S-TREK: Sequential Translation and Rotation Equivariant Keypoints for local feature extraction

Emanuele Santellani *, Christian Sormann *, Mattia Rossi †, Andreas Kuhn †, Friedrich Fraundorfer *
* Graz University of Technology, † Stuttgart Laboratory 1, SSS-EU, Sony Europe B.V.
* name.surname@icg.tugraz.at, † name.surname@sony.com

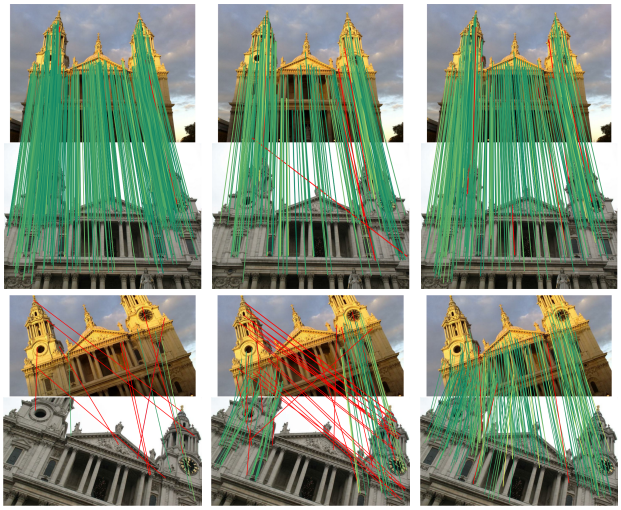
Abstract

In this work we introduce S-TREK, a novel local feature extractor that combines a deep keypoint detector, which is both translation and rotation equivariant by design, with a lightweight deep descriptor extractor. We train the S-TREK keypoint detector within a framework inspired by reinforcement learning, where we leverage a sequential procedure to maximize a reward directly related to keypoint repeatability. Our descriptor network is trained following a "detect, then describe" approach, where the descriptor loss is evaluated only at those locations where keypoints have been selected by the already trained detector. Extensive experiments on multiple benchmarks confirm the effectiveness of our proposed method, with S-TREK often outperforming other state-of-the-art methods in terms of repeatability and quality of the recovered poses, especially when dealing with in-plane rotations.

1. Introduction

Being able to find point correspondences between images has been of paramount importance since the early days of computer vision. In fact, a wide range of applications, such as Structure from Motion (SfM) [35, 1], Visual Localization [40, 33], SLAM [23, 2], object recognition [24] and object tracking [47] rely on image-to-image point correspondences.

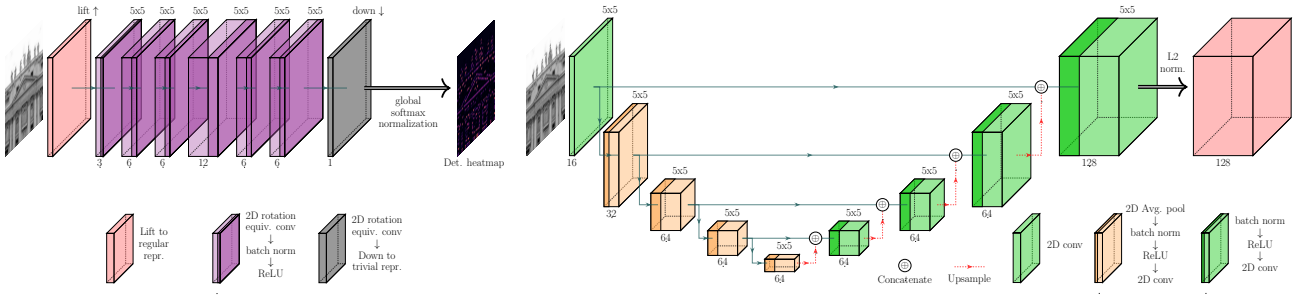
After decades where SIFT [19], SURF [6], ORB [30] and many other hand-engineered feature extractors have been ubiquitous, the community has recently experienced a fast shift toward learned methods, with several ones based on deep architectures [12, 11, 27, 44]. While many of these newly proposed methods show remarkable matching performances, the commonly used multi-layer convolutional architecture lacks one of the properties that most hand-engineered keypoint detectors have by design: rotation equivariance. Rather unexpectedly, modern detectors show poor performances when the input image undergoes in-plane rotations unless specifically trained to handle this transformation. In order to avoid this pitfall, we take advantage of the recent developments in the field of group-equivariant networks [9, 10] and design the keypoint detector of our novel feature extractor method, named S-TREK, to make use of rotation-equivariant convolutional layers. This makes our keypoints independent of the image orientation by design, regardless of the dataset used for training.



(a) DISK [44] (b) REKD [17] (c) S-TREK (ours)

Figure 1: Qualitative comparison with two state-of-the-art feature extraction methods on the Image Matching Benchmark [15] (top) and on our $\pm$ 45° rotated version of it (bottom). RANSAC inlier matches are color coded from green to yellow, representing reprojection errors equal to zero and 5px, respectively; the outlier matches are in red.

The detection of keypoints in an image is a hard selection process, where a finite set of locations is selected. Because of the non-differentiability of this process, some keypoint detection methods in the literature resort to training with with proxy losses, which are applied to the entire detection heatmaps at the network output [27, 31], while others train for keypoints and descriptors jointly [12, 20]. To train di-



(a) Keypoint detector.

(b) Descriptor extractor.

Figure 2: Overall architecture of the S-TREK feature extractor.

rectly with the keypoints location, we propose a training framework inspired by reinforcement learning, which permits to maximize a reward formulation directly related to the keypoints repeatability. Following [44, 7], we frame the keypoint detection as a probabilistic process, and present a novel *sequential sampling* procedure that does not suffer from the limitations of their sampling schemes.

In contrast to the more recent *"detect and describe"* approach, where keypoint locations and descriptors are learnt jointly using a shared backbone for both tasks [27, 12, 44, 31, 7], we follow a *"detect, then describe"* paradigm using two different networks for the two tasks, and train the descriptors only after the detector has been trained. This allows us to design the two networks to have different properties.

Recent studies have also explored new research directions, utilizing deep matching architectures like SuperGlue [32], or developing methods able to find pointwise correspondences directly from image pairs such as LoFTR [38]. However, these methods require execution for each individual image pair, which limits their applicability in scenarios where the computational resources are constrained, or when dealing with a large number of images. For these reasons, and the additional benefit of an easier integration in existing systems, local feature extraction methods remain the most popular approach in many applications.

In summary, our main contributions are:

- We propose S-TREK, a local feature extractor that combines a deep translation and rotation equivariant keypoint detector with a lightweight descriptor extractor. Our network is trained from scratch following a *"detect, then describe"* approach.
- We propose a keypoint detector training framework, inspired by reinforcement learning, based on a novel reward formulation that maximizes the repeatability metric directly. Additionally, we propose a novel sequential keypoint probability sampling strategy that overcomes the limitations of previous approaches.
- Extensive experiments show that the S-TREK detector achieves state-of-the-art repeatability on multiple benchmarks. Moreover, when equipped with our lightweight features extractor network, S-TREK provides features that are well suited for recovering accurate camera poses, especially when dealing with in-plane rotation.

2. Related works

A multitude of local feature extractors have been proposed in the last decades [43, 3, 36]. In the most classical approaches, the local feature extraction task is separated into two distinct steps: detecting keypoints and extracting descriptors. The two tasks are approached by means of specialized algorithms, obtaining keypoint-descriptor pairs as a result [14, 29, 8, 19, 6]. The early deep learning approaches have been designed following the same *"detect, then describe"* approach, with initial works addressing either keypoint detection [45, 37, 34, 5] or descriptor extraction from normalized image patches [22, 41, 42]. In [13], a log-polar sampling scheme is used to extract descriptors which result more robust to scale differences. Later works propose to design the descriptor network to output a dense volume where descriptors are sampled at keypoint locations [46, 11, 25], shifting toward a *"detect and describe"* approach where the two tasks are trained jointly [12, 27, 20, 31, 44, 7]. The rationale behind this shift is keypoint matchability: a keypoint, regardless of his repeatability, could be surrounded by a non-discriminative region (e.g. textureless patch, repetitive structures) which makes it hard to match its associated descriptor correctly. While this is true if the descriptors are computed based solely on small patches around each keypoint as in the early methods, increasing the descriptor support regions permits to assign a discriminative descriptor to most of the keypoints. Based on this observation, we adopt a *"detect, then describe"* approach and design our method with two distinct networks, each one with an architecture specialized for the specific task.

This allows us to design our keypoint detector network with rotation equivariant convolutions as in REKD [17], thus obtaining detections robust to in-plane rotation by design.

To circumvent the non-differentiability of the keypoint selection process, REKD [17] uses a window-based keypoint detection loss inspired by [5] that softens the keypoints selection by means of a spatial softmax, but requires to fix the number of keypoints used during training. Other recent methods approach the local feature learning with training frameworks that draw inspiration from reinforcement learning, such as DISK [44] and Reinforced Feature Points [7]. While these methods optimize for both keypoints and descriptors at the same time, for the aforementioned reasons we focus first on the keypoints, postponing the descriptor learning to a later stage. DISK [44] models the probabilistic sampling of keypoints by dividing the input image into a regular grid and sampling one keypoint from each cell. However, this approach has some shortcomings. For instance, even if a cell contains no stable keypoints, a keypoint will still be sampled from it. Additionally, if two valid keypoints are present in the same cell, only one of them can be sampled. Moreover, if a keypoint falls on one of the cell edges, it is likely to be sampled by both cells with only a 1px offset. In Reinforced Feature Points [7] instead the probabilistic sampling is modeled on top of an already trained SuperPoint network [11], sampling multiple times from the whole image with the danger of repeatedly sampling the same keypoint. Our approach is to use a *Sequential Sampling* procedure which solves all these issues. We do not soften the keypoint selection by averaging coordinates like [7], but by means of a probabilistic sampling. We do not require the image to be divided into cells, as we normalize the whole computed detection heatmap at once and sample from it directly, thus avoiding all the cell related issues. We only fix the max number of keypoints sampled during training, but not the minimum one; our sampling procedure has an early-stopping mechanism where no further keypoints are sampled if no stable keypoints are left in the image. Finally, we avoid the double-sampling of the same point by applying a *sampling avoidance radius* at each sampled keypoint, which forces the next sampled keypoint to lay at least N pixels from any already sampled ones.

3. Preliminaries on Equivariance and Invariance

The purpose of this section is to provide the reader with a basic understanding of the *invariance* and *equivariance* properties. Although we will mostly deal with equivariance in this paper, it is useful to agree on the vocabulary and spend a few words on what invariance means.

Given the sets X, Y and a set G of actions $g : X \to X$, a function $f : X \to Y$ is invariant with respect to G if, for any $g \in G, f[g(x)] = f(x)$.

For example, if f measures the area of a domain in the plane, such a function would be invariant with respect to translations but not with respect to changes of scale.

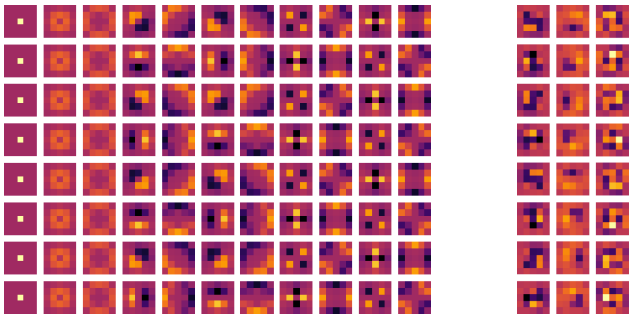


Figure 3: Basis (left) and resulting filters (right) of the first layer of our rotation equivariant keypoint detector. Each column of filters is a linear combination, with learnable weights, of the columns from the basis, and contributes to a single output channel.

If the domain and codomain of f are the same, that is $f : X \to X$, equivariance with respect to the set of actions G is defined as the property guaranteeing $f[g(x)] = g[f(x)]$.

With few approximations, the frequently used convolutional layer can be considered equivariant with respect to translations, and this property carries over to a network consisting of several layers. Formally, if $X = \mathbb{R}^{H \times W}$ is the set of input images, T is the set of translations $t : \mathbb{R}^{H \times W} \to \mathbb{R}^{H \times W}$, and f is the operation carried out by our network, the following holds true:

$$f[t(x)] = t[f(x)], \forall t \in T. \tag{1}$$

Equivalently, if the input undergoes a translation of an integer number of pixels in the x or y direction, the resulting network output will be translated by the same amount. However, the same is not true in general for other geometric transformations that our input might undergo.

In this work, we make use of special convolutional layers that guarantee the equivariance property with respect to the set of in-plane rotations T_R of the input image, where the function f represents our keypoint detector deep network f_θ, and θ represents the learnable parameters of the model.

For the purpose of this work, the preliminary introduction we just provided will be sufficient to frame and understand our contribution. We refer to [9, 21] for further details.

4. Method

In contrast to many recent works that train jointly for the two tasks of keypoint detection and descriptor extraction, our approach follows the *"detect, then describe"* paradigm. Our architecture is inherently partitioned into two parts: the *detector* and *descriptor extractor*, which are trained separately. By adopting this approach, we are able to address the

two tasks individually and employ specialized architectures for each of them.

We define a keypoint just as *repeatable* point, in other words a point that can be reliably detected again in another image. This definition also includes points surrounded by textureless areas or that lay on repetitive structures, which are notoriously difficult to match. To address this challenge, we equip our descriptor network with a very large receptive field and train the descriptors only at the exact locations where our keypoints are detected to avoid wasting descriptor space for non-repeatable regions. Conversely, we argue that the detector should concentrate solely on local structures, designing its backbone with a small receptive field.

Our definition of keypoint requires the detections to remain consistent under any transformations, whether photometric or geometric, applied to the image. While CNNs have proven to be easily trained to be robust with respect to photometric distortions, and translation equivariance is inherent in the convolutional architecture, robustness against other transformations must be encoded in the network's learned weights. Recent developments in the field of Steerable Filters and Group-equivariant networks [9, 10] have made it possible to effectively embed in-plane rotation equivariance in neural networks. In a rotation-equivariant convolution (ReCONV), instead of learning the kernel weights like in a standard convolution, the layer learns a weight for each precomputed basis and generates the convolution kernel as a weighted basis sum. This greatly reduces the number of learnable parameters compared to a standard convolution with the same kernel size. In Figure 3 we show the basis and and resulting kernels for our first *detector* layer.

4.1. Detector

Our *detector* architecture is composed by a series of ReCONV layers. The first layer transforms the input features into the chosen *regular representation* which depends on the encoded rotation cyclicity (i.e. the number of angles at which the basis are computed). The middle section applies convolutions on the *regular representation*. Finally, the last layer converts the feature representation back to a single-channel heatmap, which is then normalized with a *temperature softmax* operation. This network runs on the input gray-scale image $\boldsymbol{I} \in \mathbb{R}^{H \times W}$ and outputs the heatmap $\boldsymbol{D} \in \mathbb{R}^{H \times W}$. A scheme of the network architecture is visible in Figure 2a.

In order to train the detector, we propose a simple formulation that applies to the extracted keypoints, rather than to the output heatmap $\boldsymbol{D}$, and aims at maximizing the keypoints repeatability directly. Typically the keypoint extraction process relies on the $argmax$ operation, which prevents backpropagation to the network weights. In order to overcome this, we draw inspiration from reinforcement learning and frame the learning process as an expected reward maximization. We adopt a probabilistic approach to sample keypoints

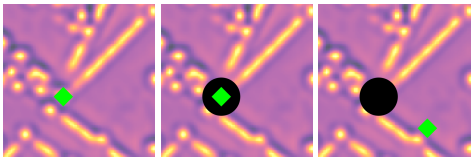


Figure 4: Sequential sampling process. A keypoint is sampled from the weight map obtained by normalizing the heatmap at the keypoint detector output. Then, all the weights in a *sampling avoidance radius* are set to 0. A new keypoint can now be sampled from the updated weight map and the process is iterated.

and use the *policy gradient* [39] algorithm to update the network weights after each training step.

4.1.1 Reward Computation

Our reward formulation is very straightforward, given two input images $\boldsymbol{I}_0, \boldsymbol{I}_1 \in \mathbb{R}^{H \times W}$ and the invertible relation $g_{0 \to 1} \colon \mathbb{R}^2 \to \mathbb{R}^2$ that projects the coordinate of a keypoint from the first image into the second image, the reward for the i-th keypoint k_i^0 in the first image is computed as:

$$r_{k_i^0} = \begin{cases} d_{\text{max}} - d & \text{if } d \leq d_{\text{max}} \\ r_n & \text{otherwise} \end{cases} \tag{2}$$

where d_{max} is a defined *reward radius*, r_n is the *negative reward* assigned to keypoints that are not considered repeatable and d is the distance between the projected keypoint k_i^0 and the closest keypoints in the second image. This reward function relates directly to the repeatability measure, defined later in Sec 6, as it is computed in an equivalent way.

4.1.2 Sequential Sampling

For the *policy gradient* [39] algorithm to work the expected reward must be computed. In our scenario, this involves a probabilistic selection of the keypoints.

The other methods in the literature [44, 7] either divide the image into cells and sample one keypoint for each cell or sample multiple keypoints independently from the normalized detection map. The main limitation of the first approach is that one keypoint is sampled from every cell, regardless of whether there are two high probability points or none. Furthermore, the grid size defines both the spacing between the keypoints and the total number of sampled keypoints. Finally, peaks that lie on the border between two adjacent cells are likely to be sampled in both just with a 1px offset. The second approach, instead is susceptible to the problem of sampling the same keypoint multiple times. To tackle both issues, we propose a *sequential sampling* procedure. First, we normalize the heatmap $\boldsymbol{D}$ by applying the *softmax* operation with temperature t:

$$\overline{\boldsymbol{D}} = \text{softmax}\left(\frac{\boldsymbol{D}}{t}\right). \tag{3}$$

Then, we treat $\overline{\boldsymbol{D}}$ as a weight map and draw N keypoints sequentially with a probability proportional to the corresponding pixel weight. To avoid sampling both the same keypoint twice or spatially close keypoints, after each sampling we set to zero all the weights in a *sampling avoidance radius* around the sampled keypoints. The sampling process stops when the sum of all the remaining weights is below a certain threshold, as this suggests that no stable keypoints are left. This allows the network to tune the number of keypoints sampled during each training iteration automatically. Figure 4 shows a graphical representation of this process.

4.1.3 Weight update

For each training iteration we run our detector network on a pair of images and then sample from the obtained heatmaps the two sets of keypoints K^0 and K^1.

We compute the reward for each keypoint and estimate the total expected reward as follows:

$$\mathbb{E}\left[R(K^0, K^1)\right] = \sum_{k \in K^0} p(k) r_k + \sum_{k \in K^1} p(k) r_k \tag{4}$$

where $p(k)$ refers to the probability of sampling the keypoints k, which is approximated with the value of the normalized heatmap $\overline{\mathbf{D}}$ at the coordinate of the keypoint.

Using the *policy gradient* [39] technique we can then estimate the gradient of the expected reward with respect to the network parameters θ as follows:

$$\begin{aligned} &\nabla_\theta \, \mathbb{E}\left[R(K^0, K^1)\right] = \\ &\underset{k \in K^0}{\text{mean}} \left[\nabla_\theta \log\left(p(k)\right) r_k\right] + \underset{k \in K^1}{\text{mean}} \left[\nabla_\theta \log\left(p(k)\right) r_k\right] \end{aligned} \tag{5}$$

and use this gradient to optimize the network weights.

4.2. Descriptor extraction

We design our *descriptor* network as a four-level U-Net [28] with a single convolutional layer per level, as shown in Figure 2b. The network is fed with the same grayscale image $\boldsymbol{I}$ and outputs a dense L2-normalized descriptor volume $\boldsymbol{V} \in \mathbb{R}^{H \times W \times d}$ where d is the descriptor dimension. The descriptor associated to a detected keypoint is simply read from the volume at the keypoint location. We employ a hinged triplet loss formulation [22]:

$$\mathcal{L}_{\text{Descriptor}} = \underset{t \in \mathcal{T}}{\text{mean}} \left[\max(0, m + s_n^t - s_p^t)\right] \tag{6}$$

where t is a triplet from the pool $\mathcal{T}$ of all the chosen anchor-positive-negative triplets, m is the margin, s_n is the anchor-negative score and s_p is the anchor-positive score. The triplets are built following the *hardest* strategy [22], where we pick the closest non-matching descriptor for each anchor.

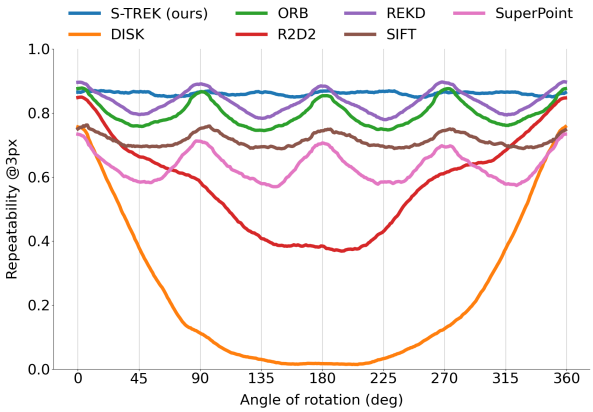


Figure 5: Repeatability as a function of image rotation angle. The curves are smoothed with a 15 degrees moving average.

5. Training Setup

We train the two networks sequentially on the subset of MegaDepth [18] provided by [44]. The dataset comprises images, camera poses, camera intrinsics and depth maps of 3D reconstructed touristic locations. The subset includes a total of 135 scenes and is the result of removing the scenes with unreliable depths and those overlapping with the Image Matching Benchmark [15]. We draw random image pairs from the provided list of 10k image pairs per scene. We rescale each image such that the shortest edge measures 512px and crop the other dimension to obtain 512×512 image patches. For the descriptor training, we apply an additional random rotation sampled from a uniform distribution [-30, 30] degrees to each image.

The first training involves only the *detector* network, where we use the ESCNN library [9] for the ReCONV and set the cyclicity to 8 (the basis are generated for each $2\pi/8$ rotation). We train from scratch using the Pytorch framework [26], Adam optimizer [16] with betas (0.9, 0.999), learning rate 1e-4 and batch size 4. We set the *sampling avoidance radius* to 6px, the reward radius d_{max} to 3px, the number of max sampled points to 1000 and start with the negative reward r_n set to 0, decreasing it from the 1000th iteration onward on with a linear slope of -1e-5. We fix the *softmax* temperature to 100. We train for 5k iterations, which takes only 5h on a single Nvidia RTX 2080Ti.

Only after the keypoint *detector* training is complete, the learned keypoints can be used to train the *descriptor* network. Consequently, the *descriptor* network can concentrate solely on distinguishing the areas surrounding the detected keypoints and ignore all other regions. We set the triplet margin m to 0.5, the descriptor dimensionality d to 128 and train with the same batch size and learning rate as above. The anchor-positive pairs are built from keypoint pairs that projects closer than 3px. We found it beneficial to start by

sampling random negatives instead of the hardest, lowering this random sampling probability exponentially and setting it to 0 after 10k iterations. We train for 90k iterations, which takes around 24h on a single Nvidia RTX 2080Ti.

6. Experiments

We evaluate our method and compare with other state-of-the-art feature extraction methods via three different experiments. The first experiment regards only the keypoint *detector* and is meant to evaluate its ability to find repeatable keypoints under in-plane rotation. The other two experiments regard two commonly used benchmarks in the literature and test the performances of keypoints and descriptors jointly. The metrics evaluated are the following:

- **Repeatability**: measures the capability of a method to detect the same keypoints in a pair of images depicting the same scene. It is computed as the fraction of keypoints from the first image whose projection into the second image has at least one keypoint from the second image within T pixel distance [11].
- **MMA**: The *Mean Matching Accuracy* is computed as the number of correct matches (up to T pixels) over the total number of matches proposed [12].
- **MS**: The *Matching Score* is computed as the number of correct matches (up to T pixels) over the average number of detected keypoints in the overlap area [11].
- **Homography accuracy AUC**: It is computed as the Area Under the Curve of the fraction of recovered homographies that have a corner error [11] below T pixels. The corner error is computed as the average distance between the reference image corners and the corners of the source image reprojected using the homography estimated from the matches.
- **mAA**: The *mean Average Accuracy* is computed as the *Area Under the Curve* of the fraction of recovered relative camera poses with an error within a specified threshold [15].

During inference, our ReCNN architecture comes at no additional cost compared to a standard CNN, and the probabilistic *sequential sampling* is replaced with a Non-Maximum suppression with a 3px radius and extracts the N keypoints with highest score. On *Hpatches* and *IMB* we run both our networks on a 5-level image pyramid with scale factor $1/\sqrt{2}$.

6.1. Repeatability under rotations

We replicate the experiment conducted in [17]. Specifically, we select the first ten images from *HPatches* [4], we generate in-plane rotated versions of these images with an increment of 1° from 0° to 360° and we center-crop them to size 224×224. As in the original experiment [17], we apply a light gaussian noise to each rotated image. We set a budget of 50 keypoints and run all the learnt methods single-scale. Keypoints are extracted from each image and the *repeatability* is computed against the angle 0° image. The results are depicted in Figure 5, where S-TREK obtains high and very stable repeatability at any angle, showing only minor oscillations. REKD [17], despite employing a similar rotation-equivariance backbone, shows a more pronounced oscillation. While SIFT [19] has lower but consistent repeatability, ORB [30] displays remarkable repeatability values but also more oscillation. Finally, while SuperPoint [11] shows modest robustness to rotation, the other deep methods perform poorly with angles higher than 90°, with DISK [44] reaching values close to zero at 180°. Additional experiments on this dataset are performed in Sec 7.1.

	Method	Rep ↑			MMA ↑			MS ↑			Hom. Acc.
		@1px	@2p	@3px	@1px	@2p	@3px	@1px	@2px	@3px	AUC@3px
standard	DISK [44]	0.29	0.45	0.54	**0.45**	**0.67**	**0.76**	**0.28**	**0.40**	**0.45**	0.438
	SuperPoint [11]	0.23	0.41	0.54	0.31	0.51	0.62	0.19	0.32	0.38	0.413
	R2D2 [27]	**0.31**	0.50	0.60	0.34	0.61	0.74	0.16	0.27	0.32	0.441
	REKD [17]	0.22	0.42	0.54	0.26	0.49	0.62	0.17	0.31	0.39	0.416
	S-TREK (ours)	**0.31**	**0.51**	**0.61**	0.35	0.58	0.71	0.22	0.35	0.42	**0.449**
± 20°	DISK [44]	0.23	0.42	0.52	**0.36**	**0.62**	**0.71**	**0.20**	**0.33**	**0.38**	0.329
	SuperPoint [11]	0.17	0.37	0.51	0.22	0.45	0.57	0.13	0.27	0.34	0.289
	R2D2 [27]	**0.27**	**0.48**	0.57	0.27	0.55	0.68	0.09	0.18	0.22	0.293
	REKD [17]	0.17	0.38	0.51	0.20	0.44	0.58	0.12	0.26	0.34	0.224
	S-TREK (ours)	**0.27**	**0.48**	**0.58**	0.30	0.56	0.67	0.17	0.30	0.36	**0.336**
± 45°	DISK [44]	0.18	0.36	0.47	0.21	0.37	0.43	0.11	0.18	0.21	0.182
	SuperPoint [11]	0.14	0.33	0.47	0.16	0.33	0.42	0.09	0.19	0.24	0.181
	R2D2 [27]	0.22	0.42	**0.53**	0.16	0.33	0.40	0.05	0.09	0.11	0.165
	REKD [17]	0.16	0.36	0.48	0.17	0.39	0.51	0.10	0.22	**0.29**	0.184
	S-TREK (ours)	**0.25**	**0.44**	**0.53**	**0.24**	**0.45**	**0.54**	**0.13**	**0.23**	0.28	**0.248**

Table 1: Comparison on HPatches - keypoints budget 2048.

6.2. HPatches

HPatches [4] is a dataset composed by two sets of pictures: planar scenes captured from different angles and static photos captured in different lighting conditions. We run our evaluation using the 108 scenes subset of [12]. Each scene is composed by one reference image and five source images; all the metrics are computed between the reference and all the sources. To test the method stability with respect to rotation, we generate two additional versions of the whole dataset applying random rotations, sampled uniformly between ±20° and ±45° to each source image. Each rotated image is cropped to match the rectangle with the largest area possible to avoid any border artifacts. This makes the rotated version of the benchmark even more challenging due to the lower overlap between each source and reference image. We compute the relative homography using the OpenCV `findHomography` function (10k iterations) with multiple RANSAC thresholds (0.125, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0). For each method, we compute the homography accuracy AUC at each RANSAC threshold and then pick the best value. For a more fair evaluation, we run all the multi-scale methods starting from the original image resolution, fix the keypoints budget to 2048 and always use the same

	Method	Repeatability@3px ↑										N. matches inliers ↑										mAA@10 ↑									
		BM	FLC	LM	LB	MC	MR	PSM	SF	SPC	**AVG**	BM	FLC	LM	LB	MC	MR	PSM	SF	SPC	**AVG**	BM	FLC	LM	LB	MC	MR	PSM	SF	SPC	**AVG**
Standard	DISK [44]	**0.58**	0.42	0.39	**0.47**	0.53	0.43	0.36	0.38	0.47	0.448	**572**	**400**	327	**350**	**556**	**393**	**265**	**367**	**408**	**404**	**0.41**	**0.69**	0.59	**0.58**	**0.53**	**0.38**	**0.26**	**0.58**	**0.59**	**0.512**
	SuperPoint [11]	0.42	0.36	0.37	0.35	0.38	0.41	0.30	0.35	0.34	0.364	113	122	115	101	102	130	48	174	82	110	0.23	0.44	0.46	0.33	0.22	0.20	0.14	0.38	0.24	0.295
	R2D2 [27]	0.50	0.40	0.36	0.38	0.53	**0.50**	0.34	0.43	0.43	0.429	215	192	220	148	229	271	150	217	172	202	0.25	0.60	0.60	0.43	0.33	0.30	0.18	0.42	0.40	0.390
	REKD [17]	0.54	**0.43**	**0.48**	0.40	0.49	0.45	**0.41**	0.41	0.44	0.450	284	298	**490**	217	273	310	212	325	254	296	0.26	0.66	**0.66**	0.41	0.37	0.28	0.18	0.49	0.50	0.423
	S-TREK (ours)	0.54	0.41	0.47	0.43	**0.54**	0.47	0.34	**0.46**	**0.48**	**0.460**	325	322	419	245	377	319	222	345	323	322	0.33	0.67	0.53	0.47	0.43	0.34	0.23	0.54	0.58	0.458
±20°	DISK [44]	**0.55**	0.39	0.39	**0.47**	0.53	0.43	0.36	0.37	**0.45**	0.438	288	198	147	195	315	224	150	206	221	216	0.16	0.44	0.39	0.37	0.35	0.22	0.17	0.38	0.40	0.320
	SuperPoint [11]	0.39	0.35	0.31	0.34	0.37	0.39	0.29	0.33	0.31	0.342	39	42	34	44	42	51	18	68	29	41	0.13	0.23	0.27	0.22	0.13	0.11	0.06	0.24	0.12	0.168
	R2D2 [27]	0.45	0.40	0.36	0.39	0.52	**0.49**	0.34	0.43	0.42	0.422	102	105	110	89	126	140	82	110	89	106	0.11	0.38	0.36	0.27	0.24	0.19	0.13	0.26	0.25	0.243
	REKD [17]	0.49	**0.42**	**0.46**	0.42	0.50	0.46	**0.40**	0.41	0.43	0.443	249	267	**386**	219	291	**273**	194	**291**	226	266	0.16	0.56	**0.58**	0.36	0.32	0.23	0.17	0.44	0.41	0.359
	S-TREK (ours)	0.52	0.41	0.43	0.46	**0.54**	0.47	0.35	**0.45**	**0.45**	**0.453**	**308**	**288**	329	**252**	**348**	258	**196**	280	**272**	**281**	**0.18**	**0.59**	0.46	**0.42**	**0.38**	**0.28**	**0.20**	**0.47**	**0.48**	**0.384**
±45°	DISK [44]	0.48	0.35	**0.37**	0.42	0.48	0.39	0.32	0.31	0.41	0.392	125	92	68	80	138	109	69	96	86	96	0.07	0.19	0.19	0.14	0.16	0.10	0.07	0.18	0.15	0.139
	SuperPoint [11]	0.37	0.33	0.30	0.31	0.35	0.36	0.28	0.31	0.30	0.323	18	18	18	20	18	25	10	27	14	19	0.04	0.08	0.11	0.07	0.04	0.04	0.02	0.09	0.04	0.059
	R2D2 [27]	0.39	0.38	0.32	0.36	0.49	**0.47**	0.32	0.39	0.40	0.391	56	53	53	45	65	71	43	56	43	54	0.04	0.15	0.16	0.10	0.10	0.08	0.06	0.11	0.08	0.098
	REKD [17]	0.46	**0.41**	**0.40**	0.39	0.50	0.45	**0.37**	0.40	0.42	0.422	178	**178**	**207**	146	**212**	**186**	**138**	**184**	145	**175**	0.09	**0.35**	**0.36**	0.20	0.21	0.13	**0.13**	**0.28**	0.21	0.218
	S-TREK (ours)	**0.49**	**0.41**	0.36	**0.44**	**0.52**	0.44	0.33	**0.43**	**0.43**	**0.428**	**201**	176	160	**150**	205	150	114	147	**154**	162	**0.10**	**0.35**	0.27	**0.21**	**0.24**	**0.16**	**0.13**	**0.28**	**0.26**	**0.222**

Table 2: Experiments on Image Matching Benchmark [15]. The keypoints budget is 2048.

Mutual-Nearest-Neighbor matching strategy.

The results of our evaluation are reported in Table 1. We can observe that the S-TREK detector holds the best repeatability values at every threshold on each rotation set, thus validating our proposed detector architecture and training scheme. Regarding MMA and MS, DISK [44] is the clear winner for the standard and $\pm 20°$ sets, with S-TREK following it closely on the $\pm 20°$ and overtaking it on the $\pm 45°$ set. For the homography recovery task, MMA is not crucial, as most of the wrong proposed matches are filtered out by RANSAC. While MS is a better predictor for the homography accuracy, it does not capture how the matches are distributed in the image, which is of primal importance for a precise homography recovery. The high homography accuracy obtained by S-TREK backs our *"detect, then describe"* approach and confirms our method capabilities to find repeatable, reliable and well distributed local features.

6.3. Image Matching Benchmark

We evaluate the performance of local features for the task of *stereo pose recovery* on the restricted keypoint category (2048 keypoints per image) of the Image Matching Benchmark [15] phototourism set. The benchmark consists of 9 scenes of famous tourism sites, each comprising 100 images with various degrees of overlaps. The online evaluation server has recently been disabled. For this reason, all the evaluations reported on this paper for methods that were not available in the online leaderboard have been run locally. Similarly to the previous experiment in Sec 6.2, we generated two additional instances of the benchmark by applying random $\pm 20°$ and $\pm 45°$ rotations, which also in this case become more challenging due to the decreased overlap between images. We run all the methods without any additional outlier rejection stage and using the Mutual-Nearest-Neighbor matcher. Where available, we use the matcher parameters and RANSAC threshold suggested by the authors. Specifically, for S-TREK we use a minimum matching score of 0.5 and a RANSAC threshold of 1.0 px.

The results for *repeatability*, *number of inlier matches* and *mAA@10* for each scene in the dataset are reported in Table 2. Similarly to the *HPatches* experiments, S-TREK is the method with the highest repeatability in all the benchmark sets. While DISK [44] obtains the best *number of inlier matches* and *mAA@10* in the standard instance, S-TREK follows it closely with a competitive 0.458 *mAA@10*. On the rotated versions of the benchmark DISK falls behind S-TREK and REKD, which achieve similar performance regarding the *number of inlier matches*. S-TREK provides the best *mAA@10* for both angle sets, confirming again the effectiveness of our approach. Figure 1 shows a qualitative comparison between the best performing methods where S-TREK is the only method which provides well distributed matches in both standard, and $\pm 45°$ benchmark instances.

7. Ablation Studies

7.1. Rotation equivariant architectures comparison

We repeat the experiment from Section 6.1 using three different keypoint detector architectures:

- **ReCNN cyclic 8**: Baseline S-TREK *detector* architecture ($\sim$20k learnable params).
- **smaller ReCNN cyclic 8**: shallow ReCNN with cyclicity 8 ($\sim$5k learnable params).
- **smaller ReCNN cyclic 16**: shallow ReCNN with cyclicity 16 ($\sim$10k learnable params).

The results are shown in Figure 6. It can be observed that the smaller ReCNN with cyclicity 8 exhibits a repeatability with lower mean and larger fluctuations than the baseline. Increasing the cyclicity to 16 smooths out the fluctuations but does not increase the mean repeatability. The baseline, i.e. S-TREK detector, is able to reduce the fluctuations without requiring higher cyclicity thanks to the deeper expressivity given by the larger number of convolutional layers (7 vs 4).

7.2. Different losses and architectures

We compare our sequential and reinforcement learning inspired keypoint training against the commonly used *peaky*

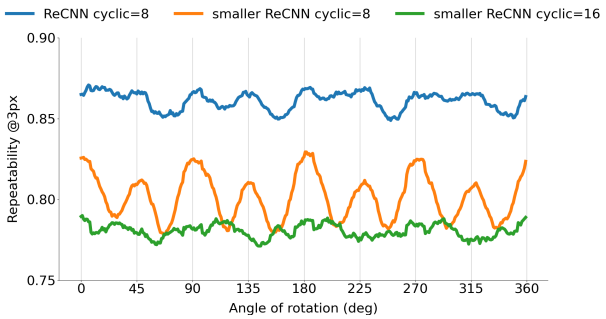


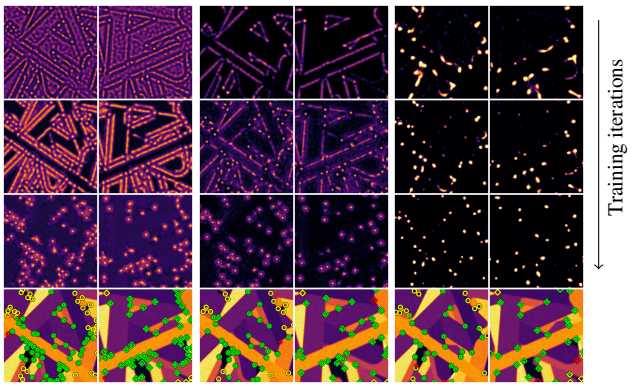
Figure 6: Comparison between keypoint detector architectures in terms of keypoint repeatability as a function of image rotation angle. ReCNN cyclic $= 8$ is the S-TREK detector.

and *similarity* losses [27, 31], which need to be used in conjunction and are applied to the detection heatmap. The *peaky* loss encourages local peaks and is controlled by a window-size parameter, while the *similarity* loss promotes similar detection heatmaps in a dense manner.

To analyse the convergence properties of the different training formulations, we train on a synthetic dataset generated by drawing random grayscale lines plus random gaussian noise. Every pair of images is obtained applying two random homography warpings to a generated lines image.

The heatmap evolution during training is shown in Figure 7 and provides insights on the convergence process. Regardless of the training strategy, the network starts by highlighting both edges and corners, progressively favouring corners more. However, in this process, the peakyness loss tends to loose good points. In contrast, our training framework remains more stable in terms of number of the detected points, regardless of the number of chosen samples. This is confirmed by the numerical values in Table 3. The middle section reports the results of our S-TREK keypoint *detector* trainings using different *peaky* window sizes. It can be noticed that the number of detected keypoints during inference varies significantly depending on the chosen *peaky* window size, ranging from 27 to 60. The first section instead shows the results with a varying number of sequential samples, for which the number of keypoints is more stable, confirming the ability of our training scheme to adapt the number of keypoints to the number of existing stable points.

In the last section of the same table we compare the results of training different standard Convolutional Networks. The *CNN same channels* refers to a standard CNN with the same number of layers and channels shown in Figure 2a, while *CNN equivalent* refers to a standard CNN with the same number of layers shown in Figure 2a where the channel count has been multiplied by the S-TREK cyclicity. The ReCNN, in conjunction with our training framework, obtains the highest *max repeatability* values while still keeping a high keypoints count, confirming the ability of our method to learn to detect the stable points available in the dataset.



(a) N. samples 200 (b) N. samples 50 (c) Peaky W=64

Figure 7: Visual comparison between different training methods on our lines dataset. Last row: input images and detected keypoints (Green - repeatable, Red - non repeatable, Yellow - non overlapping). First three rows: detector heatmaps evolution during training. Our training framework excels at finding repeatable points without any direct supervision.

Architecture	Training	n. kpts	Rep. max. ↑ @1px	@2px	@3px	Learnable Parameters
ReCNN	Sequential n. samples 50	39	0.77	0.92	0.93	20k
	Sequential n. samples 100	53	**0.78**	**0.93**	**0.95**	20k
	Sequential n. samples 200	56	0.72	0.91	0.94	20k
	Peaky + Similarity W 32	60	0.51	0.72	0.78	20k
	Peaky + Similarity W 64	36	0.58	0.82	0.88	20k
	Peaky + Similarity W 96	27	0.61	0.85	0.90	20k
CNN same channels	Sequential n. samples 100	19	0.61	0.85	0.90	6k
CNN equivalent	Sequential n. samples 100	38	0.68	0.90	0.92	377k

Table 3: Comparison between different training approaches on the validation set of our lines dataset (see Figure 7).

8. Conclusion and future works

This paper presents S-TREK, a local feature extractor method that combines a deep keypoint detector that is both translation and rotation equivariant with a lightweight deep descriptor extractor. The proposed training framework and reward formulation, inspired by reinforcement learning, maximizes the keypoint repeatability score directly. Moreover, we model the probabilistic keypoint sampling process adopting a sequential scheme that overcomes the limitation of previous approaches. Extensive experiments show that the S-TREK detector often outperforms the state-of-the-art in the repeatability metric. Paired with our learnt descriptors, S-TREK achieves competitive matching performance, especially in applications with strong in-plane rotations.

A possible direction for future improvements of the S-TREK features could be to incorporate scale equivariance in the network architecture. This could further reduce the number of learnable parameters and, therefore, the amount of data and the computational efforts required by the training.

Acknowledgement: This work has been supported by the FFG, Contract No. 881844: ”Pro2Future”.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] Josep Aulinas, Yvan Petillot, Joaquim Salvi, and Xavier Lladó. The slam problem: a survey. *Artificial Intelligence Research and Development*, pages 363–371, 2008.

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5173–5182, 2017.

[4] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5173–5182, 2017.

[5] Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key. net: Keypoint detection by handcrafted and learned cnn filters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5836–5844, 2019.

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Aritra Bhowmik, Stefan Gumhold, Carsten Rother, and Eric Brachmann. Reinforced feature points: Optimizing feature detection and description for a high-level task. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[9] Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build E(N)-equivariant steerable CNNs. In *International Conference on Learning Representations*, 2022.

[10] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[11] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[12] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 8092–8101, 2019.

[13] Patrick Ebel, Anastasiia Mishchuk, Kwang Moo Yi, Pascal Fua, and Eduard Trulls. Beyond cartesian representations for local descriptors. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 253–262, 2019.

[14] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, 1988.

[15] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching across Wide Baselines: From Paper to Practice. *International Journal of Computer Vision*, 2020.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Jongmin Lee, Byungjin Kim, and Minsu Cho. Self-supervised equivariant learning for oriented keypoint detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[18] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[19] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[20] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6589–6598, 2020.

[21] Saunders Mac Lane. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

[22] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *arXiv preprint arXiv:1705.10872*, 2017.

[23] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[24] Jim Mutch and David G Lowe. Multiclass object recognition with sparse, localized features. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 11–18. IEEE, 2006.

[25] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. *arXiv preprint arXiv:1805.09662*, 2018.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[27] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[29] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[31] Emanuele Santellani, Christian Sormann, Mattia Rossi, Andreas Kuhn, and Friedrich Fraundorfer. Md-net: Multi-detector for local feature extraction. In *Proceedings 26th International Conference on Pattern Recognition (ICPR)*, pages 1 – 6, 2022.

[32] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[33] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8601–8610, 2018.

[34] Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sattler, and Marc Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1822–1830, 2017.

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Johannes L Schonberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1482–1491, 2017.

[37] Christoph Strecha, Albrecht Lindner, Karim Ali, and Pascal Fua. Training for task specific keypoint detection. In *Joint pattern recognition symposium*, pages 151–160. Springer, 2009.

[38] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.

[39] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[40] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(7):1455–1461, 2017.

[41] Yurun Tian, Bin Fan, and Fuchao Wu. L2-net: Deep learning of discriminative patch descriptor in euclidean space. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 661–669, 2017.

[42] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11016–11025, 2019.

[43] Tinne Tuytelaars and Krystian Mikolajczyk. *Local invariant feature detectors: a survey*. Now Publishers Inc, 2008.

[44] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *arXiv preprint arXiv:2006.13566*, 2020.

[45] Yannick Verdie, Kwang Yi, Pascal Fua, and Vincent Lepetit. Tilde: A temporally invariant learned detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5279–5288, 2015.

[46] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European conference on computer vision*, pages 467–483. Springer, 2016.

[47] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009. Special Issue on Video Analysis.