Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters Revisited

Axel Barroso-Laguna and Krystian Mikolajczyk

Abstract—We introduce a novel approach for keypoint detection that combines handcrafted and learned CNN filters within a shallow multi-scale architecture. Handcrafted filters provide anchor structures for learned filters, which localize, score, and rank repeatable features. Scale-space representation is used within the network to extract keypoints at different levels. We design a loss function to detect robust features that exist across a range of scales and to maximize the repeatability score. Our Key.Net model is trained on data synthetically created from ImageNet and evaluated on HPatches and other benchmarks. Results show that our approach outperforms state-of-the-art detectors in terms of repeatability, matching performance, and complexity. Key.Net implementations in TensorFlow and PyTorch are available online.

Index Terms—Local features, keypoint detector, image matching, 3D reconstruction

1 Introduction

Research advances in local feature detectors and descriptors led to remarkable improvements in areas such as image matching, object recognition, self-guided navigation, or 3D reconstruction. Although the general direction of image matching methods is moving towards learned based systems, the advantage of learning methods over handcrafted ones has not been clearly demonstrated in keypoint detection [1]. In particular, Convolutional Neural Networks (CNNs) were able to significantly reduce matching error in local descriptors [2], despite the impractical inefficiency of the initial techniques [3], [4]. These works stimulated further research efforts and resulted in improved efficiency of CNN-based descriptors. On the contrary, CNN keypoint detector research has not advanced as much as the descriptors due to their limited performance improvements, in addition to the growing interest in dense correspondence methods that do not rely on keypoints [5], [6], [7]. However, even with the increasing popularity of detector-free methods, local detectors are still broadly used in many applications [8], [9], and emerging technologies such as augmented reality (AR) headsets, as well as AR smartphone apps, have drawn more attention to reliable and efficient keypoint detectors that could be used for surface estimation, sparse 3D reconstruction, 3D model acquisition, or objects alignment, among others [8], [9].

Traditionally, local feature detectors were based on engineered filters. For instance, approaches such as Difference of Gaussians [10], Harris-Laplace, or Hessian-Affine [11] use combinations of image derivatives to compute feature maps, which is remarkably similar to the operations in trained CNN's layers. Intuitively, with just a few layers, a network could mimic the behavior of traditional detectors by learning the appropriate values in its convolutional filters. However, unlike the success with CNNs based local image descriptors, the improvements upon handcrafted detectors offered by recently proposed fully CNN based methods [12], [13], [14], [15], [16] are limited in terms of widely accepted metrics such as repeatability. One of the reasons is their low accuracy when estimating the affine parameters of the feature regions. Robustness to scale variations seems particularly problematic while other parameters such as dominant orientation can be regressed well by CNNs [12], [17]. This motivates our novel architecture, termed Key.Net, that makes use of handcrafted and learned filters as well as a multi-scale representation. The Key.Net architecture is illustrated in Fig. 1. Introducing handcrafted filters, which act as soft anchors, makes it possible to reduce the number of parameters used by state-of-the-art detectors while maintaining performance in terms of repeatability. The model operates on multi-scale representation of full-size images and returns a response map containing the keypoint score for every pixel. The multi-scale input allows the network to propose stable keypoints across scales thus providing robustness to scale changes.

Ideally, a robust detector can propose the same features for images that undergo different geometric or photometric transformations. Many related works have focused their objective function to address this issue, although they were based either on local patches [14], [15] or global map regression loss [16], [18], [19]. In contrast, we extend the covariant constraint loss to a new objective function that combines local and global information. We design a fully differentiable operator, Multi-scale Index Proposal, that proposes keypoints at multi-scale regions. We extensively evaluate the method in popular HPatches benchmark [2] in terms of accuracy and repeatability according to the protocol from [20]. Besides experiments in HPatches, we also evaluate Key.Net in terms of 3D reconstruction and camera localization metrics.

• *The authors are with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ London, U.K. E-mail: {axel.barroso17, k.mikolajczyk}@imperial.ac.uk.*

Manuscript received 23 July 2021; revised 9 Dec. 2021; accepted 18 Jan. 2022. Date of publication 25 Jan. 2022; date of current version 5 Dec. 2022.
This work was supported in part by Chist-Era EPSRC IPALM under Grant EP/S032398/1.
(Corresponding author: Axel Barroso-Laguna.)
Recommended for acceptance by V. Lepetit.
Digital Object Identifier no. 10.1109/TPAMI.2022.3145820

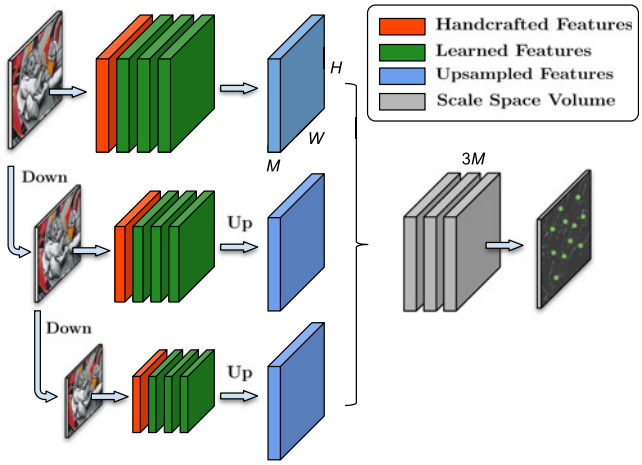


Fig. 1. The proposed Key.Net architecture combines handcrafted and learned filters to extract features at different scale levels. Feature maps are upsampled and concatenated. The last learned filter combines the Scale Space Volume to obtain the final response map.

In summary, our contributions are the following:

- A keypoint detector that combines handcrafted and learned CNN features.
- A novel multi-scale loss and operator for detecting and ranking stable keypoints across scales.
- A multi-scale feature detection with shallow architecture.

This manuscript extends the work from [21] with additional details and contributions:

- A revised and updated related work including the latest keypoint detectors and combined detector-descriptor networks.
- A new Key.Net implementation in PyTorch, including its training and evaluation scripts.
- Updated experiments reporting repeatability and matching with the latest joint detector-descriptor methods. Moreover, we extend the previous evaluation and analyze the performance of Key.Net on other practical problems, i.e., 3D reconstruction, visual localization, and camera pose estimation.

The rest of the paper is organized as follows. We review the related work in Section 2. Section 3 presents our proposed hybrid Key.Net architecture of handcrafted and learned CNNs filters, and Section 4 introduces the loss. Implementation and experimental details are given in Section 5, and the results are presented in Section 6.

2 Related Work

Many surveys extensively discuss feature detection methods [1], [22]. In this section, we present related works in two main categories: handcrafted and learned-based.

2.1 Handcrafted Detectors

Traditional feature detectors localize geometric structures through engineered algorithms, which are often referred to as handcrafted. Harris [23] and Hessian [24] detectors used first and second-order image derivatives to find corners or blobs in images. Those detectors were further extended to handle multi-scale and affine transformations [11], [25]. Later, SURF [26] accelerated the detection process by using integral images and an approximation of the Hessian matrix. Multi-scale improvements were proposed in KAZE [27] and its extension, A-KAZE [28], where the Hessian detector was applied to a non-linear diffusion scale space in contrast to the widely used Gaussian pyramid. Although corner detectors proved to be robust and efficient, other methods seek alternative structures within images. SIFT [10] looked for blobs over multiple scale levels, while SIFER [29] detected both corner and blob structures. Edge Foci [30] determined scale and position of local features by finding equidistant points to two oriented edges and MSER [31] segmented, and selected stable regions as keypoints.

2.2 Learned Detectors

The success of learned methods in general object detection and feature descriptors motivated the research community to explore similar techniques for feature detectors. FAST [32] was one of the first attempts to use machine learning to derive a corner keypoint detector. Further works extended FAST by optimizing it [33], adding a descriptor [34] or orientation estimation [35].

The latest advances in CNNs also made an impact on feature detection. TILDE [19] trained multiple piece-wise linear regression models to identify interest points that are robust under severe weather and illumination changes. [14] introduced a new formulation to train a CNN based on feature covariant constraints. The previous detector was extended in [15] by adding predefined detector anchors, showing improved stability in training. [13] presented two networks, MagicPoint, and MagicWarp, which first extracted salient points and then a parameterized transformation between pairs of images.

2.3 Key.Net Related Detectors

Besides previously introduced learned detectors, Key.Net has inspired new approaches. MSK [36] computes keypoint detection scores and scales by exploiting the information across different receptive fields. In their work, authors built an information change volume that contains the differences between features extracted by multiple receptive fields. A final convolutional block fuses the information change volume into detection and scale maps. MSK architecture is trained with the M-SIP loss function proposed in Key.Net. KeyReg [37] introduced a keypoint regressor that consists of several random forest classifier modules to compute repeatable and reliable keypoints candidates. KeyReg also combines handcrafted features with a learned classifier for keypoint detection. Finally, SobelNet [38] proposed a Gaussian filter and a Sobel edge function to detect clean corners in images. SobelNet inherited the multi-scale architecture proposed in Key.Net but replaced the handcrafted filters for a single Sobel operator.

2.4 Joint Detectors-Descriptors

MagicPoint was extended in [18] to SuperPoint, which included a salient detector and descriptor. A shallow and efficient architecture was proposed in [39] to emulate the response of KAZE [27] detector. LIFT [12] implemented an end-to-end feature detection and description pipeline, including the orientation estimation for every feature. Quadruple image patches and a ranking scheme of point responses as cost function were used in [40] to train a neural network. In [41], authors proposed a pipeline to automatically sample positive and negative pairs of patches from a

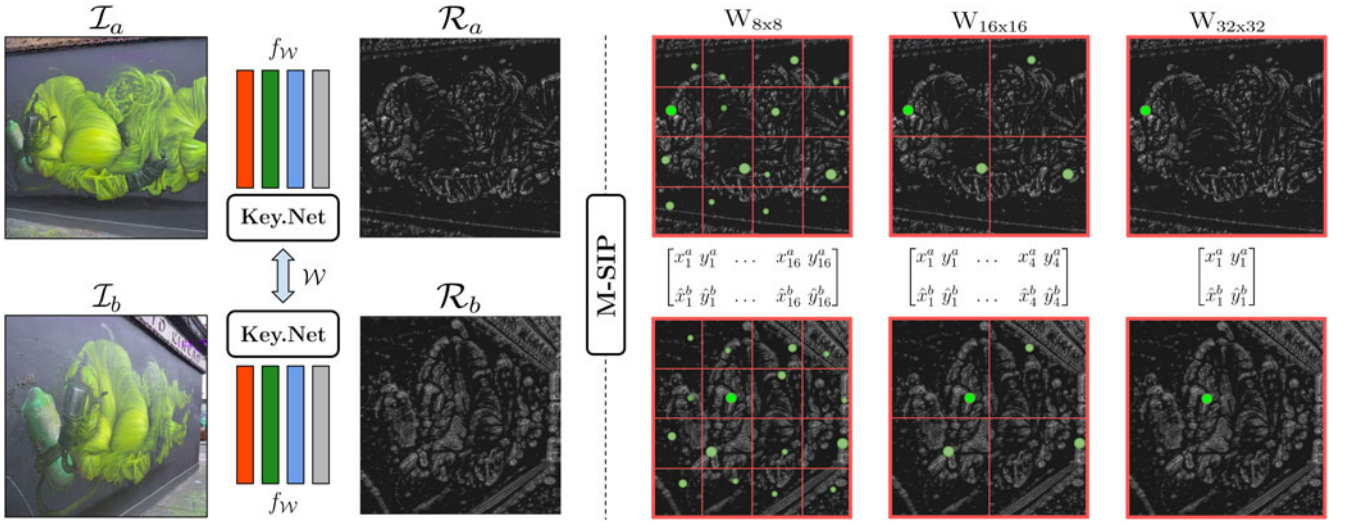


Fig. 2. Siamese training process. Image I_a and I_b go through Key.Net to generate their response maps, R_a and R_b. M-SIP proposes interest point coordinates for each one of the windows at multi-scale regions. The final loss function is computed as a regression of coordinate indexes from I_a and local maximum coordinates from I_b. Better visualize in color.

region proposal network to optimize jointly point detections and their representations. Recently, LF-Net [16] estimated the position, scale, and orientation of features by optimizing jointly the detector and descriptor. RF-Net [42] extended LF-Net by studying and using the information provided by its different receptive fields to improve its detections. D2-Net [43] and D2D [44] showed that feature detection could be directly done in the descriptor space. D2-Net demonstrated that a pre-trained network can be used for feature extraction even though it was optimized for a different task, meanwhile, D2D used pre-trained L2-Net [45] to perform detections in its feature maps. R2D2 [46] presented a dense variant of the L2-Net [45] architecture to predict descriptors and two keypoint score maps, which were each based on either keypoint repeatability or reliability scores. ASL-Feat [47] proposed a detector-descriptor architecture based on multi-level connections and deformable convolutional networks [48], [49] to have robust detections and invariant descriptors. Following the idea of invariant descriptors, HDD-Net [50] proposed a hybrid dense L2-Net [45] that aims at obtaining robustness to rotation and scale changes. Recently, DISK [51] introduced an end-to-end local detector-descriptor pipeline that overcomes the non-differentiable nature of selecting and matching keypoints by introducing principles from Reinforcement Learning. Besides independent image feature extraction, CD-UNet [52] presented a method that used two images to condition the position of interest point candidates and the invariance of its descriptor.

In addition to the above presented learned methods, CNN architectures also were deployed to optimize the matching stage. [53] learned to predict which features and descriptors were matchable. This research trend has been extended in many follow-up works [7], [54], [55], [56], [57], making CNN-based matchers a critical step in local features [58]. Furthermore, other CNNs were also studied to perform tasks beyond detection or matching. In [17], the architecture assigned orientations to interest points and AffNet [59] used the descriptor loss to predict the affine parameters of a local feature region. [60] added a new optimization step to refine the keypoint locations after the matching stage, providing higher quality in their 3D reconstructions.

3 KEY.NET ARCHITECTURE

Key.Net architecture combines successful ideas from handcrafted and learned methods namely gradient-based feature extraction, learned combinations of low-level features, and multi-scale pyramid representation.

3.1 Handcrafted and Learned Filters

The design of the handcrafted filters is inspired by the success of Harris [23] and Hessian [24] detectors, which used first and second order derivatives to compute the salient corner responses. A complete set of derivatives is called *LocalJet* [61] and they approximate the signal in the local neighborhood as known from Taylor expansion

$$I_{i_1,\ldots,i_n} = I_0 * \partial_{i_1,\ldots,i_n} g_\sigma(\vec{x}), \tag{1}$$

where g_σ denotes the Gaussian of width σ centered at $\vec{x} = \vec{0}$, and i_n denotes the direction. Higher order derivatives i.e., $n > 2$ are sensitive to noise and require large kernels, we, therefore, include derivatives and their combinations up to the second order only:

- *First Order*. From image I we derive 1st order gradients I_x and I_y. In addition, we compute $I_x * I_y$, ${I_x}^2$, and ${I_y}^2$ as in the second moment matrix of Harris detector [23].
- *Second Order*. From image I, 2nd order derivatives I_{xx}, I_{yy}, and I_{xy} are also included as in the Hessian matrix used in Hessian and DoG detectors [10], [62]. Since Hessian detector uses the determinant of the Hessian matrix we add $I_{xx} * I_{yy}$, and I_{xy}^2.
- *Learned*. A convolutional layer with M filters, a batch normalization layer and a ReLU activation function form a learned block.

The hardcoded filters reduce the number of total learnable parameters to train the architecture, improving the stability and convergence during backpropagation.

3.2 Multi-Scale Pyramid

We design our architecture to be robust to small scale changes without the need for computing several forward passes. As illustrated in Fig. 1, the network includes three scale levels of

the input image which is blurred and downsampled by a factor of 1.2. All the feature maps resulting from the handcrafted filters are concatenated and then fed into the stack of learned blocks in each of the scale levels. All three streams share the weights, such that the same type of anchors result from different levels and form the set of candidates of final keypoints. Feature maps from all scale levels are then upsampled, concatenated, and fed to the last 5×5 convolutional layer to obtain the final response map after ReLU activation function.

4 Loss Functions

In supervised training, the loss function relies on the ground truth. In the case of keypoints, ground truth is not well defined as keypoint locations are useful as long as they can be accurately detected regardless of geometric or photometric image transformation. Some learned detectors [14], [16], [40] train the network to identify keypoints without constraining their locations, where only the homography transformation between images is used as ground truth to calculate the loss as a function of keypoints repeatability.

Other works [15], [18], [19] show the benefits of using anchors to guide their training. Although anchors make the training more stable and lead to better results, they prevent the network from proposing new keypoints in case there is no anchor in the proximity.

In contrast, the handcrafted filters in Key.Net provide a weak constraint with the benefit of the anchor-based methods while allowing the detector to propose new stable keypoints. In our approach, only the geometric transformation between images is required to guide the loss.

4.1 Index Proposal Layer

This section introduces the Index Proposal (IP) layer, which is extended to its multi-scale version in Section 4.2.

Extracting coordinates for training keypoint detectors has been widely studied and showed great improvements: [12], [14], [15] extracted coordinates in a patch level, SuperPoint [18] used a channel-wise softmax to get maxima belonging to fix grids of $8x8$, and [63] used a spatial softmax layer to compute the global maxima of a feature map, obtaining one keypoint candidate per feature map. In contrast to previous methods, the IP layer can return multiple global keypoint coordinates centered on local maxima from a single image without constraining the number of keypoints to the depth of the feature map [63], or the size of the grid [18].

Similarly to handcrafted techniques, keypoint locations are indicated by local maxima of the filter response map $\mathcal{R}$ output by Key.Net. Spatial softmax operator is an effective method for extracting the location of a soft maximum within a window [12], [16], [18], [63]. Therefore, to ensure that the IP layer is fully differentiable, we rely on spatial softmax operator to obtain the coordinates of a single keypoint per window. Consider a window w_i of size $N \times N$ in $\mathcal{R}$, with the score value at each coordinate $[u, v]$ within the window, exponentially scaled and normalized

$$m_i(u,v) = \frac{e^{w_i(u,v)}}{\sum_{j,k}^{N} e^{w_i(j,k)}}. \tag{2}$$

Due to exponential scaling the maximum dominates and the expected location calculated as the weighted average $[\bar{u}_i, \bar{v}_i]$ gives an approximation of the maximum coordinates

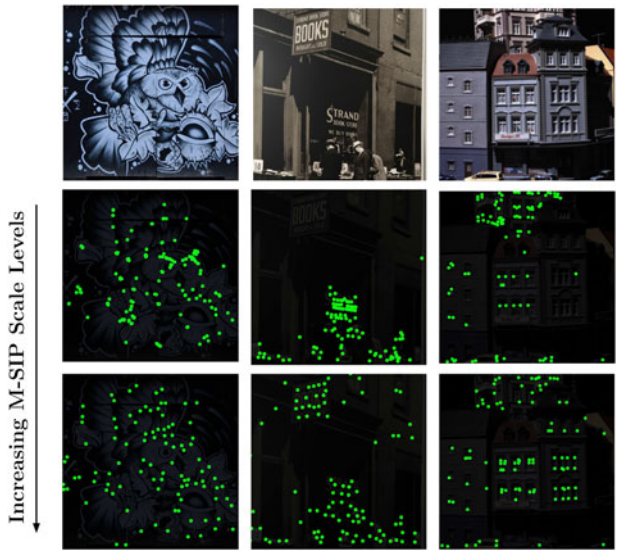


Fig. 3. Keypoints obtained after adding larger context windows to M-SIP operator. The more stable points remain as the M-SIP operator increases its window size. Feature maps in the middle row contain points around edges or non discriminative areas, while bottom row shows detections that are more robust under geometric transformations.

$$[x_i, y_i]^T = [\bar{u}_i, \bar{v}_i]^T = \sum_{u,v}^{N} [W \odot m_i, W^T \odot m_i]^T + c_w, \tag{3}$$

where W is a kernel of size $N \times N$ with index values $j = 1 : N$ along its columns, pointwise product $\odot$, and c_w is the top-left corner coordinates of window w_i. This is similar to non-maxima suppression (NMS) but unlike NMS, the IP layer is differentiable and it is a weighted average of the global maximum of the window rather than the exact location of it. Depending on the base of the power expression in Equation (2), multiple local maxima may have a more or less significant effect on the resulting coordinates.

A detector is covariant if same features are detected under varying image transformations. Covariant constraint was formulated as a regression problem in [14]. Given images I_a and I_b, and ground truth homography $H_{b,a}$ between them, the loss $\mathcal{L}$ is based on the squared difference between points extracted by IP layer and actual maximum coordinates (NMS) in corresponding windows from I_a and I_b

$$\mathcal{L}_{IP}(I_a, I_b, H, N) = \sum_i \alpha_i \| [x_i, y_i]_a^T - H[\hat{x}_i, \hat{y}_i]_b^T \|^2, \tag{4}$$

$$\text{and} \quad \alpha_i = \mathcal{R}_a(x_i, y_i)_a + \mathcal{R}_b(\hat{x}_i, \hat{y}_i)_b, \tag{5}$$

where $\mathcal{R}_a$ and $\mathcal{R}_b$ are the response map of I_a and I_b with coordinates related by the homography H. Moreover, $[x_i, y_i]$ and $[\hat{x}_i, \hat{y}_i]$ refer to the keypoint coordinates given by our IP layer and the NMS operator, respectively. We skip homogeneous coordinates for simplicity. Parameter α_i controls the contribution of each location based on its score value, thus computing the loss for significant features only. As NMS is non-differentiable, gradients are only back-propagated where IP layer is applied, therefore, we switch I_a and I_b and combine both losses to enforce consistency.

4.2 Multi-Scale Index Proposal Layer

IP layer returns one location per window, therefore, the number of keypoints per image strongly depends on the predefined

window size N, in particular, with an increasing size only a few dominant keypoints survive in the image. In [64], authors demonstrated improved performance of local features by accumulating image features not only within a spatial window but also within the neighboring scales. We propose to extend IP layer loss by incorporating multi-scale representation of a local neighborhood. Multiple window sizes encourage the network to find keypoints that exist across a range of scales. The additional benefit of including larger windows is that other keypoints within the window can act as anchors for the estimated location of the dominant keypoint. Similar idea proved successful in [65], where stable region boundaries are used.

We, therefore, propose the Multi-Scale Index Proposal (M-SIP) layer. M-SIP splits multiple times the response map into grids, each with a window size of $N_s \times N_s$ and computes the candidate keypoint position for each window as shown in Fig. 2. Our proposed loss function is the average of covariant constraint losses from all scale levels

$$\mathcal{L}_{MSIP}(I_a, I_b, H_{a,b}) = \sum_s \lambda_s \mathcal{L}_{IP}(I_a, I_b, H_{a,b}, N_s), \quad (6)$$

where s is the index of the scale level with N_s as window size, $\mathcal{L}_{IP}$ is the covariant constraint loss and λ_s is the control parameter at scale level s, that decreases proportionally to the increasing window area as larger windows lead to a larger loss, which is somewhat similar to the scale-space normalisation [11].

The combination of different scales imposes an intrinsic process of simultaneous scoring and ranking of keypoints within the network. To minimize the loss, the network must learn to give higher scores to robust features that remain dominant across a range of scales. Fig. 3 shows different response maps for increasing window size.

5 Experimental Evaluation

In this section, we present implementation details, metrics and the dataset used for evaluating the method.

5.1 Training Data

We generate a synthetic training set from ImageNet ILSVRC 2012 dataset. We apply random geometric transformations to images and extract pairs of corresponding regions as our training set. The process is illustrated in Fig. 4. The parameters of the transformations are: scale $[0.5, 3.5]$, skew $[-0.8, 0.8]$ and rotation $[-60°, 60°]$. Textureless regions are not discriminative, therefore, we discard them by checking if the response of any of the handcrafted filters is lower than a threshold. We modify the contrast, brightness, and hue value in HSV space to one of the images to improve the network's robustness against illumination changes. In addition, for each pair, we generate binary masks that indicate the common area between images. Those masks are used in training to avoid regressing indexes of keypoints that are not present in the common region. There are 12,000 image pairs of size 192×192. We use 9,000 of them as the training data and 3,000 as the validation set.

5.2 Implementation Notes

Training is performed in a siamese pipeline, with two instances of Key.Net that share the weights and are updated at the same time. As shown in Fig. 1, Key.Net uses three learned convolutional blocks, where each has $M = 8$ filters of size 5×5, with He weights initialization and L2 kernel regularizer. We compute the covariant constraint loss $\mathcal{L}_{M\text{-}SIP}$ for five scale levels, with the size of the M-SIP windows $N_s \in [8, 16, 24, 32, 40]$, and loss term $\lambda_s \in [256, 64, 16, 4, 1]$, that were determined by performing a hyperparameter search on the validation set. Larger candidate window sizes have greater mean errors between coordinate points since the maximum distance is proportional to the window size. Thus, λ_s has the largest value for the smallest window. We use a batch size of 32, Adam Optimizer with a learning rate of 10^{-3}, and a decay factor of 0.5 after 20 epochs. On average, the architecture converges in 30 epochs, 2h on a machine with an i7-7700 CPU running at 3.60GHz and an NVIDIA GeForce GTX 1080 Ti. Non-maxima suppression of 15×15 is performed at inference time during evaluation. Evaluation benchmark, synthetic data generator, Key.Net network, and loss are implemented using TensorFlow and are available on GitHub.[1] Even though results are computed with TensorFlow Key.Net, we also provide its PyTorch implementation.[2]

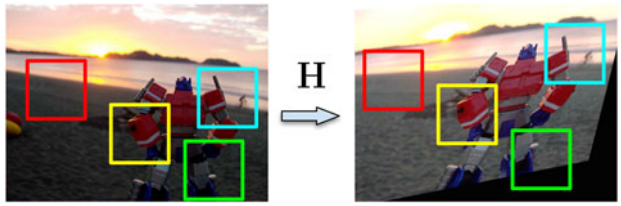


Fig. 4. We apply random geometric and photometric transformations to images and extract pairs of corresponding regions as the training set. Red crop is discarded by checking the response of the handcrafted filters.

5.3 Evaluation Metrics

Repeatability. We follow the evaluation protocol proposed in [20] and improved in the follow up works [1], [12], [14], [15]. The repeatability score for a pair of images is computed as the ratio between the number of corresponding keypoints and the lower number of keypoints detected in one of the two images. We fix the number of extracted keypoints to compare across methods and allow each keypoint to match only once as in [19], [33]. In addition, as exposed by [1], we address the bias from the magnification factor that was applied to accelerate the computation of the overlap error between multi-scale keypoints. Keypoints are identified by spatial coordinates and scales at which the features were detected. To identify corresponding keypoints we compute the Intersection-over-Union error, ϵ_{IoU}, between the areas of the two candidates. To evaluate the accuracy of keypoint location and scale independently, we perform two sets of experiments. One is based on the detected scales and the other assumes the scales are correctly detected by using the ground truth parameters. In our benchmark, we use the top 1,000 interest points that belong to the common region between images, and a match is considered correct when ϵ_{IoU} is smaller than 0.4 i.e., the overlap between corresponding regions is more than 60%. The scales are normalized as in [1], which sets the larger size in a pair of points to 30 pixels and rescales the other one accordingly. HPatches [2] dataset is used for testing. HPatches contains 116 sequences, which are split between viewpoint and illumination transformations, 59 and 57 sequences respectively. HPatches offers predefined image patches for evaluating descriptors, instead, we use full images for evaluating keypoint detectors.

1. https://github.com/axelBarroso/Key.Net
2. https://github.com/axelBarroso/Key.Net-Pytorch

M-SIP Region Sizes					
W_{8x8}	W_{16x16}	W_{24x24}	W_{32x32}	W_{40x40}	Repeatability
✓	-	-	-	-	70.5
✓	✓	-	-	-	74.6
✓	✓	✓	-	-	76.8
✓	✓	✓	✓	-	77.6
-	-	-	-	✓	65.7
-	-	-	✓	✓	71.4
-	-	✓	✓	✓	73.2
-	✓	✓	✓	✓	74.9
✓	✓	✓	✓	✓	**79.1**

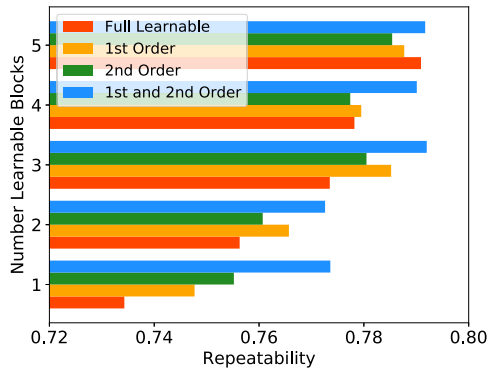


Fig. 5. *Left*: Comparison of repeatability results for the various number of levels in M-SIP operator. We show different combinations of context losses as the final loss, from smaller to larger regions. The best result is obtained when using five window sizes from 8×8 up to 40×40. *Right*: Repeatability results for different combinations of handcrafted filters and a number of learnable layers ($M = 8$ filters each). A higher number of layers leads to better results. All repeatability scores are computed on synthetic validation set from ImageNet.

Image Matching. We compute the Mean Matching Accuracy (MMA) as the ratio of correct matched and detected keypoints. We follow the benchmark proposed in [43], which computes the MMA with different pixel distance thresholds, th, for accepting a pair of matched keypoints. Similar to the previous experiment, we also use the top 1,000 interest keypoints that belong to the common region between two images. The objective of defining the total number of keypoints is twofold: it guarantees that MMA score is not boosted due to a higher number of interest points, and assures good MMA performances only if robust and repeatable keypoints are proposed. Moreover, data privacy has drawn attention to algorithms that can work with a small number of interest points since reverse-engineering attacks from which original data can be reconstructed are mitigated by reducing the total number of keypoints [66]. As in the repeatability experiment, we use the viewpoint and illumination HPatches [2] images for testing. We perform two experiments to evaluate the detector's ability to provide keypoints suitable for matching. In the first experiment, for each detector, we extract keypoint candidates and use the same descriptor network (HardNet [67]) to compute the MMA scores. The experiment allows to compare detectors from the descriptor and matching perspective, i.e., how discriminative the keypoints are. Since some detectors are optimized jointly with their descriptors, the second experiment performs the same matching experiment but using full state-of-the-art detector-descriptor pipelines.

Camera Pose. We further evaluate local features by conducting experiments in the Image Matching Challenge benchmark introduced in the Image Matching: Local Features & Beyond CVPR'20 workshop.[3] The benchmark proposes two tasks: stereo and multi-view reconstruction. In the stereo part, the task aims at matching pairs of images that share some co-visible regions. Hence, features are extracted for each pair of images and filtered by RANSAC to compute their relative pose. The multi-view task uses COLMAP [68] to build SfM reconstructions from randomly sampled small subsets of 5, 10, and 25 images. The dataset contains ten sequences of 100 images each. Each sequence displays multiple images taken under multiple conditions in terms of viewpoint, illumination, or even weather/season. Some examples of famous landmarks in the dataset are the British Museum, the Florence Cathedral, The London Bridge, or Mount Rushmore, among others. The benchmark evaluates the quality of the estimated poses in both tasks by computing the mean Average Accuracy (mAA) for multiple error thresholds, from 1 to 10 degrees. The benchmark also provides a repeatability score, offering a direct comparison of detection methods. In addition to the mAA and repeatability, the benchmark computes the matching score, number of inliers after RANSAC, number of inliers given to COLMAP, number of landmarks, number of observations per landmark, and the absolute trajectory error. We evaluate the results in the restricted keypoint (2,048 top interest points) and unrestricted descriptor regime, which still provides a fair comparison between methods at a much lower cost [58].

Visual Localization. We also perform experiments on the popular task of visual localization [69], where the goal is to estimate the absolute pose of a query image with respect to a 3D model. We use the Aachen Day-Night visual localization benchmark [70], [71] with day-night outdoor scenes. Specifically, we use the Aachen v1.1 which has 824 and 191, day and night image queries, respectively. We obtain the results using the official benchmark website,[4] COLMAP [68] software, and Kapture [72] toolbox. To assess the quality of local features, we use the top 2,048 interest points and report the number (%) of correctly localized queries under multiple acceptance thresholds.

3D Reconstruction. As seen in previous tasks, improvements in local features directly affect many practical tasks, therefore, we also report results for the 3D reconstruction task. For building the 3D reconstruction models, we use the protocol proposed in the Local Feature Evaluation Benchmark [9]. Unlike previous experiments, we do not limit the number of features. This reflects a practical scenario where Key.Net is expected to propose a high number of keypoints. We believe that small (top 1,000 points in matching), medium (top 2,048 points in camera pose and visual localization), and large (unlimited number of points in 3D reconstructions) provide the whole spectrum of keypoint regimes where Key.Net could work. We use COLMAP [68] software to build the 3D models. Due to the significant computational time involved in this task, we perform experiments for our approach otherwise for other methods we copy the results from the literature [73].

3. https://image-matching-workshop.github.io/

4. https://visuallocalization.net/

TABLE 1
Repeatability Results for Different Design Choices on Synthetic Validation Set From ImageNet

	Num. Pyramid Levels					
	1	2	3	4	5	6
Rep.	72.5	74.6	79.1	79.4	**79.5**	78.6

(a) Number of input scale levels in Key.Net.

	Spatial Softmax Base					
	1.2	1.4	2.0	e	5.0	7.5
Rep.	77.5	78.4	77.9	**79.1**	74.6	73.2

(b) Spatial softmax base used in equation 2.

We report the results in terms of registered images, sparse points, dense points, track length, and reprojection error.

6 Experiments

In this section, we present the results obtained by Key.Net in different datasets and tasks. We first show results on validation data for several variants of the proposed architecture. Next, Key.Net repeatability scores in single-scale and multi-scale are presented along with the state-of-the-art detectors on HPatches. Moreover, we evaluate the matching performance for different detectors when pairing them with a common descriptor. We extend the matching experiment by combining Key.Net with a state-of-the-art patch descriptor and compare it against the latest detector-descriptors architectures in HPatches. Besides results in planar scenes, we also test Key.Net in more general 3D reconstruction and camera pose tasks. Finally, we present a study of the number of learnable parameters and inference time of our proposed detector and compare it to other techniques.

6.1 Preliminary Analysis

We study several combinations of loss terms, different handcrafted filters, and the effects of the number of learnable layers or pyramid levels within the architecture.

M-SIP Levels are investigated in Fig. 5 (Left) showing increasing repeatability with more scale levels within M-SIP operator. In addition, we show that losses with smaller window sizes obtain higher repeatability scores. However, the best result is obtained when all levels are combined.

Filter Combinations are analyzed in Fig. 5 (Right). We show results for 1st and 2nd order filters as well as their combination. All networks have the same number of filters, however, we either freeze the first layer of 10 filters with handcrafted kernels (c.f. Section 3.1) or learn them depending on the variant of our network, e.g, in Fully Learnable Key.Net there are no handcrafted filters as all are randomly initialized and learned. The results show that the information provided by handcrafted filters is essential when the number of learnable layers is small. Handcrafted filters act as soft constraints, which directly discard areas without gradients, i.e., non-discriminative with low repeatability. However, as we add more learnable blocks, repeatability scores for combined and fully learnable networks become comparable. Naturally, gradient-based handcrafted filters are simple, and architectures with enough complexity could learn them if they were required. However, the use of engineered features leads to a smaller architecture while maintaining the performance, which is often critical for real-time applications. In summary, combining both types of filters allows to significantly reduce the number of learnable layers. We use Key.Net architecture with three learnable blocks in the next experiments.

Multiple Pyramid Levels at the input to the network also affect the detection performance as shown in Table 1 a. For a single pyramid level, only the original image is used as input. Adding pyramid levels is similar to increasing the size of the receptive fields in the architecture. Our experiment suggests that using more than three levels does not lead to significantly improved results. On the validation set, we obtain a repeatability score of 72.5% for one level, an increase of 6.6% for three, and 7.0% for five levels. We, therefore, use three levels, which achieve good performance while keeping the computational cost low.

Spatial Softmax Base in Equation (2) defines how *soft* the estimation of keypoint coordinates is. High values return the location of the global maximum within the window, while low values average local maxima. The base is varied in Table 1b. Optimum scores are obtained when using the base in Equation (2) close to the e value, which is in line with the setting used in [74].

6.2 Keypoint Detection

This section presents the results for state-of-the-art local feature detectors along with our proposed method. Table 2 shows the repeatability score, average intersection-over-union error $\bar{\epsilon}_{IoU}$, and scale range S_{range}, which is the ratio between the maximum and minimum scale values of the extracted interest points. TI and SI refer to translation (detection at a single scale only) and scale invariance (detection at multiple scales), respectively. Keypoint location is only evaluated under L by assuming a perfect scale detection, while scale and location, SL, use the actual detected scale and position for computing the repeatability and overlap error.

In addition to Key.Net, we propose Tiny-Key.Net, which is a reduced size architecture with all handcrafted filters but only one learnable layer with one filter ($M = 1$) and a single scale input. The idea behind Tiny-Key.Net is to demonstrate how far the complexity can be reduced while keeping good performance. Key.Net and Tiny-Key.Net are extended to scale invariance by evaluating the detector on several scaled images, similar to [15]. We also show results on single scale input Key.Net-SS, to compare it directly with other TI detectors such as R2D2-SS [46], DISK [51], SuperPoint [18], or TILDE [19]. We set the thresholds of algorithms such that they return at least 1,000 points per image. As MSER [31] proposes regions without scoring or ranking, we randomly pick 1,000 points to compute the results. We repeat this experiment ten times and average the results for MSER to better display its real performance.

Key.Net has the best results on viewpoint sequences, in terms of both, location and scale. Tiny-Key.Net does not perform as well as Key.Net but it is within the top three repeatability scores, after Key.Net-SS and Key.Net-MS. On illumination sequences, Key.Net-SS performs the best among TI detectors, not being affected by scale estimation errors. TCDET, which uses points detected by TILDE as anchors, is the most accurate in location estimation compared to other SI detectors, being its repeatability score

TABLE 2
Repeatability Results (%) for Translation (TI) and Scale (SI) Invariant Detectors on HPatches

		Viewpoint					Illumination				
		Repeatability		$\bar{\epsilon}_{IoU}$		S_{range}	Repeatability		$\bar{\epsilon}_{IoU}$		S_{range}
		SL	L	SL	L	SL	SL	L	SL	L	SL
Translation Invariant (TI)	FAST [32]	30.4	63.1	0.21	**0.10**	-	63.6	63.6	**0.09**	**0.09**	-
	TILDE [19]	31.0	65.1	0.20	0.15	-	**70.4**	70.4	0.11	0.11	-
	SuperPoint [18]	32.5	67.1	0.20	**0.10**	-	69.9	69.9	0.10	0.10	-
	D2-Net-SS [43]	27.1	53.3	0.22	0.17	-	64.4	64.4	0.14	0.14	-
	R2D2-SS [43]	28.9	58.6	0.20	0.13	-	68.5	68.5	0.10	0.10	-
	DISK [51]	25.5	60.1	0.21	0.11	-	64.2	64.2	0.10	0.10	-
	Tiny-Key.Net-SS	32.7	69.2	0.20	0.12	-	**70.4**	70.4	0.10	0.10	-
	Key.Net-SS	33.9	**71.2**	0.20	0.11	-	**72.5**	**72.5**	**0.08**	**0.08**	-
Scale Invariant (SI)	SIFT [10]	43.4	55.5	**0.17**	0.12	78.6	48.3	62.2	0.18	0.12	84.5
	SURF [26]	46.7	60.3	0.18	0.18	24.8	53.0	64.0	0.15	0.11	27.4
	MSER [31]	46.4	62.8	**0.12**	**0.08**	**503.7**	46.5	54.5	0.12	0.10	**524.8**
	Harris-Laplace [62]	45.1	62.0	0.20	0.13	**95.9**	52.7	62.0	0.17	**0.08**	**90.4**
	KAZE [27]	**53.3**	65.7	0.20	0.11	12.5	56.9	65.7	0.12	0.10	12.7
	AKAZE [28]	50.4	62.6	0.20	**0.10**	13.7	65.9	69.6	0.11	**0.09**	13.6
	LIFT [12]	43.4	59.4	0.20	0.13	13.3	51.6	65.4	0.18	0.12	13.8
	DNet [14]	49.4	62.2	0.21	0.14	11.4	59.1	65.1	0.14	0.13	17.1
	TCDET [15]	49.6	61.6	0.23	0.16	6.7	66.9	**71.0**	0.16	0.15	11.4
	LF-Net [16]	32.3	62.2	0.23	0.12	2.00	68.6	69.1	0.11	0.10	2.0
	D2-Net-MS [43]	25.1	50.6	0.26	0.19	4.0	52.1	62.2	0.18	0.17	4.0
	R2D2-MS [43]	48.7	59.9	0.32	0.14	3.4	62.0	65.9	**0.09**	0.10	3.4
	Tiny-Key.Net-MS	50.0	68.9	0.20	0.11	7.6	65.2	67.9	0.11	0.11	7.6
	Key.Net-MS	**53.4**	**69.5**	0.19	**0.10**	7.6	65.3	68.5	0.10	0.10	7.6

We also report average overlap error $\bar{\epsilon}_{IoU}$, and the ratio of maximum to minimum extracted scale S_{Range}. In SL, scales and locations are used to compute overlap error, meanwhile, in L, only locations are used and scales are assumed to be correctly estimated. Key.Net is the best algorithms on viewpoint, for both L and SL. On illumination sequences, translation-invariant (TI) Key.Net-SS obtains the best accuracy. Among scale-invariant (SI) detectors, TCDET is the best in L and LF-Net in SL.

higher than Key.Net-MS. Note that TILDE-based detectors were specifically designed and trained for illumination sequences. Key.Net-MS addresses the scale changes better than other methods but the errors in multi-scale sampling affect it when there is no scale change between images, i.e., illumination sequences. This has often been observed for detectors with more invariance than required by the data [22]. Handcrafted detectors have the lowest average overlap error $\bar{\epsilon}_{IoU}$ among all methods. Specifically, MSER is the best SI detector according to SL and L overlap error in viewpoint, meanwhile, Harris-Laplace [62] and Key.Net-SS obtain the lowest overlapping errors in illumination sequences. A wide range of scales S_{range} is detected by MSER, which has a great capability of extracting local features from different scales due to its feature segmentation nature. Note that Key.Net and SuperPoint are trained with synthetic images, hence, planar HPatches nature may favorite them over the latest methods that were trained with real 3D data: D2-Net [43], R2D2 [46], or DISK [51].

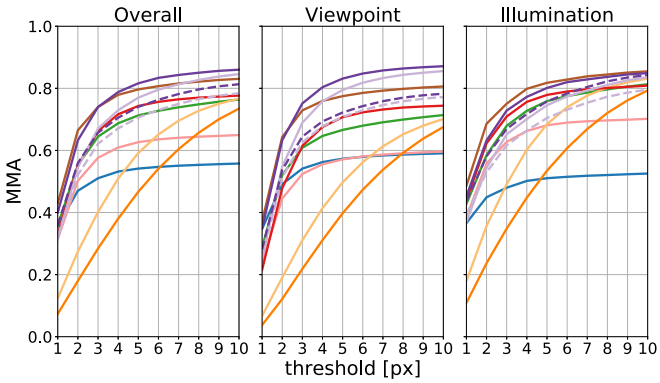
6.3 Keypoint Matching

Moreover, to demonstrate that the detected features are useful for matching, Table 3 shows matching scores for detectors combined with HardNet descriptor [67]. As our method only focuses on the detection part, and for a fair comparison, we used the same descriptor and discard the orientation for all methods that provide it. The matching score is computed as the ratio between correct and matched features at a 5-pixel threshold error. Features are matched and filtered by a mutual nearest neighbor (MNN) strategy as in [43]. We also report the number of matches after MNN as a reference. Top matching scores on viewpoint and illumination for SI methods are obtained by Key.Net-MS. Tiny-Key.Net-MS still performs competitively against other methods in viewpoint, but it falls behind on illumination sequences. Meanwhile, when looking at methods that propose single scale detections (TI), Key.Net-SS obtains the best performance on viewpoint

TABLE 3
Matching Score (%) of Best Detectors Together With HardNet

	Num. Matches	MMA (5px) View	MMA (5px) Illum
MSER [31] + HardNet [67]	184.7	24.68	39.45
SIFT [10] + HardNet [67]	337.4	71.96	68.05
HarrisLaplace [62] + HardNet [67]	330.3	64.76	66.23
AKAZE [28] + HardNet [67]	436.4	72.56	76.23
TILDE [19] + HardNet [67]	412.3	68.13	76.94
LIFT [12] + HardNet [67]	380.4	63.28	69.19
DNet [14] + HardNet [67]	373.1	64.62	72.73
TCDET [15] + HardNet [67]	385.6	58.22	77.20
SuperPoint [18] + HardNet [67]	501.8	70.96	**79.93**
LF-Net [16] + HardNet [67]	418.0	63.78	78.59
D2-Net-SS [43] + HardNet [67]	414.5	40.52	61.16
D2-Net-MS [43] + HardNet [67]	430.6	32.58	48.69
R2D2 [46] + HardNet [67]	387.6	68.59	77.66
DISK [51] + HardNet [67]	504.8	69.15	**80.41**
Tiny-Key.Net-SS + HardNet [67]	467.2	68.48	71.80
Tiny-Key.Net-MS + HardNet [67]	**509.8**	**77.31**	75.04
Key.Net-SS + HardNet [67]	489.4	70.07	76.80
Key.Net-MS + HardNet [67]	**520.8**	**81.20**	79.85

Results on HPatches sequences, both viewpoint, and illumination. Key.Net-MS architecture gets the best matching score for viewpoint, while DISK + HardNet for illumination sequences.



Methods	Num. Matches	Overall MMA	Viewp. MMA	Illum. MMA
SIFT [10]	398.7	54.16	56.31	51.05
SuperPoint [18]	510.9	71.28	65.43	75.87
LF-Net [16]	354.4	62.65	56.10	68.11
D2-Net-SS [43]	417.7	59.18	48.86	68.47
D2-Net-MS [43]	389.5	46.68	39.06	53.47
R2D2 [46]	296.0	74.32	69.46	77.88
DISK [51]	492.5	79.65	76.13	**81.78**
Tiny-Key.Net-SS	472.3	70.63	69.57	70.44
Tiny-Key.Net-MS	515.7	77.00	78.09	74.56
Key.Net-SS	499.2	74.03	70.96	75.80
Key.Net-MS	**521.4**	**81.62**	**81.68**	80.12

Fig. 6. *Left*: Mean Matching Accuracy (MMA) curves under multiple thresholds on HPatches dataset. *Right*: Number of matches after the mutual nearest neighbor filtering, and MMA results for a pixel threshold of 5 pixels. Key.Net and Tiny-Key.Net are combined with HyNet descriptor and obtain the best results in the viewpoint and overall splits. Meanwhile, DISK outperforms all the other methods on illumination sequences.

followed by DISK, while DISK and SuperPoint outperform all methods on illumination. Note also that handcrafted AKAZE (SI) performs close to the top learned methods, even outperforming R2D2 detections on average, which highlights the importance and performance of non-data-driven algorithms.

6.4 Image Matching

The latest keypoint detectors are optimized together with its descriptor side, and therefore, evaluating only the matching score of its detection with an off-the-shelf descriptor may not accurately describe its performance. Note that patch-based descriptors are trained with patches centered in corners or blobs, and current local feature extractors may find keypoint candidates beyond those structures [51]. Hence, in Fig. 6 (left), we show the mean matching accuracy (MMA) curves for state-of-the-art detectors-descriptors. In addition to the MMA curves, as in Table 3, we report in Fig. 6 (right) the number of matches after the mutual nearest neighbor (MNN) filtering and MMA at a 5-pixel threshold error. As Key.Net offers the location and the scale of a keypoint candidate, it can be paired with any keypoint descriptor. Thus, in this experiment and the following ones, we combine our detector with HyNet [73], which has the current state-of-the-art results in patch matching [73].

The left plot on Fig. 6 shows that the top method in the Overall split is Key.Net + HyNet, followed by DISK. As seen in the right table from Fig. 6, DISK outperforms all the other methods on illumination sequences; it gets a 1.66 MMA score increase compared with Key.Net + HyNet. However, Key.Net + HyNet especially excels in the viewpoint sequences, where it surpasses DISK by 5.55 MMA points. Tiny-Key.Net + HyNet falls behind DISK and regular Key.Net, but it still outperforms other state-of-the-art methods such as SuperPoint, D2-Net, or R2D2. In terms of the number of matched keypoints after MNN filtering, Key.Net and Tiny-Key.Net together with HyNet recover the largest number of mutual neighbor keypoints. That combined with the high MMA scores obtained in both viewpoint and illumination splits indicates the good performance of both methods even though they are not trained together. SuperPoint also has a high number of matches, but its MMA scores are below Tiny-Key.Net and Key.Net. Moreover, R2D2 MMA scores are close to DISK or Key.Net performance, but it provides a much-reduced number of matches. D2-Net performance is below LF-Net, and close to SIFT. However, the authors indicated that D2-Net works the best on a bigger regime of keypoint candidates [43]. We will observe D2-Net performance on a more wide regime of keypoint candidates in the following experiments. In addition to the comparison against state-of-the-art methods, we also show the performances of single and multi-scale Key.Net detections. The increment of Key.Net-MS and Tiny-Key.Net-MS methods over single scale detection excels especially in viewpoint by 9.62 points. This is expected as viewpoint sequences display different scale situations in which multi-scale detection favors not only the detections but also the descriptor. In contrast to patch-based descriptors, other TI methods, such as SuperPoint's or DISK's, are more robust against those scale changes since no scale information is used to correct or normalize its descriptors.

6.5 Camera Pose - Image Matching Challenge 2020

The previous evaluation focused on HPatches, a planar dataset with varying viewpoint and illumination conditions. Even though the great variability of HPatches, it does not present common conditions that real-world images may display; occlusions, weather/season changes, dynamic objects, among others. Therefore, Table 4 evaluates Key.Net and other state-of-the-art methods in terms of camera pose precision on famous 3D landmarks. As mentioned in Section 5.3, results in Table 4 are those available directly in the CVPR'20 Image Matching Challenge leaderboard.

Table 4 shows that best results in terms of camera pose estimation, number of inliers, and track length are obtained by DISK, being followed by Key.Net and Key.Net-Sm + HyNet methods. SIFT still shows great performance when compared to other state-of-the-art methods, and it can be even boosted when combined with HardNet descriptor, outperforming SuperPoint, D2-Net, or R2D2. SuperPoint was trained with synthetic images, and although performance on planar scenes (Table 6) is close to top performance methods, it fails on this 3D real scenario. However, [56] showed that SuperPoint can be integrated with a custom matcher and obtain state-of-the-art performance. Nevertheless, contrary to SuperPoint, Key.Net networks still outperform all other methods in terms of repeatability, even though Key.Net was trained on synthetic images. We show examples in Figs. 7 and 8.

5. https://cs.ubc.ca/research/image-matching-challenge/2020/

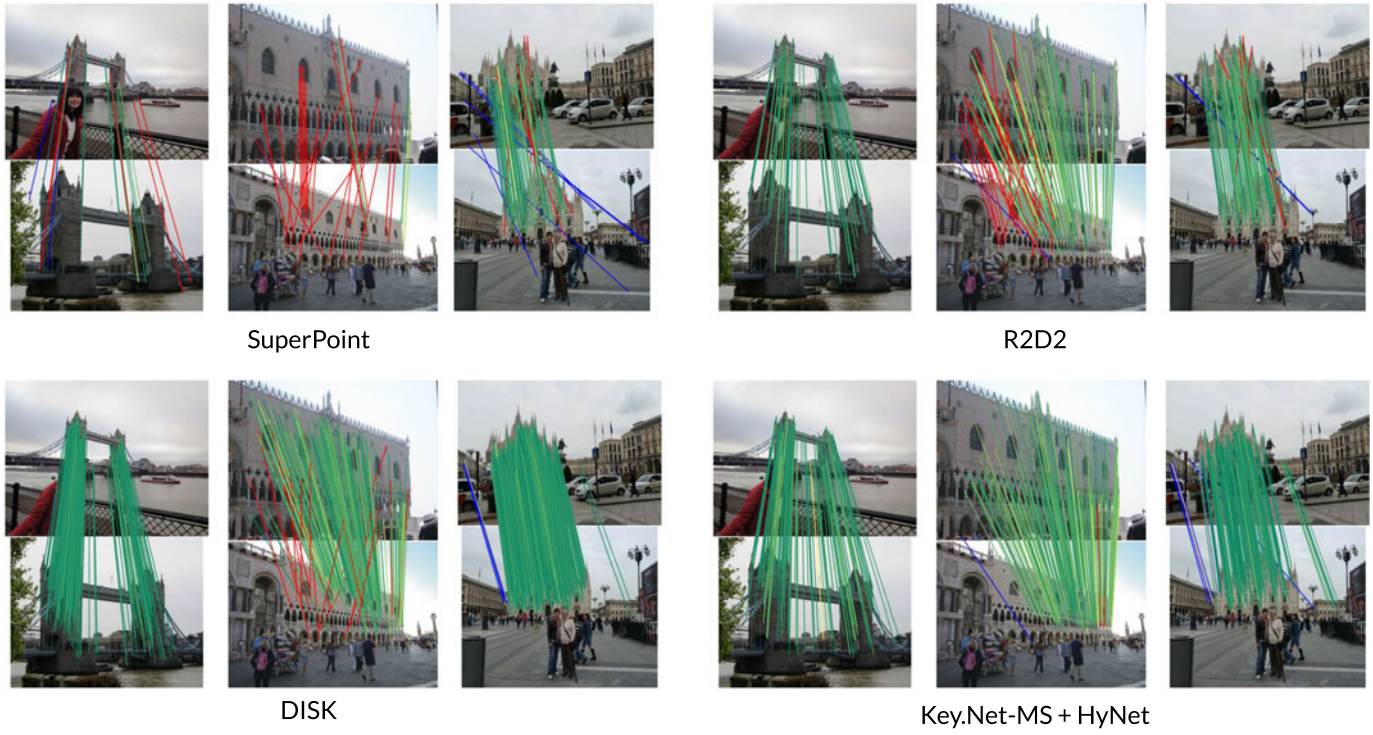


Fig. 7. Stereo results on the Image Matching Challenge [58]. The figure plots the resulting inliers for different state-of-the-art methods. Matches are displayed from green to yellow if they are correct, i.e., their reprojection error is between 0 and 5 pixels, in red if they are incorrect (reprojection error above 5 pixels), and in blue if their ground truth depth is not available. SuperPoint fails in strong viewpoint conditions, providing fewer correct matches than its competitors. R2D2 and Key.Net candidate points are proposed sparsely in repeatable structures, while DISK candidates are cluster in interest regions.

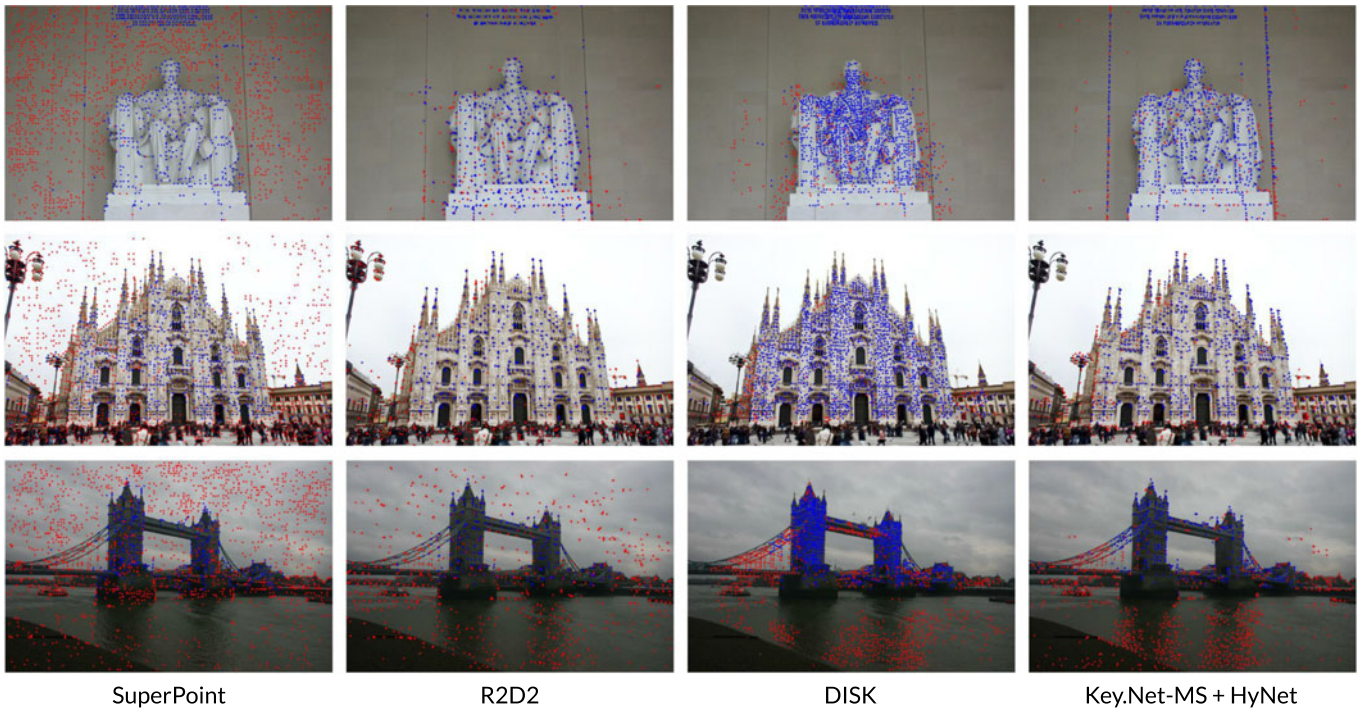


Fig. 8. Multiview results on the Image Matching Challenge [58]. After 3D reconstruction, registered keypoints by COLMAP are drawn in blue and unregistered points are in red. SuperPoint and R2D2 fail to register many of their detections due to detections on the sky or walls. Meanwhile, DISK and Key.Net can propose repeatable and reliable keypoints that are likely registered in the 3D model. Key.Net focuses on regions with strong gradients, i.e., corners, and robust DISK descriptor allow for detections on seemingly textureless regions.

TABLE 4
Camera Pose Evaluation Results on the Workshop CVPR Challenge Image Matching Competition 2020 [58]

	Stereo				Multiview				
	NI	Rep (3 pix.)	MS (3 pix.)	mAA (10°)	NM	NL	TL	ATE	mAA (10°)
SIFT [10]	126.0	33.3	0.82	0.44	193.1	1436.9	4.33	0.46	0.64
SIFT [10] + HardNet [67]	152.7	33.3	0.81	0.46	201.3	1467.9	4.31	0.47	0.64
SuperPoint [18] (NMS=4)	122.7	36.4	0.63	0.29	170.6	1185.4	4.33	0.56	0.55
D2-Net-SS [43]	147.5	20.2	0.34	0.16	364.7	1985.0	3.41	0.72	0.37
D2-Net-MS [43]	118.4	16.8	0.29	0.12	286.1	1999.4	3.01	0.77	0.28
R2D2 [46]	176.5	42.2	0.73	0.36	262.0	1188.1	4.30	0.50	0.61
DISK [51]	**404.2**	44.8	**0.85**	**0.51**	**527.5**	**2428.0**	**5.55**	**0.41**	**0.73**
Key.Net + HardNet [67]	134.4	**46.9**	0.82	0.33	195.3	1276.3	4.49	0.49	0.62
Key.Net + HyNet [73]	246.6	44.9	0.81	0.45	331.6	1621.7	4.57	0.45	0.67

The metrics refer to mean Average Accuracy (mAA), number of inliers after RANSAC (NI), repeatability (Rep), matching score (MS), number of inliers given to COLMAP (NM), number of landmarks (NL), number of observations per landmark (TL), and absolute trajectory error (ATE). DISK method obtains the best results in terms of camera pose accuracy while Key.Net achieves the highest keypoint repeatability score, indicating that its proposes are more repeatable or accurate than DISK detections. Refer to the official 2020 IMC website for all submissions.[5]

6.6 Visual Localization

Table 5 presents the results of state-of-the-art methods on the visual localization task. We report the percentage of correct localized day and night queries under multiple thresholds. First, Key.Net + HyNet reaches higher number of correctly localized queries than Key.Net + HardNet, reflecting once more that better local descriptors benefit the overall performance. In addition, we observe that R2D2 has the highest score for the restricted error threshold (.25, 2°), meanwhile, Key.Net + HyNet obtains the top scores for all the other error thresholds (.5, 5° and 5, 10°) on the day queries. On the night queries, SuperPoint and D2-Net obtain the best results overall followed by DISK and Key.Net + HyNet. While HyNet is trained on UBC dataset [75] (Liberty split), which has no night images, its combination with Key.Net still provides competitive results against the other methods that were trained on datasets more similar to Aachen Day-Night [70], e.g., R2D2 (Aachen), DISK (Megadepth), or D2-Net (Megadepth).

6.7 3D Reconstruction

Since 3D reconstruction requires high computational and time requirements, we compare against methods that have available 3D results in the literature. Therefore, as some previous discussed methods do not offer such results, e.g. DISK, we complement the 3D task evaluation with additional methods, namely CD-UNet [52] and ASLFeat [47].

TABLE 5
Visual Localization on Aachen-Day-Night v1.1 [70]

	Correctly localized queries (%)	
	(.25m, 2°) / (.5m, 5°) / (5m, 10°)	
	Day	Night
SIFT [10]	70.5 / 78.9 /86.8	15.2 / 18.3 / 25.1
SuperPoint [18]	86.3 / **94.3** / 98.7	**70.2** / **83.8** / 95.8
D2-Net [43]	78.4 / 91.0 / 97.7	51.8 / **83.8** / **97.4**
R2D2 [46]	**87.3** / 93.8 / 98.5	61.8 / 80.1 / 94.8
DISK [51]	84.8 / 93.6 / 98.7	66.5 / 82.2 / 95.8
Key.Net + HardNet [67]	86.0 / 94.1 / 98.8	61.8 / 78.5 / 93.7
Key.Net + HyNet [73]	85.8 / **94.3** / **98.9**	66.5 / 80.6 / 93.2

Results obtained when using top 2,048 keypoints.

Table 6 shows the metrics obtained when aiming at building 3D reconstructions with local feature extractors. We observe that Key.Net + HyNet success in registering a higher number of images and sparse points into the 3D model. In terms of the number of dense points, ASLFeat outperforms all the other methods, followed by Key.Net + HyNet and R2D2. On the big sequences (Madrid Metropolis and Gendarmenmarkt), ASLFeat gets the highest number of images that see a registered 3D point on average (Track Length). ASLFeat uses deformable convolutions that make its descriptor robust against the strong viewpoint presented on those sequences. Even though all the recent efforts for getting an accurate keypoint detector [18], [21], [46], [47], SIFT still obtains the smallest reprojection error in its reconstructions. However, SIFT also recovers a shorter track length, and therefore, the reprojection error is computed by triangulating fewer images than in other methods. Moreover, as shown in Fig. 9, the reconstructions produced by Key.Net register a higher number of 3D points into the model, and obtains denser and greater quality reconstructions.

6.8 Efficiency

We also compare the number of learnable parameters, indicating the complexity of the predictor, which leads to an increased risk of overfitting and the need for a large amount of training data. Table 7 shows the approximate

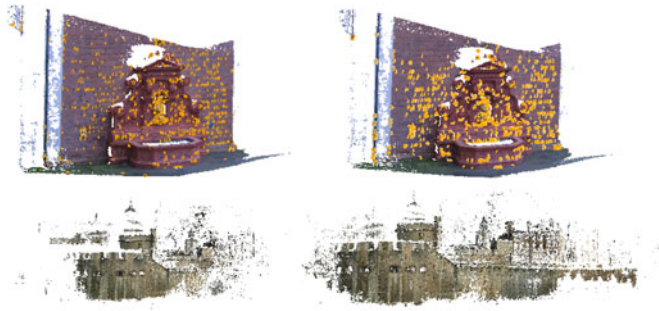


Fig. 9. Toy 3D reconstruction example for Fountain (top) and Tower of London (bottom) with SIFT (left) and Key.Net (right). In addition, for better comparison, Fountain reconstruction highlights the reconstructed sparse points. The number of registered 3D points are 1039 and 1608 in Fountain, and 544 and 1088 in Tower of London, for SIFT and Key.Net, respectively.

TABLE 6
3D Evaluation Results on ETH Dataset [9]

		Num. Reg. Images	Num. Sparse Points	Num. Dense Points	Track Length	Reproj. Error	Num. Features
Fountain 11 images	SIFT [10]	**11**	**15K**	292K	4.79	**0.39px**	11.8K
	SuperPoint [18]	**11**	7K	304K	4.93	0.81px	5.5K
	D2-Net [43]	**11**	**19K**	301K	3.03	1.40px	12.5K
	R2D2 [46]	**11**	13K	**308K**	**5.02**	1.47px	12.6K
	Key.Net + HyNet [73]	**11**	12K	**307K**	**7.81**	**0.69px**	11.9K
South Building 128 images	SIFT [10]	**128**	108K	**2.14M**	6.04	**0.54px**	13.3K
	SuperPoint [18]	**128**	125K	2.13M	**7.10**	0.83px	10.6K
	D2-Net [43]	**128**	**178K**	2.06M	3.11	1.36px	12.4K
	R2D2 [46]	**128**	**136K**	**3.31M**	5.60	1.43px	13.2K
	Key.Net + HyNet [73]	**128**	100K	2.11M	**12.03**	**0.74px**	12.9K
Madrid Metropolis 1344 images	SIFT [10]	500	116K	**1.82M**	6.32	**0.60px**	7.4K
	SuperPoint [18]	702	125K	1.14M	4.43	1.05px	2.1K
	D2-Net [43]	787	229K	0.96M	5.50	1.27px	7.7K
	R2D2 [46]	**790**	158K	1.15M	**7.26**	1.20px	12.9K
	CD-UNet [52]	702	**256K**	1.10M	6.09	1.30px	-
	ASLFeat [47]	649	129K	**1.92M**	**9.56**	**0.95px**	-
	Key.Net + HyNet [73]	**897**	**386K**	1.62M	5.87	1.05px	9.3K
Gendarmenmarkt 1463 images	SIFT [10]	1035	338K	**4.22M**	5.52	**0.69px**	8.5K
	SuperPoint [18]	1112	236K	2.49M	4.74	1.10px	2.3K
	D2-Net [43]	1225	541K	2.60M	5.21	1.30px	8.0K
	R2D2 [46]	**1226**	529K	3.80M	6.38	1.21px	13.3K
	CD-UNet [52]	1072	**570K**	2.11M	**6.60**	1.34px	-
	ASLFeat [47]	1061	320K	**4.00M**	**8.98**	**1.05px**	-
	Key.Net + HyNet [73]	**1259**	**897K**	3.58M	5.79	1.13px	10.6K

In addition to previous evaluated methods, we also test our Key.Net detector against recent ASLFeat [47] and CD-UNet [52]. The combination of Key.Net and HyNet can obtain the highest number of registered images into the 3D model. This is highly important since registered images and their camera poses are a crucial element of complex localization pipelines [76]. Key.Net + HyNet also recovers the largest number of sparse points in the model, meanwhile, other methods such as R2D2 or ASLFeat result in denser models. ASLFeat especially outperforms in terms of track length, and classical SIFT is still the most accurate method (lowest reprojection error).

number of parameters for different architectures. In Table 7 a, we only display the number of learnable parameters that are used during the inference in the detector part, meanwhile, Table 7 b shows the total number of parameters involved in the detector and descriptor. DISK and R2D2 use a common feature extractor for predicting detections and descriptors, therefore, there is not an overhead introduced by extracting descriptors. However, even though the current trend for proposing jointly detector-descriptor networks, detectors are still used by themselves in many works as an independent block [59], [66], [73], [77], [78]. Hence, there is a need for efficient and reliable detectors. We compare Key.Net against SuperPoint, which is a popular choice when it comes to keypoint selection [79], [80]. Key.Net has nearly 160 times fewer parameters, and Tiny-Key.Net has 3,300 times fewer parameters than SuperPoint with better performance in multiple benchmarks. On the other side, joint detection-description networks are more expensive than those that perform detection alone, however, R2D2 offers a good compromise of 486K learnable parameters for computing both, detection scores and descriptor maps. SuperPoint, DISK, or Key.Net + HyNet alternatives are above R2D2 but still reduced by half the number of learnable parameters in LF-Net. The inference time of an image of 600 $\times$ 600 is 5.7ms (175 FPS) and 31ms (32.25 FPS) for Tiny-Key.Net and Key.Net, respectively.

TABLE 7
Comparison of the Number of Learnable Parameters for State-of-the-Art Architectures

Number of Learnable Parameters					
LF-Net	SuperPoint	R2D2	DISK	Tiny-Key.Net	Key.Net
42k	940K	486K	1.1M	**279**	**5.9K**

(a) Number of learnable parameters on the keypoint detector side.

LF-Net	SuperPoint	R2D2	DISK	Key.Net + HyNet
2.6M	1.3M	**486K**	**1.1M**	1.3M

(b) Number of learnable parameters of the full architectures.

Tiny-Key.Net has only one learnable block with one filter, and therefore it is the detector with the fewest number of parameters, followed by regular Key.Net. When looking into detector and descriptor at the same time, the shared feature extractor in R2D2 allows the lowest number of parameters.

7 Conclusion

We have introduced a novel approach to detect local features that combines handcrafted and learned CNN filters. We have proposed a multi-scale index proposal layer that finds keypoints across a range of scales, with a loss function that optimizes the robustness and discriminating properties of the detections. We demonstrated how to compute and combine differentiable keypoint detection loss for multi-scale representation. Evaluation results on large benchmarks show that combining handcrafted and learned features as well as multi-scale analysis at different stages of the network improves the repeatability scores compared to other state-of-the-art keypoint

detection methods. Moreover, we show how the keypoint candidates proposed by our interest point detectors can be combined with different state-of-the-art patch-based descriptors to be on-par or even outperform current top-performing detectors-descriptors in image matching, 3D reconstruction, visual localization, and camera pose estimation.

We further show that excessively increasing the network's complexity does not lead to improved results. In contrast, using handcrafted filters allows to significantly reduce the complexity of the architecture leading to a fast and compact detector with 279 learnable parameters and inference of 175 frames per second.

REFERENCES

[1] K. Lenc and A. Vedaldi, "Large scale evaluation of local image feature detectors on homography datasets," in *Proc. Brit. Mach. Vis. Conf.*, 2018.

[2] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3852–3861.

[3] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3279–3286.

[4] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4353–4361.

[5] P. Truong, M. Danelljan, L. V. Gool, and R. Timofte, "Learning accurate dense correspondences and when to trust them," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5714–5724.

[6] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "LoFTR: Detector-free local feature matching with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 8918–8927.

[7] W. Jiang, E. Trulls, J. Hosang, A. Tagliasacchi, and K. M. Yi, "COTR: Correspondence transformer for matching across images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6207–6217.

[8] G. Csurka, C. R. Dance, and M. Humenberger, "From handcrafted to deep local features," 2018, *arXiv:1807.10254*.

[9] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6959–6968.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.

[11] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, pp. 63–86, 2004.

[12] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned invariant feature transform," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 467–483.

[13] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Toward geometric deep SLAM," 2017, *arXiv:1707.07410*.

[14] K. Lenc and A. Vedaldi, "Learning covariant feature detectors," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 100–117.

[15] X. Zhang, F. X. Yu, S. Karaman, and S.-F. Chang, "Learning discriminative and transformation covariant local feature detectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4923–4931.

[16] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "LF-Net: Learning local features from images," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 6234–6244.

[17] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit, "Learning to assign orientations to feature points," *CoRR*, vol. abs/1511.04273, 2015. [Online]. Available: http://arxiv.org/abs/1511.04273

[18] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 337–33712.

[19] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "TILDE: A temporally invariant learned detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5279–5288.

[20] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[21] A. Barroso-Laguna , E. Riba, D. Ponsa, and K. Mikolajczyk, "Key.Net: Keypoint detection by handcrafted and learned CNN filters," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 5835–5843.

[22] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends Comput. Graph. Vis.*, vol. 3, pp. 177–280, 2008.

[23] C. G. Harris *et al.*, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, 1988, pp. 10–5244.

[24] P. Beaudet, "Rotationally invariant image operators," in *Proc. Int. Joint Conf. Pattern Recognit.*, 1978.

[25] K. Mikolajczyk *et al.*, "A comparison of affine region detectors," *Int. J. Comput. Vis.*, vol. 65, no. 1, pp. 43–72, 2005.

[26] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understanding*, vol. 110, pp. 346–359, 2008.

[27] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 214–227.

[28] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 1281–1298.

[29] P. Mainali, G. Lafruit, Q. Yang, B. Geelen, L. Van Gool , and R. Lauwereins, "SIFER: Scale-invariant feature detector with error resilience," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 172–197, 2013.

[30] C. L. Zitnick and K. Ramnath, "Edge foci interest points," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 359–366.

[31] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image Vis. Comput.*, vol. 22, no. 10, pp. 761–767, 2004.

[32] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 430–443.

[33] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 105–119, Jan. 2010.

[34] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 2548–2555.

[35] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 2564–2571.

[36] Y. Cho *et al.*, "Learning to detect local features using information change," *IEEE Access*, vol. 9, pp. 43 898–43 908, 2021.

[37] S. Kim, M. Jeong, and B. C. Ko, "Self-supervised keypoint detection based on multi-layer random forest regressor," *IEEE Access*, vol. 9, pp. 40 850–40 859, 2021.

[38] X. Yuan, K. Hu, and S. Chen, "Realtime CNN-based keypoint detector with Sobel filter and CNN-based descriptor trained with keypoint candidates," 2020, *arXiv:2011.02119*.

[39] P. Di Febbo , C. Dal Mutto , K. Tieu, and S. Mattoccia, "KCNN: Extremely-efficient hardware keypoint detection with a compact convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 795–7958.

[40] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys, "Quad-networks: Unsupervised learning to rank for interest point detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3929–3937.

[41] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Košecká, "End-to-end learning of keypoint detector and descriptor for pose invariant 3D matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1965–1973.

[42] X. Shen *et al.*, "RF-Net: An end-to-end image matching network based on receptive field," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8124–8132.

[43] M. Dusmanu *et al.*, "D2-Net: A trainable CNN for joint detection and description of local features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8084–8093.

[44] Y. Tian, V. Balntas, T. Ng, A. Barroso-Laguna , Y. Demiris, and K. Mikolajczyk, "D2D: Keypoint extraction with describe to detect approach," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 223–240.

[45] Y. Tian, B. Fan, and F. Wu, "L2-Net: Deep learning of discriminative patch descriptor in euclidean space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6128–6136.

[46] J. Revaud, P. Weinzaepfel, C. De Souza , and M. Humenberger, "R2D2: Repeatable and reliable detector and descriptor," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 113.
[47] Z. Luo *et al.*, "ASLFeat: Learning local features of accurate shape and localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6588–6597.
[48] J. Dai *et al.*, "Deformable convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 764–773.
[49] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable ConvNets V2: More deformable, better results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9300–9308.
[50] A. Barroso-Laguna , Y. Verdie, B. Busam, and K. Mikolajczyk, "HDD-net: Hybrid detector descriptor with mutual interactive learning," in *Proc. Asian Conf. Comput. Vis.*, 2020, pp. 500–516.
[51] M. J. Tyszkiewicz, P. Fua, and E. Trulls, "DISK: Learning local features with policy gradient," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.
[52] O. Wiles, S. Ehrhardt, and A. Zisserman, "Co-attention for conditioned image matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15915–15924.
[53] W. Hartmann, M. Havlena, and K. Schindler, "Predicting matchability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 9–16.
[54] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua, "Learning to find good correspondences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2666–2674.
[55] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, and K. Moo Yi , "ACNe: Attentive context normalization for robust permutation-equivariant learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11283–11292.
[56] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4937–4946.
[57] Y. Liu, L. Liu, C. Lin, Z. Dong, and W. Wang, "Learnable motion coherence for correspondence pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3236–3245.
[58] J. Yuhe *et al.*, "Image matching across wide baselines: From paper to practice," *Int. J. Comput. Vis.*, vol. 129, pp. 517–547, 2021.
[59] D. Mishkin, F. Radenovic, and J. Matas, "Repeatability is not enough: Learning affine regions via discriminability," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 284–300.
[60] M. Dusmanu, J. L. Schönberger, and M. Pollefeys, "Multi-view optimization of local feature geometry," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 670–686.
[61] L. Florack, B. T. H. Romeny, M. Viergever, and J. Koenderink, "The Gaussian scale-space paradigm and the multiscale local jet," *Int. J. Comput. Vis.*, vol. 18, no. 1, pp. 61–75, 1996.
[62] K. Mikolajczyk and C. Schmid, "Indexing based on scale invariant interest points," in *Proc. 8th IEEE Int. Conf. Comput. Vis.*, 2001, pp. 525–531.
[63] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3D keypoints via end-to-end geometric reasoning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2063–2074.
[64] J. Dong and S. Soatto, "Domain-size pooling in local descriptors: DSP-SIFT," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5097–5106.
[65] S. Obdrzalek and J. Matas, "Object recognition using local affine frames on distinguished regions," in *Proc. Brit. Mach. Vis. Conf.*, 2002, Art. no. 3.
[66] D. Dangwal *et al.*, "Analysis and mitigations of reverse engineering attacks on local feature descriptors," 2021, *arXiv:2105.03812*.
[67] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4826–4837.
[68] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4104–4113.
[69] T. Sattler *et al.*, "Benchmarking 6DOF outdoor visual localization in changing conditions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8601–8610.
[70] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image retrieval for image-based localization revisited," in *Proc. Brit. Mach. Vis. Conf.*, 2012, Art. no. 4.
[71] Z. Zhang, T. Sattler, and D. Scaramuzza, "Reference pose generation for long-term visual localization via learned features and view synthesis," *Int. J. Comput. Vis.*, vol. 129, no. 4, pp. 821–844, 2021.
[72] M. Humenberger *et al.*, "Robust image retrieval-based visual localization using kapture," 2020, *arXiv:2007.13867*.
[73] Y. Tian, A. Barroso-Laguna, T. Ng, V. Balntas, and K. Mikolajczyk, "HyNet: Local descriptor with hybrid similarity measure and triplet loss," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.
[74] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3D keypoints via end-to-end geometric reasoning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2063–2074.
[75] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 43–57, Jan. 2010.
[76] P.-E. Sarlin *et al.*, "Back to the feature: Learning robust camera localization from pixels to pose," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3246–3256.
[77] Q. Wang, X. Zhou, B. Hariharan, and N. Snavely, "Learning feature descriptors using camera pose supervision," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 757–774.
[78] H. Yang, W. Dong, L. Carlone, and V. Koltun, "Self-supervised geometric perception," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14345–14356.
[79] H. Germain, G. Bourmaud, and V. Lepetit, "S2DNet: Learning accurate correspondences for sparse-to-dense feature matching," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 626–643.
[80] H. Germain, V. Lepetit, and G. Bourmaud, "Neural reprojection error: Merging feature learning and camera pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 414–423.

Axel Barroso-Laguna received the BSc and MSc degrees in telecommunications engineering from the Polytechnic University of Catalonia (UPC), Barcelona, Spain, and the MSc degree in computer vision from the Autonomous University of Barcelona (UAB-CVC), Bellaterra, Spain. He is currently working toward the PhD degree in computer vision with the MatchLab Imperial Group, Imperial College London, London, U.K. His current research focuses on 3D geometry and deep learning.

Krystian Mikolajczyk received the PhD degree from the Institute National Polytechnique de Grenoble, Grenoble, France. He is currently a professor with Imperial College London. He held a number of research positions with INRIA, University of Oxford and Technical University of Darmstadt, as well as faculty positions with the University of Surrey, and Imperial College London. He has served in various roles at major international conferences co-chairing BMVC 2012, 2017 and IEEE International Conference on Advanced Video and Signal-Based Surveillance 2013. In 2014, he received Longuet-Higgins Prize awarded by the Technical Committee on Pattern Analysis and Machine Intelligence of the IEEE Computer Society.