Learning a Descriptor-specific 3D Keypoint Detector

Samuele Salti
University of Bologna
samuele.salti@unibo.it

Federico Tombari
University of Bologna
federico.tombari@unibo.it

Riccardo Spezialetti
University of Bologna
riccardo.spezialetti@studio.unibo.it

Luigi Di Stefano
University of Bologna
luigi.distefano@unibo.it

Abstract

Keypoint detection represents the first stage in the majority of modern computer vision pipelines based on automatically established correspondences between local descriptors. However, no standard solution has emerged yet in the case of 3D data such as point clouds or meshes, which exhibit high variability in level of detail and noise. More importantly, existing proposals for 3D keypoint detection rely on geometric saliency functions that attempt to maximize repeatability rather than distinctiveness of the selected regions, which may lead to sub-optimal performance of the overall pipeline. To overcome these shortcomings, we cast 3D keypoint detection as a binary classification between points whose support can be correctly matched by a predefined 3D descriptor or not, thereby learning a descriptor-specific detector that adapts seamlessly to different scenarios. Through experiments on several public datasets, we show that this novel approach to the design of a keypoint detector represents a flexible solution that, nonetheless, can provide state-of-the-art descriptor matching performance.

1. Introduction

Detection of repeatable and distinctive keypoints is a fundamental task in modern computer vision when dealing with both images as well as 3D data. Keypoint detection in images finds applications in image retrieval, object detection and recognition, object and camera tracking, camera calibration and image registration, among the others. Keypoints from 3D data are useful to deal with several shape matching tasks, such as point cloud registration, 3D object recognition and pose estimation, shape retrieval and shape classification.

The standard approach in 2D and 3D computer vision involves defining keypoints by maximization of a handcrafted local saliency function [25, 13, 24]. Popular approaches for 3D keypoint detection based on such functions can be categorized into fixed-scale and adaptive-scale, some of these methods exhibiting good repeatability as well as the ability to identify regions that may be matched correctly by 3D descriptors [24]. For instance, among fixed-scale approaches, Intrinsic Shape Signatures (ISS) [29] is a widely used fast and effective proposal; the fixed-scale detector proposed by Mian *et al*. [15] is a slower alternative, particularly robust to point density variations. Among adaptive-scale detectors, MeshDoG [28] is an extension of the popular Difference of Gaussian detector [14] to scalar functions defined over a manifold approximated by a mesh; the adaptive-scale variant proposed in [15] maximizes the saliency function across scales to adaptively define the point neighborhood size; [4] is also aimed at extending the DoG operator to meshes.

However, when deploying such detectors, the saliency is unrelated to the quality of the description to be later computed at the point. Indeed, detectors are usually designed to maximize repeatability rather than distinctiveness of the regions, this possibly resulting in sub-optimal matching performance that yield a reduced number of correct correspondences. However, identification of correct correspondences is the ultimate goal of the overall detection/description process, and the importance and effectiveness of pursuing directly such goal already in the detection stage is shown, in video tracking, by the popular work of Shi and Tomasi [20], where ”good” features to detect are defined as those likely to yield correct matches between consecutive frames. Moreover, in the realm of 3D vision, diverse data acquisition modalities provide quite different levels of detail and noise, which may require careful tuning of the detector parameters to obtain meaningful keypoints across diverse datasets, or may even render a detector useless when applied to data that differ from those it was originally conceived for (*e.g*. curvature based detectors are largely affected by the amount of noise in the data).

In this work, we propose to cast detection of 3D keypoints as a binary classification problem between the classes of those points that can or cannot be effectively encoded by a pre-defined 3D descriptor, and also present an effective algorithm to define the required training set. Automatic learning of such classification function holds the potential to adaptively identify those points which are more likely to provide correct correspondences for a particular dataset without being bound to the specific structures found by a hand-crafted detector.

A few researchers investigated the use of machine learning techniques in works related to the problem of keypoint detection. As far as images are concerned, the most important contribution is arguably FAST [16], 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 used in BRISK [12] and ORB [17]. Another line of research has been focused on pruning the keypoints extracted by a standard keypoint detector so to improve the overall performance in a particular task. In [9], 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 detection algorithm, namely DoG. By using a Random Forest [3] to learn such ”matchability”, they show that the approach can improve and speed-up considerably the feature matching stage of a Structure-from-Motion pipeline. Similarly, in [22], the authors show that higher repeatability can be achieved by instructing a WaldBoost classifier to keep, among those extracted by a standard detector, only those points that are known to be useful in a given scenario. As an exemplar application of the method, they demonstrate improved image matching in an urban environment, the classifier learning to focus on stable man-made structures and ignore objects that undergo natural changes such as vegetation and clouds. A different approach is proposed in [26], where the most repeatable DoG keypoints across a set of training images are deployed to define the positive samples used to train a regressor which learns a saliency function able to highlight the same image points under drastic illumination changes caused by changes in weather, season, and time of day.

In 3D computer vision, casting keypoint detection as a binary classification problem has been proposed only when dealing with datasets that already include the definition of the points of interest, such as *e.g.* facial landmarks [5]. Similarly, the authors of [23] proposed to learn a detector from training data instead of hand-engineering a geometric saliency in order to cope with the high variability of structures selected as salient by human users in the benchmark proposed in [7], where the task is indeed not to define generic and repeatable keypoints, but to find exactly the keypoints indicated as salient by human users during the creation of the benchmark. Differently, Holzer *et al.* [10] proposed to speed up and improve the repeatability of curvature-based detectors by learning the saliency function with a Regression Forest that uses, as features, binary depth comparisons.

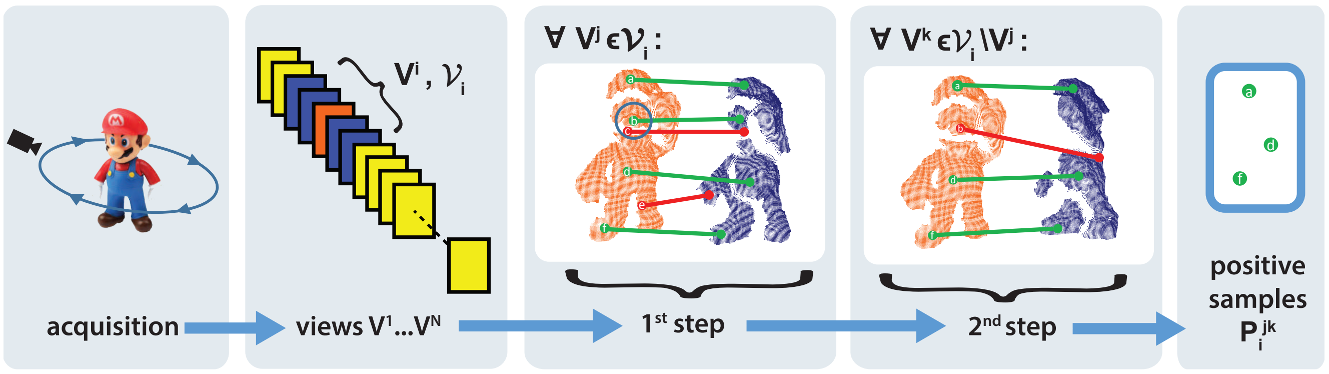
In this paper, instead, we propose to learn a classification function that acts as a keypoint detector aimed at identifying points likely to generate correct matches when encoded by a given descriptor and, accordingly, define a method to automatically generate the data required to train the classifier. Therefore, the definition of the interest points needs not to be available together with the dataset, as in [23, 5], but, peculiarly, it is data-driven and attained automatically based on the ability to match specific descriptors, thereby generating a descriptor-specific keypoint detector. The ability to learn a descriptor-specific detector is particularly relevant in the 3D vision realm, wherein, unlike prominent 2D approaches such as SIFT or SURF, state-of-the-art descriptors [8] [19], [18], [11] still lack a companion detector.

We also point out that, in contrast with the pruning approaches proposed so far for 2D detectors [22, 9], 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 saliency function employed by the selected detector. Moreover, the choice of such basic 3D detector would also turn out problematic as there is not yet an established and generally applicable algorithm for 3D data as it may be considered DoG for 2D images.

The paper is structured as follows. Section 2 provides an overview of the proposed framework to learn a descriptor-specific detector and defines how to create the training set and the experiment that we performed to validate its effectiveness before using it to learn a classification function. Section 3 presents the design of the features extracted to train our classifier and how we use at test-time to extract keypoints on a given cloud. Section 4 presents results on three publicly available datasets and compares our proposal to several state-of-the-art detectors. Section 5 reports on the insights gained while attempting to implement this novel idea for the design of a keypoint detector

2. Training a Descriptor-specific Detector

The main aim of our approach is to learn to detect 3D keypoints that yield highly distinctive 3D descriptors. The definition of the ground-truth for training, and, in particular, of the positive examples, is therefore crucial. We exploit a set of partially overlapping 2.5D views of 3D objects that are of interest in a particular dataset so to define both posi-



acquisition
views V1...VN
V^i, $\mathcal{V}_i$
$\forall V^j \in \mathcal{V}_i$:
a
b
c
d
e
f
1st step
$\forall V^k \in \mathcal{V}_i \setminus V^j$:
2nd step
positive samples P_i^{jk}

Figure 1. Overview of the definition of positive samples, from left to right: if not available, a set of 2.5D views is generated by simulating a sensor in N uniformly distributed views; for each view V^i, its overlapping views are selected; for each overlapping view, matches are established by selecting the nearest neighbor in the descriptor space for each point in V^i; matches are removed if they are wrong matches (point e) or close to a match with more similar descriptors (point c); by using every other overlapping view V^k, the positive set is refined by removing points that do not yield correct matches (point b), because their distinctiveness is not robust enough to changes in viewpoint.

tive and negative samples.

Although our framework is general and, in principle, may be applied to any 3D descriptor, in the following we rely on the SHOT descriptor, given its availability in the open source Point Cloud Library (PCL)[1] and its overall good performance [19]. Moreover, we fix the support of the descriptor and derive a fixed-scale detector that identifies distinctive points at that scale.

2.1. Definition of the training set

Given an object from a 3D dataset, let $\{V^i\}, i = 1, \dots, N$ be the set of its N 2.5D views (see Fig. 1). Should the object be provided in the dataset as a full 3D model, we would obtain 2.5D views by performing synthetic renderings, as done *e.g.* in [2, 1]. For instance, we deploy the method adopted in [1], where views are rendered from the nodes of an icosahedron centered at the centroid of the 3D model.

As illustrated in Fig.1, to identify those points that may be matched most robustly and use them as positive examples of keypoints, we seek the points of the object whose SHOT descriptors computed on different partially overlapping views turn out highly similar.

Algorithmically, we proceed as follows. For each view V^i, we compute the SHOT descriptor at each point $p \in V^i$, $SHOT_p^i$. Then, for each view V^i, we select the subset of views

$$\mathcal{V}^i = \{V^j | V^i \cap V^j \geq \tau, j = 1, \dots, N, j \neq i\} \quad (1)$$

i.e. the set of views partially overlapping with V^i according to a threshold τ. For each partially overlapping view V^j, we perform a two-step selection of points. First, for each descriptor $SHOT_p^i$ we find the point $q \in V^j$ that yields the nearest neighbor SHOT descriptor, *i.e.* $SHOT_q^j$. We then sort ascendantly the pairs of matching points (p, q) according to the Euclidean distance between the corresponding descriptors $d_{pq} = ||SHOT_p^i - SHOT_q^j||_2$. Starting from the first pair of this list of candidate positive samples C_i^j, we check whether the match is correct or not, *i.e.* if the points in the two views correspond to the same point in the 3D model. To make the learned detector robust to small shifts in the detection response, we consider a match to be correct if the distance between point p and q expressed in the same reference frame, *i.e.* transformed with the known ground-truth rotation and translation between the views, is smaller than a threshold ϵ. This check also discards points that are not in the overlapping part of the two views.

If the match is correct, we add it to the list of positive samples for this pair of views, P_i^j. We then exclude from the list of candidates C_i^j the point p as well as all the pairs of matching points that include neighbors of p, to simulate the effect of spatial non-maxima suppression that is usually performed over saliency values to obtain keypoints as local extrema. For each point p' in V^i that is close to p, *i.e.* whose 3D distance from p is smaller than a threshold ϵ_{NMS}, we remove its pair from the sorted list of matches. We then repeat the algorithm for the top entry of the list. If the match is not correct, we remove the pair from the list and repeat.

At the end of the first step, thus, we obtain a list of positive examples P_i^j for V^i when V^j is used to match SHOT descriptors. To obtain a positive set that is not influenced by the specific viewpoint change between V^i and V^j, in the second step we refine the list of positive samples by checking if such points can be robustly matched by SHOT also in those 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

[1] www.pointclouds.org

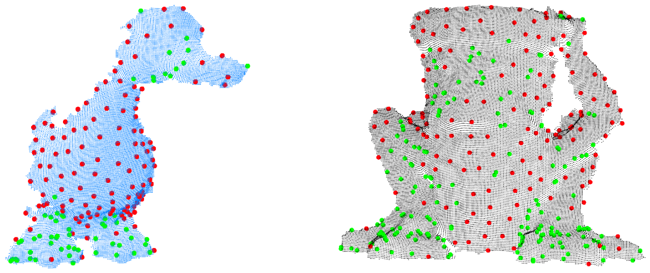


Figure 2. Example of negative (in red) and positive (in green) training samples obtained with the proposed method on two model views.

step as follows. First, we compute the positive points that are in the common part of the views, by transforming each positive sample from P_i^j with the ground-truth rotation and translation and checking if it has any neighbor in V^k within a distance ϵ. For every positive point that is in the common area, we select the nearest neighbor in the descriptor space between the descriptors from V^k and we check if it is a correct match with the test on the 3D distance between the points with respect to the threshold ϵ. Specifically, a point is required to have a correct match from just one of the V^k views in order to be considered as a positive sample. In this way, we obtain a refined set of positive samples for each view V^k, P_i^{jk}.

The final set of positive samples associated with V^i is the union of the refined positive lists across all overlapping views,

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

Finally, the set P of positive samples is given by the union of all the positive samples P^i from the views. To obtain the negative samples, several alternatives are viable:

i) implement a mechanism similar to that described above, but using now the list sorted in descending order, so that the points producing the most dissimilar descriptors become the negative samples;

ii) include in the negative set all the incorrect matches found when scrolling the original list, up to a predefined number, so that the classifier learns to avoid wrong matches producing highly similar descriptors;

iii) randomly sample the negatives from the points which are not included in the positive set, so to obtain a uniform coverage of the non-distinctive parts of the cloud.

We started with the last strategy, which is the simplest to implement, and found it to work very well. Therefore, we leave to future investigation the implementation and testing of the alternatives presented above. To enforce points to be scattered across all the non-informative parts of the view, we also remove, upon sampling a new negative sample, all its neighbors within a threshold ϵ_{neg} from the list of possible negatives. Two examples of positive and negative sample sets extracted from a model view are shown in Fig. 2. As witnessed by the Figure, positive points are unevenly distributed on the surface and do not necessarily appear on prominent, geometrically well defined points such as the right model knees and elbows or the right model tail, which would be extracted by most standard detectors.

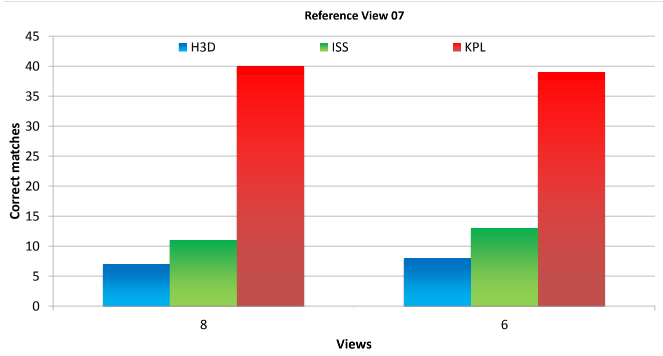


Figure 3. Correct matches obtained from the regions selected by Harris3D, ISS and the proposed training set creation (KPL).

2.2. Validation of the training set

Before defining the features to train a specific classifier, we investigate on the effectiveness of the proposed approach in defining good regions to be encoded through SHOT descriptors. In other words, we want first to find out whether, should a perfect classifier be available, the classification function defined by our training set would select regions more suitable to be matched by SHOT than those detected by standard descriptor-agnostic detectors based on geometric saliency. Purposely, we compare the number of correct correspondences attained by matching SHOT descriptors from the reference view V_i with their nearest neighbor in the partially overlapping views V^k used in the second selection step. In particular, we compare the matching performance of the SHOT descriptors computed at the points in the positive set P_i^j achieved after the first selection step and at the keypoints extracted by the ISS [29] and Harris3D detector available in PCL. To evaluate only the descriptiveness of the regions regardless of their repeatability, we rigidly move the keypoints from V_i to V_k and check if the nearest neighbor in the SHOT space is within ϵ distance from the transformed keypoint. Fig. 3 reports an exemplar result from view 7 of the *Mario* model from the *Kinect* dataset [19], which partially overlaps with view 6 and 8. We select the positive points performing the first step between view 7 and 6, then check how many correct matches we get between view 7 and 8 (left-most histogram), and then we switch roles between view 6 and 8. We can clearly see that our points are able to identify regions more suitable for the selected descriptor, even if they are learned from a different pair of views. We can therefore address the problem of defining a proper classifier aimed at learning to detect 3D keypoints yielding distinctive SHOT descriptors by using the proposed training set.

3. Design of the Classifier

We select the Random Forest [3] as the classifier to learn the keypoint detection function from our training set due to several reasons. First, it has been applied successfully to several problems in computer vision [6] and also, recently, to the problem of 3D keypoint detection [23]. Moreover, Random Forests are among the fastest classifiers to test at run-time, even when learning complex classification functions, as opposed *e.g.* to SVM with non-linear kernels. This is particularly important when using a classifier as a detector, because it has to be applied to every point of the input cloud. Finally, a Random Forest naturally enables multi-class classification and is therefore amenable to extend our framework to multiple support size so to possibly devise an adaptive-scale descriptor-specific detector.

The definition of the features used to capture the geometric structure around the points in a cloud is a crucial step in the design of the classifier. The usual features proposed when applying Random Forests to images have been simple binary features such as random intensity differences [6, 10]. However, to apply random differences to 3D neighborhoods while preserving rotation invariance requires the definition of a local Reference Frame, such as *e.g.* that used by SHOT [19] or MeshHoG [28], to correctly associate every binary test to the same points. Differently from the case of 3D descriptors, though, which are computed for a sparse subset of points of a cloud, such a costly reference frame computation would need to be performed at every point of the cloud. Moreover, point differences require interpolation to deal with views showing different points density.

Based on the above considerations, we opted for a different, rotation invariant feature, that does not require the definition of a local Reference Frame. Inspired by the robustness and descriptiveness of the SHOT descriptor, we rely on histograms of normal orientations, *i.e.* for each reference point p for which we have to compute the feature vector and for everyone of its neighbor points q within the radius r_{feat}, the angle between the normal at p and the normal at q is computed and quantized into the histogram corresponding to the sector of the sphere q belongs to. In SHOT, the sectors are obtained by dividing the sphere along the radial, elevation, and azimuth polar coordinates of points. To avoid the computation of the local Reference Frame, we change the shape of the divisions of the spherical support so 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 for spherical shells, they are inherently rotation invariant so that the calculation of a local RF is not needed. As discussed in [19], to speed up the computation and make it more robust, we do not quantize the angle between the normal at the reference point and the normal at the considered point but just their scalar product, which correspond to the cosine of the angle between the normals. To avoid quantization artifacts, bilinear interpolation is performed upon casting a vote into a histogram. Finally, the feature vector is normalized by dividing it for its Euclidean norm to limit the effect of varying point density. Fig. 4 provides a graphical overview of the feature computation process.

Given all the models of a dataset, we construct the training set and extract the histogram of orientations features at each point, setting ϵ_{neg} to obtained a balanced training set. We then train a Random Forest with T trees. When detecting keypoints on an unseen input point cloud, we apply the classifier at each point and count the number of trees T_p that classify it as keypoint. The score associated at the point is T_p/T and the point is detected as a keypoint if its score is higher then a minimum score $s_{min} \geq 0.5$ and it is a local maximum of the scores in a neighborhood of radius r_{nms}.

4. Experimental results

To experimentally validate our proposal, we measure the improvements in descriptor matching performance that can be obtained by replacing hand-crafted saliency-based detectors with our Random Forest in a standard matching pipeline.

To implement our proposal, we used the Random Forest implementation provided by OpenCV[2] and the implementation of SHOT included in PCL. As for saliency-based detectors, we selected those available in PCL, namely ISS, Harris3D and NARF [21], together with the detector proposed by Mian *et al.* [15], referred to as KPQ in the evaluation in [24], as it turned out, similarly to ISS, among the best algorithms between fixed-scale detectors. We also consider uniform sampling of points as a baseline 3D detector, given that it is normally deployed by the 3D community as a fast, albeit poorly performing, alternative. In the legends, we refer to our proposal as to KeyPoint Learning (KPL).

We tested all the detectors on three publicly available datasets already used in [24] to compare detectors: the *Laser Scanner* dataset introduced by Mian *et al.* [15]; the *Random Views* dataset, based on the Stanford 3D scanning repository[3], originally proposed in the detector evaluation [24]; the *Kinect* dataset introduced in [19]. Each dataset includes a list of models of interest, which we used to train our detector, and several scenes, where we performed keypoint

[2] `opencv.org`

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

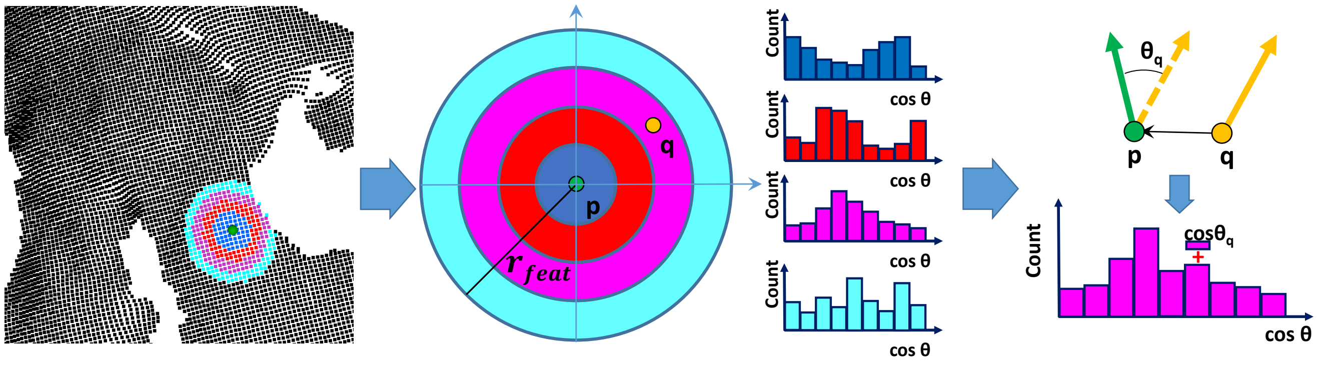


Figure 4. Overview of the feature computation. In the example, $N_r = 4$ radial subdivision are used (for ease of visualization, a 2D representation of the 3D spherical support is portrayed). Each subdivision is associated with a spherical volume and a histogram comprising N_b bins, which accumulates the cosine of the angle between the normal of p (central point) and that of each point q falling in the corresponding spherical volume.

detection. When the models are full 3D, as it is the case of *Laser Scanner* and *Random Views*, to apply our learning algorithm we sampled 42 equally spaced 2.5D views as the vertices of an icosahedron, following the methodology proposed in [1] whose implementation is also available in PCL. The scenes are 2.5D views acquired independently from the models and depicting arrangements of 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, where models are either generally known beforehand or acquired at initialization time, whereas scenes are unseen data.

The parameters used at test and training time on each dataset are reported in Table 1. As it can be seen from the Table, we used the same support size for SHOT across all the datasets, scaled to the unit of measurement of each dataset(*i.e.* meters in *Kinect* and *Random Views*, mm in *Laser Scanner*). Similarly, we used the same support size to compute both the features for our detector as well as the saliency function of the other methods.

4.1. Random Forest tuning

Random Forests are able to learn complex functions while using only a few self-explanatory parameters. However, the effect on performance and generalization ability of some of its parameters such as the tree depth and the maximum number of samples at a node to stop splitting is not clear: *e.g.* Breiman [3] suggests to let the tree grows until just 1 sample remains in a node, whereas Criminisi *et al.* [6] generally use a higher number of samples, which are used to estimate the posterior at the node.

Therefore, we investigated on the effect of the number of trees, the maximum tree depth, and the number of samples to stop splitting. To evaluate the effect on the generalization

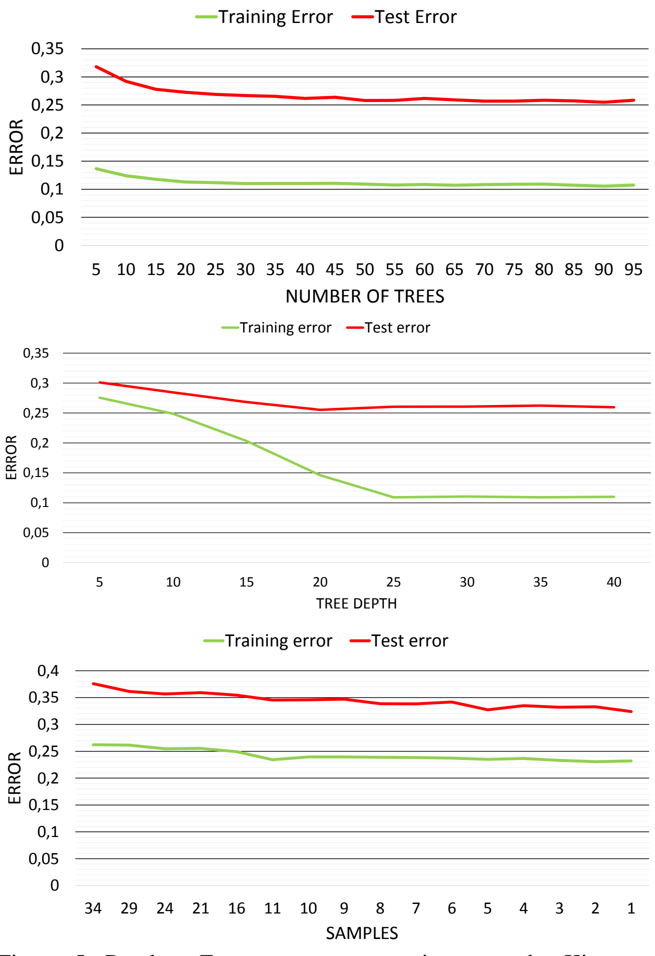


Figure 5. Random Forest parameters tuning over the Kinect dataset.

abilities of the classifier, we perform three fold cross validation at the view level, *i.e.* we used $2/3$ of the N views of

Table 1. Parameters used for each dataset. On *Random Views* we did not train a new forest, so some parameters are not specified.

Dataset	r_{SHOT}	r_{feat}	τ	ϵ	ϵ_{nms}	ϵ_{neg}	r_{nms}	s_{min}	N_b	N_r
Laser Scanner	40	20	0.85	7	4	2	4	0.8	10	5
Random Views	0.040	0.020	-	0.007	-	-	0.004	0.8	10	5
Kinect	0.040	0.020	0.24	0.01	0.02	0.015	0.02	0.8	10	5

a model to define the training set and the remaining views to test it, and we consider the training and test error.

As far as the number of trees is concerned, we vary it from 10 to 100, we checked the effect of the depth when varying it from 10 to 40 and we also let the number of samples to stop splitting vary from 1% of the points of the training set, down to 1. Fig. 5 shows that performance of the classifier does not improve when we consider 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 at a node, as limited over-fitting can be observed with smaller values, *i.e.* the test error increases although the training error decreases. For the *Laser Scanner* dataset, which features more training data (100K positive samples versus 8000 positive samples of the *Kinect* dataset), we instead found that the best performance are obtained with 100 trees, the same depth and 1 node per sample to stop splitting.

4.2. *Laser Scanner* dataset

This dataset comprises 4 full 3D models and 50 scenes, where the models occlude each other and there is also the presence of an object which is not included in the model list, so to create some clutter.

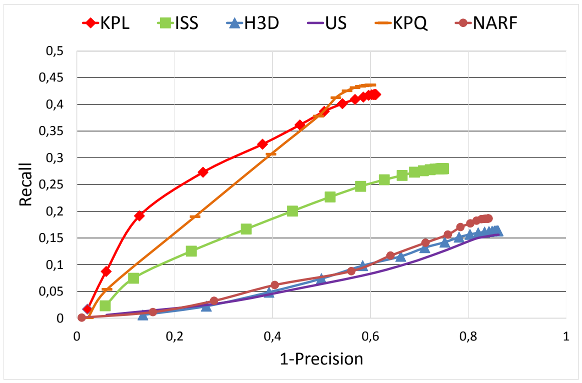


Figure 6. Results on the *Laser Scanner* dataset.

To perform the descriptor matching experiment, we first detect keypoints on all the views of all the models, compute the corresponding SHOT descriptors, and create only one kd-tree on the set of all the model descriptors. We then run the detectors on the scenes, and for each scene keypoint we establish a match with the nearest neighbor of its SHOT descriptor in the kd-tree. For each match, we check if it is a correct match and increment the true positives or false positives, accordingly. By varying the threshold on the maximum distance between SHOT descriptors to accept a match, we plot the Precision-Recall curve for the detectors, shown in Fig. 6.

The best detector among the saliency-based proposals is KPQ. This is somewhat not surprising, as the detector was originally proposed for the *Laser Scanner* dataset. ISS offers also reasonable results, whereas NARF and Harris3D perform similar to the baseline uniform sampling. Our proposal is able to identify the best regions for SHOT description, even better than the detector specifically tuned for this kind of data, which confirms our intuition that saliency-based detectors cannot select the best regions to optimize the performance of the overall detector-descriptor pipeline.

4.3. Transfer Learning On *Random Views*

This dataset comprises 6 full 3D models and 36 scenes, where the models occlude each other but there is not clutter. The models and scenes are highly detailed, but synthetic random noise has been added to the scenes. Here we consider the scenes with Gaussian noise with $\sigma = 0.1$ mesh resolution units.

The testing protocol is the same used on the *Laser Scanner* dataset. Remarkably, since the dataset presents a level of detail and noise comparable to that of the *Laser Scanner* dataset, we do not train a new Random Forest for this dataset, but used the one already learned on the previous data. This allows to test the generalization ability to unseen objects of our method.

Results are reported in Fig. 7. Overall, the relative order of the detectors is comparable with the previous dataset. However, the gap between our proposal and KPQ widens: saliency-based detectors have more difficulties in maintaining a similar performance level across different datasets. Moreover, this result shows that the way we select the training set, the feature we propose, and the selected parameters for the classifier are effective in learning a classification function with high generalization abilities.

4.4. *Kinect* dataset

This dataset comprises 7 models given as 2.5D views, as well as 17 scenes where the models are acquired under

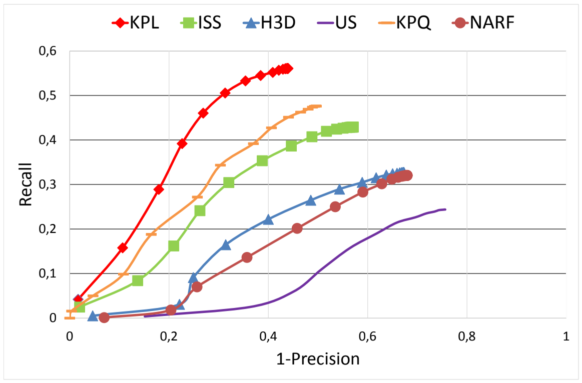


Figure 7. Results on the *Random Views* dataset.

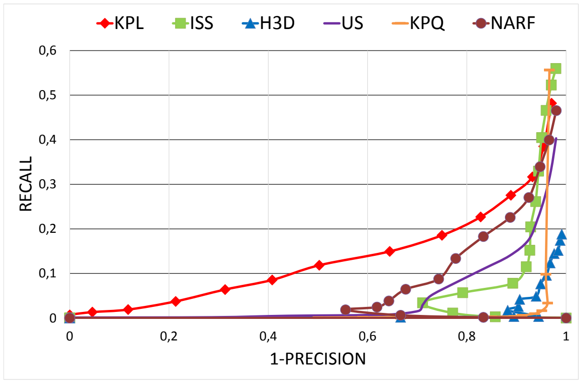


Figure 8. Results on the *Kinect* dataset.

heavy clutter and occlusions. The data is low-quality, as it has been acquired with a low-cost consumer depth camera.

To compare the evaluated detectors on this dataset, we followed the protocol already proposed in the original SHOT paper [19]. In particular, keypoints are extracted and described on each of the model present in the scene. Descriptors are then extracted on the scene, both at the location of the model keypoints and at additional keypoints extracted from clutter. Every descriptor of the scene is then matched against the set of model descriptors with the ratio criterion [14] and check for geometric correctness. By varying the threshold of the ratio test, we obtain Precision-Recall curves as in Fig. 8.

The dataset is very challenging. Results are largely worse than those obtained on the previous datasets. It is important to note how the relative order of the hand-crafted detectors changes: on the one hand, the baseline uniform sampling performs even better than KPQ, ISS, and Harris3D; on the other hand, NARF, whose performance was quite unsatisfactory on previous datasets turns out the best of the saliency-based detectors. Our KPL detector has definitely the better performance, showing the ability to adapt to different sensing modalities and the robustness of the proposed rotation-invariant features.

5. Conclusions and Future Work

The problem of detecting keypoints amenable to provide distinctive regions according to a 3D description algorithm so to improve the performance of the overall matching pipeline can be successfully solved by deploying automatic learning. The definition of positive training samples from the nearest neighbors in the descriptor space that correspond to correct 3D matches yields a classification function that identifies regions more suited to be distinctively encoded by the descriptor compared to standard hand-crafted saliences. A properly tuned Random Forest can learn such function robustly from training data by using the proposed rotation invariant geometric features. Our descriptor-specific detector adapts seamlessly to data sensed by different acquisition modality. When used on data sensed with the same modality, it also exhibits good transfer learning capacities from one dataset to another, so that the effort of training a new classifier may not be required.

Therefore, it may be possible, in future developments of this work, to learn generic detectors for specific 3D data. Moreover, the ability of the Random Forest to seamlessly handle multi-class classification enables the possibility to learn an adaptive-scale detector, that may further boost the matching performance of the pipeline. A cascade approach [27] may also be designed to quickly reject areas of the input cloud unlikely to yield good keypoints using just a few trees, thereby improving run-time performance at test time. Finally, as the underlying principle and methodology is general, we plan to apply our learning framework so to obtain detectors capable of identifying good keypoints according to other state-of-the-art 3D descriptors, *e.g*. [8, 18, 11].

References

[1] A. Aldoma, F. Tombari, L. Di Stefano, and M. Vincze. A global hypotheses verification method for 3d object recognition. In *European Conf. on Computer Vision (ECCV)*, volume 7574 of *Lecture Notes in Computer Science*, pages 511–524. Springer Berlin Heidelberg, 2012.

[2] P. Bariya and K. Nishino. Scale-hierarchical 3d object recognition in cluttered scenes. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conf. on*, pages 1657–1664, June 2010.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] U. Castellani, M. Cristani, and S. Fantoni. Sparse points matching by combining 3D mesh saliency with statistical descriptors. *Proc. Computer Graphics Forum*, pages 643–652, 2008.

[5] C. Creusot, N. Pears, and J. Austin. A machine-learning approach to keypoint detection and landmarking on 3d meshes. *International Journal of Computer Vision*, 102(1-3):146–179, 2013.

[6] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision: Vol. 7: No 2-3, pp 81-227*, 2012.

[7] H. Dutagaci, C. Cheung, and A. Godil. Evaluation of 3d interest point detection techniques via human-generated ground truth. *The Visual Computer*, 28(9):901–917, 2012.

[8] Y. Guo, F. Sohel, M. Bennamoun, M. Lu, and J. Wan. Trisi: A distinctive local surface descriptor for 3d modeling and object recognition. *Computer Graphics Theory and Applications (GRAPP), 8th International Conference on*, 2013.

[9] W. Hartmann, M. Havlena, and K. Schindler. Predicting matchability. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 9–16, June 2014.

[10] S. Holzer, J. Shotton, and P. Kohli. Learning to efficiently detect repeatable interest points in depth data. In *Computer Vision (ECCV), 2012 IEEE European Conference on*, 2012.

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

[12] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555, Nov 2011.

[13] Y. Li, S. Wang, Q. Tian, and X. Ding. A survey of recent advances in visual feature detection. *Neurocomputing*, 149, Part B(0):736 – 751, 2015.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, nov 2004.

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

[16] E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):105–119, jan 2010.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. *Computer Vision, IEEE International Conference on*, 0:2564–2571, 2011.

[18] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[19] S. Salti, F. Tombari, and L. D. Stefano. Shot: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, 125(0):251 – 264, 2014.

[20] J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593 – 600, 1994.

[21] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3d range scans taking into account object boundaries. In *Robotics and automation (icra), 2011 ieee international conference on*, pages 2601–2608. IEEE, 2011.

[22] C. Strecha, A. Lindner, K. Ali, and P. Fua. Training for task specific keypoint detection. In J. Denzler, G. Notni, and H. Se, editors, *Pattern Recognition*, volume 5748 of *Lecture Notes in Computer Science*, pages 151–160. Springer Berlin Heidelberg, 2009.

[23] L. Teran and P. Mordohai. 3D interest point detection via discriminative learning. In *ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 159–173. Springer International Publishing, 2014.

[24] F. Tombari, S. Salti, and L. DiStefano. Performance evaluation of 3d keypoint detectors. *Int. J. of Computer Vision*, 102(1-3):198–220, 2013.

[25] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, jul 2008.

[26] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *Proceedings of the Computer Vision and Pattern Recognition*, 2015.

[27] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. of Computer Vision*, 57(2):137–154, 2004.

[28] A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface feature detection and description with applications to mesh matching. *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 373–380, 2009.

[29] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *Proc. Int. Conf. on Computer Vision Workshops*, pages 1–8, 2009.