

# SKD: Keypoint Detection for Point Clouds using Saliency Estimation

Georgi Tinchev, Adrian Penate-Sanchez and Maurice Fallon

**Abstract**—We present a novel keypoint detector that uses saliency to determine the best candidates from a point cloud for tasks such as registration and reconstruction. The approach can be applied to any differentiable deep learning descriptor by using the gradients of that descriptor with respect to input 3D position to estimate an initial set of candidate keypoints. By using a neural network over the set of candidates we learn to refine the point selection until a final set of keypoints is obtained. The key intuition behind this approach is that keypoints are not extracted solely as a result of the geometry surrounding a point, but also take into account the descriptor’s response. To improve the performance of the learned keypoint descriptor we combine the saliency and the feature descriptor to allow the network to select good keypoint candidates. The approach was evaluated on two large LIDAR datasets - the Oxford RobotCar dataset and the KITTI dataset, where we obtain up to 50% improvement over the state-of-the-art in both matchability and repeatability. When performing sparse matching with the keypoints suggested by our method we achieve a higher inlier ratio and faster convergence.

## I. INTRODUCTION

A key task in robotics is the ability for a robot to localize itself in its surrounding environment. There are two prominent approaches - dense registration and sparse keypoint matching of points to a known map or object. Dense registration methods try to match every point to the map while sparse keypoint matching methods choose points with similar characteristics. Both methods have weaknesses: dense registration methods suffer from occlusions or modifications in the environment while keypoint matching requires a descriptive environment with repeatable points. Sparse keypoint matching is usually carried out in two stages: keypoint extraction and keypoint description. We present a novel method for keypoint extraction from point clouds that exploits the saliency of the point’s descriptor to propose points with higher likelihood of being matched. This follows the recently proposed *describe-then-detect* methodology [1], where we focus mainly on the *detect* part, assuming an already trained differentiable point cloud descriptor.

Keypoint extraction methods have focused on understanding the appearance of an image so as to reliably generate keypoints at the same location when presented with different images of the same object, this is known as *repeatability*. However, this approach can be affected by disturbances. For example, when extracting keypoints and descriptors

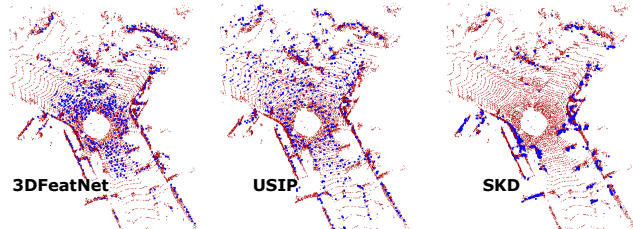


Fig. 1. Top-down view of a single point cloud (brown) and 1024 generated keypoints (blue). Our method (Salient Keypoint Detector, SKD) was not implicitly trained to ignore the ground, but rather it uses the feature signal of the descriptor and the context information to determine informative areas in the environment - corners, edges and structure.

from an element that is repetitive keypoint extraction can be performed reliably and those points will be *repeatable*. However, the descriptors of those points will not be useful due to the ambiguity caused by the repeated pattern. Recent research suggests that repeatability by itself is not sufficient for evaluating keypoint quality [2], [3]. A more reliable metric is *matchability*, as it measures how useful the combination of *keypoint and feature* is. For this reason we focus on building a keypoint detector that learns to produce points that can be reliably matched when given a deep learning descriptor. We use the gradient response of the descriptor with respect to the positional information of the input and combine it with a low dimensionality representation of the point feature to produce keypoints that generate a higher percentage of inlier matches.

Our proposed approach leverages the idea, presented in [3], that state-of-the-art keypoint detectors can be built by only looking at the feature response. Using this as an initial set of keypoints, we estimate a *saliency* score using the combination of neural network layer activations and the gradients at that layer w.r.t. the positional input to select characteristic keypoints (Fig. 1). The use of saliency has been shown to produce promising results both in images and in point clouds [4], [5], [6]. This subset of the original points can then be combined within a neural network that looks at the descriptor values and the saliency and makes a decision as to which are the best points to select. The main contributions of our work are as follows:

- The saliency-based keypoint detector (SKD): a novel method for extracting keypoints in point clouds based on the gradient response of a differentiable descriptor and the geometry of the scene.
- An environment agnostic approach: by using the gradient response rather than the actual point cloud, our approach is less vulnerable to biases in the training data. This makes our approach robust when testing on different environments and allows the model to be used

The authors are with the Oxford Robotics Institute, University of Oxford, United Kingdom. {gtinchev, adrian.mfallon}@robots.ox.ac.uk

This work was supported by EPSRC RAIN and ORCA Robotics Hubs (EP/R026084/1 and EP/R026173/1 respectively) and EU H2020 project Memory of Motion (MEMMO, project ID: 780684). M. Fallon is supported by a Royal Society University Research Fellowship.

without retraining. We demonstrate this by training on the Oxford RobotCar dataset [7] and testing on the KITTI dataset [8] without retraining.

- State-of-the-art performance: Compared to the state-of-the-art [9], [10], our approach generates two times more correct matches and achieves 40% more relative repeatability. We have evaluated our algorithm on two of the largest and most widely used LIDAR datasets - the Oxford RobotCar dataset [7] and the KITTI dataset [8]. In total these datasets contain point clouds collected from more than 300 kilometers of driving.

## II. RELATED WORK

This paper focuses on reliably obtaining keypoints from point clouds. This is a topic which is very closely related to image keypoint extraction, so we will review the literature for both fields in this section. We will also outline the relevant research on point cloud-based deep learning models and saliency methods as these topics are central to the contributions of our approach.

### A. Image Keypoint Detectors

Much of the recent work on keypoint and feature extraction from images employs deep learning architectures. In [11] a keypoint detector for depth images was presented. It used a Siamese approach by pairing two Faster-RCNN networks with a contrastive loss between them. Focusing on unsupervised keypoint learning, [12] presented an approach that generated reliable keypoints by learning from the temporal consistency of the network activations in short videos. A successful approach for obtaining good keypoints is to optimize how they respond to the content of the image and not just the geometry. TILDE [13] learned to make the points more reliable by understanding how changes in weather and lighting conditions modified the performance of a point. In LF-Net [14], the keypoint detector and the feature descriptor are jointly learned by leveraging structure from motion (SfM) sequences to generate large amounts of training data. By understanding that the job of a keypoint is to increase the probability of descriptor matching, it focused on matching performance and at the time outperformed all previous approaches. Recently, ELF [3] applied a simple approach by using the gradient response of deep learning features to produce keypoints.

R2D2 [2] not only focused on repeatability for a keypoint, but also on the reliability score based on the descriptor's response. They jointly learned the keypoint detector and the discriminativeness of the descriptor. This is similar to our proposition to look at matchability of a keypoint in addition to its repeatability. SuperPoint [15] jointly learned detection and description in a supervised task with multiple training stages on artificial images of basic structures. D2-Net [1] used a CNN with shared weights for description and detection. In contrast to previous approaches it used a methodology of *describe-than-detect*. UnsuperPoint [16] improved the training methodology of SuperPoint by requiring only a single training round. They use a self-supervised

detector and unsupervised descriptor to improve the repeatability of proposed keypoints. Their detector and descriptor also shared a common backbone. SIPs [17] introduced a metric to minimize the number of detected points with an iterative algorithm. Similar to us, they used the probability of a pixel being an inlier, but in 2D image data.

An alternative research direction was taken by GLAM-points [18] to apply reinforcement learning rewards to SIFT matches in order to maximize correct correspondences. Similarly, DISK [19] leveraged cooperative reinforcement learning to match local features with a policy gradient. In contrast to the above works, we do not jointly train a keypoint detector and descriptor, but instead focus only on the former and assume a given keypoint descriptor. Therefore, our methodology can be viewed as similar to *describe-than-detect*, with a given differentiable *describe* method. Also, we do not explicitly define a reward for each correct keypoint, nor use any non-maximum suppression (NMS) techniques.

### B. Point Cloud Keypoint Detectors

PointNet [20] and PointNet++ [21] introduced new ways to efficiently understand unstructured sets of points (point clouds). These works presented a novel layer for extracting descriptions from sets of points by learning a symmetric function and thus making the layer invariant to point order. PointCNN [22] managed to increase the performance of the basic neurons by approximating a convolution using an MLP and by performing aggregation of spatial data within each neuron. PointNetVLAD [23] presented a method to perform place recognition from point clouds. The point clouds were described using PointNet++ and NetVLAD trained with a metric learning loss to produce a feature vector. In contrast, we focus on detecting keypoints that aid descriptors such as [20], [21], [22]. Many recent papers leveraged PointNet layers in different applications [24], [6], [25]. In [6] the authors built a saliency map to understand the effect of each point on the final prediction. They tested the performance of their saliency score by performing point dropping operations to demonstrate performance better than a method based on the critical-subset theory [6]. We take inspiration from [6], but we do not perform point shifting or point dropping operations.

The most relevant methods to our work are 3DFeatNet [9] and USIP [10]. 3DFeatNet [9] uses PointNet++ as a building block and learned a detector and a descriptor by using a two stage network. The main problem with this approach is that the keypoint extraction network does not perform as well as desired. To mitigate this, a novel keypoint extraction method was recently proposed, USIP [10]. It focused on obtaining keypoints with a high level of repeatability. USIP [10] minimizes a variation of the Chamfer loss that focuses in matching points reliably while 3DFeatNet uses a triplet loss, that focuses in obtaining a descriptive feature space, but that cannot manage matching as it only focuses on the descriptor. 3DFeatNet [9] extract attention for each point in the input while we use the saliency scores obtained directly from the feature descriptor.

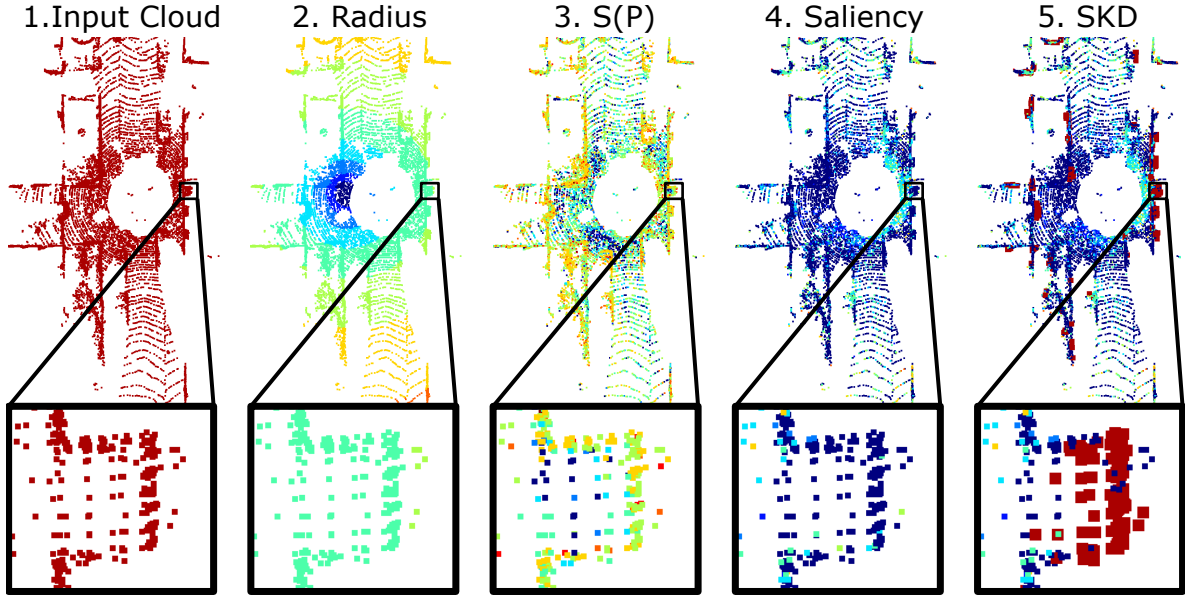


Fig. 2. **1.** Input point cloud  $P$ . **2.** Distances of each point ( $r_i$ ) from the spherical core - blue indicates closer to the median point, red - further apart. **3.** The *initial saliency*,  $S(P)$ , based on the activations and gradients w.r.t. input point cloud (1.) - blue corresponds to negative gradients, red - highly positive. **4.** The *saliency score*  $s_i$  estimated from the radius (2.) and the gradients (3.), blue - low score, red - high score. **5.** Selected keypoints (red, enlarged) on top of the saliency score. Our method picks keypoints based on the keypoint saliency score  $s_i$ , but also leverages context and the original descriptor to correctly pick out edges and corners negatively scored by  $s_i$ .

Recently, DH3D [26] introduced a hierarchical Siamese network for joint learning of local description, keypoint score, and global descriptor. It uses a coarse-to-fine re-localization technique to first find submaps in the scene and then local features for accurate pose estimation. Similarly to 3DFeatNet, they used two-phase training to first train the descriptor and then the detector. D3Feat [27] adopt the joint descriptor and detector paradigm from D2-Net and extended it to point clouds. Their descriptor is trained using a contrastive loss, while a self-supervised loss is proposed for the detector, based on the descriptor’s response. In contrast to these works our training procedure only includes training the keypoint detector while using an pre-existing descriptor.

### C. Saliency Estimation

*Saliency* has been studied as a way to quantify and understand what is relevant within a neural network. The definition of *saliency* in this context has been studied by a number of works, such as [4], [28], [29], that seek to understand why machine learning models behave as they do. Even before the growth in popularity of neural networks, methodologies such as [30] were being designed to understand why classifiers made specific decisions. This became more prevalent with the adoption of deep learning models for most perception tasks. Approaches such as [4], [29] seek to understand what a neural network finds relevant by looking at how the gradients of a given prediction behave. The interpretation was that gradients with higher magnitude in specific areas influence predictions. The use of *saliency* has been demonstrated to be capable of generating state-of-the-art keypoint extractors in images. ELF [3] computed the gradient of the feature map given an image and then used the Kapur threshold [31] to select keypoints. In a similar fashion, Grad-CAM [28] used

the gradient maps of a classification score to produce regions of interest for a given image for tasks such as classification, image captioning, and visual question answering.

## III. METHODOLOGY

We present Salient Keypoint Detection (**SKD**) - a keypoint extraction method based on the *saliency* of a point cloud. We define *saliency* as the combination of the feature activation signal at a specific layer of a pre-trained descriptor and the gradients of the same layer with respect to the input point cloud. We leverage a pre-trained descriptor network to produce robust keypoints directly from 3D data. We then project the *saliency* into the spherical coordinate system of the input point cloud and extract the most informative regions. We combine those regions with the features of the original descriptor in order to extract robust and repeatable keypoints. Using the combined feature vector, we train a neural network that learns to predict the likelihood of a point being a keypoint.

Similarly to the approach of [3] on images, we extract the gradients of the features at different levels of the architecture, and choose the best performing one. An evaluation of the performance at each layer of the pre-trained descriptor network is presented in Sec. IV-C. In contrast with [3], we select keypoints by combining the gradient information and the activations with the points in Euclidean space in order to determine critical points using the criterion defined by [6].

### A. Point Cloud Saliency

For a given point cloud  $P = \{p_i \in \mathbb{R}^3\}^N$  with  $N$  points, we extract the gradients of a pre-trained network  $\nabla F$  at a specific layer  $l$  with respect to the positional information of each point  $\{p_i \in P\}^N$ , defined as  $\nabla F_l^P \in \mathbb{R}^{\Omega_l \times N \times 3}$ . We define the *initial saliency*  $S(P)$ , as the product of the

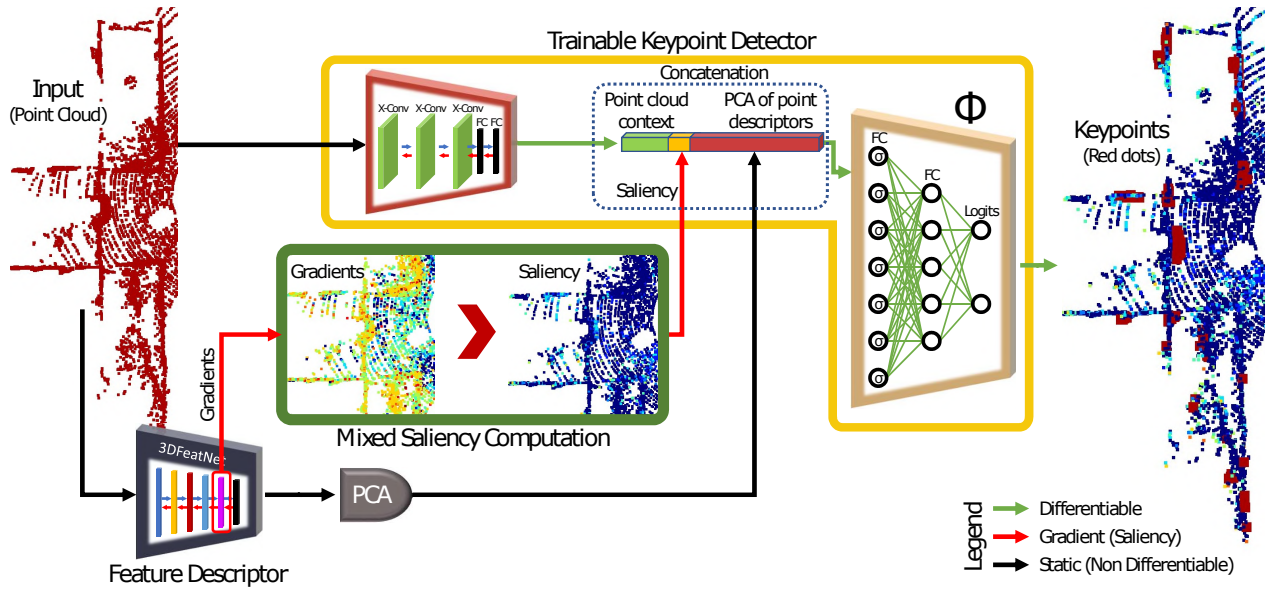


Fig. 3. Proposed network architecture. The input to our system is a raw point cloud and a feature descriptor network. Our method concatenates the saliency information from the gradients of the feature descriptor with the per-point context features and a PCA projection of the original feature descriptor. The concatenated vector is the input to two FC layers that generate per-point response at the logits layer.

feature activations of that layer  $F_l^P \in \mathbb{R}^{\Omega_l}$  with the gradients, formally defined as:

$$S(P) = F_l^P \cdot \nabla F_l^P \quad (1)$$

Where,  $S(P) \in \mathbb{R}^{N \times 3}$ . In this way the extracted initial saliency corresponds to specific points in the point cloud that have good activations and are valuable based on the gradient of that layer w.r.t. the positional input. From a geometric perspective this can be thought of as projecting the feature signal through to the input point cloud and determines the value of individual points (see example projection in Fig. 2).

We take these saliences and determine a secondary saliency score per point, defined as  $s_i$  for each point. The score weights the contribution of  $S(P)$  by the distance from the center of the point cloud in the Spherical Coordinate system. The score is formally defined as:

$$s_i = - \sum_{j=1}^3 [S_j(P) \cdot (x_{ij} - \text{median}(x_{ij}))] r_i \quad (2)$$

where  $j \in \{1, 2, 3\}$  defines each of the Cartesian coordinates  $x, y, z$  of point  $p_i \in P$  and  $r_i = \sqrt{\sum_{j=1}^3 (x_{ij} - \text{median}(x_{ij}))^2}$  is the distance of point  $i$  to the median of the spherical core of the point cloud [6]. In other words, we weight the initial saliency score by the offset from the median of the point cloud. Therefore, further away points will have higher score, and points closer to the center will have lower score (see Fig. 2). This design decision is motivated by the high density of points near the origin of the sensor (see distribution of points of 3DFeatNet vs SKD in Fig. 7). Also, note that well distributed keypoints in two point clouds will likely result in a more robust solution.

Finally, in accordance with standard deep learning practices, the input saliences are normalized to have zero mean and unit variance within a single point cloud. The final

saliency ensures a good spatial distribution of the selected points while also selecting points with good activations based on the descriptor network. Fig. 2 illustrates the saliences of an example input point cloud at each stage of computation.

### B. Network Architecture

Our network architecture is depicted in Fig. 3. It consists of three parts. The first component is the *saliency* method described in the previous section. The second component is a PCA dimensionality reduction of the original per-point features which improved the final results by producing a smoother feature space. The third part is called the *point cloud context* features. These are four X-Conv layers [22] and two fully connected layers that have been pretrained on a feature extraction task to create stable initial estimates. These layers will learn to provide a description of the context around any given point. We use a two dimensional size for the context latent space. This contextual components provide only a small performance improvement and overall are much less important than the other two components. Finally, the three components are concatenated and fed to two additional fully connected layers to produce the final keypoint prediction. The network learns to infer a score for each point — determining the probability of it being a robust and repeatable keypoint for the original descriptor. Note that our model is descriptor-agnostic and thus can be applied to any descriptor network in order to improve the performance.

1) *Training*: During training, the input to our model consists of a pair of point clouds and the ground truth transformation between them,  $(P_k, P_l, T)$ . Both  $P_k$  and  $P_l$  are  $N \times 3$  dimensional vectors, where  $N$  denotes the cardinality of the point cloud set. In addition, we assume to have a pre-trained model for the point descriptors. To train we estimate the saliency  $s_i$  and features  $f_i$  for each point in both point clouds. Due to the large dimensionality of the feature space  $f_i$ , we



performed PCA and transformed the features that explained  $\approx 90\%$  of the data. In addition, given the ground truth transformation between the two point clouds, we determine the bidirectional correspondence for each point, conditioned on the descriptor. These correspondences are used in our loss to select matching pairs of points. Note that these correspondences need not be injective nor surjective. In other words, if for a point  $i_1$  the closest neighbour under the ground truth transformation (up to a threshold of 1 m),  $T$ , is  $j_1$ , it does not mean the reverse applies, nor does that limit  $i_1$  have a unique neighbour from  $P_l$ .

For each point in both point clouds we extract *context-aware features*,  $f_c$ , and concatenate them to the saliencies  $s_i$  and features  $f_i$ . The term *context-aware* is used as we expect the layers that lead to this descriptor to contain information about the local geometry around each point. We pretrain these layers on a feature learning task to obtain stable initial features in the training. We chose a small feature space of only two dimensions in order to force the network to learn rough estimates of the shape of objects that can generalize better when moving to a different dataset. Afterwards, the full architecture is trained end-to-end with the saliency and feature concatenation. We note that the contribution of the *context-aware features* is smaller compared to  $s_i$  and  $f_i$ .

The concatenated saliencies, PCA features, and *context-aware* features are fed to two fully connected layers to estimate the probability of each of the points being a keypoint. To this end we use a softmax cross entropy loss between the stacked  $P_k, P_l$  clouds and the determined correspondences, given the ground truth transformation. Due to the smaller number of correct keypoint correspondences between the two clouds, we balance the loss function terms given the keypoint to non-keypoint ratio determined by the ground truth correspondences.

2) *Inference*: During the forward pass of the network we estimate a probability for each point being a correct keypoint. We can then either extract the top  $K$  keypoints, based on this probability or select all the keypoints above certain probability. The extracted keypoints produced by the described approach are better suited to the descriptor as the learning iterations optimize its performance. Note, that neither Non-Maximum Suppression, nor any other threshold is applied to obtain the final set of keypoints.

#### IV. RESULTS

In this section we discuss the datasets and metrics we used to evaluate our approach and then present our findings.

##### A. Datasets

In our study we used two datasets - the Oxford RobotCar [7] and the KITTI odometry dataset [8]. In order to provide a fair comparison, our experiments use the preprocessed test data provided by related previous works, [9] and [10]. We also use their evaluation scripts to make the comparison as clear as possible. We train our model using the same sequences from the RobotCar dataset as [9], and test our approach using the same test set of 3,426 point cloud

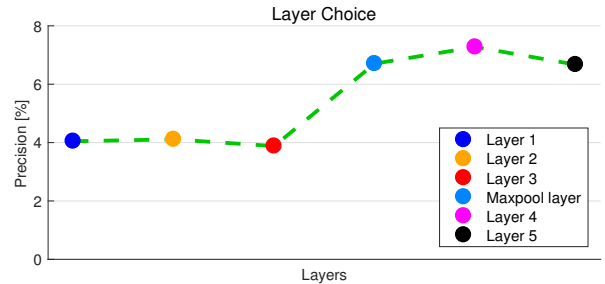


Fig. 4. Performance of each layer of 3DFeatNet expressed as percentage of correct matches if the layer was selected for gradient computation. The layer identified corresponds to the counter of layers in [9] starting from the left. In our evaluation we used the layer that maximized the performance (Layer 4) for both ELF [3] and our method.

pairs which the authors provided. Furthermore, we neither train our method nor these baselines on the KITTI dataset, in order to test the generalization ability of the approaches.

The evaluation part of the KITTI dataset, used by both [9], [10], provides only 2,369 point clouds out of the total dataset. So as to increase the size of the KITTI evaluation dataset, we extended it using the 11 *training* sequences. This is possible only because the RobotCar dataset is used for training all models. The extended dataset is processed in a dense manner: for each point cloud we align the next consecutive 10 point clouds to it using the ground truth sensor poses. By doing this we expanded the number of testing point cloud pairs from 2,831 to 207,917, which allows us to more fully study our proposed approach. Note that none of the KITTI data was used for training our method, any of the baselines, or the descriptor.

##### B. Metrics

We focus our analysis on the keypoint extraction methods and compare them using three metrics. In addition, we use the same 3DFeatNet [9] descriptor for all the methods.

The first metric uses the matching score proposed by [9] that determines correspondences within 1 m maximum distance. We detect keypoints separately for two point clouds. Given the ground truth sensor poses, we project the keypoints from the first point cloud into the second one. Keypoints that do not have a nearest neighbour bidirectional correspondence in the second point cloud are ignored from the final result (i.e. no overlap). For the rest of the keypoints, the descriptors are compared and matched to establish a correspondence. The precision is measured as the number of correct correspondences against the total number of possible matches. This metric estimates the percentage of correct correspondences based on the distance between them. Correct correspondences are thresholded to 1 m distance, as it serves as a desirable upper limit when performing registration between two point clouds.

For the second metric we chose to compare the normalized relative repeatability, as proposed in [10]. The metric compares the keypoints detected in one point cloud to the keypoints detected in its corresponding point cloud. If the keypoints overlap, below a certain distance, they are considered a match. While this saturates with a high number

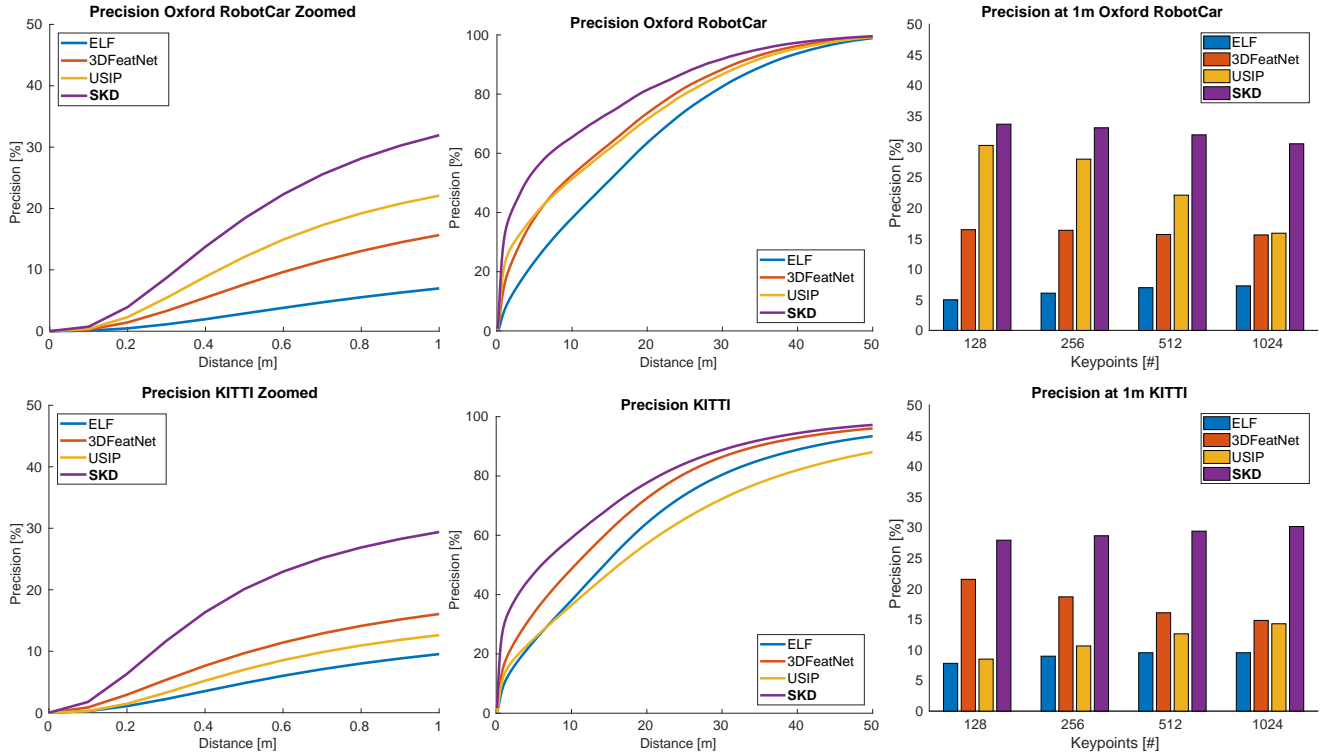


Fig. 5. Matching score evaluated on the Oxford RobotCar dataset (top row) and the KITTI dataset (bottom row). The first two columns are the percentage of matched keypoints using the 3DFeatNet [9] descriptor when varying the distance between correspondences. The first column shows a zoomed-in version of the second column within 1 m of distance, considered relevant for geometric registration. The third column shows the performance of different approaches while varying the number of keypoints.

of keypoints [3], we chose to use this metric to have a fair comparison against USIP [10]. We note that the ability to match keypoints on their own is not sufficient, hence we consider the matching score to be a more important metric.

Finally, we evaluated the geometric registration performance of our approach on both the KITTI and the Oxford RobotCar datasets using RANSAC [32]. As in [9], we consider an alignment to be successful if it differs by less than 2m and  $5^\circ$  from the ground truth sensor pose. For the successful registrations we present relative translation error (RTE), relative rotation error (RRE), success rate as a percentage of the successful registrations, average number of iterations it took RANSAC to find a suitable candidate within 99% confidence (capped at 10,000 iterations), and the inlier ratio of how many points were considered when obtaining a correct registration.

### C. Baseline Methods

Our approach is general and can be applied to any point cloud descriptor network. For simplicity we chose to use the descriptor of [9] as it is open source and easy to use, as well as being state-of-the-art point cloud descriptor. We select the best performing layer from the descriptor to generate the gradients on which we compute the saliency values, as illustrated in Fig. 4. We have compared against the learned keypoint detector methods of 3DFeatNet [10], DH3D [26], USIP [10], and a 3D interpretation of the ELF [3] detector. All the learning methods are trained on the Oxford RobotCar dataset and tested on both RobotCar and KITTI data. We have used the models provided online,

and trained our own network. For USIP, we took the models provided by the authors which they trained on the Oxford RobotCar dataset. As ELF does not need training, we took the best performing layer, according to Fig. 4. We adapt their approach to work on point cloud data by performing Non-Maximum Suppression in 3D and choosing keypoints based on the Kapur Threshold [31].

### D. Matching Score Experiments

We begin by evaluating our approach using the matching score metric on both Oxford RobotCar and KITTI datasets. Fig. 5 illustrates the performance of SKD compared to other state-of-the-art methods. The first row shows the percentage of matched keypoints on the Oxford RobotCar dataset using the 3DFeatNet descriptor for all methods. The first column is a zoomed-in version of the percentage of points detected within 1m distance that are considered relevant for matching. In the second column we show the full results as presented in [9]. The last column shows the precision at 1m distance at different numbers of keypoints — 128, 256, 512, 1024. As all the approaches are learning-based, we took the top  $K$  keypoints. This ensures that the number of keypoints are identical. The second row presents the results for the full KITTI dataset, with all the methods trained on the Oxford RobotCar dataset. We observe that our approach generalizes well without a loss in performance, and outperforms the second best approach by a significant margin. Similarly to 3DFeatNet [9], our approach’s performance does not decline with the increase of detected keypoints.

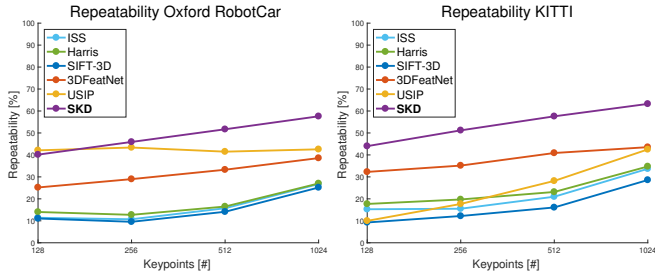


Fig. 6. Relative repeatability on the Oxford RobotCar dataset (left) and KITTI (right) with different number of extracted keypoints.

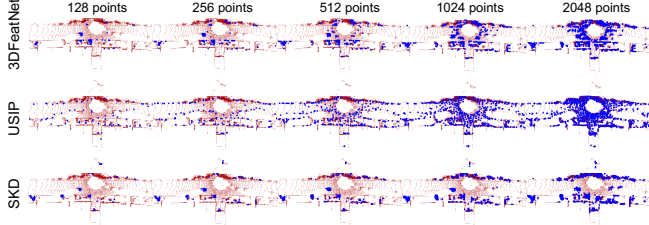


Fig. 7. Qualitative results on the KITTI dataset. For all methods the number of generated keypoints is increased from left to right. This shows what relative importance each method gives to certain areas of the point cloud.

### E. Repeatability Experiments

Next, we evaluate the repeatability of keypoints across multiple observations of the same scene. Fig. 6 shows the repeatability of SKD measured on the Oxford RobotCar dataset (left) and the KITTI dataset (right). For the Oxford RobotCar, our approach performs comparably to the second best method when extracting 128 and 256 keypoints, but it selects points that are 40% more repeatable when extracting 1024 interesting points. Furthermore, Fig. 1, 7, 8, and the supplementary video material present qualitative examples of the detection of keypoints of the top three approaches. The proposed approach has learned to select more descriptive areas of the environment. For example, our method does not select ground points, and even though we forced the method to generate large amounts of points, it still chooses points based on the high activations of the network - around building edges and corners, that in turn are easier to match.

Fig. 6 (right) illustrates the repeatability of our approach on the KITTI dataset. On this dataset, 3DFeatNet performs slightly better than on the Oxford RobotCar dataset. Interestingly, as we use the feature signal of 3DFeatNet, our keypoint detector also performs better. In addition, Fig. 7 presents qualitative evaluation of our approach on the KITTI dataset with an increased number of keypoints, where SKD consistently selects the same areas of the environment.

### F. Geometric Verification Experiments

Although our work is primarily focused on keypoint detection, we provide results that show one of the possible applications of our keypoint detector — geometric registration. We present the results of geometric registration using KITTI and Oxford RobotCar datasets in Tab. I. For the baseline algorithms, we used the code published by the authors.

SKD performs similarly to the state-of-the-art in terms of relative rotation and translation error, within the standard deviation of the best-performing method (2 mm and

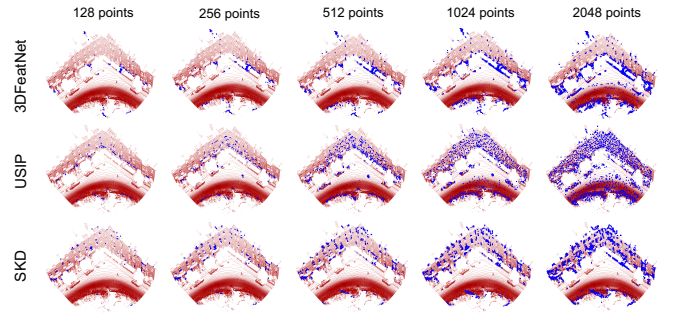


Fig. 8. Qualitative results on the Oxford RobotCar dataset. The keypoint proposals from SKD are evenly spread in geometrically interesting locations.

0.04 degrees difference with best baseline). Our method is, however, approximately nine times faster on the KITTI dataset and two times faster on the Oxford RobotCar dataset compared to the second-best approach while also having a higher percentage of inliers. We note that we did not use non-maximum suppression (NMS), which we suspect is the reason for the slightly higher errors and lower success rate. Applying NMS first and then extracting the top  $K$  keypoints would reduce the number of grouped keypoints, and as such the robust estimator could find better distributed keypoints, which will likely result in better point cloud registration accuracy. As shown in our *repeatability* and *matchability* experiments, our method is good at picking out keypoints at similar locations, however, for applications such as global registration highly repeatable and matchable points are not the only necessary requirement. However, for object pose estimation, for example, highly repeatable points are arguably more important than the distribution of points. Furthermore, not using NMS makes our detector more robust to changes in the scene.

### G. Ablation studies

Next, we studied the contribution of each component of our method by measuring the precision at 1 m distance on the Oxford RobotCar dataset, shown in Fig. 9.

First, we inspected the contribution of the saliency information alone is responsible for approximately half of the concatenated vector - dark blue compared to yellow in Fig. 9. In addition, training the neural network,  $\Phi$ , with just saliency information contributes to approximately a third of the end result - red compared to light blue. For these reasons we have decided to name our method Salient Keypoint Detector (SKD). Second, the contribution of the network,  $\Phi$ , when training with the full feature vector is substantial as is to be expected - purple compared to dark blue and yellow on Fig. 9. By analyzing our ablation study in Fig. 9 we can see that the increase in performance from the context features is only marginal. We would like to note that the performance gains, and thus the contributions of this paper, come from using saliency together with the value of the feature to determine which points to extract as keypoints. Finally the use of PCA was introduced to smooth the feature space and allow for better generalization, and improved performance, as can be seen in Fig. 9 (green vs light blue).

	Detector + Descriptor Method	RTE (m)	RRE (°)	Success Rate	Avg # iter	Inlier ratio
Oxford	ELF [3] + 3DFeatNet [9] *	$0.42 \pm 0.31$	$1.66 \pm 1.09$	86.49%	9788	5.3%
	3DFeatNet [9] + 3DFeatNet [9] *	$0.30 \pm 0.25$	$1.07 \pm 0.85$	97.64%	3083	12.9%
	USIP [10] + 3DFeatNet [9] *	$0.29 \pm 0.26$	<b><math>0.96 \pm 0.77</math></b>	<b>98.74%</b>	823	21.0%
	DH3D [26] + 3DFeatNet [9]	<b>0.28</b>	1.04	98.2%	2908	-
	<b>SKD + 3DFeatNet [9]</b>	$0.31 \pm 0.27$	$1.11 \pm 0.89$	97.64%	<b>393</b>	<b>32.7%</b>
KITTI	3DFeatNet [9] + 3DFeatNet [9] *	$0.142 \pm 0.120$	<b><math>0.533 \pm 0.410</math></b>	<b>97.80%</b>	3917	12.7%
	USIP [10] + 3DFeatNet [9] *	$0.203 \pm 0.193$	$0.637 \pm 0.517$	97.12%	5324	11.0%
	<b>SKD + 3DFeatNet [9]</b>	<b><math>0.140 \pm 0.134</math></b>	$0.579 \pm 0.480$	96.52%	<b>594</b>	<b>32.2%</b>

TABLE I: Geometric registration on the Oxford RobotCar dataset (top) and the KITTI dataset (bottom), evaluated by RANSAC. \* indicates our evaluation on the dataset using each method’s official implementation and models. Note: the used KITTI dataset is 73 times larger than the one by [9], [10].

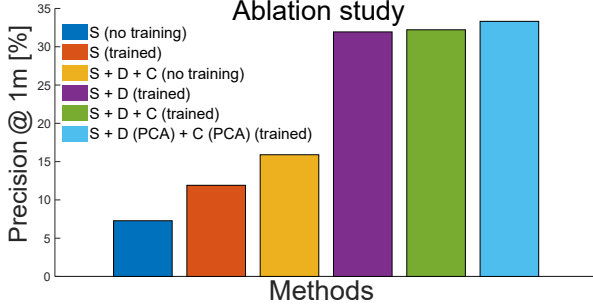


Fig. 9. S=Saliency, D=Descriptor, C=Context. Ablation studies of our method evaluated on the Oxford RobotCar dataset. We evaluate the contribution of the saliency information, and the importance of training with the original descriptor. The results displayed in the figure show the clear contribution the neural network is making to the keypoint generation.

## V. CONCLUSIONS

In this paper we present a novel method for keypoint extraction that uses saliency information to extract informative regions in a point cloud. The method concatenates signals from the gradients with respect to the positional information of the input, context-aware features, and the descriptor features and learns to predict which 3D points have a higher chance of being matched correctly. The proposed approach is descriptor-agnostic and outperforms the state-of-the-art by up to 50% in matchability and repeatability compared to the second-best method. When performing geometric registration, the algorithm achieves a higher number of inliers while also finding a correct pose estimate significantly faster than the current state-of-the-art.

## REFERENCES

- [1] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, “D2-Net: a trainable CNN for joint description and detection of local features,” in *CVPR*, 2019, pp. 8092–8101.
- [2] J. Revaud, P. Weinzaepfel, C. R. de Souza, and M. Humenberger, “R2D2: repeatable and reliable detector and descriptor,” in *NIPS*, 2019.
- [3] A. Benbihi, M. Geist, and C. Pradalier, “ELF: embedded localisation of features in pre-trained CNN,” in *ICCV*, 2019.
- [4] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” in *ICLRW*, 2014.
- [5] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” in *ICLRW*, 2015.
- [6] T. Zheng, C. Chen, J. Yuan, and K. Ren, “Pointcloud saliency maps,” in *ICCV*, 2019.
- [7] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, “1 Year, 1000km: The Oxford RobotCar Dataset,” *IJRR*, vol. 36, no. 1, pp. 3–15, 2017.
- [8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *IJRR*, vol. 32, no. 11, pp. 1231 – 1237, 2013.
- [9] Z. J. Yew and G. H. Lee, “3dfeat-net: Weakly supervised local 3d features for point cloud registration,” in *ECCV*, 2018.
- [10] J. Li and G. H. Lee, “Usip: Unsupervised stable interest point detection from 3d point clouds,” in *ICCV*, 2019.
- [11] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Koeck, “End-to-end learning of keypoint detector and descriptor for pose invariant 3d matching,” in *CVPR*, 2018.
- [12] T. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih, “Unsupervised learning of object keypoints for perception and control,” in *NeurIPS*, 2019.
- [13] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, “TILDE: A temporally invariant learned detector,” in *CVPR*, 2014.
- [14] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, “Lf-net: Learning local features from images,” in *NeurIPS*, 2018.
- [15] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *CVPR*, 2018.
- [16] P. H. Christiansen, M. F. Kragh, Y. Brodskiy, and H. Karstoft, “UnsuperPoint: end-to-end unsupervised interest point detector and descriptor,” *arXiv preprint arXiv:1907.04011*, 2019.
- [17] T. Cieslewski, K. G. Derpanis, and D. Scaramuzza, “SIPs: succinct interest points from unsupervised inlierness probability learning,” in *3DV. IEEE*, 2019, pp. 604–613.
- [18] P. Truong, S. Apostolopoulos, A. Mosinska, S. Stucky, C. Ciller, and S. D. Zanet, “GLAMPoints: greedily learned accurate match points,” in *ICCV*, 2019, pp. 10 732–10 741.
- [19] M. Tyszkiewicz, P. Fua, and E. Trulls, “DISK: learning local features with policy gradient,” *NeurIPS*, vol. 33, 2020.
- [20] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” in *CVPR*, 2017.
- [21] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” in *NeurIPS*, 2017.
- [22] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, “Pointcnn: Convolution on x-transformed points,” in *NeurIPS*, 2018.
- [23] M. A. Uy and G. H. Lee, “PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition,” in *CVPR*, 2018.
- [24] C. Xiang, C. R. Qi, and B. Li, “Generating 3D adversarial point clouds,” in *CVPR*, June 2019.
- [25] D. Xu, D. Anguelov, and A. Jain, “PointFusion: Deep sensor fusion for 3D bounding box estimation,” in *CVPR*, 2018.
- [26] J. Du, R. Wang, and D. Cremers, “DH3D: deep hierarchical 3D descriptors for robust large-scale 6DoF relocalization,” in *ECCV*. Springer, 2020, pp. 744–762.
- [27] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, “D3Feat: joint learning of dense detection and description of 3D local features,” in *CVPR*, 2020, pp. 6359–6367.
- [28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *ICCV*, Oct 2017.
- [29] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, “Sanity checks for saliency maps,” in *NeurIPS*, 2018.
- [30] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller, “How to explain individual classification decisions,” *J. Mach. Learn. Res.*, vol. 11, pp. 1803–1831, Aug. 2010.
- [31] J. Kapur, P. Sahoo, and A. Wong, “A new method for gray-level picture thresholding using the entropy of the histogram,” *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 273 – 285, 1985.
- [32] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *CACM*, vol. 24, no. 6, pp. 381–395, 1981.