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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Axel Barroso-Laguna1 | Edgar Riba2;3 | | Daniel Ponsa2 | | Krystian Mikolajczyk1 |
| 1Imperial College London | | 2Computer Vision Center | | | 3Arraiy, Inc. |
| faxel.barroso17, k.mikolajczykg@imperial.ac.uk | | | | feriba, danielg@cvc.uab.es | |

|  |
| --- |
| arXiv:1904.00889v3 [cs.CV] 12 Oct 2019 |

Abstract

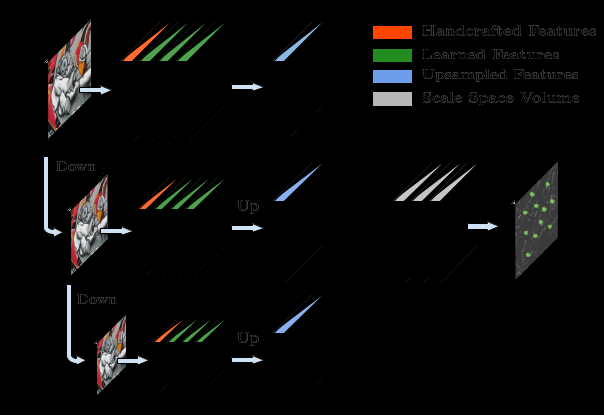
We introduce a novel approach for keypoint detection task that combines handcrafted and learned CNN filters within a shallow multi-scale architecture. Handcrafted fil-ters provide anchor structures for learned filters, which lo-calize, score and rank repeatable features. Scale-space rep-resentation is 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 benchmark. Results show that our approach out-performs state-of-the-art detectors in terms of repeatability, matching performance and complexity.

1. Introduction

Research advances in local feature detectors and descrip-tors led to remarkable improvements in areas such as im-age matching, object recognition, self-guided navigation or 3D reconstruction. Although the general direction of image matching methods is moving towards learned based sys-tems, the advantage of learning methods over handcrafted ones has not been clearly demonstrated in keypoint de-tection [[1].](#page9) In particular, Convolutional Neural Networks (CNNs) were able to significantly reduce matching error in local descriptors [[2],](#page9) despite the impractical inefficiency of the initial techniques [[3,](#page9) 4]. These works stimulated fur-ther research efforts and resulted in improved efficiency of CNN based descriptors, on the contrary, on top of the lim-ited success of learned detectors, a general trend towards dense rather than sparse representation and matching put aside local feature detectors. However, the growing pop-ularity of augmented reality (AR) headsets, as well as AR smartphone apps, has drawn more attention to reliable and efficient local feature detectors that could be used for sur-face estimation, sparse 3D reconstruction, 3D model acqui-sition or objects alignment, among others.

Traditionally, local feature detectors were based on engi-neered filters. For instance, approaches such as Difference

*H*



*W*

*M* 3*M*

Figure 1: The proposed Key.Net architecture combines handcrafted and learned filters to extract features at differ-ent scale levels. Feature maps are upsampled and concate-nated. Last learned filter combines the Scale Space Volume to obtain the final response map.

of Gaussians [[5],](#page9) Harris-Laplace or Hessian-Affine [[6]](#page9) 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 net-work could mimic the behavior of traditional detectors by learning the appropriate values in its convolutional filters. However, unlike the success with CNNs based local im-age descriptors, the improvements upon handcrafted detec-tors offered by recently proposed fully CNN based meth-ods [[7, 8, 9, 10, 11]](#page9) 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 partic-ularly problematic while other parameters such as dominant orientation can be regressed well by CNNs [[12,](#page9) 7]. This mo-tivates 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 figure [1.](#page1) Introducing handcrafted filters, which act as soft anchors, makes possible to reduce the number of parameters used by state-of-the-art detectors while maintaining the per-

formance 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 is able to propose the same fea-tures for images that undergo different geometric or photo-metric transformations. A number of related works have focused their objective function to address this issue, al-though they were based either on local patches [[9, 10]](#page9) or global map regression loss [[13, 14, 11].](#page9) In contrast, we ex-tend the covariant constraint loss to a new objective func-tion that combines local and global information. We design a fully differentiable operator, Multi-scale Index Proposal, that proposes keypoints at multi-scale regions. We exten-sively evaluate the method in recently introduced HPatches benchmark [[2]](#page9) in terms of accuracy and repeatability ac-cording to the protocol from [[15].](#page9)

In summary, our contributions are the following: a) a keypoint detector that combines handcrafted and learned CNN features, b) a novel multi-scale loss and operator for detecting and ranking stable keypoints across scales, c) a multi-scale feature detection with shallow architecture.

The rest of the paper is organized as follows. We re-view the related work in section [2.](#page2) Section [3](#page2) presents our proposed hybrid Key.Net architecture of handcrafted and learned CNNs filters and section [4](#page3) introduces the loss. Im-plementation and experimental details are given in section [5](#page5) and the results are presented in section [6.](#page6)

2. Related Work

There are many surveys that extensively discuss feature detection methods [[1, 16].](#page9) We present related works in two main categories: handcrafted and learned based.

2.1. Handcrafted Detectors

Traditional feature detectors localize geometric struc-tures through engineered algorithms, which are often re-ferred to as handcrafted. Harris [[17]](#page9) and Hessian [[18]](#page9) de-tectors 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 [[6, 19].](#page9) Later, SURF [[20]](#page9) accelerated the detection process by using integral images and an approximation of the Hes-sian matrix. Multi-scale improvements were proposed in KAZE [[21]](#page9) and its extension, A-KAZE [[22],](#page9) where Hessian detector was applied to a non-linear diffusion scale space in contrast to widely used Gaussian pyramid. Although corner detectors proved to be robust and efficient, other methods seek alternative structures within images. SIFT [[5]](#page9) looked for blobs over multiple scale levels, and MSER [[23]](#page9) seg-mented and selected stable regions as keypoints.

2.2. Learned Detectors

The success of learned methods in general object de-tection and feature descriptors motivated the research com-munity to explore similar techniques for feature detectors. FAST [[24]](#page9) was one of the first attempts to use machine learning to derive a corner keypoint detector. Further works extended FAST by optimizing it [[25],](#page9) adding a descriptor [[26]](#page9) or orientation estimation [[27].](#page9)

Latest advances in CNNs also made an impact on feature detection. TILDE [[14]](#page9) trained multiple piece-wise linear regression models to identify interest points that are robust under severe weather and illumination changes. [[9]](#page9) intro-duced a new formulation to train a CNN based on feature covariant constraints. Previous detector was extended in

1. by adding predefined detector anchors, showing im-proved stability in training. [[8]](#page9) presented two networks, MagicPoint, and MagicWarp, which first extracted salient points and then a parameterized transformation between pairs of images. MagicPoint was extended in [[13]](#page9) to Su-perPoint, which included a salient detector and descriptor. LIFT [[7]](#page9) implemented an end-to-end feature detection and description pipeline, including the orientation estimation for every feature. Quadruple image patches and a rank-ing scheme of point responses as cost function were used in [[28]](#page9) to train a neural network. In [[29],](#page9) authors proposed a pipeline to automatically sample positive and negative pairs of patches from a region proposal network to optimize jointly point detections and their representations. Recently, LF-Net [[11]](#page9) estimated position, scale and orientation of fea-tures by optimizing jointly the detector and descriptor.

In addition to the above presented learned detectors, CNN architectures also were deployed to optimize the matching stage. [[30]](#page9) learned to predict which features and descriptors were matchable. More recently, [[31]](#page9) intro-duced a network to learn to find good correspondences for wide-baseline stereo. Furthermore, other CNNs also stud-ied to perform tasks beyond detection or matching. In [[12],](#page9) the architecture assigned orientations to interest points and AffNet [[32]](#page9) used the descriptor loss to learn to predict the affine parameters of a local feature.

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 fea-tures and multi-scale pyramid representation.

3.1. Handcrafted and Learned Filters

The design of the handcrafted filters is inspired by the success of Harris [[17]](#page9) and Hessian [[18]](#page9) detectors, which used first and second order derivatives to compute the salient corner responses. A complete set of derivatives is

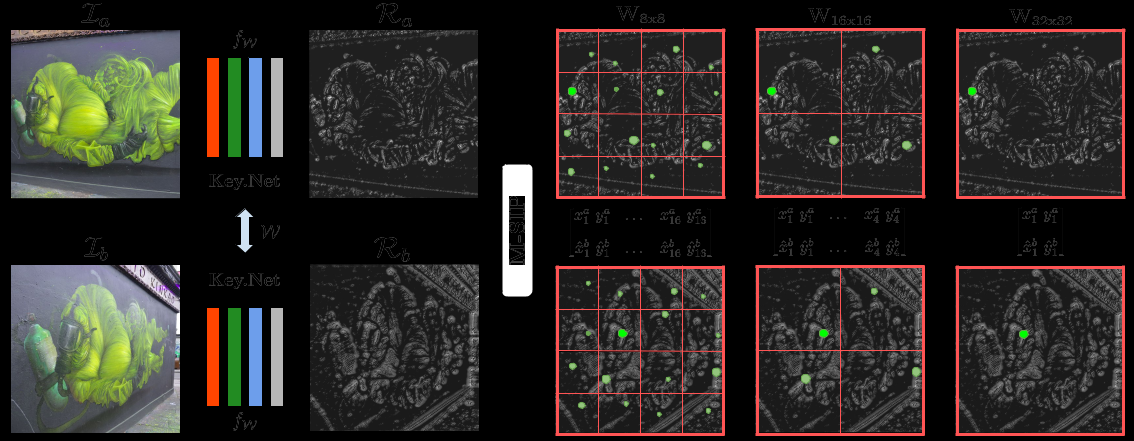


Figure 2: Siamese training process. Image Ia and Ib go through Key.Net to generate their response maps, Ra and Rb. M-SIP proposes interest point coordinates for each one of the windows at multi-scale regions. The final loss function is computed as a regression of coordinate indexes from Ia and local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

called LocalJet [[33]](#page9) and they approximate the signal in the local neighborhood as known from Taylor expansion:

|  |  |
| --- | --- |
| Ii1;:::;in = I0 @i1;:::;in g (~x); | (1) |

where g denotes the Gaussian of width centered at ~x = ~

0, and in 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 gradi-ents Ix and Iy. In addition, we compute Ix Iy, Ix2 and Iy2 as in the second moment matrix of Harris de-tector [[17].](#page9)

Second Order. From image I, 2nd order derivatives Ixx, Iyy and Ixy are also included as in the Hessian matrix used in Hessian and DoG detectors [[34,](#page9) 5]. Since Hessian detector uses the determinant of the Hessian matrix we add Ixx Iyy and Ixy2 .

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 figure [1,](#page1) the network includes three scale levels of the input image which is blurred and down-sampled by a factor of 1:2. All the feature maps result-ing from the handcrafted filters are concatenated to feed the

stack of learned filters in each of the scale levels. All three streams share the weights, such that the same type of an-chors result from different levels and form the set of candi-dates for final keypoints. Feature maps from all scale levels are then upsampled, concatenated and fed to the last convo-lutional filter to obtain the final response map.

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 detec-tors [[9, 28, 11]](#page9) train the network to identify keypoints with-out constraining their locations, where only the homogra-phy transformation between images is used as ground truth to calculate the loss as a function of keypoints repeatability.

Other works [[14, 13, 10]](#page9) show the benefits of using an-chors 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 meth-ods while allowing the detector to propose new stable key-points. 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.](#page4)

Extracting coordinates for training keypoint detectors has been widely studied and showed great improvements: [[7, 9, 10]](#page9) extracted coordinates in a patch level, SuperPoint

where Ra and Rb are the response map of Ia and Ib with coordinates related by the homography Hb;a. We skip ho-mogeneous coordinates for simplicity. Parameter i con-trols the contribution of each location based on its score

i

and i = Ra(xi; yi)a

Hb;a[^xi; y^i]Tb k2;

where W is a kernel of size N N with index values j = 1 : N along its columns, pointwise product , and cw is the top-left corner coordinates of window wi. 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 lo-cation of it. Depending on the base of the power expression in equation [2,](#page4) multiple local maxima may have a more or less significant effect on the resulting coordinates.

A detector is covariant if same features are detected un-der varying image transformations. Covariant constraint was formulated as a regression problem in [9]. Given im-ages Ia and Ib, and ground truth homography Hb;a between them, the loss L is based on the squared difference between points extracted by IP layer and actual maximum coordi-nates (NMS) in corresponding windows from Ia and Ib :

X

N

X

= [ui; vi]T = [W mi; W T mi]T

u;v

Due to exponential scaling the maximum dominates and the expected location calculated as the weighted average [ui; vi] gives an approximation of the maximum coordinates:

mi(u; v) =

[[13]](#page9) used a channel-wise softmax to get maxima belonging to fix grids of 8x8, and [[35]](#page9) 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 is able to return multiple global key-point coordinates centered on local maxima from a single image without constraining the number of keypoints to the depth of the feature map [[35]](#page9) or the size of the grid [[13].](#page9)

Similarly to handcrafted techniques, keypoint locations are indicated by local maxima of the filter response map R output by Key.Net. Spatial softmax operator is an effec-tive method for extracting the location of a soft maximum within a window [[7, 35, 11, 13].](#page9) Therefore, to ensure that the IP layer is fully differentiable, we rely on spatial soft-max operator to obtain the coordinates of a single keypoint per window. Consider a window wi of size N N in R, with the score value at each coordinate [u; v] within the win-dow, exponentially scaled and normalized:

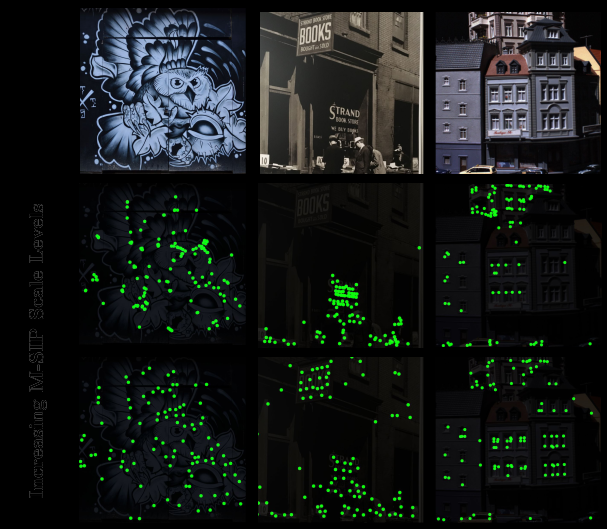
ewi(u;v)

:

PNj;k ewi(j;k)

[xi; yi]T

Figure 3: Keypoints obtained after adding larger context windows to M-SIP operator. The points that are more sta-ble remain as the M-SIP operator increases its window size.



1. Feature maps in the middle row contain points around edges or non discriminative areas, while bottom row shows detec-tions that are more robust under geometric transformations.

value, thus computing the loss for significant features only. As NMS is non-differentiable, gradients are only back-

+cw; (3) propagated where IP layer is applied, therefore, we switch

Ia and Ib and combine both losses to enforce consistency.

LIP (Ia; Ib; Ha;b; N) = ik[xi; yi]Ta

+ Rb(^xi; y^i)b;

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 increas-ing size only a few dominant keypoints survive in the im-age. In [[36],](#page9) 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. Multi-ple 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 [[37],](#page9) 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 Ns Ns and

1. computes the candidate keypoint position for each window as shown in figure [2.](#page3) Our proposed loss function is the av-erage of covariant constraint losses from all scale levels:

X

LMSIP (Ia; Ib; Ha;b) = sLIP (Ia; Ib; Ha;b; Ns); (5)

s

where s is the index of the scale level with Ns as window size, LIP is the covariant constraint loss and s is the con-trol 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 [[6].](#page9)

The combination of different scales imposes an intrinsic process of simultaneous scoring and ranking of keypoints within the network. In order to minimize the loss, the net-work must learn to give higher scores to robust features that remain dominant across a range of scales. Figure [3](#page4) shows different response maps for increasing window size.

5. Experimental Settings

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

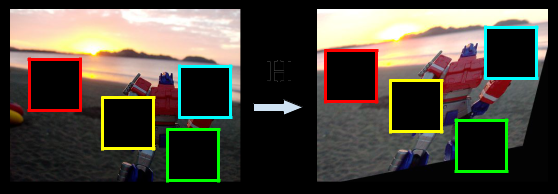
5.1. Training Data

We generate a synthetic training set from ImageNet ILSVRC 2012 dataset. We apply random geometric trans-formations to images and extract pairs of corresponding re-gions as our training set. The process is illustrated in fig-ure [4.](#page5) The parameters of the transformations are: scale [0:5; 3:5], skew [ 0:8; 0:8] and rotation [ 60 ; 60 ]. Tex-tureless regions are not discriminative, therefore, we dis-card them by checking if the response of any of the hand-crafted filters is lower than a threshold. We modify the con-trast, brightness and hue value in HSV space to one of the images to improve network’s robustness against illumina-tion changes. In addition, for each pair, we generate bi-nary masks that indicate the common area between images. Those masks are used in training to avoid regressing in-dexes of keypoints that are not present in the common re-gion. There are 12,000 image pairs of size 192 192. We use 9,000 of them as the training data and 3,000 as valida-tion set.

5.2. Evaluation Metrics

We follow the evaluation protocol proposed in [[15]](#page9) and improved in the follow up works [[7, 9, 10,](#page9) 1]. Repeatability score for a pair of images is computed as the ratio between the number of corresponding keypoints and the lower num-ber 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 [[25, 14].](#page9) In addition, as exposed by [[1],](#page9) 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 iden-tify corresponding keypoints we compute the Intersection-over-Union error, IoU , between the areas of the two can-didates. To evaluate the accuracy of keypoint location and

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



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 top 1,000 interest points that belong to the common region between images and a match is considered correct when IoU is smaller than 0.4 i.e., the overlap between corresponding regions is more than 60%. The scales are normalized as in [[1],](#page9) which sets the larger size in a pair of points to 30 pixels, and rescales the other one accordingly. Non-maxima suppres-sion of 15 15 is performed at inference time during eval-uation. HPatches [[2]](#page9) dataset is used for testing. HPatches contains 116 sequences, which are split between viewpoint and illumination transformations, 59 and 57 sequences re-spectively. HPatches offers predefined image patches for evaluating descriptors, instead, we use full images for eval-uating keypoint detectors.

5.3. Implementation Notes

Training is performed in a siamese pipeline, with two instances of Key.Net that share the weights and are up-dated at the same time. Each convolutional layer has M

* 8 filters of size 5 5, with He weights initialization and L2 kernel regularizer. We compute the covariant con-straint loss LM-SIP for five scale levels, with the size of the M-SIP windows Ns 2 [8; 16; 24; 32; 40] and loss term s 2 [256; 64; 16; 4; 1], that were determined by perform-ing a hyperparameter search on the validation set. Larger candidate window sizes have greater mean errors between coordinate points since the maximum distance is propor-tional to the window size. Thus, s has the largest value for the smallest window. We use a batch size of 32, an 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 a NVIDIA GeForce GTX 1080 Ti. Evalua-tion benchmark, synthetic data generator, Key.Net network, and loss are implemented using TensorFlow and are avail-able on GitHub[1](#page5).
* https://github.com/axelBarroso/Key.Net

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | M-SIP Region Sizes | | |  |  |
| W8x8 | W16x16 | W24x24 | W32x32 | W40x40 | Repeatability |
| X | - | - | - | - | 70.5 |
| X | X | - | - | - | 74.6 |
| X | X | X | - | - | 76.8 |
| X | X | X | X | - | 77.6 |
| - | - | - | - | X | 65.7 |
| - | - | - | X | X | 71.4 |
| - | - | X | X | X | 73.2 |
| - | X | X | X | X | 74.9 |
| X | X | X | X | X | 79.1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 5 | Full Learnable | |  |  |  |  |
| Blocks |  | 1st Order |  |  |  |  |  |
|  |  | 2nd Order |  |  |  |  |  |
| Learnable |  | 4 | 1st and 2nd Order | |  |  |  |  |
|  |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |
| Number |  | 2 |  |  |  |  |  |  |
|  |  | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | 0.72 | | 0.74 | 0.76 | 0.78 | 0.80 | |  |
|  |  |  |  | Repeatability |  |  |  |  |

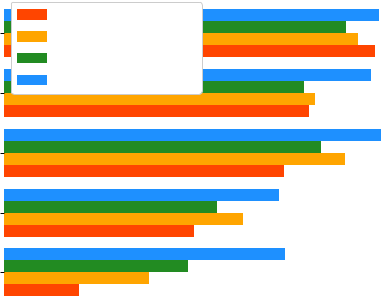


Figure 5: Left: Comparison of repeatability results for several levels in the M-SIP operator. We show different combinations of context losses as the final loss, from smaller to larger regions. The best result is obtained when using five window sizes from 8 8 up to 40 40. Right: Repeatability results 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.

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.](#page4)

Table 1: Repeatability results for different design choices on synthetic validation set from ImageNet.

6. Results

In this section, we present the experiments and discuss the results. We first show results on validation data for sev-eral variants of the proposed architecture. Next, Key.Net repeatability scores in single-scale and multi-scale are pre-sented along with the state-of-the-art detectors on HPatches. Moreover, we evaluate the matching performance, the num-ber of learnable parameters and inference time of our pro-posed detector and compare 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 figure [5](#page6) (Left) showing in-creasing repeatability with more scale levels within M-SIP operator. In addition, we show how the loss with smaller window size N improves repeatability. However, the best result is obtained when all levels are combined.

Filter Combinations are analyzed in figure [5](#page6) (Right). We

show results for 1st and 2nd order filters as well as their combination. All networks have the same number of fil-ters, however, we either freeze first layer of 10 filters with handcrafted kernels (c.f. section [3.1)](#page2) or learn them depend-ing 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 informa-tion 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 gra-dients, i.e. non-discriminative with low repeatability. How-ever, as we add more learnable blocks, repeatability scores for combined and fully learnable networks become compa-rable. Naturally, gradient-based handcrafted filters are sim-ple, 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 appli-cations. 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 [1a.](#page6) 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 exper-iment 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](#page4) defines how soft the es-timation of keypoint coordinates is. High values return the

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Viewpoint | | | | | | |  |  |  |  |  |  |  |  |  |  | Illumination | | | | | |  |  |  |  |  |
|  |  | Repeatability | | | |  |  |  | IoU | | | |  | Srange | | |  | Repeatability | | | | |  |  | IoU | | | |  | Srange | | |  |
|  |  | SL |  | L | |  |  |  | SL |  | L | |  |  | SL | |  |  | SL |  | L | |  |  | SL |  | L | |  |  | SL | |  |
|  |  |  |  |  |  |  | |  |  |  |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |
| SIFT-SI [[5]](#page9) | 43.1 | | 57.6 | |  |  | | 0.18 | | 0.12 | |  | 78.6 | | |  | 47.8 | | | 60.4 | |  | 0.18 | | | 0.12 | | | 84.5 | | | |  |
| SURF-SI [[20]](#page9) | 46.7 | | 60.3 | |  | 0.18 | | | | 0.18 | |  | 24.8 | | |  | 53.0 | | | 64.0 | |  | 0.15 | | | 0.11 | | | 27.4 | | | |  |
|  |  |  |  |  |  |  | | |  | 0.10 | |  |  | |  |  |  | |  |  |  |  | 0.09 | | | 0.09 | | |  | |  |  |  |
| FAST-TI [[24]](#page9) | 30.4 | | 63.1 | |  | 0.21 | | | |  | - | | |  | 63.6 | | | 63.6 | |  | - | | |  |  |
|  |  |  |  |  |  |  | | | |  |  |  |  | |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |
| MSER-SI [[23]](#page9) | 56.4 | | 62.8 | |  | 0.12 | | | | 0.08 | |  | 503.7 | | | | 46.5 | | | 54.5 | |  | 0.12 | | | 0.10 | | | 524.8 | | | |  |
| Harris-Laplace-SI [[34]](#page9) | 45.1 | | 62.0 | |  | 0.20 | | | | 0.13 | |  |  | | 95.9 |  | 52.7 | | | 62.0 | |  | 0.17 | | | 0.08 | | | 90.4 | | |  |  |
|  |  |  |  |  |  |  | | | |  | |  |  | |  |  |  | |  |  |  |  |  | |  |  | | |  | |  |  |  |
| KAZE-SI [[21]](#page9) | 53.3 | | 65.7 | |  | 0.20 | | | | 0.11 | |  |  | | 12.5 |  | 56.9 | | | 65.7 | |  | 0.12 | | | 0.10 | | | 12.7 | | | |  |
| AKAZE-SI [[22]](#page9) | 54.0 | | 65.6 | |  | 0.19 | | | | 0.10 | |  | 13.5 | | |  | 64.9 | | | 69.1 | |  | 0.11 | | | 0.09 | | | 13.6 | | | |  |
|  |  |  |  |  |  |  | | | |  |  |  |  | | |  | 70.4 | | |  |  |  |  | |  |  |  |  |  | | |  |  |
| TILDE-TI [[14]](#page9) | 31.0 | | 65.1 | |  | 0.20 | | | | 0.15 | |  | - | | |  | 70.4 | |  | 0.11 | | | 0.11 | |  | - | | |  |  |
|  |  |  |  |  |  |  | | | |  | |  |  | | |  |  | |  |  |  |  |  | |  |  | | |  | | | |  |
| LIFT-SI [[7]](#page9) | 43.4 | | 59.4 | |  | 0.20 | | | | 0.13 | |  | 13.3 | | |  | 51.6 | | | 65.4 | |  | 0.18 | | | 0.12 | | | 13.8 | | | |  |
| DNet-SI [[9]](#page9) | 49.4 | | 62.2 | |  | 0.21 | | | | 0.14 | |  | 11.4 | | |  | 59.1 | | | 65.1 | |  | 0.14 | | | 0.13 | | | 17.1 | | | |  |
| TCDET-SI [[10]](#page9) | 49.6 | | 61.6 | |  | 0.23 | | | | 0.16 | |  | 6.7 | | |  | 66.9 | | |  | 71.0 |  | 0.16 | | | 0.15 | | | 11.4 | | |  |  |
| SuperPoint-TI [[13]](#page9) | 33.3 | | 67.1 | |  | 0.20 | | | | 0.17 | |  | - | | |  | 69.9 | | | 69.9 | |  | 0.10 | | | 0.10 | | | - | | |  |  |
|  |  |  |  |  |  |  | | | |  | |  |  | | |  |  | | |  | |  |  | |  |  | | |  | | |  |  |
| LF-Net-SI [[11]](#page9) | 32.3 | | 62.2 | |  | 0.23 | | | | 0.12 | |  | 2.00 | | |  | 68.6 | | | 69.1 | |  | 0.10 | | | 0.10 | | | 2.0 | | |  |  |
|  |  |  |  |  |  |  | | | |  | |  |  | | |  |  | | |  | |  |  | |  |  | | |  | | |  |  |
| Tiny-Key.Net-SI |  | 57.8 | 70.3 | |  | 0.20 | | | | 0.12 | |  | 7.6 | | |  | 56.1 | | | 62.8 | |  | 0.14 | | | 0.11 | | | 7.6 | | |  |  |
| Key.Net-TI | 34.2 | | 71.5 | |  | 0.20 | | | | 0.11 | |  | - | | |  | 72.0 | | | 72.0 | |  | 0.10 | | | 0.10 | | | - | | |  |  |
|  |  | |  |  |  |  | | | |  | |  |  | | |  |  | | |  | |  |  | |  |  | | |  | | |  |  |
| Key.Net-SI | 60.5 | | 73.2 | |  | 0.19 | | | | 0.14 | |  | 7.6 | | |  | 61.3 | | | 66.2 | |  | 0.12 | | | 0.10 | | | 7.6 | | |  |  |

Table 2: Repeatability results (%) for translation (TI) and scale (SI) invariant detectors on HPatches. We also report average overlap error IoU and ratio of maximum to minimum extracted scale SRange. 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 and Tiny-Key.Net are the best algorithms on viewpoint, for both L and SL. On illumination sequences, translation invariant Key.Net-TI obtains the best accuracy. Among scale invariant SI detectors, TCDET is the best in L and LF-Net in SL.

location of the global maximum within the window, while low values average local maxima. The base is varied in ta-ble [1b.](#page6) Optimum scores are obtained when using the base in equation [2](#page4) close to the e value, which is in line with the setting used in [[35].](#page9)

6.2. Keypoint Detection

This section presents the results for state-of-the-art local feature detectors along with our proposed method. Table [2](#page7) shows the repeatability score, average intersection-over-union error IoU and scale range Srange, which is the ratio between the maximum and minimum scale values of the ex-tracted interest points. Suffixes -TI and -SI, refer to trans-lation (detection at a single scale only) and scale invariance (detection at multiple scales), respectively. Keypoint loca-tion is only evaluated under L by assuming correct scale detection, while scale and location (SL) use the actual de-tected scale and location 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 [[10].](#page9) We also show results on single scale input Key.Net-TI, to compare it directly with other TI detectors such as SuperPoint or TILDE. We set the thresholds of algorithms such that they return at least 1,000 points per image. As MSER proposes regions without scor-ing or ranking, we randomly pick 1,000 points to compute the results. We repeat this experiment ten times and aver-age the results for MSER. 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-TI and Key.Net-SI.

On illumination sequences, Key.Net-TI 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 com-pared to other SI detectors. Note that TILDE based detec-tors were specifically designed and trained for illumination sequences. LF-Net is the best SI detector according to SL overlap, not suffering much from incorrect scale estima-tions. However, its repeatability decreases the most from L to SL among all SI detectors on viewpoint sequences. Key.Net-SI addresses the scale changes better than the other methods but the errors in multi-scale sampling affect it

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Matching Score | | | | | |
|  | View | | | Illum | | |
|  |  |  |  |  |  |  |
| MSER [[23]](#page9) + HardNet [[38]](#page9) | 11.7 | |  | 18.8 | |  |
| SIFT [[5]](#page9) + HardNet [[38]](#page9) | 23.2 | |  | 24.8 | |  |
| HarrisLaplace [[34]](#page9) + HardNet [[38]](#page9) | 30.0 | |  | 31.7 | |  |
| AKAZE [[22]](#page9) + HardNet [[38]](#page9) | 36.4 | |  | 41.4 | |  |
| TILDE [[14]](#page9) + HardNet [[38]](#page9) | 32.3 | |  | 39.3 | |  |
| LIFT [[7]](#page9) + HardNet [[38]](#page9) | 30.3 | |  | 32.8 | |  |
| DNet [[9]](#page9) + HardNet [[38]](#page9) | 33.5 | |  | 34.7 | |  |
| TCDET [[10]](#page9) + HardNet [[38]](#page9) | 27.6 | |  | 36.3 | |  |
| SuperPoint [[13]](#page9) + HardNet [[38]](#page9) | 37.4 | |  |  | 43.0 |  |
| LF-Net [[11]](#page9) + HardNet [[38]](#page9) | 26.9 | |  | 43.8 | |  |
|  |  |  |  |  | |  |
| LIFT [[7]](#page9) | 21.8 | |  | 26.5 | |  |
| SuperPoint [[13]](#page9) | 38.0 | |  | 41.5 | |  |
|  |  |  |  |  | |  |
| LF-Net [[11]](#page9) | 23.0 | |  | 29.1 | |  |
|  |  | |  |  | |  |
| Tiny-Key.Net + HardNet [[38]](#page9) | 37.9 | |  | 37.3 | |  |
| Key.Net + HardNet [[38]](#page9) | 38.4 | |  | 39.7 | |  |

Table 3: Matching score (%) of best detectors together with HardNet and state-of-the-art detector/descriptors. Results on HPatches sequences, both viewpoint, and illumination. Key.Net architecture gets the best matching score for view-point, while LF-Net+HardNet for illumination sequences.

when there is no scale change between images i.e. illumina-tion sequences. This has often been observed for detectors with more invariance than required by the data. Handcrafted detectors have the lowest average overlap error IoU among all detectors. A wide range of scales Srange is detected by MSER, which has a great capability of extracting local fea-tures from different scales due to its feature segmentation nature.

6.3. Keypoint Matching

Moreover, in order to demonstrate that the detected fea-tures are useful for matching, table [3](#page8) shows matching scores for detectors combined with HardNet descriptor [[38].](#page9) 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. In addi-tion, we include in the table LIFT [[7],](#page9) SuperPoint [[13]](#page9) and LF-Net [[11]](#page9) with their descriptors, but ignoring their ori-entation estimation. SuperPoint and LF-Net have 256 de-scriptor dimension, while dimension of HardNet [[38]](#page9) and LIFT is 128. Matching score is computed as the ratio between features matched and detected (top 1,000). Top matching scores is obtained by Key.Net on viewpoint, and LF-Net+HardNet on illumination. Feature detectors that were optimized jointly with a descriptor [[7, 13, 11]](#page9) have better matching score than regular learned detectors on il-

Number of Learnable Parameters

TCDET SuperPoint LF-Net Tiny-Key.Net Key.Net

548k 940k 39k 280 5.9k

Table 4: Comparison of the number of learnable parameters for state-of-the-art architectures. Tiny-Key.Net has only one learnable block with one filter.

lumination sequences, but not on viewpoint. Handcrafted AKAZE performs close to the top learned methods for both viewpoint and illumination sequences.

6.4. Efficiency

We also compare the number of learnable parameters, in-dicating then the complexity of the predictor, which leads to an increasing risk of overfitting and need for a large amount of training data. Table [4](#page8) shows the approximate number of parameters for different architectures. Learnable parame-ters that are not used during inference in the detector part are not counted for SuperPoint and LF-Net detectors. The highest complexity is from SuperPoint with 940k learnable parameters. Key.Net has nearly 160 times fewer parameters and Tiny-Key.Net has 3,100 times fewer parameters than SuperPoint with better repeatability for viewpoint scenes. The inference time of an image of 600 600 is 5.7ms (175 FPS) and 31ms (32.25 FPS) for Tiny-Key.Net and Key.Net, respectively.

7. Conclusions

We have introduced a novel approach to detect local fea-tures 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 benchmark show that combining handcrafted and learned features as well as multi-scale analysis at different stages of the net-work improves the repeatability scores compared to other state-of-the-art keypoint detection methods.

We further show that excessively increasing 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 detector with 280 learnable parameters and inference of 175 frames per sec-ond. Proposed detectors lead to state-of-the-art matching performance when used with a descriptor on viewpoint.

References

1. Karel Lenc and Andrea Vedaldi. Large scale evaluation of lo-cal image feature detectors on homography datasets. BMVC, 2018.
2. Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krys-tian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. CVPR, 2017.
3. Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Suk-thankar, and Alexander C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. CVPR, 2015.
4. Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. CVPR, 2015.
5. David G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 2004.
6. Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. ICCV, 2004.
7. Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. ECCV, 2016.
8. Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabi-novich. Toward geometric deep slam. arXiv preprint arXiv:1707.07410, 2017.
9. Karel Lenc and Andrea Vedaldi. Learning covariant feature detectors. ECCV, 2016.
10. Xu Zhang, Felix X. Yu, Svebor Karaman, and Shih-Fu Chang. Learning discriminative and transformation covari-ant local feature detectors. CVPR, 2017.
11. Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. LF-Net: Learning Local Features from Images. NIPS, 2018.
12. Kwang Moo Yi, Yannick Verdie, Pascal Fua, and Vincent Lepetit. Learning to assign orientations to feature points. CVPR, 2016.
13. Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabi-novich. Superpoint: Self-supervised interest point detection and description. CVPR Workshop, 2017.
14. Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Tilde: a temporally invariant learned detector. CVPR, 2015.
15. Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. TPAMI, 2005.
16. Tinne Tuytelaars and Krystian Mikolajczyk. Local invari-ant feature detectors: a survey. Foundations and Trends in Computer Graphics and Vision, 2008.
17. Chris Harris and Mike Stephens. A combined corner and edge detector. Alvey Vision Conference, 1988.
18. Paul Beaudet. Rotationally invariant image operators. ICPR, 1978.
19. Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. IJCV, 2005.
20. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). Computer Vision and Image Understanding, 2008.
21. Pablo Fernandez´ Alcantarilla, Adrien Bartoli, and Andrew J. Davison. Kaze features. ECCV, 2012.
22. Pablo Fernandez´ Alcantarilla, Jesus´ Nuevo, and Adrien Bar-toli. Fast explicit diffusion for accelerated features in non-linear scale spaces. BMVC, 2013.
23. Jiri Matas, Chum Ondrej, Urban Martin, and Pajdla Toms. Robust wide-baseline stereo from maximally stable extremal regions. Image and Vision Computing, 2004.
24. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. ECCV, 2006.
25. Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. TPAMI, 2010.
26. Stefan Leutenegger, Chli Margarita, and Siegwart Roland. Brisk: Binary robust invariant scalable keypoints. ICCV, 2011.
27. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. ICCV, 2011.
28. Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sat-tler, and Marc Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. CVPR, 2017.
29. Georgios Georgakis, Srikrishna Karanam, Ziyan Wu, Jan Ernst, and Jana Kosecka. End-to-end learning of key-point detector and descriptor for pose invariant 3d matching. CVPR, 2018.
30. Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. CVPR, 2014.
31. Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. CVPR, 2018.
32. Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Repeata-bility is not enough: Learning affine regions via discrim-inability. ECCV, 2018.
33. Luc Florack, Bart Ter Haar Romeny, Max Viergever, and Jan Koenderink. The gaussian scale-space paradigm and the multiscale local jet. IJCV, 2002.
34. Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. ICCV, 2001.
35. Supasorn Suwajanakorn, Noah Snavely, Jonathan Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. NIPS, 2018.
36. Jingming Dong and Stefano Soatto. Domain-size pooling in local descriptors: Dsp-sift. CVPR, 2017.
37. Stepan Obdrzalek and Jiri Matas. Object recognition using local affine frames on distinguished regions. BMVC, 2002.
38. Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor’s mar-gins: Local descriptor learning loss. NIPS, 2017.