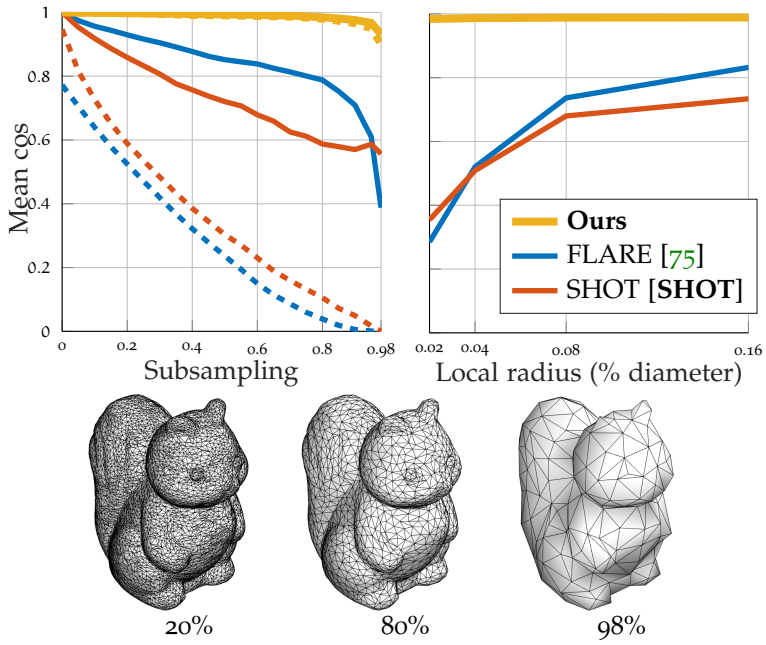


Figure 10.6: *Top left*: LRF repeatability under increasing subsampling, from 0% (no subsampling) to 98% (severe). We report results obtained with local radius $r = 0.02$ (dashed) and $r = 0.16$ (solid); all shapes have unit diameter. *Top right*: Comparisons at increasing radius, averaged over all subsampling levels. *Bottom*: example of subsampled shapes used in these tests.

Robustness to noise is achieved by averaging the gradient over a local neighborhood (10.2), similar to existing approaches. A crucial difference, however, is that our LRFs also leverage the smoothness of the *function* $f : \mathcal{M} \to \mathbb{R}$ in contrast to the smoothness of the 3D object $\mathcal{M}$ itself. This way, we shift the problem of dealing with a noisy geometric domain to a far easier task of choosing a smooth enough function on top of it. In Fig. 10.7 we show a full quantitative evaluation at increasing amounts of surface noise.

Symmetry disambiguation is another property unique to our method. Choosing an asymmetric f (*e.g.*, distance to a point) directly endows descriptors constructed on top of $\mathcal{L}_f$ with symmetry-awareness. The lack of such property is considered a crucial drawback in shape analysis applications, leading to ambiguous solutions in most top-performing shape matching pipelines.

10.4 EXPERIMENTAL RESULTS

We evaluate GFrames in different applications and settings, including rigid surface registration and deformable matching.

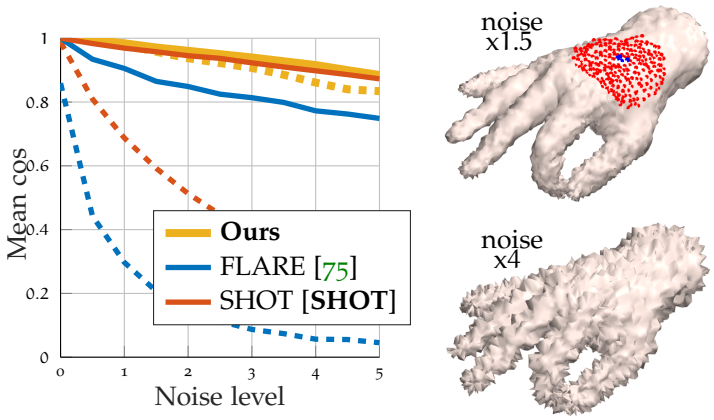


Figure 10.7: LRF repeatability at increasing surface noise (expressed as a multiplier of mesh resolution), obtained with radius $r = 0.02$ (dashed) and $r = 0.16$ (solid). Our results are better than FLARE and comparable with SHOT while using a much smaller radius; for comparison, on the top hand we plot the neighborhood at $r = 0.02$ (in blue) and $r = 0.16$ (in red). Due to the use of much smaller radius, our LRFs are much more robust to clutter and partiality.

10.4.1 *LRF repeatability and rigid shape matching*

Datasets

We use real scans of 4 objects (*Armadillo*, *Bunny*, *Buddha* and *Dragon*), from the Stanford 3D Scanning Repository [15], we refer to this dataset as Stanford Views. Some views of these objects are depicted in Fig. 10.8. Ground-truth transformations are available for all the scans.

Problem formulation

We evaluate the proposed method through the repeatability of the LRF and the efficiency of the descriptor built on the LRF. LRF repeatability at corresponding points on different shapes is assessed via the *MeanCos* and *ThCos* metrics [75]. The former represents the alignment error of both the $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ axes, while the latter indicates whether the two reference systems are aligned or not. More specifically, *ThCos* is the percentage of LRFs with a *MeanCos* value above a certain threshold (we used the value of 0.97). To evaluate the LRFs at corresponding keypoints we build point cloud pairs, as in Sec. 4.3.2 and in Sec. 7.3.2. For the sake of completeness we recapitulate the process. For each of M scans for a given test model, we extract a set of uniformly distributed keypoints, and take all possible $N = \frac{M(M-1)}{2}$ view

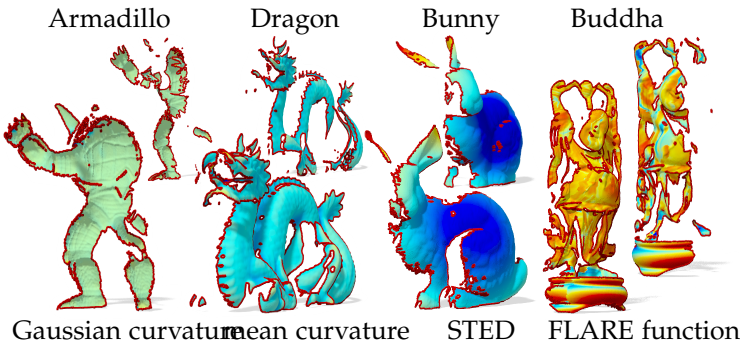


Figure 10.8: Example views from the Stanford repository. On each object we plot one of the four scalar functions used for the rigid matching experiments. Note how, despite baseline curvatures appear almost constant, they still exhibit enough gradient to outperform the SHOT LRF in most of our tests (compare with Fig. 10.9).

pairs (V_i, V_j). Due to partial overlap, a keypoint belonging to V_i may have no correspondence in V_j. Hence, given the ground-truth transformations G_i, G_j bringing V_i, V_j into a canonical frame, we compute the set:

$$\mathcal{O}_{i,j} = \{k_i : \|\mathbf{G}_i k_i - \mathcal{N}(\mathbf{G}_i k_i, \mathbf{G}_j V_j)\| \leqslant \epsilon_{ovr}\}, \tag{10.6}$$

containing the keypoints in V_i that have a corresponding point in V_j. Here, $\mathcal{N}(G_i k_i, G_j V_j)$ denotes the nearest neighbor of $G_i k_i$ in the transformed view $G_j V_j$, and ϵ_{ovr} is set to 2.5ρ[1]. If the number of points in $\mathcal{O}_{i,j}$ is less than 20% of the keypoints in V_i, the pair (V_i, V_j) is discarded due to insufficient overlap; otherwise, keypoint correspondences are established via nearest neighbor search in $\mathbb{R}^3$. Then, given a pair of corresponding keypoints $(k_i, k_j) \in V_i \times V_j$, we compute their LRFs and evaluate their repeatability according to the *MeanCos* and *ThCos* metrics.

Choice of the scalar function

The freedom of choosing f is a big advantage, allowing us to better adapt to the task at hand. As baseline choices, we use the aforementioned mean (Ours mean) and Gaussian (Ours Gauss) curvature. In addition, we use the following two functions:

STED (sum of total Euclidean distances) is simply defined as the sum $f(x_i) = \sum_{j=1}^{n} \|x_i - x_j\|_2$.

[1]Point cloud resolution ρ is the average Euclidean distance from each point to its nearest neighbor.

FLARE originally proposed in [75], it is computed at each point p as the average of the signed distances to the tangent plane defined by the normal $\mathbf{n}(p)$, computed only on a subset of points lying at the periphery of the support region. An example of each scalar function is shown in Fig. 10.8.

Results

In Fig. 10.9, we compare our LRF construction to that used in the state-of-the-art SHOT [98] and FLARE [75] methods. Our scalar functions result effective in achieving a repeatable LRF. Ours FLARE is consistently better than SHOT and FLARE LRFs on both metrics, while Ours STED outperforms them in terms of MeanCos. Note how Ours FLARE always outperforms the original FLARE method, highlighting the usefulness of relying on gradient-based LRFs for better repeatability. Ours STED tends to be less repeatable in terms of ThCos; the STED function is more sensitive to scan overlap, as we noticed a significant improvement with the increase of the overlap. A qualitative comparison on two views of a room from the RGB-D SLAM dataset [77] is further shown in Fig. 10.11, confirming the large improvement produced by our approach also on this type of real-world data. Finally, Fig. 10.10 reports a comparison in terms of descriptor matching, where the SHOT descriptor is used on top of each LRF. These results confirm the trend of the previous tests; Ours STED and Ours FLARE exhibit best accuracy, with the former having larger error on the Buddha model, which has smaller overlap compared to the other objects.

10.4.2 *Deformable shape matching*

Datasets

We adopt real-world as well as synthetic datasets: TOSCA [38] (seven classes of synthetic animal and human meshes undergoing non-rigid deformations), FAUST [92] (100 meshes from scans of ten different human subjects in different poses), TOPKIDS [126] (15 synthetic human meshes in different poses, with severe topological artifacts in areas of self-contact). Examples of shapes used in our experiments are shown in Fig. 10.12. All datasets come with ground-truth correspondence; for cross-dataset experiments, the ground-truth is estimated using the state-of-the-art shape registration method FARM [177].

Problem formulation

We applied the descriptors constructed using LRF to the problem of deformable shape matching. Pointwise correspondence is established by nearest neighbors in descriptor space. The obtained correspondences are then evaluated according to the Princeton protocol [59], computing the percentage of matches that fall into geodesic radius r (represented as a fraction of the shape geodesic diameter) from the ground truth correspondence.

Choice of the scalar function

We adopt the same baseline as the rigid setting (mean and Gaussian curvature) plus two functions specific to this task (see examples in Fig. 10.13):
Fiedler vector is the first non-constant eigenfunction of the Laplace-Beltrami operator of the surface. Except for a sign ambiguity (simply solved by taking the square of the function), it is fully intrinsic and thus invariant to isometries.
Discrete Time Evolution Process (DEP) is a recent intrinsic point descriptor that is stable to non-isometric distortions, missing parts, and topological noise [178].

Results

We consider four different settings of deformable shape matching (see Fig. 10.14):
Isometric deformations: We test on 8 pairs of deformable animals (TOSCA, dog and horse categories) and 20 scans of a human subject in different poses (FAUST intra). We report an improvement of ~ 10% over the de-facto LRF.
Non-isometric deformations: We test on 20 pairs of different poses and different *subjects* (FAUST inter), demonstrating similar performance to the previous setting.
Topological noise: We evaluate on 15 poses of a synthetic human subject undergoing severe topological variations, *e.g.*, gluing hands to the body (TOPKIDS). In this challenging setting, the advantage of our model (Ours DEP and Ours Fied) over the baseline is even more pronounced.
Different connectivity and resolution: We compose a hybrid dataset of human shape pairs from SMPL [110], TOSCA and SPRING [101]; see the last three columns of Fig. 10.12 for examples. This experiment is particularly challenging due to the differences in mesh connectivity and density among

the various models. Such as setting is a notoriously tough nut for existing LRFs and local descriptors, and is not frequently considered in existing benchmarks. Nevertheless, we still outperform the baseline.

In Fig. 10.15 we show the matching error in one standard (FAUST) and one challenging (TOPKIDS) case. Finally, on TOSCA we also evaluate the LRF repeatability (MeanCos) over all 64 pairs of the dog and horse classes. SHOT achieves an average score of 0.76, while Ours Fied reaches up to 0.90 (close to ideal). A qualitative evaluation of this result is shown for a dog pair in Fig. 10.1 using Ours DEP.

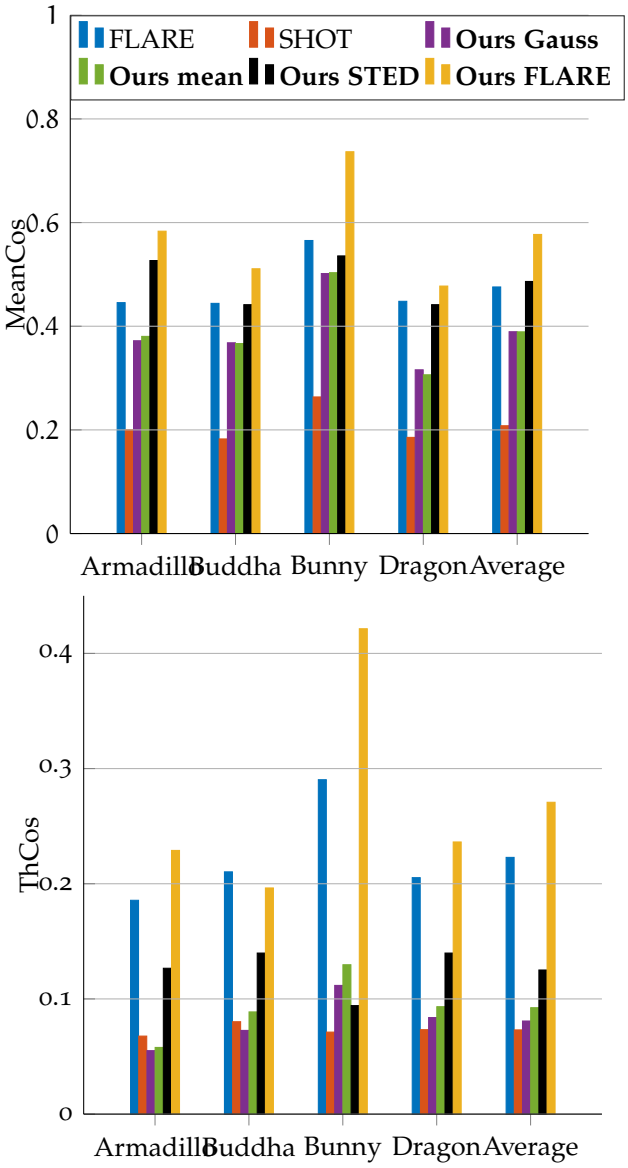


Figure 10.9: LRF repeatability on the Stanford Views dataset (the higher the better). Here, SHOT denotes the LRF of the SHOT descriptor.

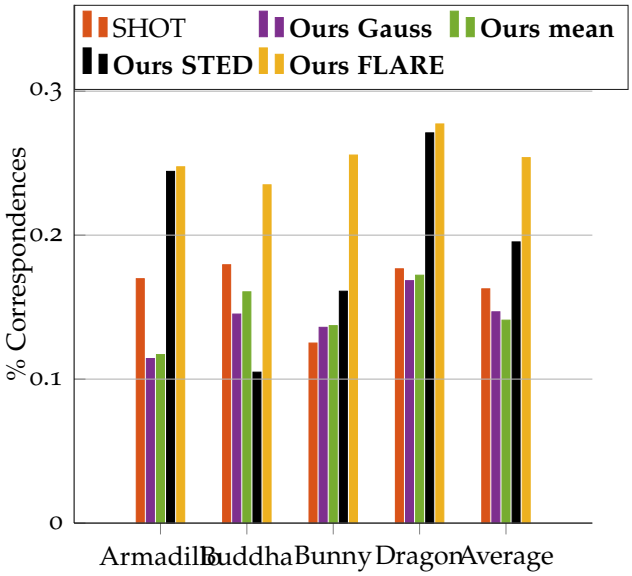


Figure 10.10: Descriptor matching results using the SHOT *descriptor* computed on different LRFs (among which the SHOT LRF itself). The y-axis denotes the percentage of matches whose Euclidean distance from the ground truth is less than 7mm.

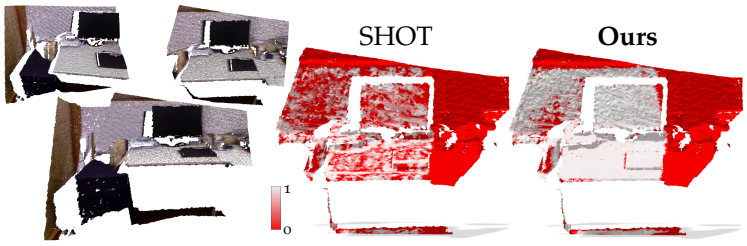


Figure 10.11: LRF repeatability on two views of a room (depicted on the left; their alignment is on the bottom). MeanCos error is encoded as a heat map, growing from white to red. Most of the error of our LRFs comes from incomplete overlap of the two views.

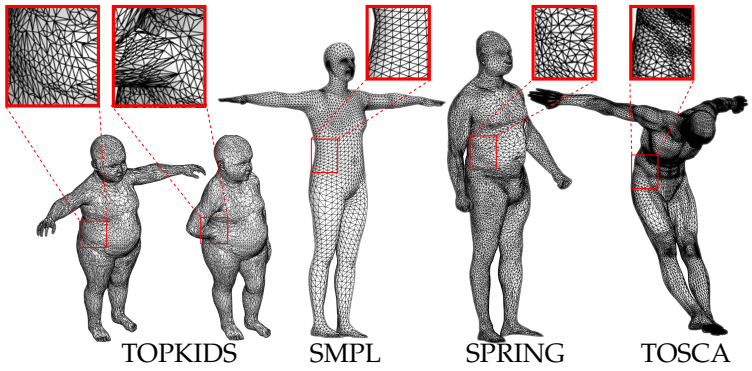


Figure 10.12: Representative data used in the deformable matching tests. TOPKIDS exhibit topological gluing at self-contacts (arm touching the body). Shapes from SMPL, SPRING, and TOSCA are used in *cross-dataset* matching experiments; the zoom-ins highlight the difference in mesh density and connectivity.

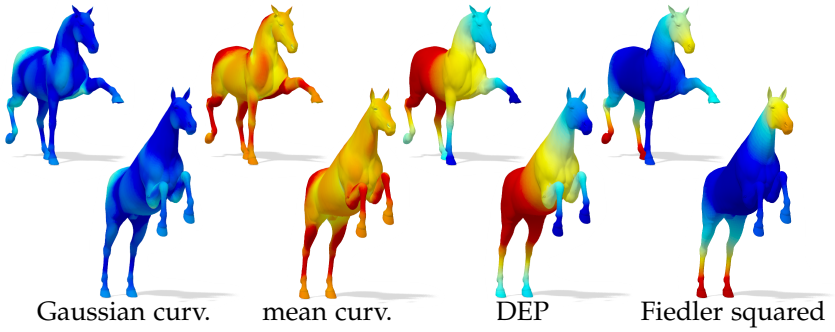


Figure 10.13: The four scalar functions used in the deformable setting. Their gradient has few singular points, which do not strongly affect the quality of the resulting LRF.

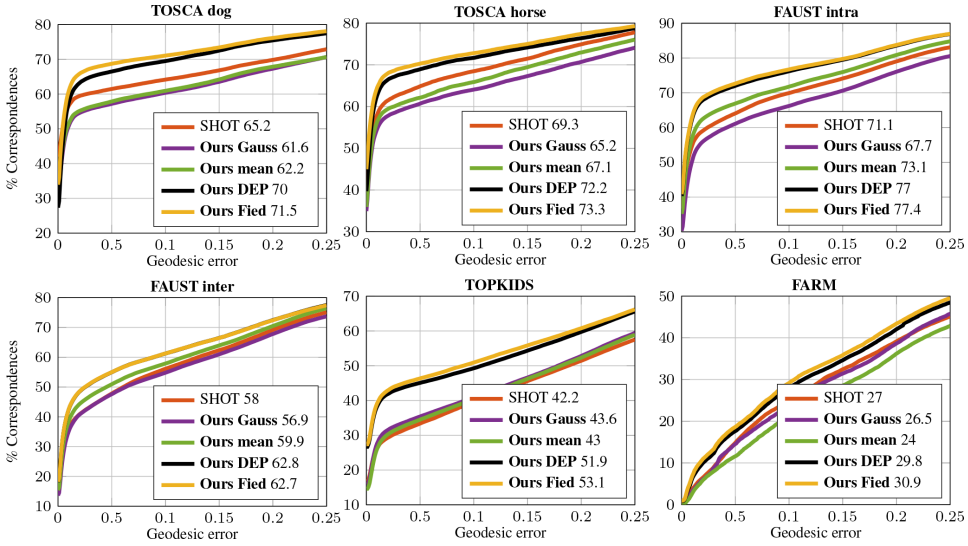


Figure 10.14: Error rates for deformable matching on different datasets. The y-axis represents the percentage of matches for which the geodesic distance from the ground truth is less than the value reported on the x-axis. The numbers in the legend denote the AUC.

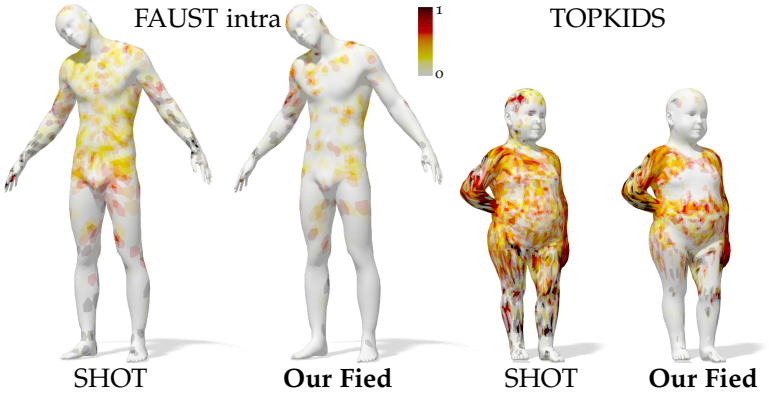


Figure 10.15: Qualitative comparisons on a standard (left) and challengin (right) case. Pointwise matching error is encoded as a heatmap, growing from white to dark red.

11

LEARNING TO ORIENT SURFACES BY SELF-SUPERVISED SPHERICAL CNNS

Humans naturally develop the ability to mentally portray and reason about objects in what we perceive as their *neutral*, *canonical* pose, and this ability is key for correctly recognizing and manipulating objects as well as reasoning about the environment. Indeed, mental rotation abilities have been extensively studied and linked with motor and spatial visualization abilities since the 70s in the experimental psychology literature [4, 6, 108].

In chapter Chap. 10, we investigated a possible solution to achieve pose invariance for surface matching, because, similar to humans, many robotic and computer vision systems require neutralizing pose variations when processing 3D data and images in many important applications such as robotic grasping and manipulation, scene understanding for augmented reality, obstacle avoidance and path planning for driver-less cars, to mention a few. In these domains, two main approaches have been pursued so to define pose-invariant methods to process 3D data: pose-invariant operators and canonical pose estimation. Pioneering work applying deep learning to point clouds, such as PointNet [148, 149], achieved invariance by sampling the range of all possible poses at training time through data augmentation. This approach, however, does not generalize to poses not seen during training. Hence, invariant operators like rotation-invariant convolutions were introduced, allowing to train on a reduced set of poses (ideally one, the unmodified data) and test on the full spectrum of rotations [111, 163, 187, 196, 201, 202]. Canonical pose estimation, instead, follows more closely the human path to invariance and exploits the geometry of the surface to estimate an intrinsic 3D reference frame which rotates with the surface. Transforming the input data by the inverse of the 3D orientation of such reference frame brings the surface in a pose-neutral, canonical coordinate system wherein pose invariant processing and reasoning can happen. While humans have a preference for a canonical pose matching one of the usual poses in which they encounter an object in everyday life, in machines this paradigm does not need to favour any actual reference pose over others: as

illustrated in Fig. 11.1, an arbitrary one is fine as long as it can be repeatably estimated from the input data.

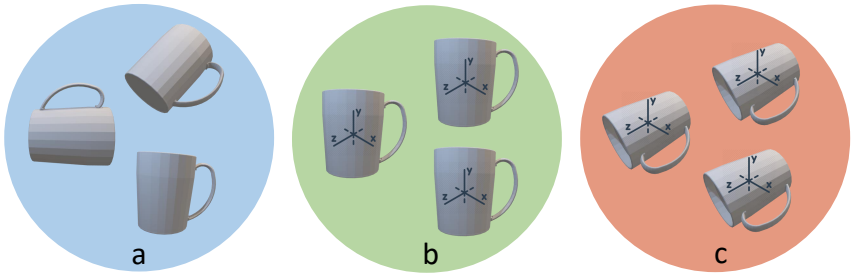


Figure 11.1: Canonical poses in humans and machines. Randomly rotated mugs are depicted in (a). To achieve rotation-invariant processing, *e.g.* to check if they are the same mug, humans mentally neutralize pose variations preferring an *upright* canonical pose, as illustrated in (b). A machine may instead use any canonical reference pose, even unnatural to humans, *e.g.* like in (c).

Despite mental rotation tasks being solved by a set of unconscious abilities that humans learn through experience, and the huge successes achieved by deep neural networks in addressing analogous unconscious tasks in vision and robotics, the problem of estimating a canonical pose is still solved solely by the handcrafted proposals, that we have thoroughly discussed in Chap. 10 [63, 68, 83, 98, 157, 191]. This may be due to convnets, the standard architectures for vision applications, reliance on the convolution operator in Euclidean domains, which possesses only the property of *equivariance* to translations of the input signal. However, the essential property of a canonical pose estimation algorithm is equivariance with respect to 3D rotations because, upon a 3D rotation, the 3D reference frame which establishes the canonical pose of an object should undergo the same rotation as the object. We also point out that, although, in principle, estimation of a canonical reference frame is suitable to pursue pose neutralization for whole shapes, in past literature it has been intensively studied mainly to achieve rotation-invariant description of local surface patches. In this chapter, we explore the feasibility of using deep neural networks to learn to pursue rotation-invariance by estimating the canonical pose of a 3D surface, be it either a whole shape or a local patch. Purposely, we leverage our insights in defining the concept of self-orienting descriptor, Sec. 7.2.4, and rely on Spherical CNNs which possess the property of equivariance w.r.t. 3D rotations by design to build *Compass*, a self-supervised methodology that learns to orient 3D shapes. As the proposed method computes feature maps living in $SO(3)$, *i.e.* feature map coordinates define 3D rotations, and does so by rotation-equivariant

operators, any salient element in a feature map, *e.g.* its argmax, may readily be used to bring the input point cloud into a canonical reference frame. However, due to non-linear activations and discretization artifacts, Spherical CNNs turn out to be not perfectly rotation-equivariant [160]. Moreover, the input data may be noisy and, in case of 2.5D views sensed from 3D scenes, affected by self-occlusions and missing parts. To overcome these issues, we propose a robust end-to-end training pipeline which mimics sensor nuisances by data augmentation and allows the calculation of gradients with respect to feature maps coordinates. In the experimental results, we adopt dataset with rigid-transformation and compare the repeatability of the LRFs computed by Compass, against several challenging baseline considering as benchmarks many of the datasets presented until now such as: 3DMatch, ETH and Stanford Views. Furthermore, we showcase that Compass can be adopted to solve the task of rotation-invariant global shape classification.

11.1 RELATED WORK

The main state-of-the-art methods for LRF have been covered in Sec. 9.1. Compass differs sharply from these methods because it learns the cues necessary to canonically orient a surface without making a priori assumptions on which details of the underlying geometry may be effective to define a repeatable canonical pose. Thus, we consider some of the learned approaches adopted to achieve invariance to rotation, especially in the context of rotation-invariant shape classification. PointNets [148, 149] employ a transformation network to predict an affine rigid motion to apply to the input point clouds in order to correctly classify global shapes under rigid transformations. In [163], Esteves *et al.* prove the limited generalization of PointNet to unseen rotations and define the Spherical convolutions to learn an invariant embedding for mesh classification. In parallel, Cohen *et al.* [160] use Spherical correlation to map Spherical inputs to $SO(3)$ features then processed with a series of convolutions on $SO(3)$. Similarly, PRIN [187] proposes a network based on Spherical correlations to operate on spherically voxelized point clouds. SFCNN [196] re-defines the convolution operator on a discretized sphere approximated by a regular icosahedral lattice. Differently, in [202], Zhang *et al.* adopt low-level rotation invariant geometric features (angles and distances) to design a convolution operator for point cloud processing. Deviating from this line of work on invariant convolutions

and operators, we show how rotation-invariant processing can be effectively realized by preliminary transforming the shape to a canonical pose learned by our method.

11.2 LEARNING TO ORIENT FROM SPHERICAL SIGNALS

In this section, we recap some concepts about the Spherical CNNs, followed by a detailed description of our method.

11.2.1 *Background*

All the background concepts about Spherical CNNs are exposed in Sec. 7.2.1. For more details, we point readers to [160, 163].

11.2.2 *Compass*

Our problem can be formalized as follows. Given the set of 3D point clouds, $\mathcal{P}$, and two point clouds $\mathcal{V}, \mathcal{T} \in \mathcal{P}$, with $\mathcal{V} = \{p_{\mathcal{V}_i} \in \mathbb{R}^3 \mid p_{\mathcal{V}_i} = (x, y, z)^T\}$ and $\mathcal{T} = \{p_{\mathcal{T}_i} \in \mathbb{R}^3 \mid p_{\mathcal{T}_i} = (x, y, z)^T\}$, we indicate by $\mathcal{T} = R\mathcal{V}$ the application of the 3D rotation matrix $R \in SO(3)$ to all the points of $\mathcal{V}$. We then aim at learning a function, $g : \mathcal{P} \to SO(3)$, such that:

$$\mathcal{V}_c = g^{-1}(\mathcal{V}) \cdot \mathcal{V} \tag{11.1}$$

$$g(\mathcal{T}) = R \cdot g(\mathcal{V}). \tag{11.2}$$

We define the rotated cloud, $\mathcal{V}_c$, in (11.1) to be the canonical, pose-neutral version of $\mathcal{V}$, *i.e.* the function g outputs the inverse of the 3D rotation matrix that brings the points in $\mathcal{V}$ into their canonical reference frame. (11.2) states the equivariance property of g: if the input cloud is rotated, the output of the function should undergo the same rotation. As a result, two rotated versions of the same cloud are brought into the same canonical reference frame by (11.1). Due to the equivariance property of Spherical CNNs layers, upon a rotation of the input signal each feature map does rotate accordingly. This means that one could just track any distinctive feature map value to establish a canonical orientation satisfying (11.1) and (11.2). Indeed, defining as Φ the composition of S^2 and $SO(3)$ correlation layers in our network, if the last layer produces the feature map $[\Phi(f_{\mathcal{V}})]$ when processing

the spherical signal $f_{\mathcal{V}}$ for the cloud $\mathcal{V}$, the same network will compute the feature map $[L_R\Phi(f_{\mathcal{V}})] = [\Phi(L_R f_{\mathcal{V}})] = [\Phi(f_{\mathcal{T}})]$ when processing the rotated cloud $\mathcal{T} = R\mathcal{V}$, with spherical signal $f_{\mathcal{T}} = L_R f_{\mathcal{V}}$. Hence, if for instance we select the maximum value of the feature map as the distinctive value to track, and the location of the maximum is at $Q_{\mathcal{V}}^{max} \in SO(3)$ in $\Phi(f_{\mathcal{V}})$, the maximum will be found at $Q_{\mathcal{T}}^{max} = RQ_{\mathcal{V}}^{max}$ in the rotated feature map. Then, by letting $g(\mathcal{V}) = Q_{\mathcal{V}}^{max}$, we get $g(\mathcal{T}) = RQ_{\mathcal{V}}^{max}$, which satisfies (11.1) and (11.2). Therefore, we realize function g by a Spherical CNN and we utilize the argmax operator on the feature map computed by the last correlation layer to define its output. In principle, equivariance alone would guarantee to satisfy (11.1) and (11.2). Unfortunately, while for continuous functions the network is exactly equivariant, this does not hold for its discretized version, mainly due to feature map rotation, which is exact only for bandlimited functions [160]. Moreover, equivariance to rotations does not hold for altered versions of the same cloud, *e.g.* when a part of it is occluded due to viewpoint changes. We tackle these issues using a self-supervised loss computed on the extracted rotations when aligning a pair of point clouds to guide the learning, and an ad-hoc augmentation to increase the robustness to occlusions. Through the use of a soft-argmax layer, we can back-propagate the loss gradient from the estimated rotations to the positions of the maxima we extract from the feature maps and to the filters, which overall lets the network learn a robust g function. Unlike in Sec. 7.2.4, where we averaged all the neighboring rotations near the maximum value of the feature map, here we provide an end-to-end learned pipeline to tackle the problem.

From point clouds to spherical signals: Spherical CNNs require spherical signals as input. We adopt the same approach we used when learning a descriptor from spherical signals, Sec. 7.2.2. We transform point coordinates from the input Euclidean reference system into a spherical one and then constructing a quantization grid within this new coordinate system [187]. The i-th cell of the grid is indexed by three spherical coordinates $(\alpha[i], \beta[i], d[i]) \in S^2 \times \mathbb{R}$ where $\alpha[i]$ and $\beta[i]$ represent the azimuth and inclination angles of its center and $d[i]$ is the radial distance from the center. Then, the K cells along the radial dimension with constant azimuth α and inclination β are seen as channels of a K-valued signal at location (α, β) onto the unit sphere S^2. The resulting K-valued spherical signal $f : S^2 \rightarrow \mathbb{R}^K$ measures the density of the points within each cell $(\alpha[i], \beta[i])$ at distance $d[i]$.

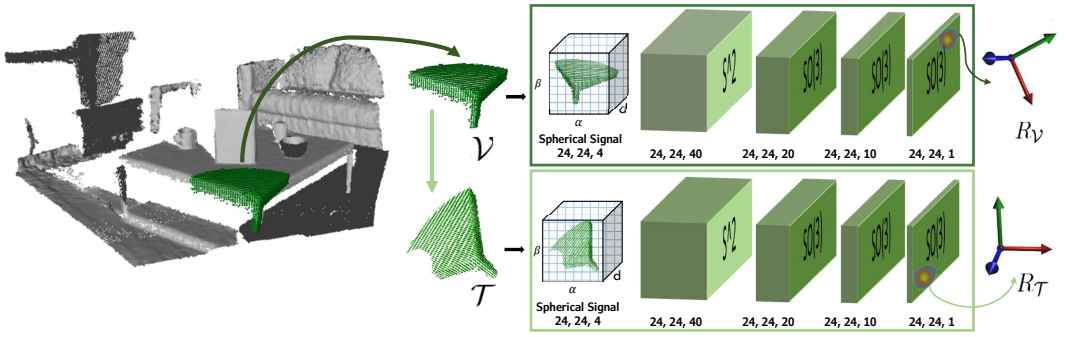


Figure 11.2: Training pipeline. We illustrate the pipeline for local patches, but the same apply for point clouds representing full shapes. During training we apply the network on a randomly extracted 3D patch, $\mathcal{V}$, and on its augmented version, $\mathcal{T}$, in order to extract the aligning rotation $R_{\mathcal{V}}$ and $R_{\mathcal{T}}$, respectively. At test time only one branch is involved. The numbers below the spherical signal indicate the number of cells along α, β and d, while the triplets under the layers indicate input bandwidth, output bandwidth and number of channels.

Training pipeline: An illustration of the Compass training pipeline is shown in Fig. 11.2. During training, our objective is to strengthen the equivariance property of the Spherical CNN, such that the locations selected on the feature maps by the argmax function vary consistently between rotated versions of the same point cloud. To this end, we train our network with two streams in a Siamese fashion [30]. In particular, given $\mathcal{V}, \mathcal{T} \in \mathcal{P}$, with $\mathcal{T} = R\mathcal{V}$ and R a known random rotation matrix, the first branch of the network computes the aligning rotation matrix for $\mathcal{V}$, $R_{\mathcal{V}} = g^{-1}(\mathcal{V})$, while the second branch the aligning rotation matrix for $\mathcal{T}$, $R_{\mathcal{T}} = g^{-1}(\mathcal{T})$. Should the feature maps on which the two maxima are extracted be perfectly equivariant, it would follow that $R_{\mathcal{T}} = RR_{\mathcal{V}} = R^{\star}_{\mathcal{T}}$. For that reason, the degree of misalignment of the maxima locations can be assessed by comparing the actual rotation matrix predicted by the second branch, $R_{\mathcal{T}}$, to the ideal rotation matrix that should be predicted, $R^{\star}_{\mathcal{T}}$. We can thus cast our learning objective as the minimization of a loss measuring the distance between these two rotations. A natural geodesic metric on the $SO(3)$ manifold is given by the angular distance between two rotations [84]. Indeed, any element in $SO(3)$ can be parametrized as a rotation angle around an axis. The angular distance between two rotations parametrized as rotation matrices R and S is defined as the angle that parametrizes the rotation SR^{T} and corresponds to the length

along the shortest path from R to S on the $SO(3)$ manifold [43, 84, 145, 203]. Thus, our loss is given by the angular distance between $R_{\mathcal{T}}$ and $R^{\star}_{\mathcal{T}}$:

$$\mathcal{L}(R_{\mathcal{T}}, R^{\star}_{\mathcal{T}}) := \cos^{-1}\left(\frac{(\mathrm{tr}(R_{\mathcal{T}}{}^{\mathsf{T}} R^{\star}_{\mathcal{T}}) - 1)}{2}\right). \tag{11.3}$$

Soft-argmax: The result of the argmax operation on a discrete $SO(3)$ feature map returns the location i, j, k along the α, β, γ dimensions corresponding to the ZYZ Euler angles, where the maximum correlation value occurs. To optimize the loss in (11.3), the gradients w.r.t. the i, j, k locations of the feature map where the maxima are detected have to be computed. To render the argmax x operation differentiable we add a soft-argmax operator [51, 167] following the last $SO(3)$ layer of the network. Let us denote as $\Phi(f_{\mathcal{V}})$ the last $SO(3)$ feature map computed by the network for a given input point cloud $\mathcal{V}$. A straightforward implementation of a soft-argmax layer to get the coordinates $C_R = (i, j, k)$ of the maximum in $\Phi(f_{\mathcal{V}})$ is given by

$$C_R(\mathcal{V}) = \text{soft-argmax}(\tau\Phi(f_{\mathcal{V}})) = \sum_{i,j,k} \text{softmax}(\tau\Phi(f_{\mathcal{V}}))_{i,j,k}(i, j, k), \tag{11.4}$$

where softmax$(\cdot)$ is a 3D spatial softmax. The parameter τ controls the temperature of the resulting probability map and (i, j, k) iterate over the $SO(3)$ coordinates. A soft-argmax operator computes the location $C_R = (i, j, k)$ as a weighted sum of all the coordinates (i, j, k) where the weights are given by a softmax of a $SO(3)$ map Φ. Experimentally, this proved not effective. As a more robust solution, we scale the output of the softmax according to the distance of each (i, j, k) bin from the feature map argmax. To let the bins near the argmax contribute more in the final result, we smooth the distances by a Parzen function [3] yielding a maximum value in the bin corresponding to the argmax and decreasing monotonically to 0 .

Learning to handle occlusions: In real-world settings, rotation of an object or scene (*i.e.* a viewpoint change) naturally produces occlusions to the viewer. Recalling that the second branch of the network operates on $\mathcal{T}$, a randomly rotated version of $\mathcal{V}$, it is possible to improve

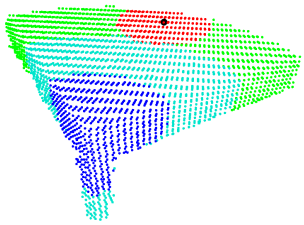


Figure 11.3: Local support of a keypoint depicting the corner of a table, divided in 3 shells. Randomly selected point in black; removed points in red.

robustness of the network to real-world occlusions and missing parts by augmenting $\mathcal{T}$. A simple way to handle this problem is to randomly select a point from $\mathcal{T}$ and delete some of its surrounding points. In our implementation, this augmentation happens with an assigned probability. $\mathcal{T}$ is divided in concentric spherical shells, with the probability for the random point to be selected in a shell increasing with its distance from the center of $\mathcal{T}$. Additionally, the number of removed points around the selected point is a bounded random percentage of the total points in the cloud. An example can be seen in Fig. 11.3.

Network Architecture: The network architecture comprises 1 S^2 layer followed by 3 $SO(3)$ layers, with bandwidth $B = 24$ and the respective number of output channels are set to 40, 20, 10, 1. The input spherical signal is computed with $K = 4$ channels.

11.3 EXPERIMENTAL RESULTS

We evaluate Compass on two challenging tasks. The first one is the estimation of LRFs for local surface patches, whilst in the second task, the canonical pose provided by our method is instead used to perform highly effective rotation-invariant shape classification by leveraging a simple PointNet classifier.

11.3.1 *LRF repeatability*

Datasets

We conduct experiments on three heterogeneous datasets: 3DMatch Sec. 7.3.1, ETH Sec. 7.4.1 and Stanford Views Sec. 10.4.1.

Problem formulation

We follow a similar protocol to the one we used in Sec. 10.4.1 to compute the repeatability of the local reference frame estimated at corresponding keypoints in different views of the same scene. Unlike in GFrames, where the main novelty was the proposal of a stable $\hat{\mathbf{x}}$ axis direction, with Compass we produce a set of three orthogonal unit vectors, thus we need to consider $\hat{\mathbf{z}}$ in the ThCos metric as well. All the datasets provide several 2.5D scans, *i.e.* fragments, representing the same *model*, *i.e.* an object or a scene depending on the dataset, acquired from different viewpoints. All N fragments belonging to a test model can be grouped into pairs, where each pair $(\mathcal{F}_s, \mathcal{F}_t)$, $\mathcal{F}_s = \{p_{s_i} \in \mathbb{R}^3\}$ and $\mathcal{F}_t = \{p_{t_i} \in \mathbb{R}^3\}$, has an area of overlap. A set of correspondences, $\mathcal{C}_{s,t}$, can be computed for each pair $(\mathcal{F}_s, \mathcal{F}_t)$ by applying the known rigid ground-truth transformation, $G_{t,s} = \left[R_{t,s}|t_{t,s}\right] \in SE(3)$, which aligns $\mathcal{F}_t$ to $\mathcal{F}_s$ into a common reference frame. $\mathcal{C}_{s,t}$ is obtained by uniformly sampling points in the overlapping area between $\mathcal{F}_s$ and $\mathcal{F}_t$. Finally, the percentage of repeatable LRFs, $Rep_{s,t}$, for $(\mathcal{F}_s, \mathcal{F}_t)$, can be calculated as follows:

$$Rep_{s,t} = \frac{1}{|\mathcal{C}_{s,t}|} \sum_{k=1}^{|\mathcal{C}_{s,t}|} I\Big(\big(\hat{\mathbf{x}}(p_{s_k}) \cdot R_{t,s}\hat{\mathbf{x}}(p_{t_k}) \geqslant \rho\big) \wedge \big(\hat{\mathbf{z}}(p_{s_k}) \cdot R_{t,s}\hat{\mathbf{z}}(p_{t_k}) \geqslant \rho\big) \Big), \tag{11.5}$$

where $I(\cdot)$ is an indicator function, $(\cdot)$ denotes the dot product between two vectors, and ρ is a threshold on the angle between the corresponding axes, 0.97 radians in our experiments. *Rep* measures the percentage of reference frames which are aligned, *i.e.* differ only by a small angle along all axes, between the two views. The final value of *Rep* for a given model is computed by averaging on all the pairs.

Test-time adaptation

Due to the self-supervised nature of Compass, it is possible to use the test set to train the network without incurring in data snooping, since there is no external ground-truth information involved. This test-time training can be carried out very quickly, right before the test, to adapt the network to unseen data and increase its performance, especially in transfer learning scenarios. This is common practice with self-supervised approaches [204].

Table 11.1: LRF repeatability on the 3DMatch dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)				
	SHOT [58]	FLARE [75]	TOLDI [157]	3DSN [191]	Compass
Kitchen	0.189	**0.330**	0.171	0.181	0.315
Home 1	0.251	0.354	0.243	0.236	**0.397**
Home 2	0.226	0.339	0.213	0.214	**0.365**
Hotel 1	0.194	**0.385**	0.213	0.216	0.370
Hotel 2	0.193	**0.405**	0.223	0.226	0.393
Hotel 3	0.240	0.407	0.261	0.276	**0.446**
Study	0.186	0.351	0.195	0.192	**0.356**
Lab	0.220	0.310	0.198	0.223	**0.361**
Mean	0.212	0.360	0.215	0.220	**0.375**

Experimental setup

We train Compass on 3DMatch following the standard procedure of the benchmark, with 48 scenes for training and 6 for validation. From each point cloud, we uniformly pick a keypoint every 10 cm, the points within 30 cm are used as local surface patch and fed to the network. Once trained, the network is tested on the test split of 3DMatch. The network learned on 3DMatch is tested also on ETH and Stanford Views, using different radii to account for the different sizes of the models in these datasets: respectively 100 cm and 1.5 cm. We also apply test-time adaptation on ETH and Stanford Views: the test set is used for a quick 2-epoch training with a 20% validation split, right before being used to assess the performance of the network. We use Adam [96] as optimizer, with 0.001 as the learning rate when training on 3DMatch and for test-time adaptation on Stanford Views, and 0.0005 for adaptation on ETH. We compare our method against the same baseline we used for GFrames in Sec. 10.4.1 plus two recent and established LRFs proposals: TOLDI [157] and a variant of TOLDI recently proposed in [191] that here we refer to as 3DSN. Unfortunately, our current implementation of GFrames could not process the large point clouds of 3DMatch and ETH due to memory limits, and we can show results for GFrames only on Stanford Views.

Quantitative results

Tab. 11.1 reports *Rep* on the 3DMatch test set. Compass outperforms the most competitive baseline FLARE, with larger gains over the other baselines.

Table 11.2: LRF repeatability on the ETH dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)					
	SHOT [58]	FLARE [75]	TOLDI [157]	3DSN [191]	Compass	Compass (adapted)
Gazebo Summer	0.293	**0.345**	0.241	0.241	0.337	0.330
Gazebo Winter	0.266	0.268	0.170	0.196	0.292	**0.303**
Wood Autumn	0.253	0.210	0.157	0.174	0.288	**0.307**
Wood Summer	0.279	0.236	0.171	0.198	0.314	**0.329**
Mean	0.273	0.264	0.185	0.202	0.308	**0.317**

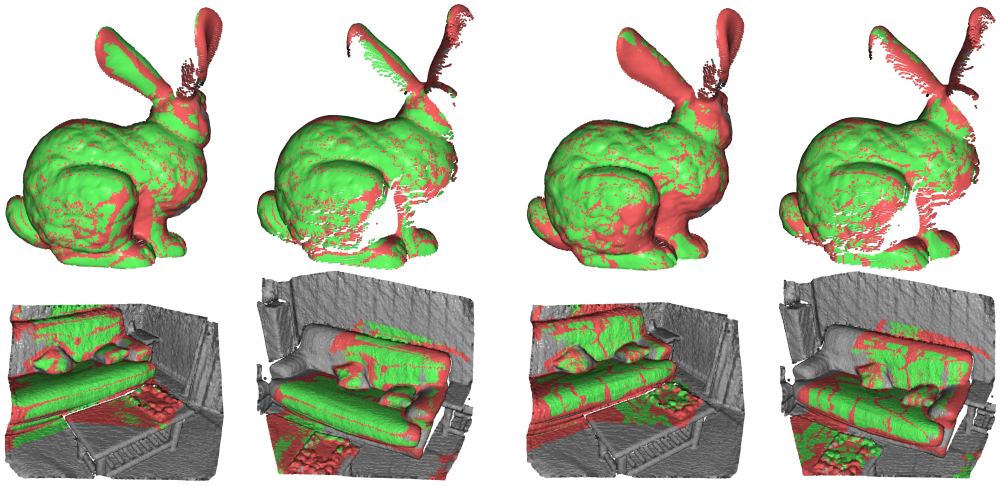
Table 11.3: LRF repeatability on the Stanford Views dataset. Best result for each row in bold.

	LRF Repeatability (*Rep* ↑)						
	SHOT [58]	FLARE [75]	TOLDI [157]	3DSN [191]	GFrames	Compass	Compass (adapted)
Armadillo	0.127	0.185	0.156	0.141	0.168	0.340	**0.359**
Buddha	0.134	0.194	0.202	0.192	0.181	0.312	**0.344**
Bunny	0.106	0.379	0.232	0.172	0.426	0.440	**0.463**
Dragon	0.161	0.207	0.201	0.188	0.251	0.352	**0.384**
Mean	0.132	0.241	0.197	0.173	0.256	0.361	**0.388**

Results reported in Tab. 11.2 for ETH and in Tab. 11.3 for Stanford Views confirm the advantage of a data-driven model like Compass over hand-crafted proposals: while the relative rank of the baselines changes according to which of the assumptions behind their design fits better the traits of the dataset under test, with SHOT taking the lead on ETH and our previous proposal GFrames on Stanford Views, Compass consistently outperforms them. Remarkably, this already happens when using pure transfer learning for Compass, *i.e.* the network trained on 3DMatch: in spite of the large differences in acquisition modalities and shapes of the models between training and test time, Compass has learned a robust and general notion of canonical pose for a local patch. This is also confirmed by the slight improvement achieved with test-time augmentation, which however sets the new state of the art on these datasets.

Qualitative results

We provide qualitative results to show the effectiveness of Compass at computing the canonical pose for local surface patches. Given a pair of fragments, we visualize in both fragments at each point the accuracy of the estimated LRF using two different metrics. In particular, in Fig. 11.4 we show the repeatability of the estimated LRFs, in Fig. 11.5 the angular distance between two rotations used as loss to train out network. In both figures, we visualize the results yielded by Compass alongside FLARE, which offers high performance across all datasets and is the second best on 3DMatch.



(a) Compass (b) FLARE

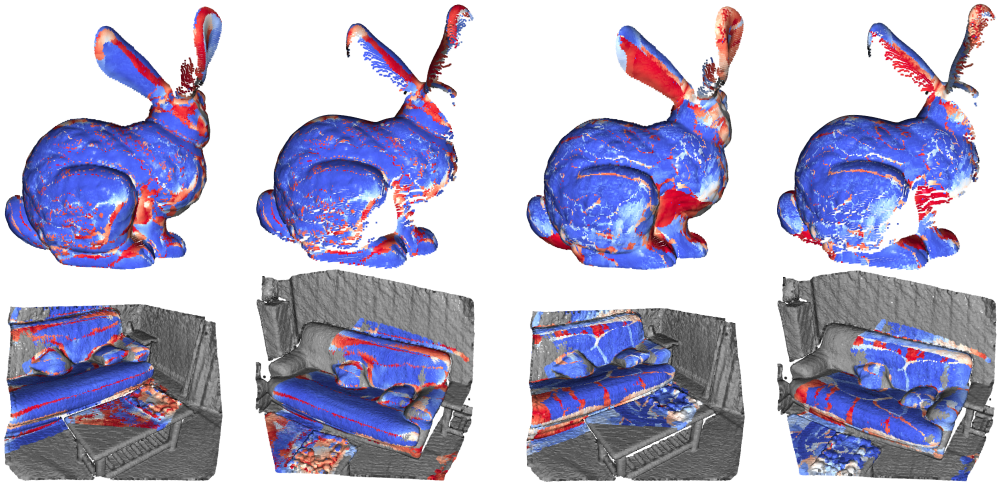
Figure 11.4: Visualization of repeatability at corresponding points of two fragments, with repeatable LRFs in green, non-repeatable ones in red and non-overlapping areas in gray. First row: a pair of fragments from Stanford Views, second row: a pair of fragments from 3DMatch. (a) and (b): results yielded by Compass and FLARE, respectively.

We can observe how Compass tends to yield larger areas in which the LRFs are accurately estimated, *i.e.* either green or blue ones, depending on the considered metric. It is worth pointing out how this is particularly evident across those challenging fragment areas affected by large missing parts in one of the two views, like, *e.g.* the left ear of the Bunny in the fragments taken from the Stanford Views dataset.

11.3.2 *Rotation-invariant Shape Classification*

Datasets

We test our model on the ModelNet40 [117] shape classification benchmark. This dataset has 12,311 CAD models from 40 man-made object categories, split into 9,843 for training and 2,468 for testing. In our trials, we actually use the point clouds sampled from the original CAD models provided by the authors of PointNet. We also performed a qualitative evaluation of the transfer learning performance of Compass by orienting clouds from the ShapeNet [104] dataset.



(a) Compass (b) FLARE

Figure 11.5: Visualization of the angular error between the LRFs estimated at corresponding points of two fragments, with lower errors in blue, higher errors in red and non-overlapping areas in gray. First row: a pair of fragments from Stanford Views. Second row: a pair of fragments from 3DMatch. (a) and (b): results yielded by Compass and FLARE, respectively.

Problem formulation

Object classification is a central task in computer vision applications, and the main nuisance that methods processing 3D point clouds have to withstand is rotation. To show the general applicability of our proposal and further assess its performance, we wrap Compass in a shape classification pipeline. Hence, in this experiment, Compass is used to orient full shapes rather than local patches. To stress the importance of correct pose neutralization, as shape classifier we rely on a simple PointNet [148], and Compass is employed at train and test time to canonically orient shapes before sending them through the network.

Experimental setup

We train Compass on ModelNet40 using 8,192 samples for training and 1,648 for validation. Once Compass is trained, we train PointNet following the settings in [148], disabling t-nets, and rotating the input point clouds to reach the canonical pose learned by Compass. We followed the protocol described in [187] to assess rotation-invariance of the selected methods: we do not augment the dataset with rotated versions of the input cloud when training PointNet; we then test it with the original test clouds, *i.e.* in the

canonical pose provided by the dataset, and by arbitrary rotating them. We use Adam [96] as optimizer, with 0.001 as the learning rate.

Table 11.4: Classification accuracy on the ModelNet40 dataset when training with no rotation augmentation. NR column reports the accuracy attained when testing on the cloud in the canonical pose provided by the dataset and AR column when testing under arbitrary rotations. Best result for each column in bold.

Classification Accuracy (*Acc.* %)		
Method	NR	AR
PointNet [148]	88.45	12.47
PointNet++ [149]	89.82	21.35
Point2Sequence [193]	92.60	10.53
Kd-Network [144]	86.20	8.49
Spherical CNN [160]	81.73	55.62
DeepSets [158]	88.73	9.72
LDGCNN [201]	92.91	17.82
SO-Net [173]	**94.44**	9.64
PRIN [187]	80.13	70.35
Compass + PointNet	80.51	**72.20**

Quantitative results

Results are reported in Tab. 11.4. Results for all the baselines come from [187]. PointNet fails when trained without augmenting the training data with random rotations and tested with shapes under arbitrary rotations. Similarly, in these conditions most of the state-of-the-art methods cannot generalize to unseen rotations. If, however, we first neutralize the pose by Compass and then we run PointNet, it gains almost 60 points and achieves 72.20 accuracy, outperforming the state-of-the-art on the arbitrarily rotated test set. This shows the feasibility and the effectiveness of pursuing rotation-invariant processing by canonical pose estimation. In Fig. 11.6, we present some models from ModelNet40, randomly rotated and then oriented by Compass. The models estimate a very consistent canonical pose for each object class, despite the large shape variations within the classes.

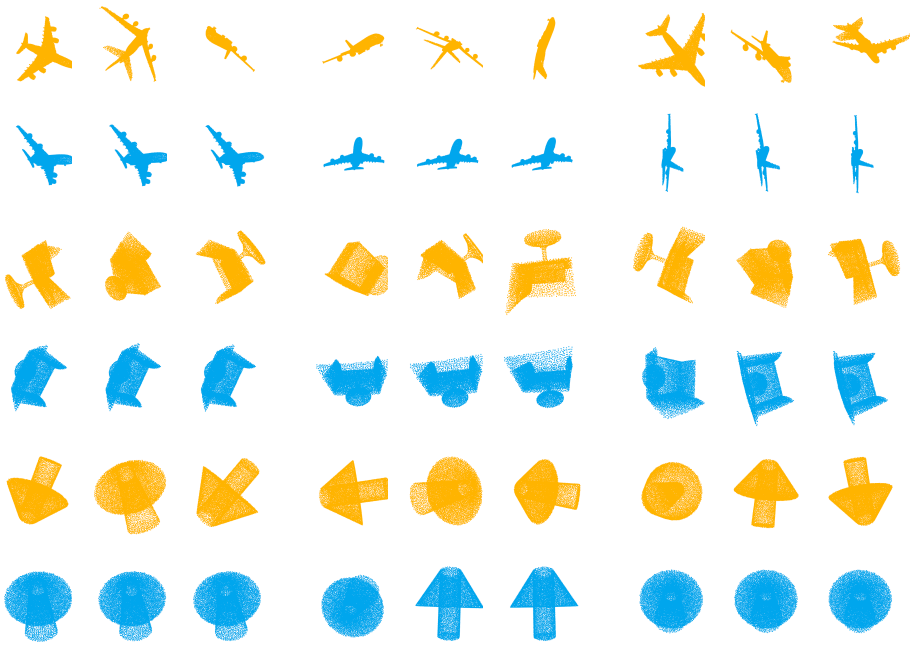
Finally, to assess the generalization abilities of Compass for full shapes as well, we performed qualitative transfer learning tests on the ShapeNet dataset, reported in Fig. 11.6. Even if there are different classes, the model trained on ModelNet40 is able to generalize to an unseen dataset and recovers similar canonical poses for the same object.

(a) ModelNet40 (b) ShapeNet

Figure 11.6: Qualitative results on ModelNet40 and ShapeNet in transfer learning. Top row: randomly rotated input cloud. Bottom row: cloud oriented by Compass.

Qualitative results

We add more qualitative results on the ShapeNet dataset, which complement those reported in Fig. 11.6. We stress the generalization capability of our model by adopting three different configurations to generate the training data. For this experiment, we consider only three categories: airplane, chair and lamp. The results of this study are shown in Fig. 11.7. In the first column, (a), we present results for a *category-specific* training, *i.e.* learning to orient only one category. Thus, we train one network for each category and then we test on the test split of the same category. In (b), we present results for a category-agnostic network, *i.e.* a single model trained on samples from the three categories. Finally, in (c) we show the orientation results of transferring to ShapeNet a model trained on the ModelNet40 dataset. From these results, we observe how the canonical pose can be often correctly recovered under random rotations, and how for each triplet of rotated objects (colored in yellow) the estimated canonical pose (in blue) is consistent, even in a transfer learning strategy. Interestingly, looking at the fourth and sixth row of the (b) and (c) cases, where the model has to define a canonical orientation for more than one category at once, the canonical pose learned by the network seems to be similar across the chair and lamp categories, which have as the first principal direction the direction of gravity. This suggests that our network may generalize the concept of canonical pose across objects of different categories that share a similar geometric structure.



(a) Category-specific training (b) Category-agnostic training (c) Transfer learning

Figure 11.7: Qualitative results on ShapeNet dataset under different training strategies. Clouds in yellow represent randomly rotated input clouds and the blue ones represent those oriented by Compass. In (a), we present orientation results after training Compass with examples belonging only to a specific category from ShapeNet; in (b), the orientation results after training Compass with a training set comprising *airplanes*, *chairs* and *lamps* together; and, in (c) the orientation results from the model trained on the ModelNet40 dataset and tested on the ShapeNet dataset.

12

CONCLUSIONS

In this part, we have proposed two different procedures to establish a robust local reference frame. In Chap. 10, we introduced a LRF suitable for rigid and non rigid 3D shape matching applicable to meshes and point clouds. The latest study on LRF repeatability was carried out in [75] and to get a more comprehensive overview of the problem, in Chap. 10, we have shown the limits of the de-facto standard LRF, SHOT, on datasets with rigid and non rigid transformations in the context of shape matching. As main novel contribution, we designed the tangent component of the LRF as the intrinsic gradient of a scalar function defined on the surface; different designs are possible depending on the task at hand, as we showcased in the experimental results session Sec. 10.4. On the one hand, the flexibility of our approach lies in the freedom of choosing a scalar function on top of which a stable LRF, and in turn repeatable descriptors, can be constructed. On the other hand, the main limitation is to be found in the requirement for the chosen function to have limited high frequency content, which may lead to unstable gradients; this excludes, for instance, the adoption of highly detailed texture or oscillatory functions obtained, *e.g.*, by wave propagation. To overcome the problematic design choice of picking the most suitable scalar function for a given task, we envision the adoption of our LRF construction in deep learning pipelines, where the scalar function itself may be learned in an end-to-end fashion.

In Chap. 11, we tackled the same problem but by a data-driven solution. Thus, we realize the first end-to-end learned pipeline that defines and recovers a canonical orientation for 3D surfaces. Unlike our proposal in Chap. 10, where we relied on intrinsic properties of the shape, in Chap. 11, instead, we leveraged the equivariant property of Spherical CNNs and, avoiding explicit supervision, we let the network define the best-suited canonical pose for the underlying geometry of the surfaces. Another important contribution of the work carried out in Chap. 11, is definition of an *ad-hoc* data augmentation that can be adopted to robustly handle occlusions in the real world 3D vision scenarios. Without having to be tied to local surface properties, *e.g.* surface normals, we were able to expand the range of experimental setup and, we

have thus demonstrated the ability of Compass to successfully orient global shapes in the task of rotation-invariant shape classification. Moreover, the effectiveness of our learned approach has allowed us to set the new state-of-the-art performance on the challenging Stanford Views dataset, where we outperformed our previous proposal, *e.g.* GFrames. For the future, we would like to investigate the possibility to deploy a unified Spherical CNN architecture acting as both descriptor, *e.g.* Chap. 7, and LRF estimator, *e.g.* Chap. 11, for a deep learning feature matching pipeline.

We hope that the research in the field of canonical orientation accomplished in this part will raise the interest and stimulate further studies about this topic.

Part IV

LEARNING TO RECONSTRUCT 3D OBJECTS

13

INITIAL REMARKS

This dissertation is about lifting the knowledge on the 3D geometry present in our world from humans to machines. A robot needs more than 2D visual perception to complete a simple task such as opening a jar of peanut butter. The 3D structure of the jar has to be recognized and possibly segmented in its functional parts to accomplish the task. So far, we have dealt with 3D object reconstruction, surface registration and shape classification, which assume as input 3D data such as point clouds and meshes. However, an intelligent system may be equipped with only a single camera and acquire 2D visual information as RGB images. Moreover, as technology became more compact and affordable, the popularity of smartphones in the world grew exponentially facilitating the spread of high-resolution devices at very low cost. As a consequence, in our mobile phones there is a plethora of applications based on complex computer vision algorithms. Based on these considerations, in this last part, we will tackle a different problem, where 3D data will become the outcome of our algorithm, instead of the input, as in the previous parts. In this regard, we will propose a method to perform 3D object reconstruction given as input a set of non-overlapping images of an object. The goal of 3D object reconstruction from RGB data is to recover the 3D structure of an object from one or multiple 2D images. This is a long standing ill-posed problem and providing an effective solution can have a key role for applications being developed, or just envisioned, in the field of virtual and augmented reality, robot manipulation and grasping, rapid prototyping/reverse engineering and autonomous driving. Existing works in the field can be categorized into two main categories: *single-view 3D object reconstruction* and *multi-view 3D object reconstruction*. While the latter is a well-known traditional computer vision problem taking into account overlapping views around the same object, the former has only been recently addressed successfully thanks to the availability of large 3D CAD model repositories [86, 100, 104, 180] and to the introduction of deep learning based methods. Multi-view 3D object reconstruction has been addressed in the past by a long line of geometry-based techniques such as *structure from motion* (SfM) [147], *multi-view/photometric stereo* (MVS) [25], *simultaneous localization and*

mapping (SLAM) [105] and *shape-from-X methods*, *i.e. shape-from-silhouette*, or *shape-by-space-carving* [12].

Multi-view methods try to recover the lost information about the depth in 2D images by leveraging the insight that seeing the same point of an object or a scene from different perspectives let us understand its underlying geometry. Thus, effective solutions typically require a large set of images acquired with a high-precision system made up by perfectly-calibrated cameras or large overlap areas between the images. Although these methods resulted to high quality 3D reconstructions, require operative conditions that are not always feasible.

The recent breakthroughs powered by deep learning, have led to a new generation of methods that are able to estimate the 3D shape of an object from a single or multiple RGB images [192]. The intuition behind this line of works leverages the prior knowledge about geometric structures of an object to guess the invisible part of it, similarly to as how we humans do. Indeed, recent proposals rely on a neural network to reconstruct the entire shape of an object given a single image [136, 164, 179, 183, 194]. These methods use the expressive power of neural networks to guess the occluded part of objects and reconstruct the full object in a canonical pose. Due to the viewpoint invariant reconstruction, the structures visible from different input images are often discarded. As a result, the output shapes suffer from a lack of realistic details. Even though the shape predicted are visually compelling as examined in [184], different regions can correspond to the occluded part of the object, hence the real 3D shape cannot be determined given only a single-view. Recent works have investigated if what is performed is shape reconstruction or shape retrieval [197].

The work that we are going to present in Chap. 14, tries to create a bridge between classic computer vision techniques and the more recent deep learning methods. Starting from sparse views of the object, we first align them into a common coordinate system by estimating the relative pose between all the pairs. Then, inspired by the traditional voxel carving, we generate an occupancy grid of the object taken from the silhouette on the images and their relative poses. Finally, we refine the initial reconstruction to build a clean 3D model which preserves the details from each viewpoint.

13.1 RELATED WORK

In this section we provide a brief overview of the main methods aimed at inferring the 3D structure of objects from one or multiple RGB image, we considering mainly deep learning approaches.

13.1.1 *Single-view 3D object reconstruction*

Single-view object reconstruction methods hallucinate the invisible parts of a shape by implicitly memorizing the statistical distribution of the training data. A central role for the quality of the estimated shape is played by the representation chosen to encode it. One of the more common choice is to produce a shape as 3D voxel volume [109, 121, 123, 135, 136, 140, 152, 156]. While other 3D parametrization are indeed possible: octrees [150, 151, 154], point clouds [138, 168, 174, 175], mesh [164, 170, 183, 188], depth images [131, 137, 179], classification boundaries [189, 194] and signed distance function [195]. Generation approaches were also explored for single-view reconstruction including geometric primitive deformation [132, 164, 169, 172, 183], combination of *Generative Adversarial Networks* (GANs) [93] and *Variational Autoencoders* (VAEs) [85] in [135] and re-projection consistency [136, 152, 171, 182]. However, all the above mentioned works are based on the assumption that invisible part of an object can be safely hallucinated given the shape priors learned from training data. Unfortunately, the occluded part of an object can not have a deterministic shape, as examined in [184, 198]. As a result, the inferred shapes tend to suffer from over smoothed surfaces without fine details. Single-view reconstruction methods can achieve good results, especially when used on the ShapeNet dataset due to the high presence of symmetric shapes.

13.1.2 *Multi-view 3D object reconstruction*

Extracting 3D geometry from multiple views is a well researched topic in computer vision. Methods based on *multi-view stereo* (MVS) [25], *structure from motion* (SfM) [7, 29, 34, 147] and *SLAM* [105], require to match features among images. Unfortunately, the matching process becomes difficult when the viewpoints are separated by wide baselines and the images are poorly textured. The dawn of deep learning has fostered several methods for multi-

view 3D reconstruction. Multi-view geometry cues can be exploited during the training of single-view prediction systems as a supervisory signal [129, 136, 168, 174, 182], or during training and inference of multi-view systems to enrich the learned representation by capturing view-dependent details [121, 142, 198, 199]. Kar *et al*. in [142] learn a machine for multi-view stereopsis [21]. In [199], a coarse shape generated as in [183] is refined iteratively by moving each vertex to the best location according to the features pooled from multiple views. Unfortunately, both methods require exact groundtruth camera poses. A different way of avoiding the need for camera poses, is to reconstruct the object from each view in a canonical reference system and let the net learn view-dependent shape priors. Choy *et al*. in [121] propose a unified framework to create a voxelized 3D reconstruction from a sequence of images, or just a single image, captured from uncalibrated viewpoints. As soon as more views are provided to the network, the reconstruction is incrementally refined by means of *Recurrent Neural Networks* (RNNs). However, due to the permutation-variant nature of RNNs, the produced results might be inconsistent, and, in addition, are time-consuming. An alternative way, explored in [153], is to pool only maximum values. To overcome these limitations in [200] Xie *et al*. introduce a learned context-aware fusion module that acts on coarse 3D volumes generated on each input images in parallel. In [198], Wei *et al*. propose to learn a generative model conditioned by multiple random input vectors. With different random inputs, they can predict multiple plausible shapes from each view to overcome the ambiguity of the invisible parts. The final model is obtained by taking the intersection of the predicted shapes on each single-view image.

14

A DIVIDE ET IMPERA APPROACH FOR 3D OBJECT RECONSTRUCTION FROM NON-OVERLAPPING VIEWS

The knowledge of the three-dimensional structure of objects in our surroundings is a paramount information for perception systems to accomplish tasks such as object interaction and modeling. Although humans can guess the 3D shape of an object of a known category from a single glance, this is a quite complicated task for a machine that requires the ability to estimate the pose of an object and to reconstruct its 3D shape. In this chapter, we are going to introduce our solution to infer the 3D geometry of an object from multiple RGB images. Multi-view 3D object reconstruction such as *i.e.* SfM, MVS and SLAM are successful in handling many scenarios, but they all rely on feature matching between images that can be difficult when multiple viewpoints are separated by large baselines and the amount of textured regions in the images is low. Recent deep learning based methods have been proposed to overcome these limitations [131, 137, 142, 198–200]. Notably, most of them rely on the unrealistic assumption of employing ground-truth camera poses while inferring shapes [131, 137, 142, 199] or predicting models only oriented in a canonical reference frame [190, 200]. Indeed, the CAD models available in the public datasets to operate in the field, such as ShapeNet [103] and ModelNet40 [116] share the same canonical reference frame so that the up direction is aligned with the the gravity, *i.e.* all the chairs are upright.

A less explored task in literature is the reconstruction of the 3D shape of an object observed from multiple non-overlapping views [121, 198, 200]. This task has several important practical applications since it relaxes the need of acquiring images around an object as a temporal sequence or according to a predefined set of viewpoints. In practice, this can simplify the acquisition skills of robots or humans when reconstructing the 3D model of an object. Moreover, the presence of several, sometimes non-overlapping images sparsely taken from an object is a common situation in scenarios such as e-commerce, where products are advertised with a few images and large database already exist.

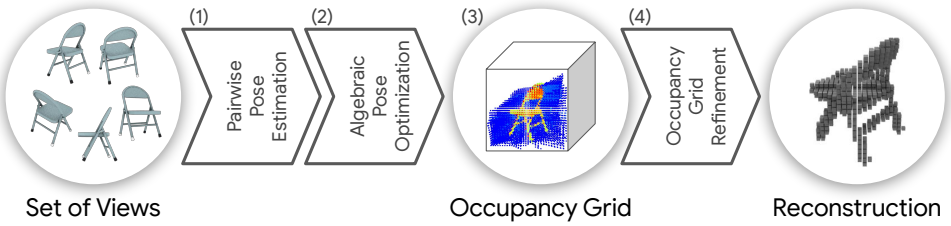


Figure 14.1: Given a set of views of the object, our framework (1) estimates the relative poses between pairs of images; (2) algebraically optimize the pose of each image; (3) build an occupancy grid from poses and silhouettes; and, (4) refine the occupancy grid to reconstruct the full model.

A reason why this direction has not been explored much in literature is that, as mentioned, the majority of traditional multi-view approaches require overlap among view pairs in order to, *e.g.* match keypoints or apply photometric constraints. At the same time, for single-view completion methods to be applied for this task, they would require the estimation of the 6D pose of each viewpoint, which is currently missing. Hence, our proposal in this chapter addresses the problem of multi-view object reconstruction with unknown camera poses and in arbitrary reference systems. We relax the assumption of previous methods that views must be overlapping and propose a two-stage learned pipeline for shape generation using color images captured from a limited number of views. A schematic overview of our reconstruction pipeline is shown in Fig. 14.1. In the first stage, we focus on estimating the pose of each view. To do so, we construct a set of all possible pairwise views and estimate the relative poses between them through a CNN-based pose estimation. Then, we rearrange the pairs in a fully-connected graph and convert it to a overdetermined system that can be further refined by an algebraic pose optimization stage. In the second step, we rely on the estimated poses to build a first rough approximation of the 3D object reconstruction by projecting the object silhouettes from every views to a shared occupancy grid. Finally, we refine the initial occupancy map with a 3D CNN to produce the final voxelized model estimation. The clear separation between the pose estimation and the identity estimation by means of 3D object reconstruction motivates the name PoseIDoNet. To break up the complexity of multi-view object reconstruction, we adopt a well-established strategy in computer science and develop small independent pieces that individually solve simpler problems. We rely both on deep learning models, but also on geometrical constraints to effectively solve different steps of a full 3D reconstruction pipeline and propose a method for multi-view 3D

object reconstruction from non-overlapping views without the need of poses or without relying on a canonical orientation of the model. We show a comprehensive experimental session dealing with relative pose estimation between pairs of images and the 3D object reconstruction on the reference benchmark ShapeNet.

14.1 RELATED WORK

Considering the classification of state-of-the-art methods for 3D object reconstruction made in Sec. 13.1, our work can be included in the multi-view category. However, contrary to these methods, we propose a viewpoint-variant modelling in order to enrich the learned representation grabbing the visible geometry from each current viewpoint which is challenging if the camera poses are unknown. Moreover, while several works targeting multi view object reconstruction from sparse views exist, importantly, our method can deal with both overlapping and non overlapping images and as such represents a first step towards this scenario. Before we go any further, we briefly expose methods aimed at recovering both the shape and the pose of an object.

14.1.1 *Shape and pose recovering*

Researches have also proposed methods for simultaneous 3D shape reconstruction and camera pose estimation from single-view [168, 180, 182]. Tulsiani *et al*. [182] regress multiple poses hypothesis for each sample, forcing the learnt distribution to be similar to a prior distribution to deals with the ambiguity of unsupervised pose estimation. In contrast Insafutdinov *et al*. [168] train an ensemble of pose regressors and use the best model as a teacher for knowledge distillation to a single student model. Finally, in [180] the pose estimation is treated as a classification problem by discretizing the azimuth and elevation angles.

14.2 POSEIDONET

Given a set of N images acquired each from a different viewpoint around the same object $\mathcal{I} = \{I_i\}_{i=1}^{N}$, the objective is to reconstruct the 3D shape of the object. To do so, we need to estimate the pose of each view so to orient the

images with respect to each other. The method is composed of two main tasks: relative poses estimation and shape reconstruction. We solve this two tasks with three independent components: (1) the relative pose estimation, that takes every permutation of image pairs in $\mathcal{I}$ by means of a task-specific neural architecture (see Sec. 14.2.1); (2) the graph-based rectification of the bidirectional relative poses, aimed at refining the previously computed set of pairwise poses, as well as to calculate the absolute poses with respect to a reference image (see Sec. 14.2.2); and, (3) the reconstruction pipeline, that starts from building an occupancy grid from the image silhouettes and poses then refines it through a 3D CNN (see Sec. 14.2.3). Influenced by *classical* 3D reconstruction frameworks, we predict relative poses instead of the absolute ones to avoid aligning the 3D CAD models within an object class into a single canonical coordinate system. In addition, we can easily deal with different object classes without requiring a common reference alignment.

14.2.1 *Pose estimation*

From a set of images of an object, recovering its 3D geometry requires the images to be correlated either through their relative or absolute pose. The absolute poses have the disadvantage of requiring a canonical reference coordinate for every object. The canonical system is usually handpicked, which imposes a strong constraint to the system that might not be optimal for many use cases. On the other hand, predicting the relative poses between all pairs of views removes this constraint. Due to this, the first stage of the pipeline relies on the *relative pose estimation*. We first arrange the set of input images in a fully connected graph by establishing the links of all the possible permutations of the image pairs. Unlike classical 3D reconstruction, our approach does not require overlapping regions between two images. Thus, we can effectively leverage all the permutations. After connecting all the pairs, we estimate the relative pose transformation considering one image as source I_s and the other as target I_t. In general, the 6D camera pose can be decomposed into a 3D rotation $\mathbf{R}$ and a 3D translation $\mathbf{t}$. In our scenario, we consider that the distance between the camera and the object is fixed and we teach a neural network to regress only the rotation. Nevertheless, the same method can be easily extended to regress also a translation. We rely on unit quaternions to parameterize the regressed rotation and denote it as $\tilde{q} \in \mathbb{R}^4$. Therefore, we aim at learning a function $Pose : (I_s, I_t) \rightarrow \mathbb{R}^4$.

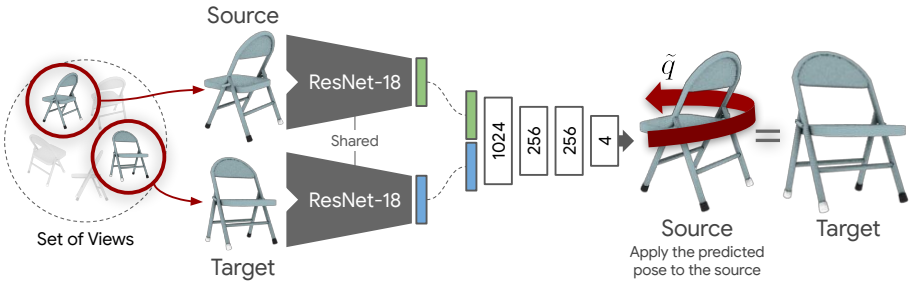


Figure 14.2: Schematic representation of our pose estimation architecture. A ResNet-18 based encoder extracts a view-point invariant image descriptor on both the source and target image. These two embeddings are then concatenated and examined by a fully-connected based network that will outputs the relative rotation between the input images.

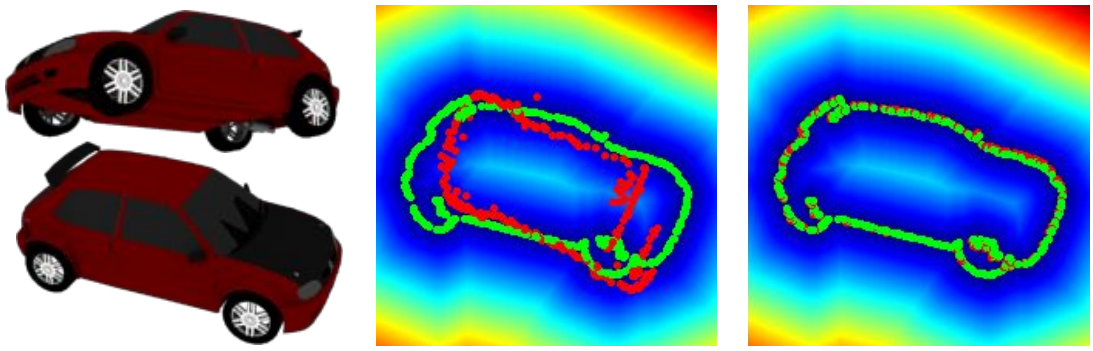
Particularly, we approximate this function as a deep neural network $\mathcal{N}$ that takes the source and target images (I_s and I_t, respectively) and regress a unit quaternion as $\tilde{q} = \mathcal{N}(I_s, I_t)$. A schematic representation of the architecture is depicted in Fig. 14.2 where we employ a siamese ResNet-18 to extract a 1024-dimensional deep embedding from each view independently. Then, we concatenate the two representations and elaborate the resulting 2048-dimensional vector with a stack of fully connected layers to finally regress the unit quaternion at the last layer.

Contour loss

We generate the training set for the network starting from the 3D model of the object. A pair of images is created by rendering the model according to the camera projection matrix π from random views with a fixed t^*, producing the ground truth relative pose q^* from I_s to I_t. We adapt the contour loss from [176] to supervise the regression of the pose of the network. Here, to find the contours, we project the point cloud to the target image and sample a sparse set of points on the image boundaries. The set of 3D points on the contours is denoted as $\mathcal{V}_t := \{v \in \mathbb{R}^3\}$. Using the contours on the target image, we build the distance transform $\mathcal{D}_t$. Consequently, with the predicted pose $\tilde{q}$, we define the loss as:

$$\mathcal{L}_{\text{contours}}(\tilde{q}, \mathcal{D}_t, \mathcal{V}_t) := \sum_{v \in \mathcal{V}_t} \mathcal{D}_t \left[\pi \left(\tilde{q} q^{*-1} (v - t^*) q^* \tilde{q}^{-1} + t^* \right) \right] \tag{14.1}$$

with q^{-1} being the conjugate quaternion. The function measures the contours alignment after transforming the points from the target to source using the ground truth and from the source to the target using the prediction.



(a) Input Views (b) $\mathcal{L}_{\text{contours}}$ only (c) $\mathcal{L}_{\text{contours}} + \mathcal{L}_{\text{angular}}$

Figure 14.3: A failure case of the contour loss, we show on the left (a) a couple of input views and on the right (b-c) the ground truth contour (in green) and the contour oriented according to the pose estimated by the network (in red) drawn over the corresponding distance transform. The network trained only with $\mathcal{L}_{\text{contours}}$ (b) outputs a completely wrong pose due to ambiguity in the distance transform. Adding $\mathcal{L}_{\text{angular}}$ (c) solves the ambiguity and align the contours well.

Although one can argue that we can simply take the contour points on the source in order to avoid transforming back and forth, we need to consider that there are instances when the source and the target have a large relative pose. In this case, the contour from the source and the distance transform from the target do not match.

Angular loss

Despite being very effective, we found out that learning a relative pose by aligning the object contours can easily fall into local minima for objects with an ellipse-like structure, *e.g.* cars. Unlike [176] that uses the contour loss for small pose changes between two images, *i.e.* between 4° and 45°, the relative pose estimation in our work requires to handle large pose differences. An example of such local minima is depicted in Fig. 14.3 with a rotation of almost 180°. The contour loss then have problems in distinguishing the front and back side of the car by just looking at the contours.

Taking advantage of the ground truth pose q^*, we can directly supervised the network with the angular difference between rotations thus establishing:

$$\mathcal{L}_{\text{angular}}(q^*, \tilde{q}) := 1 - \text{Re}\left(\frac{q^* \tilde{q}^{-1}}{\|q^* \tilde{q}^{-1}\|}\right) \tag{14.2}$$

where Re denotes the real part of the quaternion.

Complete loss

The loss to train the network is a weighted combination of Eq. 14.1 and Eq. 14.2

$$\mathcal{L}_{\text{pose}} = \alpha \cdot \mathcal{L}_{\text{angular}} + \beta \cdot \mathcal{L}_{\text{contours}} \tag{14.3}$$

Later in Tab. 14.2, we show the relative contribution of the two components to our pose estimation.

14.2.2 *Pose optimization*

Based on the predictions from Sec. 14.2.1, we can build a fully-connected graph that connects all the images through the relative poses. For every pair of images, we have a bi-directional link since we can predict the pose with one of them as the source while the other as target and vice versa. However, errors introduced from the prediction hinders us from directly using these poses for reconstruction. Even when investigating a pair of images, there is no guarantee that the pose of one direction is the inverse of the other.

To solve this problem, we propose to optimize the entire graph and imposing one relative pose per pair. Taking the inspiration from the bundle adjustment [19] to fix the poses using least-squares, we optimize the algebraic solution of:

$$\arg\min_{\hat{q}_i} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \left\| \mathbf{R}(\hat{q}_j) \cdot \mathbf{R}(\hat{q}_i)^{-1} - \mathbf{R}(\tilde{q}_{i,j}) \right\|^2 \tag{14.4}$$

to find $\hat{q}_i$ which is the absolute pose of a the i-th view. Here, $\tilde{q}_{i,j}$ is the prediction where the i-th image is the source and the j-th as the target. We also convert the quaternions to rotation matrix through the function $\mathbf{R}(\cdot)$ and assume that $\hat{q}_1$ is the identity rotation that signifies the reference coordinate system of the absolute poses. Geometrically, we can interpret the difference in rotation matrices in Eq. 14.4 as the distance between the vectors pointing towards x-, y- and z-axis of the two rotations.

(a) Silhouette (b) Visual Cone (c) Rotated Visual Cone

Figure 14.4: Ray casting for a single view. We show (a) the silhouette and (b-c) two visualization of the same visual cones – (b) one oriented according to the camera view point and (c) the other according to a different viewpoint.

14.2.3 *Identity Reconstruction from an Occupancy grid*

At this point, we have the set of images $\mathcal{I} = \{I_i\}_{i=1}^N$ as input and the corresponding set of absolute poses $\mathcal{Q} = \{\hat{q}_i\}_{i=1}^N$ from the optimization in Sec. 14.2.2 with $\hat{q}_1$ as the identity rotation. Now, we have all the data to reconstruct the object.

Similar to visual hull or voxel carving [12], we want to exploit the idea of shape-from-silhouette, wherein, given a 3D grid, the voxels outside the silhouette of at least one image is carved out of the grid, ending up with the shape of object. However, using this method is not feasible for two reasons. First, these methods usually require a large number of views of the object to achieve a good reconstruction. For instance, the problem of the reconstruction from a single view is illustrated in Fig. 14.4 where the effects of ray casting become visible when the grid is rotated. The shape of the object becomes more detailed only when we increase the number of views. In our case, we must be able to also handle a handful of input images. The second is that the errors introduced from the pose estimation generate small misalignment among the images, removing more voxels that are on the object. Note that one of the applications of visual hull is a multi-camera studio where all the cameras are professionally calibrated.

Instead of the hard thresholding in visual hull, we propose to utilize an occupancy grid to serve as the input to a network which then refines the initial reconstruction. We denote the set of silhouette as $\mathcal{S} = \{S_i\}_{i=1}^N$, where the pixel values are either 0 if it is outside the object and 1 if it is on the

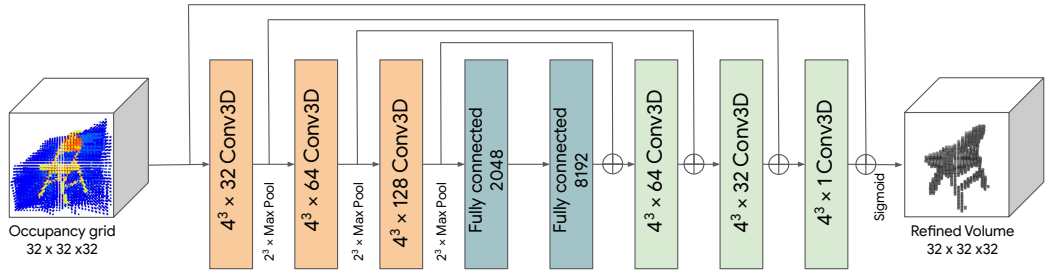


Figure 14.5: Architecture of the occupancy grid refiner network that predicts the final model.

object. Therefore, we compute the values of the voxels in the occupancy grid are calculated as the weighted average:

$$V(x) = \frac{\sum_{i=1}^{N} w_i \cdot S_i \left(\pi\left(\hat{q} \cdot x \cdot \hat{q}^{-1} + t^*\right)\right)}{\sum_{i=1}^{N} w_i} \tag{14.5}$$

where x is the centroid of the specified voxel and w_i is the weight assigned to the i-th view. Every voxel in the grid ranges from 0 to 1 where 0 corresponds to a voxel that is not visible from any view and 1 corresponds to a voxel that fall within the silhouette of all views. In our implementation, we assume that the sum of all weights is one and $w_2 = w_3 = \cdots = w_N$ while only w_1 changes. The motivation behind this assumption is to give more importance to the first view, the reference one, since the output of the model needs to be aligned to this one. Therefore, w_1 is generally weighted higher than the other weights. The dimension of the occupancy grid is $32 \times 32 \times 32$.

This coarse representation of the occupancy grid can be refined with the help of deep learning. Inspired by [200], we use the 3D CNN sketched in Fig. 14.5 to refine the raw occupancy map and predict the reconstruction. To generate the ground truth voxels, we reorient the 3D model with respect to the reference view and then discretize it to a grid of 32^3 voxels. The value of the voxels in the ground truth occupancy map is either 0 (free) or 1 (full), which is denoted as p_i^*. This is differentiated from the predicted occupancy map $\tilde{p}_i$.

The network is trained to minimize the mean value of the voxel-wise binary cross entropies between the predicted occupancy map and the reoriented ground truth model

$$\mathcal{L}_{\text{refiner}} = \frac{1}{n_v} \sum_{i=1}^{n_v} [p_i^* \log(\tilde{p}_i) + (1 - p_i^*) \log(1 - \tilde{p}_i)] \tag{14.6}$$

where $n_v = 32^3$ is the number of voxels. At test time, the occupancy map predicted by the network can be binarized with a simple 0.3 threshold to obtain the final reconstructed 3D model.

14.3 EXPERIMENTAL RESULTS

In this section, we evaluate our relative pose estimation network as well as the overall pipeline in 3D object reconstruction from multiple views.
Dataset: The experiments are conducted using the 3D models from the ShapeNetCore.v1 [104] dataset the standard reference benchmark. Similar to [168, 182], we focus on the three most challenging categories: *airplanes*, *cars* and *chairs*. To render the images for each object, we use the toolkit provided by [182] following the same data generation procedure. For each model, we render five random views with random light source positions and random camera azimuth and elevation, sampled uniformly from $[0°, 360°)$ and $[-20°, 40°]$, respectively. To conduct a fair comparison, we use the same train and test split provided in [168, 182] for the pose estimation part, while the split provided by [121, 200] for the shape estimation.

14.3.1 *Relative Pose Estimation*

Problem formulation

To measure the pose error, we use the same metrics as Tulsiani *et al*. [182]: (1) the accuracy, defined as the percentage of samples for which the error between the predicted pose and the ground truth is less than 30°; and, (2) the median error. The error between two rotations represented with quaternions is calculated through the angular difference:

$$\theta(q^*, \tilde{q}) = 2 \operatorname{acos} \left(\frac{q^* \tilde{q}^{-1}}{\| q^* \tilde{q}^{-1} \|} \right), \tag{14.7}$$

where $\tilde{q}$ and q^* are the predicted and ground truth quaternion, respectively. To evaluate the accuracy of estimated poses, we compute the relative poses of every possible pairs of views given the ground truth absolute poses. Then, we directly compare the predicted relative poses with the ground truth one.

Method	*Airplane*		*Car*		*Chair*		Mean	
GT poses [182]	**0.79**	10.70	0.90	7.40	0.85	11.20	0.85	9.77
MVC [182]	0.69	14.30	0.87	5.20	0.81	7.80	0.79	9.10
DPCD [168]	0.75	8.20	0.86	5.00	**0.86**	8.10	0.82	7.10
Ours	**0.79**	6.49	0.92	3.32	0.85	7.18	**0.86**	5.66
Ours optimized	**0.79**	**6.40**	**0.93**	**3.10**	0.85	**7.00**	**0.86**	**5.50**
Ours one net all categories	0.77	6.80	0.90	3.40	0.82	6.80	0.83	5.70

Table 14.1: Quantitative evaluation for pose prediction, for each category we report on the left the accuracy and on the right the median error. The best results are highlighted in bold.

Experimental setup

The pose estimation network and refiner models are independently trained. To train the pose network, for each model we render five views, compute all the possible permutations of two views, and use each pair as a training example. The network is trained with 224×224 RGB images and batch size of 24. Then, the loss functions are weighted by $\alpha = 0.1$ and $\beta = 0.9$. We used a fixed learning rate of 0.001 and the Adam optimizer [96]. We compare our relative pose estimation against Tulsiani *et al*. (MVC [182]), as well as Insafutdinov and Dosovitskiy (DPCD [168]). Both methods regress the absolute camera pose from a collection of images of the same object using a category-specific network. We use the results reported by the authors in their respective papers. As for MVC, we also report the upper bound of the method when trained with supervision (GT poses). Notably, both methods need to align the canonical pose learned by the network to the canonical orientation of the dataset before starting the evaluation while we don't. Moreover, both methods use different solutions to handle the problem of the shape similarity when looked from different camera views. By formulating the problem as relative pose regression, our network can easily handle this situations without using a specific mechanism.

Quantitative results

In Tab. 14.1, we report both the accuracy (on the left) and the median error (on the right) per category and averaged across all categories. Ours denotes the result achieved by the raw predictions of our relative pose estimation network without any kind of optimization. Even considering just the raw

$\mathcal{L}_{\text{angular}}$	$\mathcal{L}_{\text{contours}}$	Accuracy	Median
✗	✓	0.45	78.23
✓	✗	0.84	8.24
✓	✓	**0.86**	**5.50**

Table 14.2: Ablation study on angular and contour losses. The best results are highlighted in bold.

predictions, we are already able to achieve performance better than all the considered competitors both in terms of averaged scores and per-category performances. We ascribe this result to our choice of regressing relative poses between views rather than the absolute pose of a single view with respect to an implicit reference. The advantage of regressing relative poses is particularly evident when comparing the median error where our method without optimization improves the state of the art by $-1.44°$. Applying the pose optimization described in Sec. 14.2.2 (Ours optimized), we increase the performance even more obtaining the best overall results. In the last row of Tab. 14.1, we report the performance of our pose estimation model when training a category agnostic network rather than a different model for each category and applying the pose optimization. The network needs to solve a much harder task in this case as testified by the small drop in performance (-0.03 in accuracy and $+0.2$ in the median error). However, even in this case, our proposal remains competitive in terms of accuracy or lower in median error with respect to any competitor with the advantage of having a single class-agnostic model. In Tab. 14.2, we report an ablation study on the contribution of the two loss functions described in Sec. 14.2.1 when training the relative pose estimation model. When training with $\mathcal{L}_{\text{contours}}$ only, our model unfortunately is not able to achieve satisfactory performance while training with $\mathcal{L}_{\text{angular}}$ only can already provide relatively good results. However, by mixing the two loss functions, we are able to achieve the best overall results increasing the accuracy by $+0.02$ and especially decreasing the median error by $-2.77°$.

Qualitative results

We provide in Fig. 14.6 the qualitative results to show the effectiveness of our pose estimation network. We compare the results obtained by estimating the relative pose for pairs of images taken from the objects belonging to different

categories in different poses. Given the input pairs shown in the first and second columns of Fig. 14.6, the third column in the figure compares the point clouds of the CAD model aligned according to the ground truth pose and the predicted one, in green and red, respectively. Here, the alignment between the two point clouds verifies the accuracy we achieved in Tab. 14.1. To provide a better visualization of the quality of the estimated poses, the figures in the last column encode the misalignment error computed as the per point Euclidean Distance between the point cloud rotated according to the ground truth matrix and the point cloud oriented with the predicted one. Note that the errors are normalized according to the maximum error. From these results, we can observe that our network produces good alignments even in cases where the rotation between the source and target image is large such as the first, fourth and fifth row. This is clearly evident in the point clouds in column (c) where they are almost completely overlapping and the misalignment error in column (d) which is consistently low for most points.

14.3.2 *3D object reconstruction*

Problem formulation

For the shape estimation, we employ two different metrics: (1) the Intersection Over Union (IoU) and (2) the Chamfer Distance. The former provides an evaluation of the goodness of the output volume while the latter is highly correlated to the human judgment [180]. To compute the IoU, we binarize the predicted occupancy maps using a fixed threshold th and compute:

$$IoU = \frac{\sum_{i,j,k} I(\tilde{p}_{i,j,k} > th)\ I(p^*_{i,j,k})}{\sum_{i,j,k} I\left[I(\tilde{p}_{i,j,k} > th) + I(p^*_{i,j,k})\right]} \tag{14.8}$$

where $\tilde{p}_{i,j,k}$ and $p^*_{i,j,k}$ are the occupancy probability of the predicted and ground truth voxel at (i, j, k), respectively, while $I(\cdot)$ is an indicator function. Note that higher IoU values indicate better reconstruction results. The Chamfer Distance $d_{Chamfer}$ compares two point clouds. Given a ground truth $P^* = \{\mathbf{x}^*_n\}$ and a predicted $\tilde{P} = \{\tilde{\mathbf{x}}_n\}$ point clouds, this metric is defined as:

$$d_{Chamfer}(P^*, \tilde{P}) = \frac{1}{|\tilde{P}|} \sum_{\tilde{\mathbf{x}} \in \tilde{P}} \min_{\mathbf{x} \in P^*} \|\tilde{\mathbf{x}} - \mathbf{x}^*\|_2 + \frac{1}{|P^*|} \sum_{\mathbf{x}^* \in P^*} \min_{\tilde{\mathbf{x}} \in \tilde{P}} \|\mathbf{x}^* - \tilde{\mathbf{x}}\|_2 \tag{14.9}$$

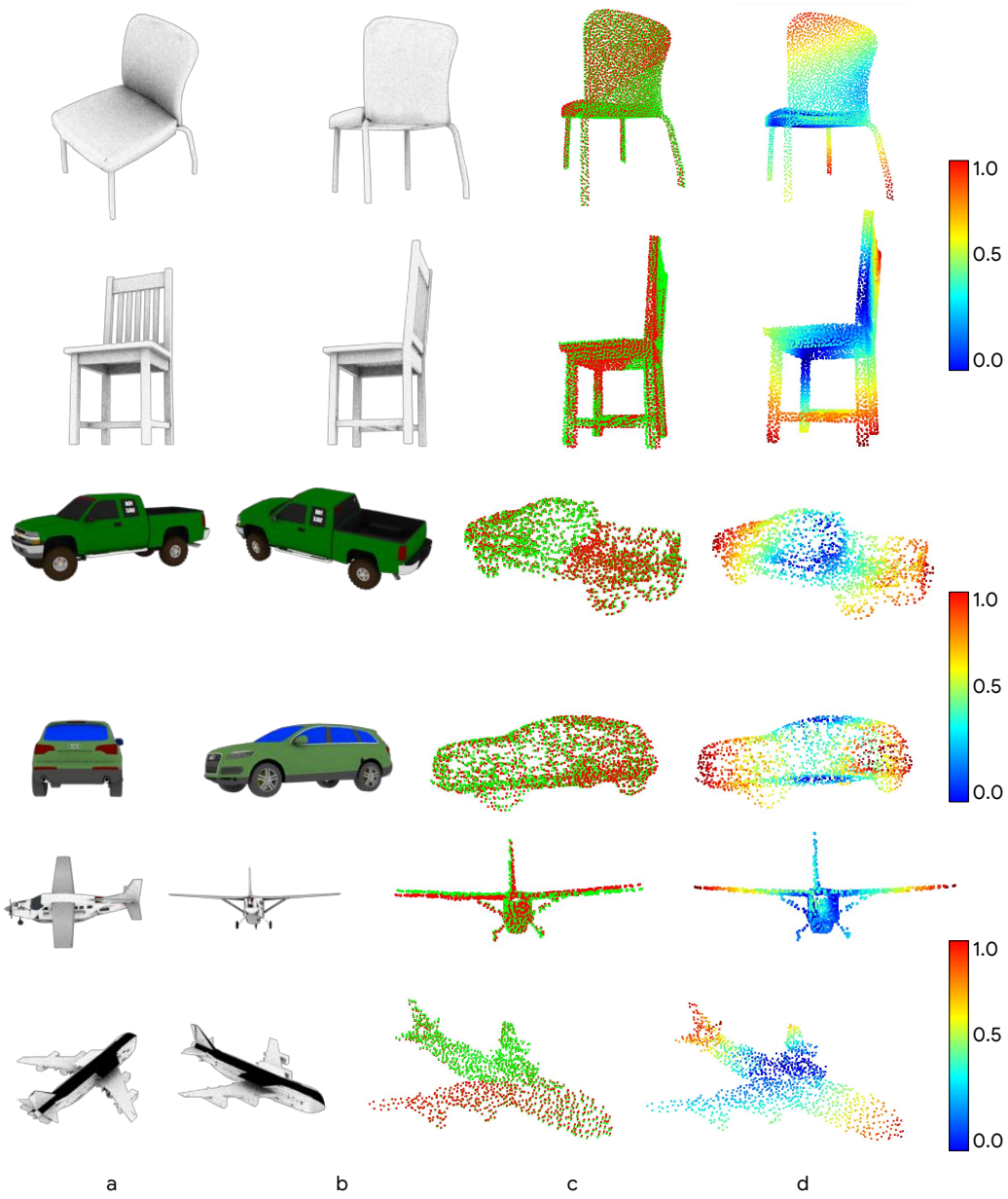


Figure 14.6: Qualitative results for the pose estimation network. We show on the left (a, b) the source and target input image, on the right (c) the CAD model point cloud, in *green*, oriented according to the ground truth pose, and the same point cloud, in *red*, oriented with the pose predicted by our method. In the last column (d), we visualize the misalignment error between the two models as a heat map ranging from blue (perfect alignment) to red (maximum misalignment).

In Eq. 14.9, the first term measures the precision of the predicted point cloud by measuring the average distance between the predicted point and the closest ground truth one while the second measures how well the predicted point cloud covers the ground truth by measuring the average distance between a ground truth point and the closest predicted one. We binarize

the output of the refiner network using a threshold $th = 0.3$. Then, for the IoU, the ground truth voxel grid is computed using the point cloud of the 3D model oriented by the ground truth pose from the canonical view to the reference view. For the Chamfer Distance computation, we convert the predicted voxel grid into a point cloud and compute the error against the reoriented ground truth cloud uniformly sampled according to [174].

Experimental setup

To train the refiner network we use the ground truth pose and silhouettes extracted from the images. We take a set of 5 views and poses per model, we randomly select one view as reference, then perturb the ground truth poses by a maximum of 10° and finally build the occupancy grid. The refiner network take as input an occupancy grid of $32 \times 32 \times 32$ and outputs a same sized grid. We train the model with batch size 16, fixed learning rate of $0,001$ and the Adam optimizer [96]. Given an unconstrained set of views our method is able to reconstruct the 3D model of an object aligned with one of the views provided as input. Unfortunately, there isn't any work in the literature that addresses the same exact settings, as structure from motion methods rely on the assumption of having overlapping views, while most 3D reconstruction methods reconstruct a model only in an arbitrary reference view. Therefore, to provide some insightful comparison, we have chosen the most similar work in the literature as competitors: multi-view reconstruction solutions that do not require overlapping views or pose as inputs, but reconstruct the model only in a canonical reference view in form of a voxel grid. Thus, we compare against Pix2Vox [200] and 3D-R2N2 [121]. We use the public available implementations kindly provided by the authors. To have a fair comparison, all methods take the same number of input images. Both the pose estimation and the refiner networks are trained using one model for all the categories.

Quantitative results

In Tab. 14.3, we compare our full pipeline against Pix2Vox and 3D-R2N2. We evaluate three variants of our method: PoseIDoNet denotes the full reconstruction pipeline, PoseIDoNet GT Poses denotes the 3D reconstruction part of our pipeline evaluated with input occupancy grid built according to ground truth poses, finally, PoseIDoNet Canonical extends the previous by taking as input and producing as output an occupancy grid oriented

Method	*Airplane*		*Car*		*Chair*		Mean	
3D-R2N2 [121]	0.585	3.77	0.851	3.58	0.575	4.21	0.670	3.86
Pix2Vox [200]	0.723	3.20	**0.876**	3.54	0.612	3.77	0.737	3.50
PoseIDoNet	0.538	4.81	0.627	3.93	0.510	4.75	0.559	4.50
PoseIDoNet GT poses	0.654	3.85	0.659	3.63	0.592	4.06	0.635	3.85
PoseIDoNet Canonical	**0.732**	**2.92**	0.874	**3.51**	**0.648**	**3.39**	**0.751**	**3.28**

Table 14.3: Quantitative evaluation for shape prediction, for each category we report the average IoU on the left and the Chamfer distance between normalized point clouds multiplied by 100. The best results are highlighted in bold.

according to the dataset canonical orientation. Comparing the performance of PoseIDoNet to the competitors we can see how for the harder task we picked we are not able to reconstruct models as accurately as the competitors that assume a canonical view. This is true also when considering the variants of our model that takes ground truth poses as input, even if the performance increases and get closer to those of the competitors we still perform slightly worse.

We argue that this slightly inferior performance are not due to an inferior 3D reconstruction pipeline but due to the task that we are trying to solve being harder than the one the competitors are addressing (*i.e.*, arbitrarily aligned reconstruction vs canonically aligned reconstruction). To verify this claim we train an additional variant of our pipeline that reconstruct the 3D model in a fixed canonical reference frame (PoseIDoNet Canonical) to have a fairer comparison with the competitors. In this settings our 3D reconstruction strategy achieves clearly better results and surpasses the current state of the art proving that our solution has the ability to achieve very detailed and refined reconstructions. Unfortunately relying on the assumption of having a canonical orientation for all the objects we wish to reconstruct works well for academic datasets, but does not scale to real world applications. For this reason we believe that our reconstructions from an arbitrary viewpoint provide a way more valuable output even if sacrificing a little accuracy in the 3D reconstruction. In Fig. 14.7, we report a visual comparison between the reconstruction obtained by different methods on the three classes of the ShapeNet test set. Our reconstructions are obtained by PoseIDoNet and correspond to a model aligned to one of the input views. To ease the visualization and the comparison with the other methods, we

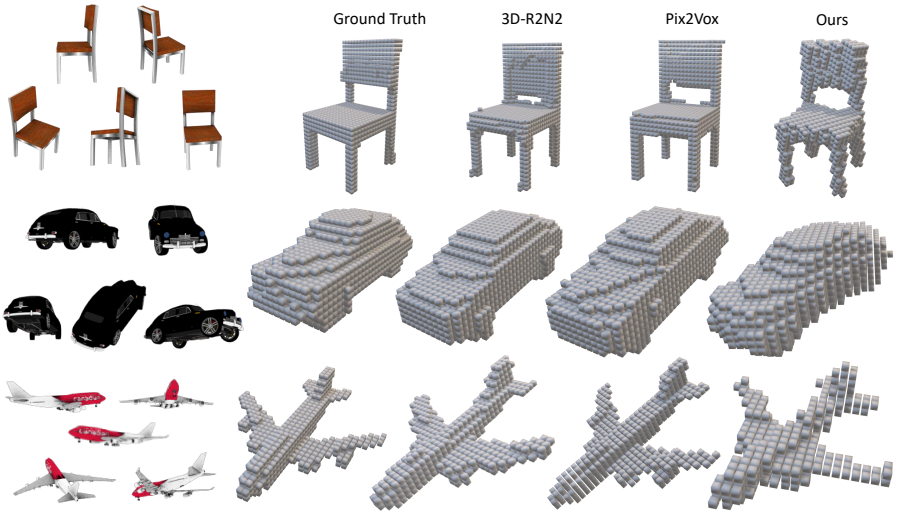


Figure 14.7: Comparison of multi-view reconstructions methods on the ShapeNet test set. On the left we show the 5 RGB views used as input for every model. Ours refers to PoseIDoNet.

manually re-aligned the model to the reference frame. Our reconstructions can better keep some of the fine details present in the views, for example the hole in the chair back or the smooth shapes of the car. However, since we are reconstructing a shape aligned to one of the view instead of using a canonical reference, we might lose some precision if straight surfaces do not align well with the voxel grid. This might result in aliasing like artifacts (*i.e.*, staircase effect in 3D) like the back of the chair and its front legs in our reconstruction.

Qualitative results

We show more qualitative results of the reconstruction obtained by our pipeline for the three categories considered: airplanes in Fig. 14.8, cars in Fig. 14.9 and chairs in Fig. 14.10. Due to the *view-dependent* reconstruction, the voxel grid obtained to supervise our model in training has a lower spatial resolution compared to the voxel grid available in the ShapeNet dataset. Indeed, to get a voxel grid w.r.t. the reference view, we first align the point cloud of the CAD model to the reference view, then we voxelize it in a $32 \times 32 \times 32$ grid. As a result, the predicted shapes have a smaller spatial resolution since they need to be aligned with the input RGB images, and therefore our refiner model outputs a shape with the same lower spatial resolution. This difference is even more clear if we look at the reconstruction in Fig. 14.11 where PoseIDoNet Canonical is trained using the voxel grid available in ShapeNet. Considering the chair reconstructed by our method

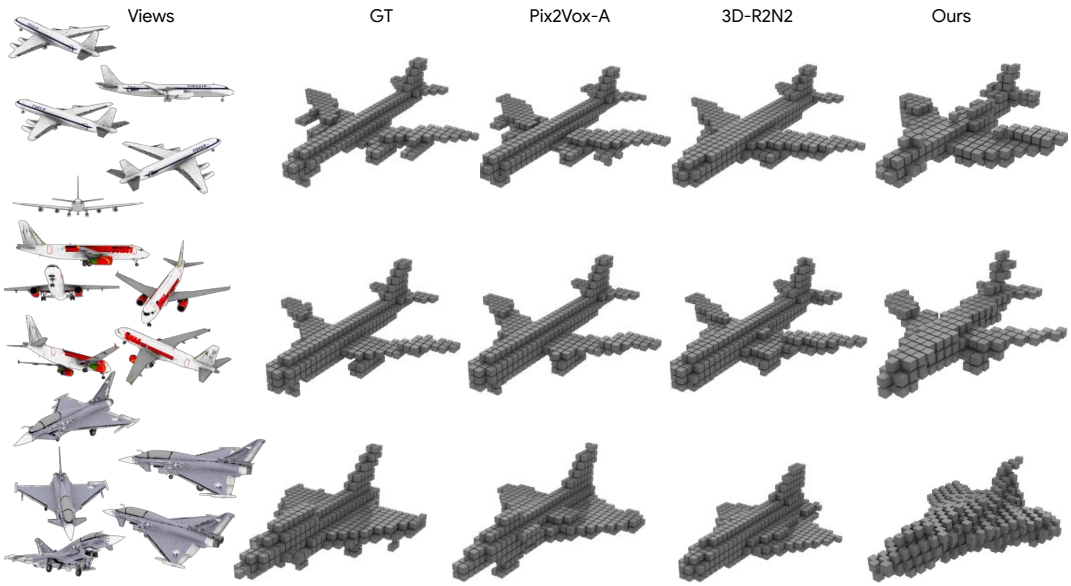


Figure 14.8: Comparison of multi-view reconstructions methods on the ShapeNet test set for the airplane category. On the left we show the 5 RGB views used as input for every method. We also report the results for the two main competitors Pix2Vox [200] and 3D-R2N2 [121]. Ours refers to PoseIDoNet.

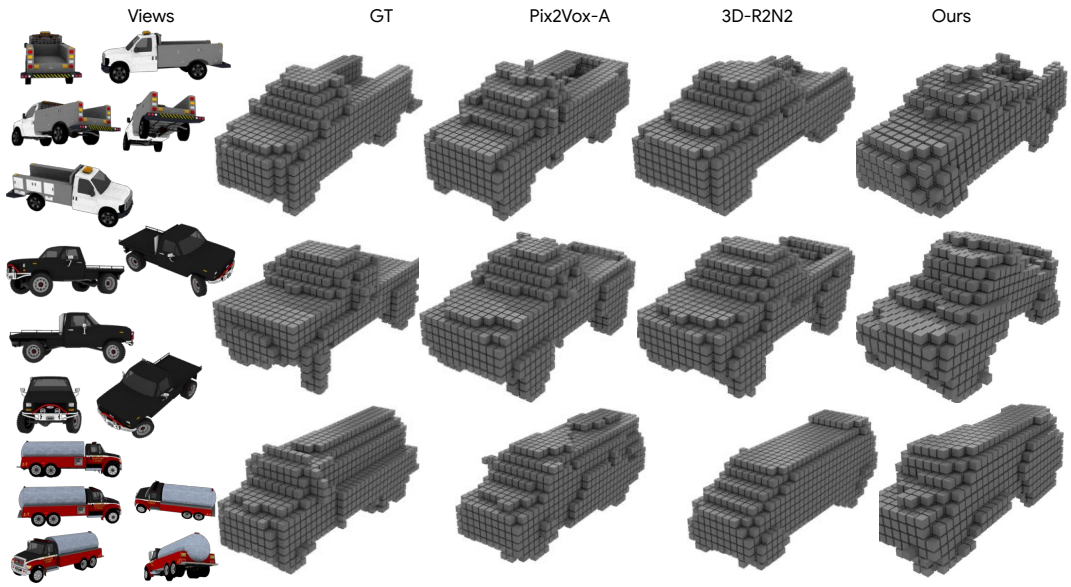


Figure 14.9: Comparison of multi-view reconstructions methods on the ShapeNet test set for the car category. On the left, we show the 5 RGB views used as input for every method. We also report the results for the two main competitors Pix2Vox [200] and 3D-R2N2 [121]. Ours refers to PoseIDoNet.

in Fig. 14.10, we show that we are able to maintain the fine details of the models that the other reconstruction solutions are completely discarding

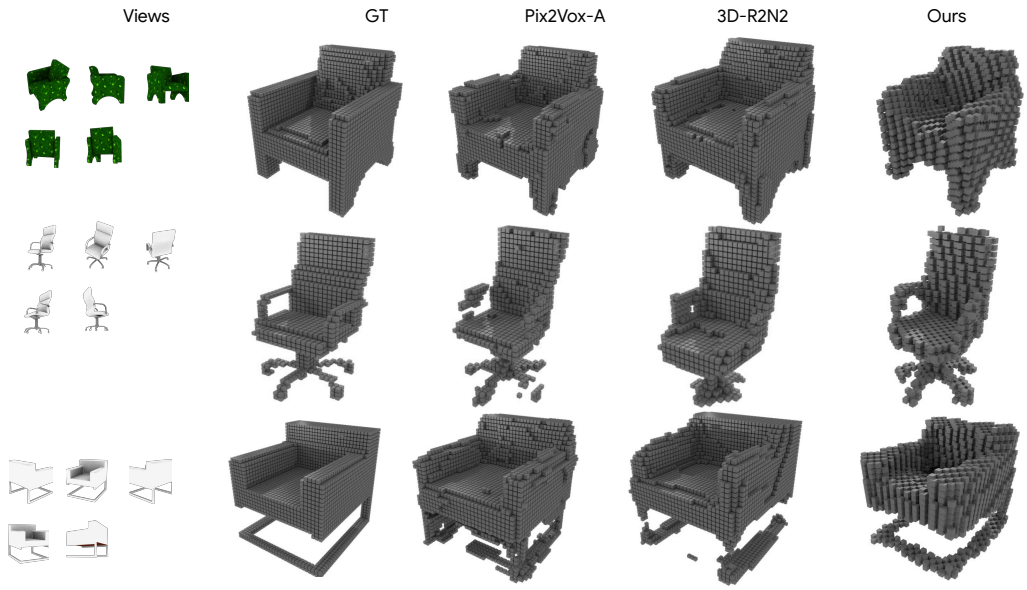


Figure 14.10: Comparison of multi-view reconstructions methods on the ShapeNet test set for the chair category. On the left we show the 5 RGB views used as input for every method. We also report the results for the two main competitors Pix2Vox [200] and 3D-R2N2 [121]. Ours refers to PoseIDoNet.

or incorrectly reconstructing. Considering the second row, our method is the only one to correctly reconstruct the arm rests and legs of the chair, while the competing methods either produce detached parts in the models or fill parts that should be empty. The same holds for the unusual shape of the legs of the chair in the third row. Our method is the only one to produce a correct reconstruction while the competitors either reconstruct a more canonical shape with four legs or result in detached parts. Similar consideration holds for the reconstruction of planes in Fig. 14.8 and trucks in Fig. 14.9. Our method correctly approximate the shape of the models while, at the same time, keeps some of the fine details like the flash light and the rear floor for the truck in the first row.

Qualitative results using the canonical orientation

This section compares the generated shapes of two variants of our reconstruction pipeline – the standard one reconstructing in an arbitrary reference frame aligned with one of the input views, *i.e.* PoseIDoNet; and, the one that always reconstruct a model in a canonical reference frame, *i.e.* PoseIDoNet Canonical. The purpose of this comparison is to clarify the performance of the second stage of our method, which is composed of building the

occupancy grid and the refiner network. For PoseIDoNet Canonical, we first build an occupancy grid oriented as the canonical orientation of the ShapeNet dataset, then we refine this volume using the refiner network. As already pointed out in 14.3.2, to supervise our network, for PoseIDoNet Canonical, we use as ground truth the voxel grid available from the ShapeNet dataset. When comparing the reconstruction of PoseIDoNet Canonical to PoseIDoNet, we can see how the reconstruction on a fixed reference frame results in more detailed and smooth models. We ascribe this difference to the task learned being simpler than reconstruction w.r.t. an arbitrary viewpoint. When comparing the reconstructions of PoseIDoNet Canonical to those of the competing methods, we can see once again how our method is able to obtain a similar quality of the reconstructed shape, while, at the same time, maintaining the fine details like: the legs and arm rests in the chair, the exhaust pipes in the truck or the propellers in the plane. We believe that these results clearly show how our 3D reconstruction from silhouettes projected in an occupancy grid is as effective (or more) than the alternatives proposed in the literature. However, reconstruction w.r.t. an arbitrary reference frame is a much harder task and this is reflected in slightly less detailed models for PoseIDoNet pipeline without the canonical frame reconstruction. We

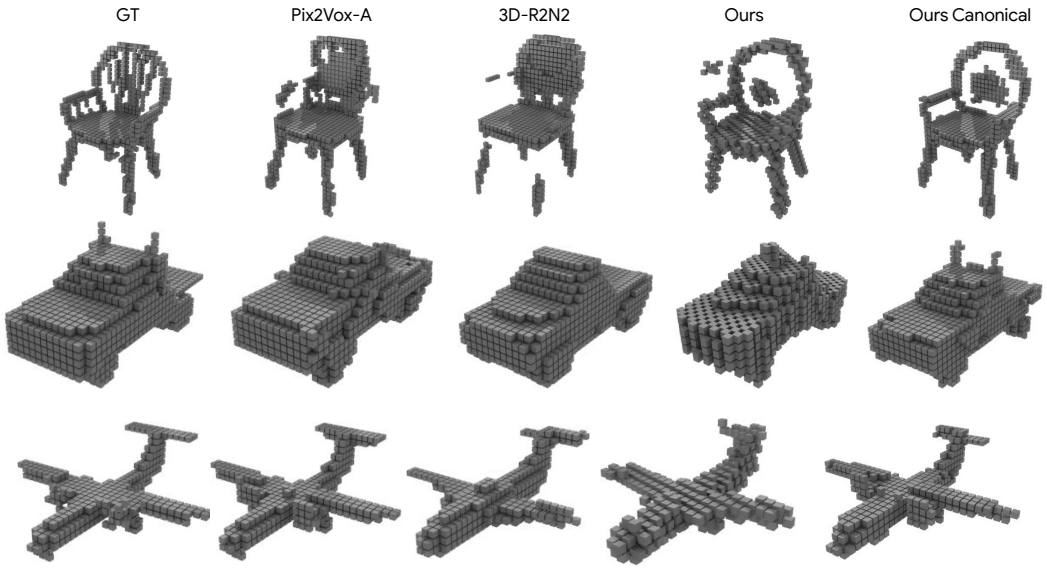


Figure 14.11: Comparison of multi-view reconstructions methods on the ShapeNet test set. On the left, we show the ground truth voxel grid. We also report the results for the two main competitors Pix2Vox [200] and 3D-R2N2 [121]. Ours and Ours Canonical refer to PoseIDoNet and PoseIDoNet Canonical, respectively.

will provide more details on how the reconstruction change with respect to the view choose as reference in 14.3.2.

Qualitative results with changing reference view

In Fig. 14.12 we show how our pipeline is able to reconstruct different models with respect to the reference view considered when building the occupancy grid. First of all, we would like to point out that our method can correctly reconstruct a model that is nicely aligned to each of the views when it is selected as reference. Secondly, we want highlight how the fine details on the reconstructions depends on how well the corresponding details are visible in the original reference image. For example the reconstruction of the legs of the first chair is more detailed when selecting as reference view one of the two rightmost one where the geometrical structure of the leg is cleanly visible. The same consideration can be extended to the second chair where the shape that mostly highlights the peculiar shape of the backrest is the one obtained when considering as reference the third view.

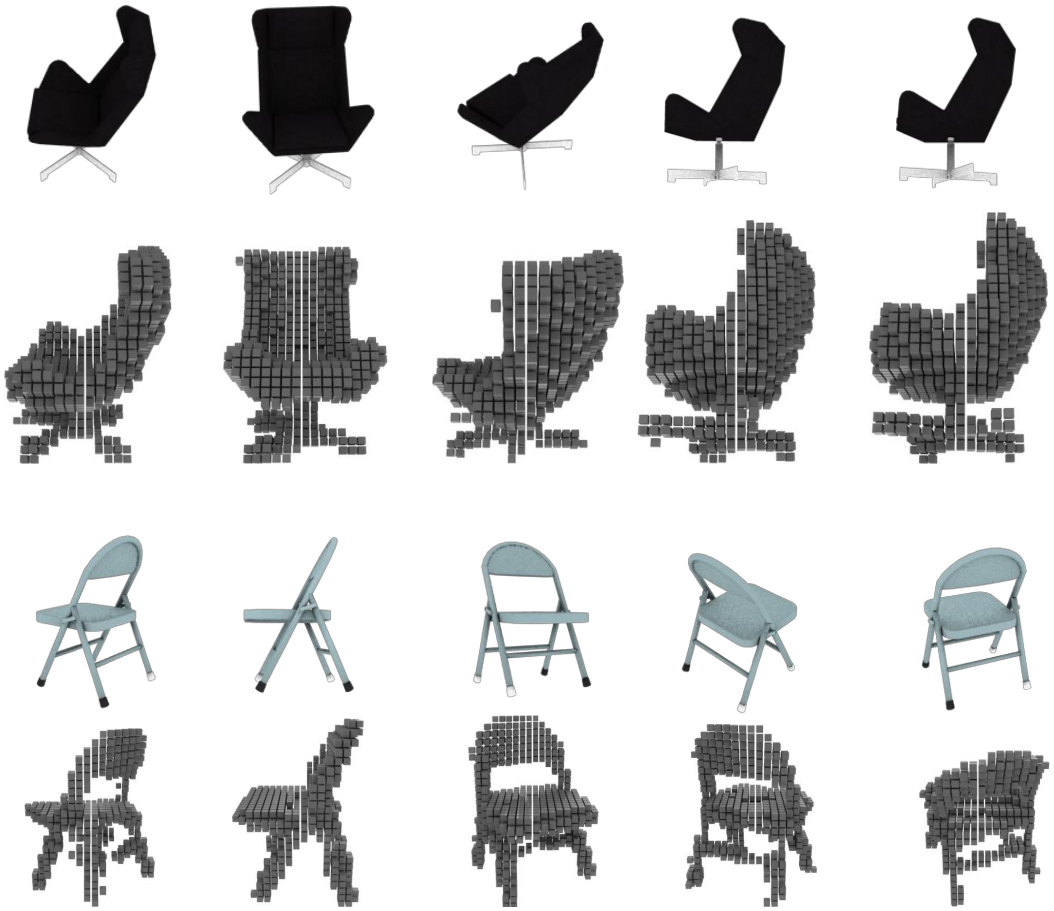


Figure 14.12: Comparison of multi-view reconstructions methods on the ShapeNet test set for the chair category performed by PoseIDoNet. In each column, we depict the voxel grid produced using a different reference view.

15

CONCLUSIONS

In the concluding part of this thesis, we have tackled the problem of recovering the geometry of an object from RGB images. Thus, in Chap. 14 we described a novel approach for 3D object reconstruction from non-overlapping images sparsely taken from an object. Contrarily to state-of-the-art proposals, our solution is the first to propose a 3D reconstruction pipeline that does not require training models from a dataset with a predefined canonical orientation and is able to reconstruct them aligned to any arbitrary input view. This characteristic is crucial both to scale to real datasets as well as to apply this technology to augmented reality applications. Our pipeline is built by stand alone components that address common problems in computer vision and can be easily reused for different tasks. In the future, we plan to continue the development of our solution following two main paths: end-to-end optimization of the whole pipeline for a specific class of objects and extension to different 3D representations. Moreover, we also plan to test our standalone components with different solutions taken from the recent literature in this field, for example we can combine our relative pose estimation network with the 3D multi view reconstruction from [142, 199].

Part V

FINAL REMARKS

16

CONCLUSIONS

This thesis has been concerned with the topic of understanding the 3D structures of objects and environments present in the real world through 3D computer vision. An autonomous agent moving inside an environment needs a proper knowledge of the space in order to navigate it and interact with objects. We have mainly discussed two problems: how to effectively deploy machine/deep learning methods to improve each of the step of the 3D feature matching pipeline, a crucial building block for many applications that rely on automatic estimation of similarity between shapes, and 3D object reconstruction from images, a traditional computer vision problem that is increasing in popularity thanks to the progress of deep learning. We have started this dissertation approaching the problem of 3D keypoint detection. In Chap. 3, we have shown how casting keypoint detection as binary classification has proven to be a successful and effective approach. In particular, the definition of a saliency function aimed at maximizing the overall performance of the feature matching pipeline turned out to be a valid strategy to increase the number of correctly matched descriptors in the context of 3D object recognition. Moreover, the experimental results proved that the keypoint learning framework can be successfully coupled with 3D descriptors that encodes different properties of the surface. Based on these results, we then tried to answer the long-standing problem of the definition of the best 3D descriptor-detector pair considering two 3D computer vision applications, 3D object recognition and surface registration. For most of the state-of-the-art proposals in the field of local 3D descriptors, we learned the corresponding detector and investigated how different pairs behave on datasets addressing the two above mentioned applications. As main outcome of this performance evaluation, we have shown that using a pair of learned descriptor-detector can achieve state-of-the-art performance on the Laser Scanner dataset in 3D object recognition.

Following the pattern defined by the feature matching pipeline, in Chap. 7, we then moved to the problem of describing local 3D keypoints and, we tackled it with a more modern deep learning based approach. Specifically, we combined two recent innovations, *i.e.* Spherical CNNs and plane folding

decoders, to learn a local equivariant embedding that can be oriented only at test time to achieve invariance to rotation. To this end, we have explored two possible directions. The first one, relies on leveraging the peculiar properties of Spherical CNNs to define a self-orienting descriptor, while the other consists in using an off-the-shelf LRF. Despite the elegance of the former, the latter turned out to be the most powerful. When used in the context of surface registration, our method is particularly effective. Indeed, we outperformed all the learned unsupervised 3D descriptors, and obtained competitive results against the latest supervised method, *i.e.* 3DSmoothNet. Through a deeper study of the design choices of the Spherical CNNs architecture, we demonstrated the power of our method to generalize both in indoor and outdoor environments, reaching the state-of-the-art results on both the 3DMatch and ETH datasets.

Moving on to Part III, we have shown how local descriptors suffer from a huge drop in performance when the LRF is not repeatable. We address this issue in Chap. 10, by proposing a novel LRF and comparing the performance of the original SHOT descriptor, Sec. 6.1.6, against to SHOT descriptor constructed on top of our proposed LRFs. In the experimental setup, we addressed rigid as well as deformable shape matching. In Chap. 11, we tackled a similar problem, but we solved it in a more effective way the by leveraging the equivariant property of Spherical CNNs. Thus, we realized the first learned framework to canonically orient local as well as and global shapes. As a result, we outperformed our previous proposal on the Stanford Views dataset and obtained a more repeatable and robust LRF. The state-of-the-art results on 3DMatch, ETH and Stanford Views in terms of LRF repeatability, together with the results presented in Chap. 7, reinforced the claim that Spherical CNNs are a very effective deep learning machinery for point cloud processing.

Finally, in Chap. 14 we have explored the idea of reconstructing 3D objects from multiple views by designing a two steps pipeline. Our solution consists in a pipeline that does not require known camera poses to aggregate the geometric structures present in the different images in the final reconstructed model, neither requires training models from a dataset with a predefined canonical orientation. The latter aspect proved crucial for the reconstructions, as evidenced by the results presented in Sec. 14.3.2. Thanks to having separated the 3D object reconstruction into two smaller problems, relative pose estimation between non-overlapping images and a occupancy grid refinement, each component of our pipeline can be reused as a plug-and-

play module for many different computer vision applications. Although in the experimental setup we relied on strong assumptions, *e.g.* the relative poses are rotation-only without any translations and images are required to feature a clean background, our proposal has set a new baseline on the ShapeNet dataset for view dependent 3D object reconstruction from multiple views without ground-truth camera poses.

In this thesis, we have analyzed how to apply data-driven approaches to solve 3D computer vision problems. Although the use of deep learning algorithms has boosted the performance of the feature matching pipeline, we believe there are still many questions that deserve to be answered. Differently from the images field, it is not yet clear what is the best parameterization to adopt while learning on 3D data in order to extract effective features. Moreover, robustness to geometric transformation and noise is an important issue, especially because most of the labeled datasets for 3D deep learning are made up of synthetic CAD objects. In our opinion data scarcity could obstacle the spread of deep learning in 3D computer vision, although, the wide spread solution of adopting synthetic datasets to train neural networks may help, the problem of domain shift when working on point clouds is critical and no valid proposal has yet addressed it in literature. The creation of huge sets of labeled point clouds or meshes is cumbersome. For this reason, we believe that the future of 3D deep learning is unsupervised. An additional open problem that scholars will have to face in forthcoming years is the scalability towards large datasets (*e.g.* large scenes, many objects) and in general computational efficiency, 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.

All these problems to overcome describe a research field that needs new solutions to effective deploy deep learning models able to operate in real world settings. We hope that the work carried out in this thesis will provide useful insights to future researchers in the field.

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] CE Shannon. 'A Mathematical Theory of Communication, Bell System Technical Journal, vol.' In: *27(1948)* (1948) (cit. on p. 65).

[2] Ming-Kuei Hu. 'Visual pattern recognition by moment invariants'. In: *IRE transactions on information theory* 8.2 (1962), pp. 179–187 (cit. on p. 65).

[3] Emanuel Parzen. 'On estimation of a probability density function and mode'. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076 (cit. on p. 137).

[4] Roger N Shepard and Jacqueline Metzler. 'Mental rotation of three-dimensional objects'. In: *Science* 171.3972 (1971), pp. 701–703 (cit. on p. 131).

[5] Jon Louis Bentley. 'Multidimensional binary search trees used for associative searching'. In: *Communications of the ACM* 18.9 (1975), pp. 509–517 (cit. on p. 6).

[6] Steven G Vandenberg and Allan R Kuse. 'Mental rotations, a group test of three-dimensional spatial visualization'. In: *Perceptual and motor skills* 47.2 (1978), pp. 599–604 (cit. on p. 131).

[7] Shimon Ullman. 'The interpretation of structure from motion'. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 203.1153 (1979), pp. 405–426 (cit. on p. 153).

[8] Martin A Fischler and Robert C Bolles. 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography'. In: *Communications of the ACM* 24.6 (1981), pp. 381–395 (cit. on p. 97).

[9] Christopher G Harris, Mike Stephens et al. 'A combined corner and edge detector.' In: *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244 (cit. on p. 13).

[10] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle. *Surface reconstruction from unorganized points*. Vol. 26. 2. ACM, 1992 (cit. on p. 91).

[11] James R Driscoll and Dennis M Healy. 'Computing Fourier transforms and convolutions on the 2-sphere'. In: *Advances in applied mathematics* 15.2 (1994), pp. 202–250 (cit. on p. 82).

[12] Aldo Laurentini. 'The visual hull concept for silhouette-based image understanding'. In: *IEEE Transactions on pattern analysis and machine intelligence* 16.2 (1994), pp. 150–162 (cit. on pp. 152, 162).

[13] Jianbo Shi et al. 'Good features to track'. In: *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE. 1994, pp. 593–600 (cit. on p. 16).

[14] Takayuki Itoh and Koji Koyamada. 'Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists'. In: *IEEE Transactions on Visualization and Computer Graphics* 1.4 (1995), pp. 319–327 (cit. on p. 72).

[15] Brian Curless and Marc Levoy. 'A volumetric method for building complex models from range images'. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM. 1996, pp. 303–312 (cit. on p. 122).

[16] Torben Risbo. 'Fourier transform summation of Legendre series and D-functions'. In: *Journal of Geodesy* 70.7 (1996), pp. 383–396 (cit. on p. 87).

[17] Chin Seng Chua and Ray Jarvis. 'Point signatures: A new representation for 3d object recognition'. In: *International Journal of Computer Vision* 25.1 (1997), pp. 63–85 (cit. on pp. 63, 114, 115).

[18] Andrew E. Johnson and Martial Hebert. 'Using spin images for efficient object recognition in cluttered 3D scenes'. In: *IEEE Transactions on pattern analysis and machine intelligence* 21.5 (1999), pp. 433–449 (cit. on pp. 15, 17, 30, 43, 44, 61, 63, 68, 79).

[19] Bill Triggs, Philip F McLauchlan, Richard I Hartley and Andrew W Fitzgibbon. 'Bundle adjustment—a modern synthesis'. In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372 (cit. on p. 161).

[20] Marcello Demi, Marco Paterni and Antonio Benassi. 'The first absolute central moment in low-level image processing'. In: *Computer Vision and Image Understanding* 80.1 (2000), pp. 57–87 (cit. on p. 65).

[21] Kiriakos N Kutulakos and Steven M Seitz. 'A theory of shape by space carving'. In: *International journal of computer vision* 38.3 (2000), pp. 199–218 (cit. on p. 154).

[22] 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*. 2001, pp. 831–837 (cit. on p. 63).

[23] Leo Breiman. 'Random forests'. In: *Machine learning* 45.1 (2001), pp. 5–32 (cit. on pp. 17, 24, 31).

[24] Robert Osada, Thomas Funkhouser, Bernard Chazelle and David Dobkin. 'Shape distributions'. In: *ACM Transactions on Graphics (TOG)* 21.4 (2002), pp. 807–832 (cit. on p. 61).

[25] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003 (cit. on pp. 151, 153).

[26] Niloy J Mitra and An Nguyen. 'Estimating surface normals in noisy point cloud data'. In: *Proc. Symp. Computational Geometry*. 2003 (cit. on p. 118).

[27] 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*. Springer. 2004, pp. 224–237 (cit. on pp. 44, 45, 51, 63, 72).

[28] David G Lowe. 'Distinctive image features from scale-invariant keypoints'. In: *International journal of computer vision* 60.2 (2004), pp. 91–110 (cit. on pp. 15, 41, 67).

[29] Matthew Brown and David G Lowe. 'Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets.' In: *3DIM*. Vol. 5. 2005, pp. 56–63 (cit. on p. 153).

[30] Sumit Chopra, Raia Hadsell and Yann 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 (cit. on pp. 75, 136).

[31] Eugene Zhang, Konstantin Mischaikow and Greg Turk. 'Feature-based Surface Parameterization and Texture Mapping'. In: *ACM Trans. Graph.* 24.1 (Jan. 2005), pp. 1–27 (cit. on p. 118).

[32] Herbert Bay, Tinne Tuytelaars and Luc Van Gool. 'Surf: Speeded up robust features'. In: *European conference on computer vision*. Springer. 2006, pp. 404–417 (cit. on p. 15).

[33] Raia Hadsell, Sumit Chopra and Yann LeCun. 'Dimensionality reduction by learning an invariant mapping'. In: *null*. IEEE. 2006, pp. 1735–1742 (cit. on p. 69).

[34] Noah Snavely, Steven M Seitz and Richard Szeliski. 'Photo tourism: exploring photo collections in 3D'. In: *ACM transactions on graphics (TOG)*. Vol. 25. ACM. 2006, pp. 835–846 (cit. on p. 153).

[35] Hui Chen and Bir Bhanu. '3D free-form object recognition in range images using local surface patches'. In: *Pattern Recognition Letters* 28.10 (2007), pp. 1252–1262 (cit. on p. 63).

[36] Babak Taati, Michel Bondy, Piotr Jasiobedzki and Michael Greenspan. 'Variable dimensional local shape descriptors for object recognition in range data'. In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8 (cit. on p. 39).

[37] Rasmus Bro, Evrim Acar and Tamara G Kolda. 'Resolving the sign ambiguity in the singular value decomposition'. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 22.2 (2008), pp. 135–140 (cit. on pp. 106, 107, 115).

[38] Alexander M Bronstein, Michael M Bronstein and Ron Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Springer Science & Business Media, 2008 (cit. on p. 124).

[39] Umberto Castellani, Marco Cristani, Simone Fantoni and Vittorio Murino. 'Sparse points matching by combining 3D mesh saliency with statistical descriptors'. In: *Computer Graphics Forum*. Vol. 27. 2. Wiley Online Library. 2008, pp. 643–652 (cit. on p. 15).

[40] 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*. Springer. 2008, pp. 440–453 (cit. on pp. 63, 109, 114).

[41] Edward Rosten, Reid Porter and Tom Drummond. 'Faster and better: A machine learning approach to corner detection'. In: *IEEE transactions on pattern analysis and machine intelligence* 32.1 (2008), pp. 105–119 (cit. on p. 17).

[42] 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*. IEEE. 2008, pp. 3384–3391 (cit. on pp. 44, 63, 65).

[43] Du Q Huynh. 'Metrics for 3D rotations: Comparison and analysis'. In: *Journal of Mathematical Imaging and Vision* 35.2 (2009), pp. 155–164 (cit. on p. 137).

[44] Helmut Pottmann, Johannes Wallner, Qi-Xing Huang and Yong-Liang Yang. 'Integral Invariants for Robust Geometry Processing'. In: *Computer Aided Geometric Design* 26.1 (2009), pp. 37–60 (cit. on p. 114).

[45] 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*. IEEE. 2009, pp. 3212–3217 (cit. on pp. 15, 17, 30, 43, 44, 61, 63, 66, 79, 114).

[46] Christoph Strecha, Albrecht Lindner, Karim Ali and Pascal Fua. 'Training for task specific keypoint detection'. In: *Joint Pattern Recognition Symposium*. Springer. 2009, pp. 151–160 (cit. on p. 18).

[47] Jian Sun, Maks Ovsjanikov and Leonidas Guibas. 'A concise and provably informative multi-scale signature based on heat diffusion'. In: *Computer graphics forum*. Vol. 28. 5. Wiley Online Library. 2009, pp. 1383–1392 (cit. on pp. 55, 63, 114).

[48] Yu Zhong. 'Intrinsic shape signatures: A shape descriptor for 3d object recognition'. In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE. 2009, pp. 689–696 (cit. on pp. 12, 15, 23, 43, 63).

[49] Andrea Albarelli, Emanuele Rodolà and Andrea Torsello. 'Loosely distinctive features for robust surface alignment'. In: *Proc. ECCV*. 2010 (cit. on p. 114).

[50] Prabin Bariya and Ko Nishino. 'Scale-hierarchical 3d object recognition in cluttered scenes'. In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 1657–1664 (cit. on p. 19).

[51] Olivier Chapelle and Mingrui Wu. 'Gradient descent optimization of smoothed information retrieval metrics'. In: *Information retrieval* 13.3 (2010), pp. 216–235 (cit. on p. 137).

[52] 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*. Ieee. 2010, pp. 998–1005 (cit. on pp. 70, 74, 77).

[53] 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*. 2010, pp. 249–256 (cit. on p. 75).

[54] 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*. Springer. 2010, pp. 589–602 (cit. on p. 63).

[55] Ajmal Mian, Mohammed Bennamoun and Robyn Owens. 'On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes'. In: *IJCV* 89.2-3 (2010), pp. 348–361 (cit. on pp. 15, 29, 32, 107, 114).

[56] 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*. IEEE. 2010, pp. 2155–2162 (cit. on p. 61).

[57] Bastian Steder, Radu Bogdan Rusu, Kurt Konolige and Wolfram Burgard. 'NARF: 3D range image features for object recognition'. In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Vol. 44. 2010 (cit. on pp. 13, 15, 29, 43).

[58] Federico Tombari, Samuele Salti and Luigi Di Stefano. 'Unique signatures of histograms for local surface description'. In: *European conference on computer vision*. Springer. 2010, pp. 356–369 (cit. on pp. 43, 44, 63, 64, 66, 77, 79, 114, 140, 141).

[59] Vladimir G Kim, Yaron Lipman and Thomas Funkhouser. 'Blended Intrinsic Maps'. In: *TOG* 30.4 (2011), p. 79 (cit. on p. 125).

[60] Stefan Leutenegger, Margarita Chli and Roland Siegwart. 'BRISK: Binary robust invariant scalable keypoints'. In: *2011 IEEE international conference on computer vision (ICCV)*. Ieee. 2011, pp. 2548–2555 (cit. on p. 17).

[61] 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*. IEEE. 2011, pp. 127–136 (cit. on p. 69).

[62] Maks Ovsjanikov, Qi-Xing Huang and Leonidas Guibas. 'A condition number for non-rigid shape matching'. In: *Computer Graphics Forum*. Vol. 30. 5. Wiley Online Library. 2011, pp. 1503–1512 (cit. on p. 55).

[63] Alioscia Petrelli and Luigi Di Stefano. 'On the repeatability of the local reference frame for partial shape matching'. In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2244–2251 (cit. on pp. 4, 77, 105, 109, 115, 132).

[64] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary R Bradski. 'ORB: An efficient alternative to SIFT or SURF.' In: *ICCV*. Vol. 11. 1. Citeseer. 2011, p. 2 (cit. on p. 17).

[65] Ivan Sipiran and Benjamin Bustos. 'Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes'. In: *The Visual Computer* 27.11 (2011), p. 963 (cit. on pp. 13, 43).

[66] 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*. IEEE. 2011, pp. 809–812 (cit. on pp. 29, 66).

[67] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli and Markus Vincze. 'Tutorial: Point cloud library: Three-dimensional object recognition and 6 DoF pose estimation'. In: *IEEE Robotics & Automation Magazine* 19.3 (2012), pp. 80–91 (cit. on pp. 19, 23, 29, 30, 91).

[68] 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*. Springer. 2012, pp. 113–122 (cit. on pp. 61, 132).

[69] Luıs A Alexandre. '3D descriptors for object and category recognition: a comparative evaluation'. In: *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*. Vol. 1. 3. 2012, p. 7 (cit. on p. 30).

[70] Jens Behley, Volker Steinhage and Armin B Cremers. 'Performance of histogram descriptors for the classification of 3D laser range data in urban environments'. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 4391–4398 (cit. on p. 30).

[71] Antonio Criminisi, Jamie Shotton, Ender Konukoglu et al. 'Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning'. In: *Foundations and Trends® in Computer Graphics and Vision* 7.2–3 (2012), pp. 81–227 (cit. on pp. 24, 26, 31).

[72] Helin Dutagaci, Chun Pan Cheung and Afzal Godil. 'Evaluation of 3D interest point detection techniques via human-generated ground truth'. In: *The Visual Computer* 28.9 (2012), pp. 901–917 (cit. on p. 18).

[73] Stefan Holzer, Jamie Shotton and Pushmeet Kohli. 'Learning to efficiently detect repeatable interest points in depth data'. In: *European Conference on Computer Vision*. Springer. 2012, pp. 200–213 (cit. on pp. 18, 19, 26).

[74] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'Imagenet classification with deep convolutional neural networks'. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on p. 70).

[75] Alioscia Petrelli and Luigi Di Stefano. 'A repeatable and efficient canonical reference for surface matching'. In: *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE. 2012, pp. 403–410 (cit. on pp. 77, 91, 109, 115, 120–122, 124, 140, 141, 147).

[76] François Pomerleau, Ming Liu, Francis Colas and Roland Siegwart. 'Challenging data sets for point cloud registration algorithms'. In: *The International Journal of Robotics Research* 31.14 (2012), pp. 1705–1711 (cit. on p. 94).

[77] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard and Daniel Cremers. 'A benchmark for the evaluation of RGB-D SLAM systems'. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 573–580 (cit. on p. 124).

[78] Federico M Sukno, John L Waddington and Paul F Whelan. 'Comparing 3D descriptors for local search of craniofacial landmarks'. In: *International Symposium on Visual Computing*. Springer. 2012, pp. 92–103 (cit. on p. 30).

[79] Walter Wohlkinger, Aitor Aldoma, Radu B Rusu and Markus Vincze. '3dnet: Large-scale object class recognition from cad models'. In: *2012 IEEE international conference on robotics and automation*. IEEE. 2012, pp. 5384–5391 (cit. on p. 30).

[80] Andrei Zaharescu, Edmond Boyer and Radu Horaud. 'Keypoints and local descriptors of scalar functions on 2D manifolds'. In: *International Journal of Computer Vision* 100.1 (2012), pp. 78–98 (cit. on pp. 15, 26, 29, 63).

[81] Clement Creusot, Nick Pears and Jim Austin. 'A machine-learning approach to keypoint detection and landmarking on 3D meshes'. In: *International journal of computer vision* 102.1-3 (2013), pp. 146–179 (cit. on pp. 18, 19).

[82] 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*. 2013, pp. 86–93 (cit. on pp. 61, 63).

[83] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu and Jianwei Wan. 'Rotational projection statistics for 3D local surface description and object recognition'. In: *International journal of computer vision* 105.1 (2013), pp. 63–86 (cit. on pp. 17, 26, 43, 44, 79, 108, 114, 115, 132).

[84] Richard Hartley, Jochen Trumpf, Yuchao Dai and Hongdong Li. 'Rotation averaging'. In: *International journal of computer vision* 103.3 (2013), pp. 267–305 (cit. on pp. 136, 137).

[85] Diederik P Kingma and Max Welling. 'Auto-encoding variational bayes'. In: *arXiv preprint arXiv:1312.6114* (2013) (cit. on p. 153).

[86] Joseph J Lim, Hamed Pirsiavash and Antonio Torralba. 'Parsing ikea objects: Fine pose estimation'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2992–2999 (cit. on p. 151).

[87] Emanuele Rodolà, Andrea Albarelli, Filippo Bergamasco and Andrea Torsello. 'A scale independent selection process for 3d object recognition in cluttered scenes'. In: *International journal of computer vision* 102.1-3 (2013), pp. 129–145 (cit. on p. 29).

[88] 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*. 2013, pp. 2930–2937 (cit. on pp. 70, 90, 96).

[89] Federico Tombari, Samuele Salti and Luigi Di Stefano. 'Performance evaluation of 3D keypoint detectors'. In: *International Journal of Computer Vision* 102.1-3 (2013), pp. 198–220 (cit. on pp. 15, 16, 28, 29, 33, 35, 43).

[90] 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*. 2013, pp. 1625–1632 (cit. on pp. 70, 90, 96).

[91] Aitor Aldoma, Thomas Fäulhammer and Markus Vincze. 'Automation of "ground truth" annotation for multi-view RGB-D object instance recognition datasets'. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 5016–5023 (cit. on p. 41).

[92] Federica Bogo, Javier Romero, Matthew Loper and Michael J Black. 'FAUST: Dataset and Evaluation for 3D Mesh Registration'. In: *Proc. CVPR*. 2014 (cit. on p. 124).

[93] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. 'Generative adversarial nets'. In: *Advances in neural information processing systems*. 2014, pp. 2672–2680 (cit. on p. 153).

[94] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu and Jianwei Wan. '3D object recognition in cluttered scenes with local surface features: a survey'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2270–2287 (cit. on pp. 61, 63, 64).

[95] Wilfried Hartmann, Michal Havlena and Konrad Schindler. 'Predicting matchability'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 9–16 (cit. on pp. 17, 18).

[96] Diederik P Kingma and Jimmy Ba. 'Adam: A method for stochastic optimization'. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 72, 75, 89, 94, 140, 144, 165, 169).

[97] Kevin Lai, Liefeng Bo and Dieter Fox. 'Unsupervised feature learning for 3D scene labeling'. In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE. 2014, pp. 3050–3057 (cit. on pp. 70, 90).

[98] Samuele Salti, Federico Tombari and Luigi Di Stefano. 'SHOT: Unique signatures of histograms for surface and texture description'. In: *Computer Vision and Image Understanding* 125 (2014), pp. 251–264 (cit. on pp. 15, 17, 23, 24, 26, 29, 30, 41, 43–46, 52, 55, 61, 63, 66, 79, 113–115, 120, 124, 132).

[99] Leizer Teran and Philippos Mordohai. '3d interest point detection via discriminative learning'. In: *European Conference on Computer Vision*. Springer. 2014, pp. 159–173 (cit. on pp. 2, 18, 19, 24).

[100] Yu Xiang, Roozbeh Mottaghi and Silvio Savarese. 'Beyond pascal: A benchmark for 3d object detection in the wild'. In: *IEEE winter conference on applications of computer vision*. IEEE. 2014, pp. 75–82 (cit. on p. 151).

[101] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis and R. Yang. 'Semantic Parametric Reshaping of Human Body Models'. In: *Proc. 3DV*. 2014 (cit. on p. 125).

[102] Tolga Birdal and Slobodan Ilic. 'Point pair features based object detection and pose estimation revisited'. In: *3D Vision (3DV), 2015 International Conference on*. IEEE. 2015, pp. 527–535 (cit. on p. 74).

[103] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi and Fisher Yu. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015 (cit. on p. 155).

[104] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su et al. 'Shapenet: An information-rich 3d model repository'. In: *arXiv preprint arXiv:1512.03012* (2015) (cit. on pp. 142, 151, 164).

[105] Jorge Fuentes-Pacheco, José Ruiz-Ascencio and Juan Manuel Rendón-Mancha. 'Visual simultaneous localization and mapping: a survey'. In: *Artificial Intelligence Review* 43.1 (2015), pp. 55–81 (cit. on pp. 152, 153).

[106] Elad Hoffer and Nir Ailon. 'Deep metric learning using triplet network'. In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92 (cit. on pp. 72, 73).

[107] Max Jaderberg, Karen Simonyan, Andrew Zisserman et al. 'Spatial transformer networks'. In: *Proc. NIPS*. 2015 (cit. on p. 115).

[108] Petra Jansen and Jan Kellner. 'The role of rotational hand movements and general motor ability in children's mental rotation performance'. In: *Frontiers in Psychology* 6 (2015), p. 984 (cit. on p. 131).

[109] Abhishek Kar, Shubham Tulsiani, Joao Carreira and Jitendra Malik. 'Category-specific object reconstruction from a single image'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1966–1974 (cit. on p. 153).

[110] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll and Michael J. Black. 'SMPL: A Skinned Multi-person Linear Model'. In: *TOG* 34.6 (2015), 248:1–248:16. DOI: 10.1145/2816795.2818013 (cit. on p. 125).

[111] J. Masci, D. Boscaini, M. Bronstein and P. Vandergheynst. 'Geodesic convolutional neural networks on Riemannian manifolds'. In: *Proc. 3dRR*. 2015 (cit. on pp. 114, 131).

[112] Daniel Maturana and Sebastian Scherer. 'Voxnet: A 3D convolutional neural network for real-time object recognition'. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 922–928 (cit. on pp. 69, 79).

[113] Samuele Salti, Federico Tombari, Riccardo Spezialetti and Luigi Di Stefano. 'Learning a descriptor-specific 3D keypoint detector'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2318–2326 (cit. on pp. 45, 50, 55).

[114] 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*. 2015, pp. 945–953 (cit. on p. 79).

[115] Yannick Verdie, Kwang Yi, Pascal Fua and Vincent Lepetit. 'TILDE: a temporally invariant learned detector'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5279–5288 (cit. on p. 18).

[116] 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*. 2015, pp. 1912–1920 (cit. on p. 155).

[117] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang and J. Xiao. '3D ShapeNets: A deep representation for volumetric shapes'. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920 (cit. on p. 142).

[118] Davide Boscaini, Jonathan Masci, Emanuele Rodolà and Michael Bronstein. 'Learning shape correspondence with anisotropic convolutional neural networks'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3189–3197 (cit. on pp. 114, 115).

[119] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein and Daniel Cremers. 'Anisotropic diffusion descriptors'. In: *Computer Graphics Forum* 35.2 (2016), pp. 431–441 (cit. on p. 115).

[120] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst. 'Geometric deep learning: going beyond Euclidean data'. In: *arXiv:1611.08097* (2016) (cit. on p. 114).

[121] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen and Silvio Savarese. '3d-r2n2: A unified approach for single and multi-view 3d object reconstruction'. In: *European conference on computer vision*. Springer. 2016, pp. 628–644 (cit. on pp. 153–155, 164, 169, 170, 172–174).

[122] Luca Cosmo, Emanuele Rodolà, Jonathan Masci, Andrea Torsello and Michael M Bronstein. 'Matching deformable objects in clutter'. In: *Proc. 3DV*. 2016 (cit. on p. 115).

[123] Rohit Girdhar, David F Fouhey, Mikel Rodriguez and Abhinav Gupta. 'Learning a predictable and generative vector representation for objects'. In: *European Conference on Computer Vision*. Springer. 2016, pp. 484–499 (cit. on p. 153).

[124] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan and Ngai Ming Kwok. 'A comprehensive performance evaluation of 3D local feature descriptors'. In: *International Journal of Computer Vision* 116.1 (2016), pp. 66–89 (cit. on pp. 41, 43, 51, 52, 61, 62, 77).

[125] Maciej Halber and Thomas Funkhouser. 'Structured global registration of rgb-d scans in indoor environments. arXiv preprint'. In: *arXiv preprint arXiv:1607.08539* 2.3 (2016), p. 9 (cit. on p. 70).

[126] Z. Lähner, E. Rodolà, M. M. Bronstein, D. Cremers, O. Burghard, L. Cosmo, A. Dieckmann, R. Klein and Y. Sahillioğlu. 'Matching of Deformable Shapes with Topological Noise'. In: *Proc. 3DOR*. 2016 (cit. on p. 124).

[127] Xinyu Lin, Ce Zhu, Qian Zhang and Yipeng Liu. '3d keypoint detection based on deep neural network with sparse autoencoder'. In: *arXiv preprint arXiv:1605.00129* (2016) (cit. on pp. 18, 19).

[128] 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*. 2016, pp. 5648–5656 (cit. on p. 69).

[129] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg and Nicolas Heess. 'Unsupervised learning of 3d structure from images'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 4996–5004 (cit. on p. 154).

[130] Nima Sedaghat, Mohammadreza Zolfaghari, Ehsan Amiri and Thomas Brox. 'Orientation-boosted voxel nets for 3d object recognition'. In: *arXiv preprint arXiv:1604.03351* (2016) (cit. on p. 105).

[131] Maxim Tatarchenko, Alexey Dosovitskiy and Thomas Brox. 'Multi-view 3d models from single images with a convolutional network'. In: *European Conference on Computer Vision*. Springer. 2016, pp. 322–337 (cit. on pp. 153, 155).

[132] Shubham Tulsiani, Abhishek Kar, Joao Carreira and Jitendra Malik. 'Learning category-specific deformable 3d models for object reconstruction'. In: *IEEE transactions on pattern analysis and machine intelligence* 39.4 (2016), pp. 719–731 (cit. on p. 153).

[133] 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*. IEEE. 2016, pp. 323–332 (cit. on pp. 69, 70, 90).

[134] 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*. 2016, pp. 1544–1553 (cit. on p. 79).

[135] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman and Josh Tenenbaum. 'Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling'. In: *Advances in neural information processing systems*. 2016, pp. 82–90 (cit. on p. 153).

[136] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo and Honglak Lee. 'Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision'. In: *Advances in Neural Information Processing Systems*. 2016, pp. 1696–1704 (cit. on pp. 152–154).

[137] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D Kulkarni and Joshua B Tenenbaum. 'Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1511–1519 (cit. on pp. 153, 155).

[138] Haoqiang Fan, Hao Su and Leonidas J. Guibas. 'A Point Set Generation Network for 3D Object Reconstruction From a Single Image'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017 (cit. on p. 153).

[139] Maciej Halber and Thomas Funkhouser. 'Fine-to-coarse global registration of RGB-D scans'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1755–1764 (cit. on p. 90).

[140] Christian Häne, Shubham Tulsiani and Jitendra Malik. 'Hierarchical surface prediction for 3d object reconstruction'. In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 412–420 (cit. on p. 153).

[141] Alexander Hermans, Lucas Beyer and Bastian Leibe. 'In defense of the triplet loss for person re-identification'. In: *arXiv preprint arXiv:1703.07737* (2017) (cit. on p. 76).

[142] Abhishek Kar, Christian Häne and Jitendra Malik. 'Learning a multi-view stereo machine'. In: *Advances in neural information processing systems*. 2017, pp. 365–376 (cit. on pp. 154, 155, 177).

[143] Marc Khoury, Qian-Yi Zhou and Vladlen Koltun. 'Learning compact geometric features'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 153–161 (cit. on pp. 43, 45, 48–51, 53, 63, 71, 77, 79, 84, 105, 115).

[144] Roman Klokov and Victor Lempitsky. 'Escape from cells: Deep kd-networks for the recognition of 3D point cloud models'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 863–872 (cit. on pp. 79, 144).

[145] Siddharth Mahendran, Haider Ali and René Vidal. '3d pose regression using convolutional neural networks'. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, pp. 2174–2182 (cit. on p. 137).

[146] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda and M. M. Bronstein. 'Geometric deep learning on graphs and manifolds using mixture model CNNs'. In: *Proc. CVPR*. 2017 (cit. on pp. 114, 115).

[147] Onur Özyeşil, Vladislav Voroninski, Ronen Basri and Amit Singer. 'A survey of structure from motion*.' In: *Acta Numerica* 26 (2017), pp. 305–364 (cit. on pp. 151, 153).

[148] Charles R Qi, Hao Su, Kaichun Mo and Leonidas J Guibas. 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660 (cit. on pp. 70, 74, 79, 115, 131, 133, 143, 144).

[149] 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*. 2017, pp. 5099–5108 (cit. on pp. 79, 131, 133, 144).

[150] Gernot Riegler, Ali Osman Ulusoy and Andreas Geiger. 'Octnet: Learning deep 3d representations at high resolutions'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3577–3586 (cit. on p. 153).

[151] Maxim Tatarchenko, Alexey Dosovitskiy and Thomas Brox. 'Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2088–2096 (cit. on pp. 79, 153).

[152] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros and Jitendra Malik. 'Multi-view supervision for single-view reconstruction via differentiable ray consistency'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2626–2634 (cit. on p. 153).

[153] Meng Wang, Lingjing Wang and Yi Fang. '3DensiNet: A robust neural network architecture towards 3D volumetric object prediction from 2D image'. In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 961–969 (cit. on p. 154).

[154] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun and Xin Tong. 'O-cnn: Octree-based convolutional neural networks for 3d shape analysis'. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), p. 72 (cit. on p. 153).

[155] Chao-Yuan Wu, R Manmatha, Alexander J Smola and Philipp Krahenbuhl. 'Sampling matters in deep embedding learning'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2840–2848 (cit. on p. 79).

[156] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman and Josh Tenenbaum. 'Marrnet: 3d shape reconstruction via 2.5 d sketches'. In: *Advances in neural information processing systems*. 2017, pp. 540–550 (cit. on p. 153).

[157] Jiaqi Yang, Qian Zhang, Yang Xiao and Zhiguo Cao. 'TOLDI: An effective and robust approach for 3D local shape description'. In: *Pattern Recognition* 65 (2017), pp. 175–187 (cit. on pp. 63, 75, 110, 132, 140, 141).

[158] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov and Alexander J Smola. 'Deep sets'. In: *Advances in neural information processing systems*. 2017, pp. 3391–3401 (cit. on p. 144).

[159] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao and Thomas Funkhouser. '3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1802–1811 (cit. on pp. 44, 62, 63, 69, 71, 79, 89, 91, 97, 99, 100).

[160] Taco S Cohen, Mario Geiger, Jonas Köhler and Max Welling. 'Spherical CNNs'. In: *arXiv preprint arXiv:1801.10130* (2018) (cit. on pp. 4, 78, 80–83, 93, 133–135, 144).

[161] Haowen Deng, Tolga Birdal and Slobodan Ilic. 'PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 602–618 (cit. on pp. 62, 63, 73, 77–79, 84, 90–92, 105).

[162] Haowen Deng, Tolga Birdal and Slobodan Ilic. 'PPFNet: Global Context Aware Local Features for Robust 3D Point Matching'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 195–205 (cit. on pp. 62, 63, 70, 80, 88, 105, 115).

[163] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia and Kostas Daniilidis. 'Learning SO(3) Equivariant Representations with Spherical CNNs'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–68 (cit. on pp. 78, 83, 86, 131, 133, 134).

[164] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell and Mathieu Aubry. 'A Papier-Mâché Approach to Learning 3D Surface Generation'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on pp. 152, 153).

[165] 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*. 2018, pp. 216–224 (cit. on pp. 78, 88).

[166] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov and Niloy J Mitra. 'PCPNet Learning Local Shape Properties from Raw Point Clouds'. In: *Computer Graphics Forum* 37.2 (2018), pp. 75–85 (cit. on p. 115).

[167] Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal and Jan Kautz. 'Improving landmark localization with semi-supervised learning'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1546–1555 (cit. on p. 137).

[168] Eldar Insafutdinov and Alexey Dosovitskiy. 'Unsupervised learning of shape and pose with differentiable point clouds'. In: *Advances in Neural Information Processing Systems*. 2018, pp. 2802–2812 (cit. on pp. 153, 154, 157, 164, 165).

[169] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros and Jitendra Malik. 'Learning category-specific mesh reconstruction from image collections'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 371–386 (cit. on p. 153).

[170] Hiroharu Kato, Yoshitaka Ushiku and Tatsuya Harada. 'Neural 3D Mesh Renderer'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on p. 153).

[171] Hiroharu Kato, Yoshitaka Ushiku and Tatsuya Harada. 'Neural 3d mesh renderer'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3907–3916 (cit. on p. 153).

[172] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher Choy and Silvio Savarese. 'Deformnet: Free-form deformation network for 3d shape reconstruction from a single image'. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 858–866 (cit. on p. 153).

[173] Jiaxin Li, Ben M Chen and Gim Hee Lee. 'So-net: Self-organizing network for point cloud analysis'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9397–9406 (cit. on p. 144).

[174] Chen-Hsuan Lin, Chen Kong and Simon Lucey. 'Learning efficient point cloud generation for dense 3D object reconstruction'. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018 (cit. on pp. 153, 154, 169).

[175] Priyanka Mandikal, Navaneet Murthy, Mayank Agarwal and R Venkatesh Babu. '3D-LMNet: Latent Embedding Matching for Accurate and Diverse 3D Point Cloud Reconstruction from a Single Image'. In: *arXiv preprint arXiv:1807.07796* (2018) (cit. on p. 153).

[176] Fabian Manhardt, Wadim Kehl, Nassir Navab and Federico Tombari. 'Deep model-based 6d pose refinement in rgb'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 800–815 (cit. on pp. 159, 160).

[177] Riccardo Marin, Simone Melzi, Emanuele Rodolà and Umberto Castellani. *FARM: Functional Automatic Registration Method for 3D Human Bodies*. 2018 (cit. on p. 124).

[178] Simone Melzi, Maks Ovsjanikov, Giorgio Roffo, Marco Cristani and Umberto Castellani. 'Discrete Time Evolution Process Descriptor for Shape Analysis and Matching'. In: *TOG* 37.1 (Jan. 2018), 4:1–4:18. ISSN: 0730-0301 (cit. on p. 125).

[179] Stephan R. Richter and Stefan Roth. 'Matryoshka Networks: Predicting 3D Geometry via Nested Shape Layers'. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018 (cit. on pp. 152, 153).

[180] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum and William T Freeman. 'Pix3d: Dataset and methods for single-image 3d shape modeling'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2974–2983 (cit. on pp. 151, 157, 167).

[181] Alessio Tonioni, Samuele Salti, Federico Tombari, Riccardo Spezialetti and Luigi Di Stefano. 'Learning to detect good 3D keypoints'. In: *International Journal of Computer Vision* 126.1 (2018), pp. 1–20 (cit. on pp. 45, 52).

[182] Shubham Tulsiani, Alexei A Efros and Jitendra Malik. 'Multi-view consistency as supervisory signal for learning shape and pose prediction'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2897–2905 (cit. on pp. 153, 154, 157, 164, 165).

[183] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu and Yu-Gang Jiang. 'Pixel2mesh: Generating 3d mesh models from single rgb images'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 52–67 (cit. on pp. 152–154).

[184] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T Freeman and Joshua B Tenenbaum. 'Learning shape priors for single-view 3d completion and reconstruction'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 646–662 (cit. on pp. 152, 153).

[185] Jiaqi Yang, Yang Xiao and Zhiguo Cao. 'Toward the Repeatability and Robustness of the Local Reference Frame for 3D Shape Matching: An Evaluation'. In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3766–3781 (cit. on p. 106).

[186] Yaoqing Yang, Chen Feng, Yiru Shen and Dong Tian. 'FoldingNet: Point Cloud Auto-Encoder via Deep Grid Deformation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 206–215 (cit. on pp. 74, 78, 88).

[187] Yang You, Yujing Lou, Qi Liu, Lizhuang Ma, Weiming Wang, Yuwing Tai and Cewu Lu. 'PRIN: Pointwise Rotation-Invariant Network'. In: *arXiv preprint arXiv:1811.09361* (2018) (cit. on pp. 83, 131, 133, 135, 143, 144).

[188] Zhiqin Chen, Andrea Tagliasacchi and Hao Zhang. 'BSP-Net: Generating Compact Meshes via Binary Space Partitioning'. In: *arXiv preprint arXiv:1911.06971* (2019) (cit. on p. 153).

[189] Zhiqin Chen and Hao Zhang. 'Learning implicit fields for generative shape modeling'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948 (cit. on p. 153).

[190] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton and Andrea Tagliasacchi. 'Cvxnets: Learnable convex decomposition'. In: *arXiv preprint arXiv:1909.05736* (2019) (cit. on p. 155).

[191] Zan Gojcic, Caifa Zhou, Jan D Wegner and Andreas Wieser. 'The perfect match: 3d point cloud matching with smoothed densities'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5545–5554 (cit. on pp. 62, 75, 77, 79, 94, 95, 97, 99, 100, 105, 132, 140, 141).

[192] Xianfeng Han, Hamid Laga and Mohammed Bennamoun. 'Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era'. In: *IEEE transactions on pattern analysis and machine intelligence* (2019) (cit. on p. 152).

[193] Xinhai Liu, Zhizhong Han, Yu-Shen Liu and Matthias Zwicker. 'Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-based Sequence to Sequence Network'. In: *Thirty-Third AAAI Conference on Artificial Intelligence*. 2019 (cit. on p. 144).

[194] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin and Andreas Geiger. 'Occupancy networks: Learning 3d reconstruction in function space'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470 (cit. on pp. 152, 153).

[195] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe and Steven Lovegrove. 'Deepsdf: Learning continuous signed distance functions for shape representation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174 (cit. on p. 153).

[196] Yongming Rao, Jiwen Lu and Jie 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 (cit. on pp. 131, 133).

[197] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun and Thomas Brox. 'What Do Single-view 3D Reconstruction Networks Learn?' In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3405–3414 (cit. on p. 152).

[198] Yi Wei, Shaohui Liu, Wang Zhao and Jiwen Lu. 'Conditional Single-view Shape Generation for Multi-view Stereo Reconstruction'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9651–9660 (cit. on pp. 153–155).

[199] Chao Wen, Yinda Zhang, Zhuwen Li and Yanwei Fu. 'Pixel2mesh++: Multi-view 3d mesh generation via deformation'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1042–1051 (cit. on pp. 154, 155, 177).

[200] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou and Shengping Zhang. 'Pix2Vox: Context-Aware 3D Reconstruction From Single and Multi-View Images'. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2019 (cit. on pp. 154, 155, 163, 164, 169, 170, 172–174).

[201] Kuangen Zhang, Ming Hao, Jing Wang, Clarence W de Silva and Chenglong Fu. 'Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features'. In: *arXiv preprint arXiv:1904.10014* (2019) (cit. on pp. 131, 144).

[202] Zhiyuan Zhang, Binh-Son Hua, David W Rosen and Sai-Kit Yeung. 'Rotation invariant convolutions for 3D point clouds deep learning'. In: *2019 International Conference on 3D Vision (3DV)*. IEEE. 2019, pp. 204–213 (cit. on pp. 131, 133).

[203] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang and Hao Li. 'On the continuity of rotation representations in neural networks'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5745–5753 (cit. on p. 137).

[204] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen and Johannes Kopf. 'Consistent Video Depth Estimation'. In: 39.4 (2020) (cit. on p. 139).