Int J Comput Vis (2018) 126:1–20
https://doi.org/10.1007/s11263-017-1037-3

Learning to Detect Good 3D Keypoints

Alessio Tonioni[1] · **Samuele Salti**[2] · **Federico Tombari**[3] · **Riccardo Spezialetti**[1] · **Luigi Di Stefano**[1]

Received: 19 November 2016 / Accepted: 20 July 2017 / Published online: 8 August 2017

Abstract The established approach to 3D keypoint detection consists in defining effective handcrafted saliency functions based on geometric cues with the aim of maximizing keypoint repeatability. Differently, the idea behind our work is to learn a descriptor-specific keypoint detector so as to optimize 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. We propose a machine learning framework that allows for defining examples of good surface patches from the training data and leverages Random Forest classifiers to realize both *fixed-scale* and *adaptive-scale* 3D keypoint detectors. Through extensive experiments on standard datasets, we show how feature matching performance improves significantly by deploying 3D descriptors together with companion detectors learned by our methodology with respect to the adoption of established state-of-the-art 3D detectors based on hand-crafted saliency functions.

Keywords 3D Keypoint Detection · 3D Descriptors · Machine Learning · Surface Matching

Communicated by Ko Nishino.

Samuele Salti and Federico Tombari: This work was done when at DISI, University of Bologna.

✉ Federico Tombari
tombari@in.tum.de

Alessio Tonioni
alessio.tonioni2@unibo.it

Samuele Salti
samuele.salti@fleetmatics.com

Riccardo Spezialetti
riccardo.spezialetti@unibo.it

Luigi Di Stefano
luigi.distefano@unibo.it

[1] DISI, University of Bologna, Bologna, Italy

[2] Fleetmatics Research, Florence, Italy

[3] CAMP, Technical University of Munich, Munich, Germany

1 Introduction

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 (Lowe 2004) and SURF (Bay et al. 2008), 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 (Tombari et al. 2013), 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 (Johnson and Hebert 1999), arguably the most influential paper in the field, as well as later popular methods such as SHOT (Salti et al. 2014) and FPFH (Rusu et al. 2009).

Akin to image features (Tuytelaars and Mikolajczyk 2008; Li et al. 2015), the standard paradigm for extracting 3D keypoints from point clouds or meshes relies on maximizing a hand-crafted saliency function computed within a local neighborhood of each data point (Tombari et al. 2013). According to the taxonomy in Tombari et al. (2013), 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) (Zhong 2009) is a widely-used, fast and effective proposal; the fixed-scale detector introduced by Mian et al. (2010) is a slower alternative, particularly robust to point density variations; NARF (Steder et al. 2011) is a method specifically conceived for 2.5D data such as range images. Among adaptive-scale detectors, MeshDoG (Zaharescu et al. 2009) is an extension of the popular Difference of Gaussian detector (Lowe 2004) to scalar functions defined over a manifold approximated by a mesh; the adaptive-scale variant proposed in Mian et al. (2010) maximizes the saliency function across scales to adaptively define the neighborhood size; Castellani et al. (2008) is another proposal aimed at extending the DoG operator to meshes. Performance of 3D keypoint detectors is usually measured in terms of repeatability (Tombari et al. 2013).

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 (1994), 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 Tombari et al. (2013), 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 hand-crafted 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 (Guo et al. 2013a, b; Salti et al. 2014; Rusu et al. 2009; Johnson and Hebert 1999). In particular, throughout our experimental evaluation, we will consider descriptors as diverse as SHOT (Salti et al. 2014), FPFH (Rusu et al. 2009) and Spin Images (Johnson and Hebert 1999) 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 hand-crafted detectors.

The paper is structured as follows. Section 2 reviews related work concerning machine learning approaches for keypoint detection. Section 3 describes the proposed framework to create the training set for a fixed-scale keypoint detector, which entails the definition of both exemplar keypoints and negative samples for a chosen descriptor; we also report on experiments aimed at validating the effectiveness of the proposed framework in sifting out good features for the chosen descriptor. Section 4 addresses the design of the classifier, delineating its feature space as well as how the

trained classifier is run at test-time to extract keypoints from a point cloud. Section 5 shows how the framework described in Sects. 3 and 4 can be extended in order to pursue adaptive-scale keypoint detection. Section 6 presents the experimental results, where our approach is realized for three descriptors [SHOT (Salti et al. 2014), FPFH (Rusu et al. 2009), Spin Images (Johnson and Hebert 1999)] and compared to the most relevant hand-crafted detectors on four publicly available datasets acquired by diverse sensing modalities. We also 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. Finally, Sect. 7 draws concluding remarks and highlights possible avenues of research along the lines set forth in this paper.

2 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 (Rosten et al. 2010), 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 (Leutenegger et al. 2011) and ORB (Rublee et al. 2011). 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 Hartmann et al. (2014), 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 (Breiman 2001) 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 Strecha et al. (2009), 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 Verdie et al. (2015), 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 (Strecha et al. 2009; Hartmann et al. 2014; Verdie et al. 2015), 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 (Creusot et al. 2013; Teran and Mordohai 2014; Lin et al. 2016). In particular, Creusot et al. (2013) investigates on the use of linear (LDA) and non-linear (AdaBoost) classifiers to detect facial landmarks in 3D meshes. Similarly, the authors of Teran and Mordohai (2014) propose to learn a classification forest to better cope with the high variability of the structures annotated as salient within the dataset proposed in Dutagaci et al. (2012), 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 Teran and Mordohai (2014), the authors of Lin et al. (2016) 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 Dutagaci et al. (2012). Differently, Holzer et al. (2012) 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, in this work 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 Creusot et al. (2013), Teran and Mordohai (2014), Lin et al. (2016), nor approximate a hand-crafted criterion, as in Holzer et al. (2012). Instead, it is peculiarly data-driven and attained automatically based on the ability to match specific descriptors, thereby generating a descriptor-specific keypoint detector.

Finally, we point out that this paper consolidates and extends the results reported in Salti et al. (2015), the additional material concerning mainly the design of the adaptive-scale detector and the experimental validation of the approach with multiple descriptors.

3 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 Sect. 5, then, we extend the methodology to learn adaptive-scale detectors.

3.1 Definition of the Training Set

Let $\mathcal{V} = \{V^i\}, i = 1, \ldots, N$, be a set of calibrated 2.5D views of objects in a 3D dataset (Fig. 1). We refer to them as 2.5D because they are the result of a perspective projection, either real or simulated, and as calibrated because the ground-truth roto-translations to bring each pair of views into a common reference frame are known. If the dataset features 3D models instead of their 2.5D views, the views can be obtained by performing synthetic renderings, as also done, e.g., in Bariya and Nishino (2010), Aldoma et al. (2012b). For instance, for our datasets of 3D models we deploy the method in Aldoma et al. (2012b), where 2.5D views are rendered from the nodes of an icosahedron centered at the centroid of the model.

The algorithm to select good points to learn a descriptor-specific keypoint detector is presented in Algorithm 1 For each point p in each view V^i, we first compute the selected 3D descriptor, denoted as D^i_p (lines 3–5). Then, for each V^i, we select the subset

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

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 selection procedure exemplified in Fig. 1.

The aim of the first step is to identify a set of candidate positive samples, P^j_i, consisting of the points of V^i that yield good matches when seen from the vantage point associated with V^j. Therefore, 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 Euclidean distance between descriptors. We then remove the fist pair from the list and check whether the match is correct, i.e. if the points in the two views represent the same 3D point. If the match is correct, we insert p into P^j_i and we also remove from the list all pairs that include its neighbors within radius ϵ_{NMS} (lines 23–25). The procedure is then repeated until the list becomes empty. To make the learned detector robust to small shifts, we regard a match as correct even if the points in the two views are not exactly the same 3D point, but their distance is smaller than a threshold ϵ (line 21). Removal of neighbors is executed to mimic the effect of the spatial non-maxima suppression usually performed by hand-crafted detectors to prevent multiple responses around salient structures.

In the second step, the aim is to select more robust keypoints by alleviating the dependence of the positive samples

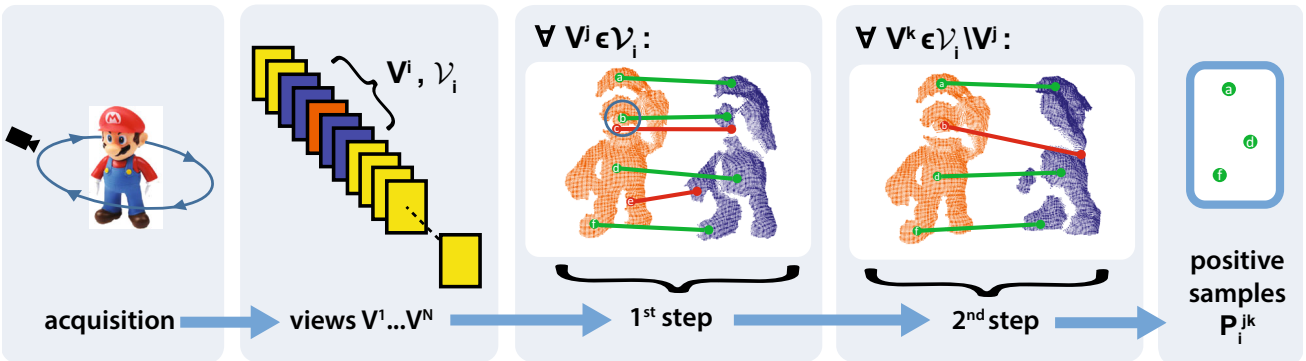

Fig. 1 From *left to right* if not provided within the dataset, a set of calibrated 2.5D views is attained by simulating a 3D sensor in N uniformly distributed vantage points around the object; for each view V^i, those other views exhibiting a sufficient overlap are selected; correspondences are established 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)

Algorithm 1: Training set for fixed-scale detectors.

Function `GetPositiveSamples(`$\mathcal{V}$, σ, τ, ϵ, ϵ_{NMS}`)`**:**
  input : Calibrated views $\mathcal{V} = \{V^i\}$, descriptor radius σ, overlap threshold τ, radius to accept a match ϵ, radius to simulate non-maxima suppression ϵ_{NMS}
  output: Set of positive training samples P
  $P \leftarrow \varnothing$;
  foreach $V^i \in \mathcal{V}$ **do**
    $D^i_p \leftarrow$ descriptor at point $p \in V^i$;
  end
  foreach $V^i \in \mathcal{V}$ **do**
    $P_i \leftarrow \varnothing$;
    $\mathcal{V}^i \leftarrow \{V^j \in \mathcal{V} \setminus V^i | V^j \cap V^i \geq \tau\}$;
    foreach $V^j \in \mathcal{V}^i$ **do**
      $dists \leftarrow \varnothing$;
      `// First step`
      foreach $p \in V^i$ **do**
        $q \leftarrow$ point $\in V^j$ with most similar descriptor to D^i_p;
        $dist^{p,q} \leftarrow$ distance between descriptors at p and q;
        $dists \leftarrow dists \cup \{ dist^{p,q} \}$;
      end
      sort $dists$ in ascending order ;
      $P^j_i \leftarrow \varnothing$;
      while *$dists$ is not empty* **do**
        $dist^{p,q} \leftarrow$ first element of $dists$;
        express q in V^i reference system ;
        if `EuclideanDistance(`p, q`)` $< \epsilon$ **then**
          $P^j_i \leftarrow P^j_i \cup \{ p \}$;
          foreach *neighbor* $p' \in V^i$ *within radius* ϵ_{NMS} *from* p **do**
            remove $dist^{p',q}$ from $dists$;
          end
        end
      end
      `// Second step`
      $\tilde{P}^j_i \leftarrow \varnothing$;
      foreach $V^k \in \mathcal{V}^i \setminus \{V^j\}$ **do**
        $P^{jk}_i \leftarrow \varnothing$;
        foreach $p \in P^j_i$ **do**
          express p in V^k reference system ;
          if *nearest neighbor of* p *is closer than* ϵ **then**
            $q \leftarrow$ point $\in V^k$ with most similar descriptor to D^i_p;
            if `EuclideanDistance(`p, q`)` $< \epsilon$ **then**
              $P^{jk}_i \leftarrow P^{jk}_i \cup \{ p \}$;
            end
          end
        end
        $\tilde{P}^j_i \leftarrow \tilde{P}^j_i \cup P^{jk}_i$;
      end
      $P_i \leftarrow P_i \cup \tilde{P}^j_i$;
    end
    $P \leftarrow P \cup P_i$;
  end
  return P ;

Function `TrainingSetFixedScale(`$\mathcal{V}$, σ, τ, ϵ, ϵ_{NMS}, ϵ_{neg}`)`**:**
  input : Calibrated views $\mathcal{V} = \{V^i\}$, descriptor radius σ, overlap threshold τ, radius to accept a match ϵ, radius to simulate non-maxima suppression ϵ_{NMS}, radius to sample negative samples ϵ_{neg}
  output: Sets of positive and negative training samples P, N
  $P =$ `GetPositiveSamples(`$\mathcal{V}$, σ, τ, ϵ, ϵ_{NMS}`)` ;
  `// Negative samples`
  $N \leftarrow | P |$ random points from $\mathcal{V} \setminus P$, with distance at least ϵ_{neg} from each other;
  return P, N ;

on the specific viewpoint change between V^i and V^j. Therefore, we refine the list of candidates by keeping only those points that can be matched correctly also in other views overlapping with V^i. More precisely, for each $V_k \in \mathcal{V}^i \setminus \{V^j\}$, we obtain a refined set of positive samples, referred to as P_i^{jk}, by retaining only 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. To compute P_i^{jk}, we first select the points in P_i^j that belong to the area of overlap between V^i and V^k (lines 31–33). For every point in P_i^j that lies in the area of overlap, we check if the match is correct according to the criterion used in the first step: we find the nearest neighbor of D_i^p in the descriptor space among all descriptors computed in V^k and check if it corresponds to the same 3D point of p (lines 34–37). If the match is correct, we insert p in P_i^{jk}.

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

$$\tilde{P}_i^j = \bigcup_{\substack{k=1 \\ k \neq i,j}}^{|\mathcal{V}^i|} P_i^{jk} \tag{2}$$

Therefore, any point in P_i^j, i.e. a point in V^i that gets correctly matched in V^j, is also in 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

$$P_i = \bigcup_{\substack{j=1 \\ j \neq i}}^{|\mathcal{V}^i|} \tilde{P}_i^j \tag{3}$$

Finally, the set of positive samples extracted from the dataset, P, is given by the union of all the positive samples obtained by its N views

$$P = \bigcup_{i=1}^{N} P_i \tag{4}$$

To obtain the negative samples, we randomly sample from all the views a set of points which have not been included in P. Upon sampling a new negative from a view, we also remove from the set of candidate negatives all its neighbours lying within a distance threshold ϵ_{neg}, so to cover all areas unfavorable to surface matching by the given descriptor. We use the threshold ϵ_{neg} as the main parameter to create a globally balanced training set featuring negative samples gathered from all the views belonging to the dataset.

3.2 Validation of the Training Set

Figure 2 depicts exemplar positive and negative training samples extracted from two views of two different objects. Interestingly, positive samples are unevenly distributed across surfaces and do not necessarily appear on the intuitively prominent structures, such as e.g., tail of the top object or the knees and elbows of the bottom 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. Conversely, as exemplified in the close-up views of Fig. 2, our method can sift-out automatically certain salient local geometries that, given the chosen descriptor, turn out both distinctive as well as resilient to viewpoint changes. Among other structures unlikely selected as positive samples by our procedure are similar salient patches due to symmetries and repetitive patterns. Indeed, such similar structures may be easily mismatched when co-occurring within the area of overlap between the views. The adaptive-scale formulation described in Sect. 5 helps selecting similar salient patches located nearby which, though ambiguous at a small scale, turn out discriminative when considering more context.

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 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

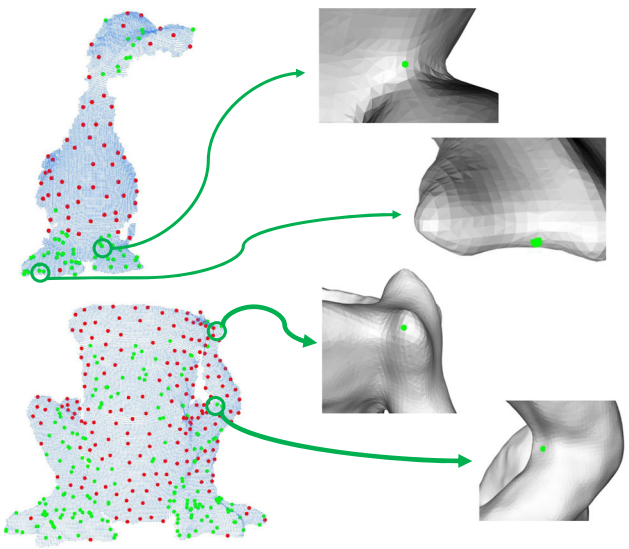

Fig. 2 Exemplar positive (*green*) and negative (*red*) training samples obtained by the proposed method on two views dealing with different objects using SHOT (Salti et al. 2014) as the given 3D descriptor (Color figure online)

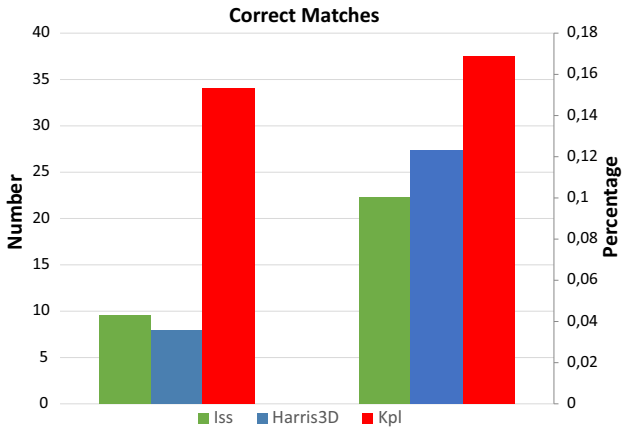

Fig. 3 Number and percentage of correct matches obtained by computing the SHOT descriptor (Salti et al. 2014) on the regions selected by Harris3D, ISS and our proposed approach (referred to as KPL)

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 Sect. 3.1. 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 Sect. 3.1, 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 (Zhong 2009) and Harris3D (Aldoma et al. 2012a).

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 we rely on SHOT (Salti et al. 2014) 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 correct correspondences obtained on 35 views extracted from all the models belonging to the *Kinect* dataset (Salti et al. 2014). Results in Fig. 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.

[1] www.pointclouds.org.

4 Design of the Classifier

We rely on Random Forest (Breiman 2001) to learn the 3D keypoint detector from the sets of positive and negative training samples collected through the procedure described in Sect. 3.1. Indeed, Random Forests have been employed quite successfully to tackle a number of computer vision problems (Criminisi et al. 2012), including 3D keypoint detection (Teran and Mordohai 2014). 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 Sect. 5.

As far as features are concerned, we propose features inspired by SHOT (Salti et al. 2014) but able to achieve rotation invariance without computing a Local Reference Frame (LRF). Figure 4 provides a graphical overview of the feature computation process, while in Algorithm 2 we report its pseudo-code. 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 local RF 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

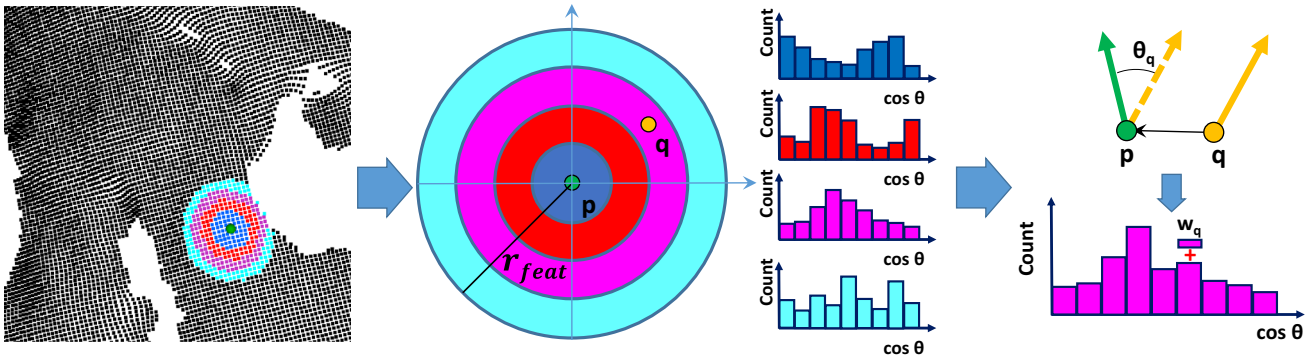

Fig. 4 Overview of the feature computation process. In the example, $N_r = 4$ subdivisions along the radial coordinate are used to split the spherical support around p into equally many sectors (for ease of visualization, a 2D representation of the 3D spherical support is portrayed). For each sector (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)

Sect. 3.1. 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. $s(p) \geq s_{min}$.

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 (Criminisi et al. 2012; Holzer et al. 2012). 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 (Salti et al. 2014), MeshHoG (Zaharescu et al. 2009), ROPS (Guo et al. 2013a). 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.

5 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.

As described in Algorithm 3, to define the training set, we first extract positive samples from the models at each predefined scale following the same procedure as described in Sect. 3.1. 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. 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 Sect. 3.1. We set ϵ_{neg} such that the

Algorithm 2: Compute features to classify a keypoint.

```
Function ComputeFeature(p, r_feat, N_r, N_b):
    input : Point p to be classified, radius r_feat, number of radial shells N_r, number of bins N_b
    output: Feature vector feat
    feat ← a vector of zeros of length N_r * N_b ;
    n_p ← normal at p;
    Q ← set of neighbors of p within radius r_feat;
    foreach q ∈ Q do
        // Compute data for q
        n_q ← normal at q;
        dist ← EuclideanDist(p,q);
        cosine ← DotProduct(n_p, n_q) ;
        // Compute index into feat
        continuous_shell_index ← dist/r_feat * N_r ;
        continuous_cosine_index ← (cosine + 1)/2 * N_b ;
        // Add interval end to last quantization interval
        if continuous_shell_index ≥ N_r then
            continuous_shell_index = N_r − ϵ
        end
        if continuous_cosine_index ≥ N_b then
            continuous_cosine_index = N_b − ϵ
        end
        shell_index ← ⌊continuous_shell_index⌋ ;
        cosine_index ← ⌊continuous_cosine_index⌋ ;
        index ← shell_index * N_b + cosine_index;
        // Bilinear interpolation
        dist_from_radial_bin_center ← continuous_shell_index − (shell_index + 0.5);
        dist_from_cosine_bin_center ← continuous_cosine_index − (cosine_index + 0.5);
        radial_weight ← 1 - Abs(dist_from_radial_bin_center) ;
        cosine_weight ← 1 - Abs(dist_from_cosine_bin_center) ;
        // Update feat
        if continuous_shell_index ≥ 0.5 and continuous_shell_index ≤ N_r − 0.5 then
            if dist_from_radial_bin_center < 0 then
                rad_index ← (shell_index − 1) * N_b + cosine_index;
            else
                rad_index ← (shell_index + 1) * N_b + cosine_index;
            end
            feat [rad_index ] ← feat [rad_index ] + (1 − radial_weight) ;
            feat [index ] ← feat [index ] + radial_weight;
        else
        end
        feat [index ] ← feat [index ] + 1 ;
        if continuous_cosine_index ≥ 0.5 and continuous_cosine_index ≤ N_b − 0.5 then
            if dist_from_cosine_bin_center < 0 then
                cos_index ← shell_index * N_b + cosine_index − 1 ;
            else
            end
            cos_index ← shell_index * N_b + cosine_index + 1;
            feat [cos_index ] ← feat [cos_index ] + (1 − cosine_weight) ;
            feat [index ] ← feat [index ] + cosine_weight;
        else
        end
        feat [index ] ← feat [index ] + 1 ;
    end
    // Normalize histogram of each shell
    for i= 0 ...(N_r − 1) do
        entries of feat of shell i ← entries of feat of shell i / EuclideanNorm (entries of feat of shell i) ;
    end
    return feat
```

Algorithm 3: Compute training set for adaptive-scale detectors.

Function `TrainingSetAdaptiveScale(`$\mathcal{V}$, Σ, τ, ϵ, ϵ_{NMS}, ϵ_{neg}`)`**:**
- **input** : Views $\mathcal{V} = \{V^i\}$, set of scales Σ, overlap threshold τ, radius to accept a match ϵ, radius to simulate non-maxima suppression ϵ_{NMS}, radius to sample negative sample ϵ_{neg}
- **output**: Set of negative training samples and positive training samples for each scale samples

foreach *scale* $\sigma \in \Sigma$ **do**
  `// For each sample compute also the average distance between descriptors` d_{avg}
  samples [σ], d_{avg} [σ] = `GetPositiveSamples(`$\mathcal{V}$, σ, τ, ϵ, ϵ_{NMS}`)` ;
end
`// First refinement`
foreach *view* $V^i \in \mathcal{V}$ **do**
  foreach *point* $p \in V^i$ **do**
    S_point $\leftarrow$ { σ | $p \in$ samples [σ] } ;
    d_avg_point $\leftarrow$ { d_{avg} [σ] [p] | $\sigma \in$ S_point } ;
    if | S_point | > 1 **then**
      scale_to_keep $\leftarrow \text{argmin}_{\sigma \in \text{S\_point}}$ d_avg_point[σ] ;
      foreach $\sigma \in$ S_point $\setminus$ / scale_to_keep / **do**
        remove p from samples [σ] ;
      end
    end
  end
end
`// Second refinement`
samples_per_class $\leftarrow \min_{\sigma \in \Sigma} |\text{samples}[\sigma]|$;
foreach $\sigma \in \Sigma$ **do**
  if |samples[σ]| > samples_per_class **then**
    sort samples[σ] ascending according to $d_{avg}[\sigma]$;
    remove last |samples[σ]| − samples_per_class points ;
  end
end
`// Negative samples`
sample samples_per_class negative samples from $\mathcal{V} \setminus$ samples with radius ϵ_{neg} and add them to samples;
return samples ;

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 (Sect. 4 and Algorithm 2) 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, $s(p) \geq s_{min}$, and it turns out a local maximum of the saliency while consider-

ing 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 (Sect. 6.1). In practice the runtime 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 Tombari et al. (2013), 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.

6 Experimental Results

As this work deals with improving feature matching performance by learning a descriptor-specific detector, rather than gathering descriptor-agnostic repeatability measurements [as, e.g., in Tombari et al. (2013)] 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 (Steder et al. 2011), alongside with the algorithm proposed by Mian et al. (2010) and referred to as KPQ in Tombari et al. (2013), which, together with ISS, turned out particularly effective in the experiments carried out in Tombari et al. (2013). 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 Mian et al. 2010) and MeshDoG (Zaharescu et al. 2009). 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 Tombari et al. (2013): the *Laser Scanner* dataset introduced by Mian et al. (2010), the *Random Views* dataset, based on the Stanford 3D scanning repository,[3] and the *Kinect* dataset proposed in Salti et al. (2014). The fourth, referred to here as *Venezia 3D* dataset,[4] was introduced in Rodolà et al. (2013). 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 Aldoma et al. (2012b) 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 either known beforehand or acquired at initialization time and scenes being unseen data.

As for descriptors, we use SHOT (Salti et al. 2014), which has been reported to obtain good results in a number of comparative evaluations addressing object recognition (Alexandre 2012; Aldoma et al. 2012a), 3D object classification and retrieval (Proença et al. 2013; Wohlkinger et al. 2012), semantic segmentation of point clouds (Behley et al. 2012), localization of medical landmarks (Sukno et al. 2012). Moreover, we learn to detect good 3D keypoints for Spin Images (Johnson and Hebert 1999) and FPFH (Rusu et al. 2009), 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 paper. We rely on the implementation available in PCL for all descriptors.

Tables 1 and 2 report the parameters used at test and training time on each dataset in the experiments pursuing

[2] www.opencv.org.

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

[4] http://www.dsi.unive.it/~rodola/data.html.

Table 1 Parameters related to fixed-scale detection experiments

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

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

Table 2 Parameters dealing with adaptive scale detection experiments

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

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

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.

6.1 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 (2001) suggests to grow a tree until just 1 sample is left in a node, whereas Criminisi et al. (2012) 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. Figure 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. 100 K positive samples compared to the 8000 positive samples

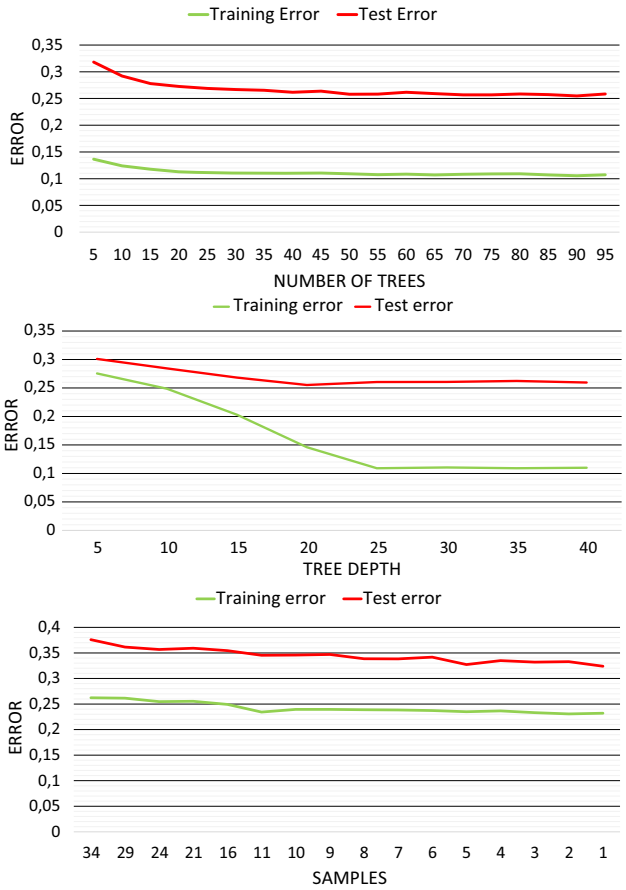

Fig. 5 Hyperparameter optimization on the Kinect dataset

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

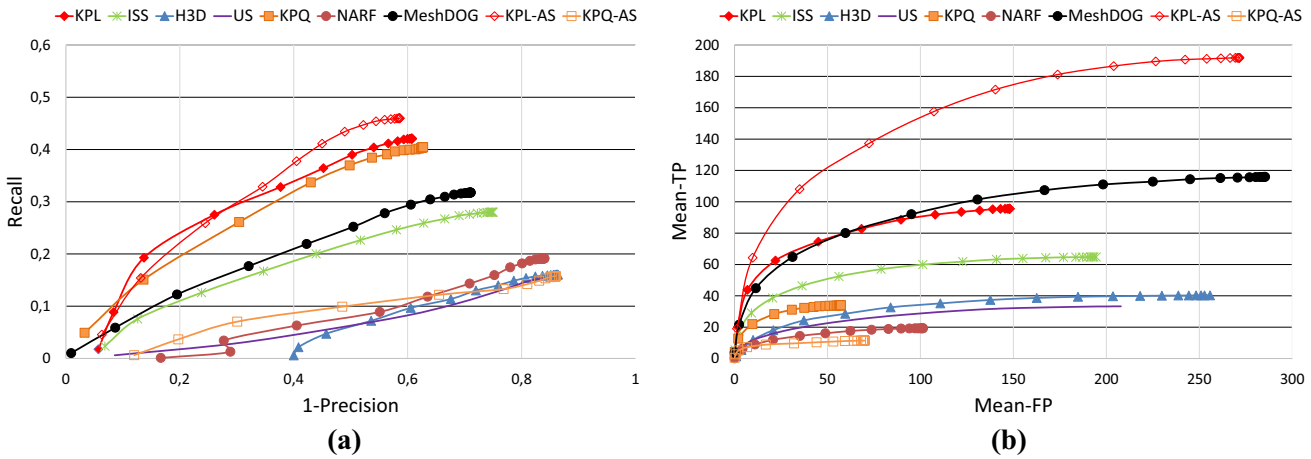

Fig. 6 Results provided on the *Laser Scanner* dataset by the keypoint detector learned for SHOT. Precision-Recall curves (**a**), Mean True Positives versus Mean False Positives (**b**)

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.

6.2 Results on the *Laser Scanner* Dataset

This dataset includes 4 full-3D models as well as 50 scenes in which models occlude each other and the presence of objects not belonging to the model set creates some clutter.

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 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. 6a. According to the same methodology, we also compute Mean True Positive versus Mean False Positive curves (e.g., Fig. 6b), as proposed in Tombari et al. (2013) 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.

Figure 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 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

[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.

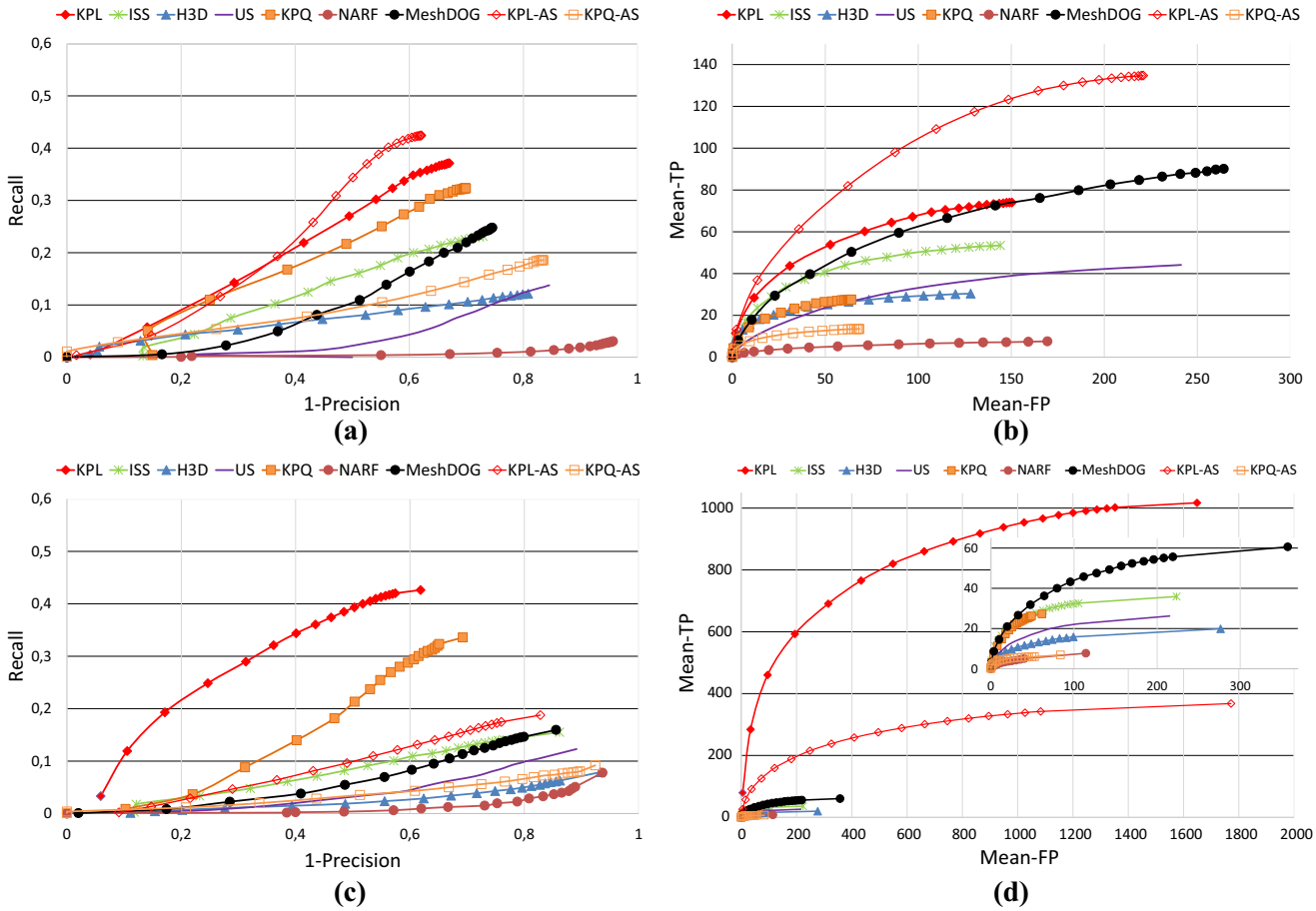

Fig. 7 Results provided on the *Laser Scanner* dataset by the keypoint detector learned for Spin Images (**a**, **b**) and FPFH (**c**, **d**). Precision-Recall curves (**a**–**c**), Mean True Positives versus Mean False Positives (**b**–**d**)

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. Figure 6b 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.

Figure 7a, 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. 7c, d show that, when using FPFH, KPL and KPL-AS outperform, respectively, all fixed-scale and adaptive-scale detectors. Overall, the results in Fig. 7 vouch for the generality of the methodology proposed in this paper. 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. 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. Fig. 8a, 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. Fig. 8c, 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.

6.3 Transfer Learning on the *Random Views* Dataset

The *Random Views* dataset comprises 6 full-3D models alongside with 36 scenes in which models occlude each other but there is no clutter. Both models and scenes are highly detailed, with synthetic noise added to scenes. Here we con-

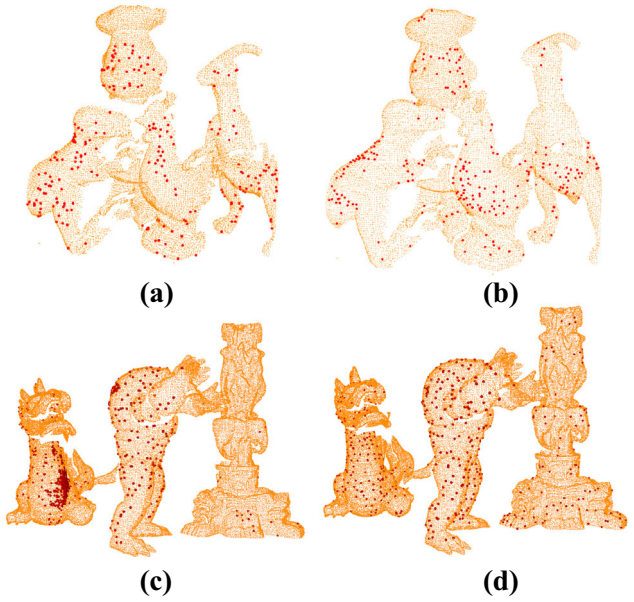

Fig. 8 Exemplar keypoints extracted by the detectors learned for SHOT (**a**, **c**), Spin Images (**b**) and FPFH (**d**) on scenes of the *Laser Scanner* dataset (**a**–**b**) and *Random Views* dataset (**c**–**d**)

sider the scenes corrupted by Gaussian noise with $\sigma = 0.1$ mesh resolution units.

The performance assessment protocol is the same as described in Sect. 6.2. 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 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. 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 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. 9b).

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. 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, Figs. 9 and 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.

6.4 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 (Sect. 6.2) or transferring a learned detector from one dataset to another (Sect. 6.3), 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 Sects. 3 and 4, 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 Sect. 5 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* (Taati et al. 2007) and *Venezia 3D*]. Figure 11a–d shows four of the models used to train the *universal detector*, one for each dataset. About 1,250,000 positive and negative samples were used to create

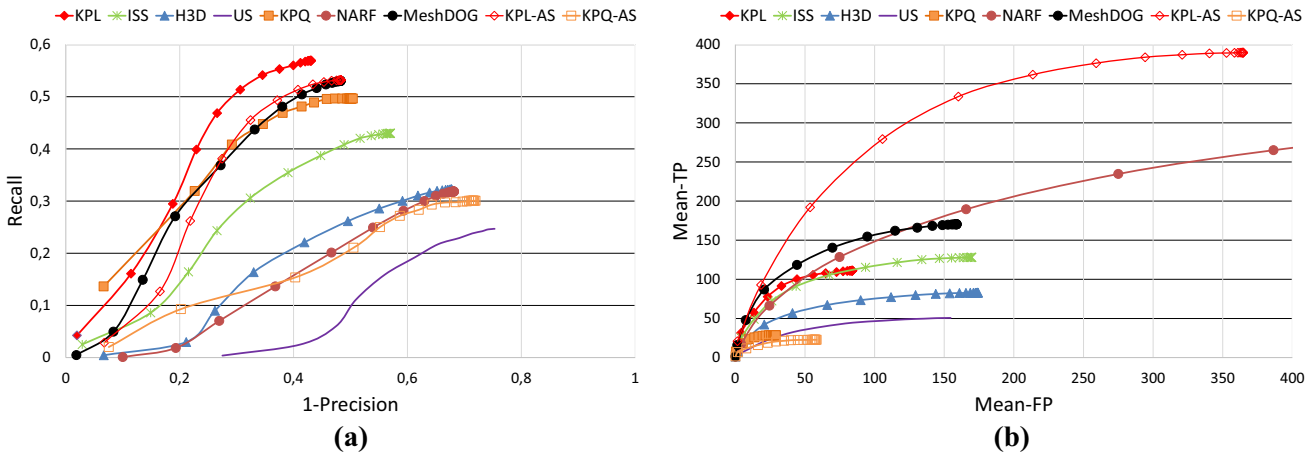

Fig. 9 Results on *Random Views* by the keypoint detector learned for SHOT on *Laser Scanner*. Precision-Recall curves (**a**), Mean True Positives versus Mean False Positives (**b**)

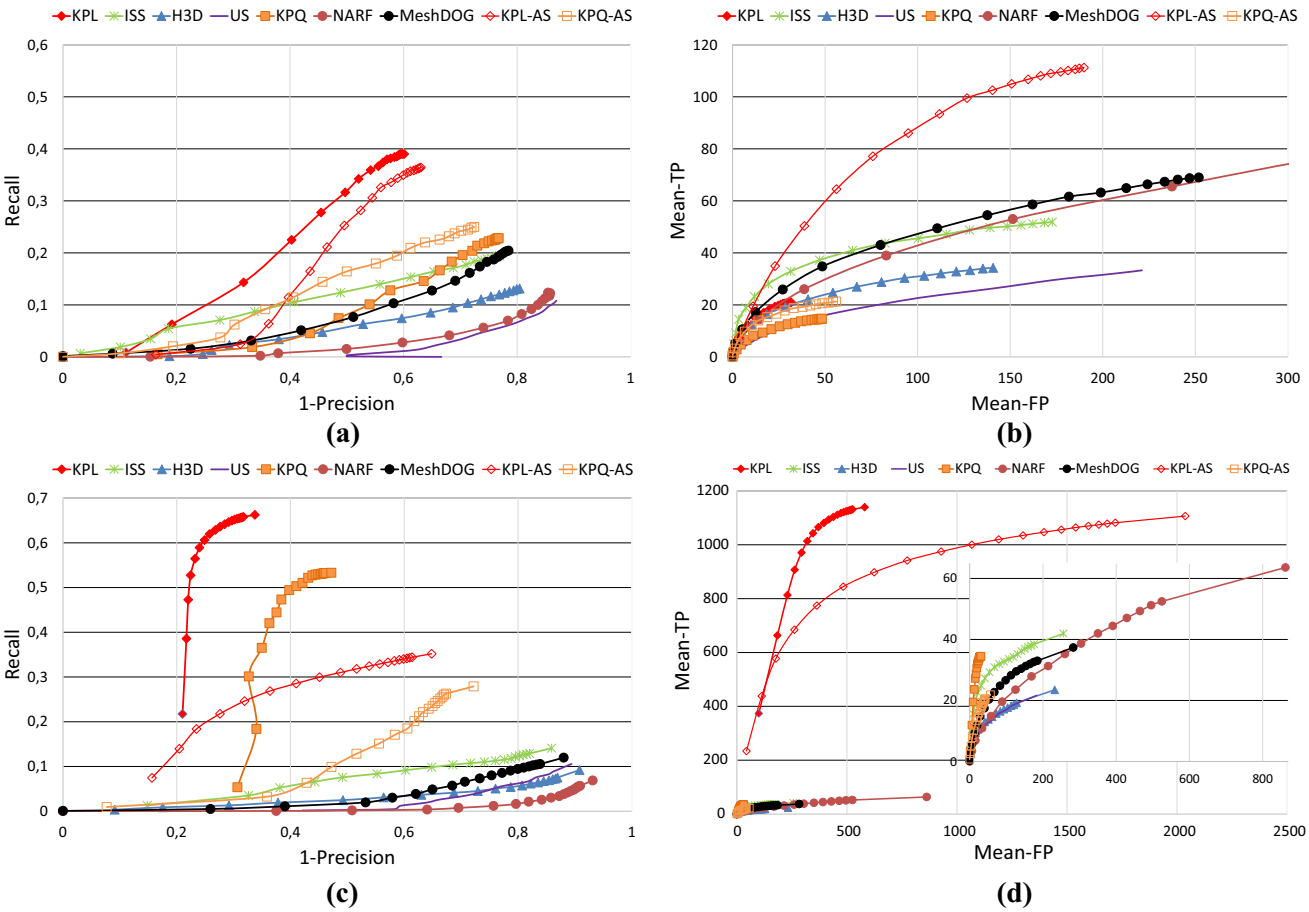

Fig. 10 Results on *Random Views* by the keypoint detector learned for Spin Images (**a**, **b**) and FPFH (**c**, **d**) on *Laser Scanner*. Precision-Recall curves (**a**–**c**), Mean True Positives versus Mean False Positives (**b**–**d**)

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

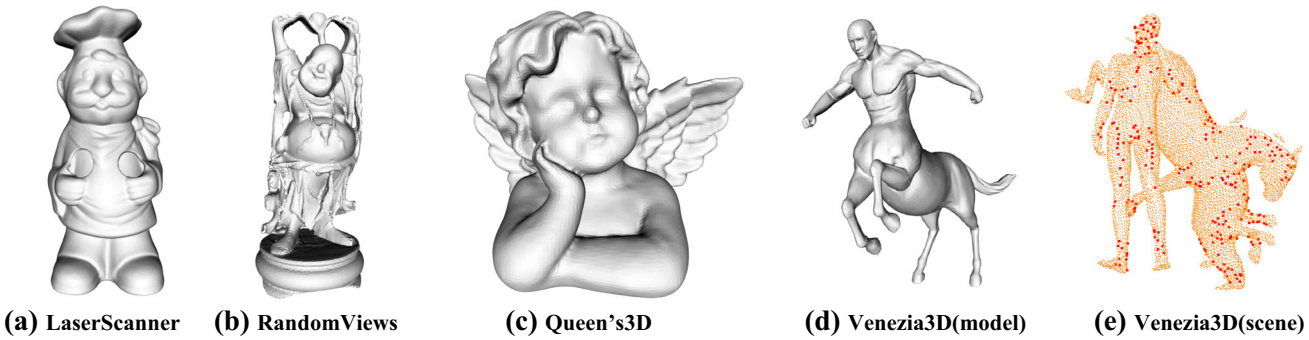

Fig. 11 Universal detector: some models used to for training (**a**–**d**) and keypoints detected on a scene of the *Venezia 3D* dataset (**e**)

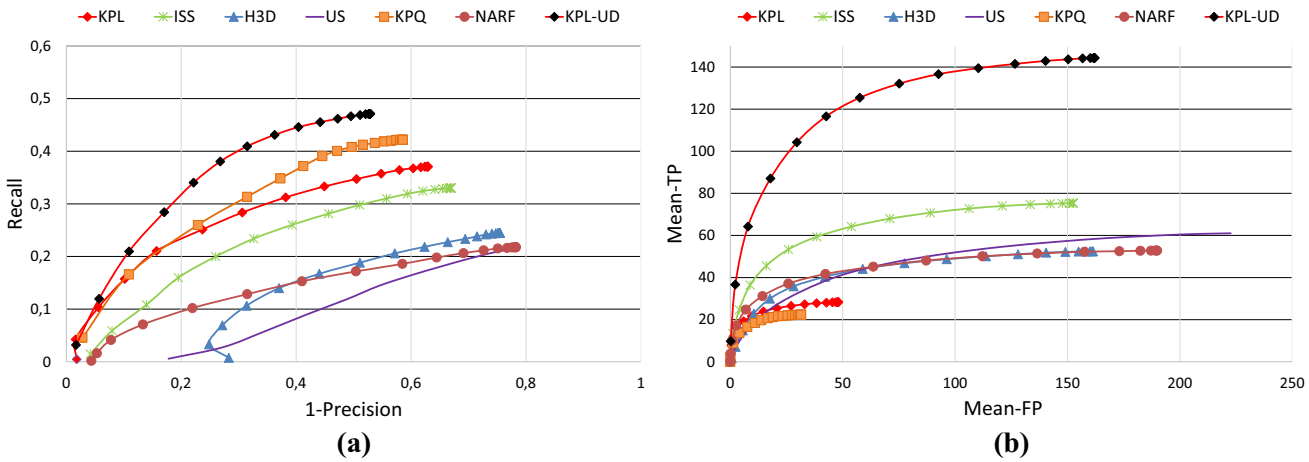

Fig. 12 Results on *Venezia 3D* for the universal detector trained for SHOT. Precision-Recall curves (**a**), Mean True Positives versus Mean False Positives (**b**)

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. 12, in addition a qualitative example is also shown in Fig. 11e. 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. 12 we compare the universal detector (denoted 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.

6.5 *Kinect* Dataset

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 (Salti et al. 2014). Accordingly, first keypoints are extracted and described from all model views. Then, descriptors are extracted from scenes, both at the locations of model key-

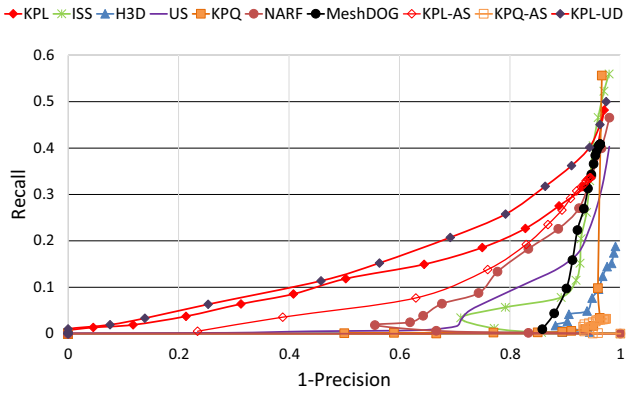

Fig. 13 Results on the *Kinect* dataset by the keypoint detector learned for SHOT

points as well as from clutter. Every descriptor computed from the scene is then matched against the set of model descriptors by the ratio criterion (Lowe 2004) and checked for geometric correctness. Varying the threshold of the ratio test, allows for drawing Precision-Recall curves as in Fig. 13.

The *Kinect* dataset is particularly challenging, such that, as also found in a recent experimental evaluation (Guo et al. 2016), the performance of the feature matching pipelines decreases dramatically with respect to previous datasets [see also Fig. 5 in Guo et al. (2016)] It is worth noticing how, in the results dealing with matching SHOT descriptors reported in Fig. 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 paper.

Given that the training set for the *Kinect* dataset consists of fewer points compared to the other datasets, e.g., ∼8000 positive samples versus ∼100,000 in *Laser Scanner*, and motivated by the results obtained with the lidar-based *universal detector* (Sect. 6.4), 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* (Aldoma et al. 2014) datasets, both including real 2.5D calibrated views acquired by low-cost consumer depth camera, so as to obtain ∼70,000 positive training samples and to train the Random Forest according to the standard methodology and parameters (Table 2). Figure 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 Sect. 6.4 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.

7 Conclusions and Future Work

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 (Salti et al. 2014), the proposed keypoint learning methodology is by no means bound to work with this specific descriptor only. Indeed, it may be leveraged successfully with any 3D descriptor, as demonstrated by the experiments dealing with other prominent descriptors such as FPFH (Rusu et al. 2009) and Spin Images (Johnson and Hebert 1999). To facilitate usage of our method and development of learned detectors for other 3D descriptors and datasets, we publicly released both the trained models as well and the source code to train and test detectors.[6]

A promising direction for future work concerns deformable shape matching (Sun et al. 2009; Ovsjanikov et al. 2011). 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 Sun et al. (2009), 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 Ovsjanikov et al. (2011) 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

[6] http://github.com/CVLAB-Unibo/Keypoint-Learning.

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 the principle advocated in this paper is novel also to the domain of image matching, another worthwhile topic for further investigation deals with learning to detect good keypoints for a given local image descriptor. This would entail gathering as positive samples those pixels that withstand matching alongside with transformations induced by camera pose variations and/or major nuisances such as illumination changes or diversity in the image sensing modality.

References

Aldoma, A., Fäulhammer, T., & Vincze, M. (2014). Automation of "ground truth" annotation for multi-view RGB-D object instance recognition datasets. In *Proceedings of international conference on intelligent robots and systems (IROS)*.

Aldoma, A., Marton, Z., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., et al. (2012a). Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics and Automation Magazine (RAM)*, *19*(3), 80–91.

Aldoma, A., Tombari, F., Di Stefano, L., & Vincze, M. (2012b). A global hypotheses verification method for 3d object recognition. In *European conference on computer vision (ECCV), Lecture Notes in Computer Science* (Vol. 7574, pp. 511–524). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-33712-3_37.

Alexandre, L. (2012). 3d descriptors for object and category recognition: A comparative evaluation. In *IROS workshop on color-depth camera fusion in robotics*.

Bariya, P., & Nishino, K. (2010). Scale-hierarchical 3d object recognition in cluttered scenes. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1657–1664. doi:10.1109/CVPR.2010.5539774.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, *110*(3), 346–359.

Behley, J., Steinhage, V., & Cremers, A. (2012). Performance of histogram descriptors for the classification of 3d laser range data in urban environments. In *International conference on robotics and automation*.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/A:1010933404324.

Castellani, U., Cristani, M., & Fantoni, S. (2008). Sparse points matching by combining 3D mesh saliency with statistical descriptors. In *Proceedings of computer graphics forum*, pp. 643–652.

Creusot, C., Pears, N., & Austin, J. (2013). A machine-learning approach to keypoint detection and landmarking on 3d meshes. *International Journal of Computer Vision*, *102*(1–3), 146–179. doi:10.1007/s11263-012-0605-9.

Criminisi, A., Shotton, J., & Konukoglu, E. (2012). Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision, 7*(2–3), 81–227. http://research.microsoft.com/apps/pubs/default.aspx?id=158806.

Dutagaci, H., Cheung, C., & Godil, A. (2012). Evaluation of 3d interest point detection techniques via human-generated ground truth. *The Visual Computer*, *28*(9), 901–917. doi:10.1007/s00371-012-0746-4.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Kwok, N. M. (2016). A comprehensive performance evaluation of 3d local feature descriptors. *International Journal of Computer Vision*, *116*(1), 66–89.

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., & Wan, J. (2013a). Rotational projection statistics for 3d local surface description and object recognition. *International Journal of Computer Vision*, *105*(1), 63–86.

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., & Wan, J. (2013b). Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. In *8th international conference on computer graphics theory and applications (GRAPP)*.

Hartmann, W., Havlena, M., & Schindler, K. (2014). Predicting matchability. In *2014 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 9–16. doi:10.1109/CVPR.2014.9.

Holzer, S., Shotton, J., & Kohli, P. (2012). Learning to efficiently detect repeatable interest points in depth data. In *2012 IEEE European conference on computer vision (ECCV)*.

Johnson, A., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(5), 433–449.

Leutenegger, S., Chli, M., & Siegwart, R. (2011). BRISK: Binary robust invariant scalable keypoints. In *2011 IEEE international conference on computer vision (ICCV)*, pp. 2548–2555. doi:10.1109/ICCV.2011.6126542.

Li, Y., Wang, S., Tian, Q., & Ding, X. (2015). A survey of recent advances in visual feature detection. *Neurocomputing*, *149 Part B*, 736–751. http://www.sciencedirect.com/science/article/pii/S0925231214010121

Lin, X., Zhu, C., Zhang, Q., & Liu, Y. (2016). 3d keypoint detection based on deep neural network with sparse autoencoder. arXiv preprint arXiv:1605.00129.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Mian, A. S., Bennamoun, M., & Owens, R. A. (2010). On the repeatability and quality of keypoints for local feature-based 3D object retrieval from cluttered scenes. *International Journal of Computer Vision*, *89*(2–3), 348–361.

Ovsjanikov, M., Huang, Q., & Guibas, L. (2011). A condition number for non-rigid shape matching. In *Eurographics symposium on geometry processing*, Vol. 30.

Proença, P. F., Gaspar, F., & Dias, M. S. (2013). Good appearance and shape descriptors for object category recognition. In *Advances in visual computing. Lecture notes in computer science* (Vol. 8033, pp. 385–394). Springer: Berlin, Heidelberg.

Rodolà, E., Albarelli, A., Bergamasco, F., & Torsello, A. (2013). A scale independent selection process for 3D object recognition in cluttered scenes. *International Journal of Computer Vision*, *102*(1–3), 129–145. doi:10.1007/s11263-012-0568-x.

Rosten, E., Porter, R., & Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(1), 105–119.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *IEEE international conference on computer vision*, pp. 2564–2571. http://doi.ieeecomputersociety.org/10.1109/ICCV.2011.6126544.

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. In *International conference*

on robotics and automation, pp. 3212–3217. doi:10.1109/ROBOT.2009.5152473.

Salti, S., Tombari, F., & Di Stefano, L. (2014). SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, *125*, 251–264. doi:10.1016/j.cviu.2014.04.011, http://www.sciencedirect.com/science/article/pii/S1077314214000988.

Salti, S., Tombari, F., Spezialetti, R., & Di Stefano, L. (2015). Learning a descriptor-specific 3D keypoint detector. In *The IEEE international conference on computer vision (ICCV)*, pp. 2318–2326.

Shi, J., & Tomasi, C. (1994). Good features to track. In *1994 IEEE conference on computer vision and pattern recognition (CVPR'94)*, pp. 593–600.

Steder, B., Rusu, R. B., Konolige, K., & Burgard, W. (2011). Point feature extraction on 3d range scans taking into account object boundaries. In *2011 IEEE international conference on robotics and automation (ICRA)* (pp. 2601–2608). IEEE.

Strecha, C., Lindner, A., Ali, K., & Fua, P. (2009). Training for task specific keypoint detection. In J. Denzler, G. Notni, & H. Se (Eds.), *Pattern recognition: Lecture notes in computer science* (Vol. 5748, pp. 151–160). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-03798-6_16.

Sukno, F., Waddington, J., & Whelan, P. (2012). Comparing 3d descriptors for local search of craniofacial landmarks. In *International symposium on visual computing (ISVC)*.

Sun, J., Ovsjanikov, M., & Guibas, L. (2009). A concise and provably informative multi-scale signature based on heat diffusion. In *Eurographics symposium on geometry processing*, Vol. 28.

Taati, B., Bondy, M., Jasiobedzki, P., & Greenspan, M. (October 2007). Variable dimensional local shape descriptors for object recognition in range data. In *Proceedings of the 11th IEEE international conference on computer vision*; Rio de Janeiro, Brazil, Vol. 1421, p. 18.

Teran, L., & Mordohai, P. (2014). 3D interest point detection via discriminative learning. In *ECCV 2014. Lecture notes in computer science* (Vol. 8689, pp. 159–173). Springer. doi:10.1007/978-3-319-10590-1_11.

Tombari, F., Salti, S., & DiStefano, L. (2013). Performance evaluation of 3d keypoint detectors. *International Journal of Computer Vision*, *102*(1–3), 198–220. doi:10.1007/s11263-012-0545-4.

Tuytelaars, T., & Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, *3*(3), 177–280.

Verdie, Y., Yi, K. M., Fua, P., & Lepetit, V. (2015). TILDE: A temporally invariant learned DEtector. In *Proceedings of the computer vision and pattern recognition*.

Wohlkinger, W., Aldoma, A., Rusu, R., & Vincze, M. (2012). 3dnet: Large-scale object class recognition from cad models. In *International conference on robotics and automation (ICRA)*.

Zaharescu, A., Boyer, E., Varanasi, K., & Horaud, R. (2009). Surface feature detection and description with applications to mesh matching. In *Proceedings of international conference on computer vision and pattern recognition (CVPR)*, pp. 373–380.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *Proceedings of international conference on computer vision workshops*, pp. 1–8.