Learning to Detect and Match Keypoints with Deep Architectures

Hani Altwaijry[1,2]
haltwaijry@cs.cornell.edu
Andreas Veit[1,2]
aveit@cs.cornell.edu
Serge Belongie[1,2]
sjb344@cornell.edu

[1] Cornell University
Ithaca, NY, USA
[2] Cornell Tech
New York, NY, USA

Abstract

Feature detection and description is a pivotal step in many computer vision pipelines. Traditionally, human engineered features have been the main workhorse in this domain. In this paper, we present a novel approach for learning to detect and describe keypoints from images leveraging deep architectures. To allow for a learning based approach, we collect a large-scale dataset of patches with matching multiscale keypoints. The proposed model learns from this vast dataset to identify and describe meaningful keypoints. We evaluate our model for the effectiveness of its learned representations for detecting multiscale keypoints and describing their respective support regions.

1 Introduction

The extraction of effective features is a key step in many machine learning and computer vision algorithms and their applications. In computer vision, one form of feature extraction is concerned with the detection and description of important image regions. Traditionally, these features are extracted using hand engineered detectors and descriptors. Approaches adopting this paradigm are generally referred to as *keypoint-based* or *feature-based* approaches.

Recently, the reintroduction of neural networks into many computer vision tasks broadly replaced hand-engineered feature-based approaches. Neural network based approaches generally learn the feature extraction as part of an end-to-end pipeline. While these approaches have shown great success in tasks such as scene recognition, object detection and classification, other tasks such as structure-from-motion still depend on purely engineered features, *e.g.* SIFT [18], to detect and describe keypoints.

In this paper, we propose a model that learns what constitutes a good keypoint, is capable of capturing keypoints at multiple scales and learns to decide whether two keypoints match. We achieve multiscale keypoint detection with a fully-convolutional network that recursively applies convolutions to regresses keypoint scores. With each successive convolution, the network evaluates image patches, i.e., keypoints, at a larger scale. By extracting the keypoint feature map after each convolution we obtain a feature map that resembles a keypoint scale-space. To learn descriptors for keypoint matching, we leverage a triplet network to learn an

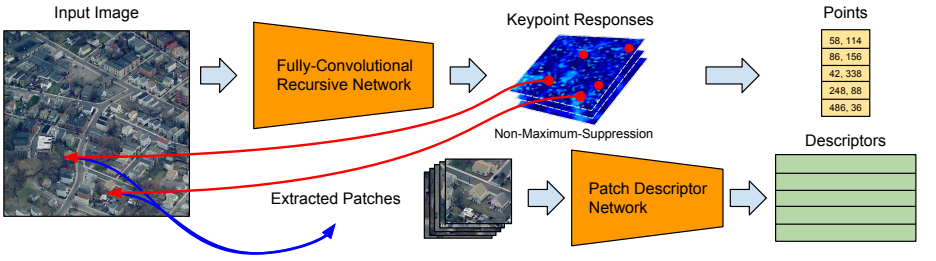

Figure 1: Proposed architecture for learning to detect and describe keypoints at multiple-scales. Given an image, a fully-convolutional recursive network outputs a scale-pyramid of keypoint responses, which are used to extract patches. Then, the patches are described by a patch descriptor network.

embedding where patches of matching keypoints are closer to each other than non-matching patches. Figure 1 provides an overview of our proposed model.

There is currently no large-scale dataset for learning both keypoint detectors and descriptors from image patches. Furthermore, finding training examples to train deep neural networks for this task poses a serious challenge, as collecting human annotated examples would be prohibitively expensive. Therefore, we create our own dataset by following a self-supervised approach, where we utilize structure-from-motion to build a large database of keypoints and matching image patches. Although those feature matches were determined originally with engineered features, structure-from-motion also factors in the underlying geometry. We only consider those keypoints that went through rigorous geometric filtering, which allows the learning of features that extend upon their engineered counterparts.

To create our supervisory examples, we collect a dataset of aerial oblique imagery and construct a large-scale model of 1.3 million 3D points using VisualSFM [28, 29]. We used those 3D points to extract matching patches exhibiting varying photometric differences, including: scale, illumination, perspective. Those patches formed the basis from which our deep neural network model was able to learn to detect and match keypoints.

We evaluate the proposed model both quantitatively and qualitatively and show that it is capable of identifying keypoints at multiple scales as well as matching them.

Our main contributions are:

1. We propose a novel approach capable of learning to detect multiscale keypoints and descriptors for effective correspondence matching.

2. We introduce a large-scale dataset composed of over 2.5 million matching image patches at varying scales.

2 Related Work

2.1 Feature Extraction and Description

The computer vision literature has served up a large number of engineered feature extractors and descriptors, such as SIFT [18], HOG [9], SURF [4], ORB [7], and BRISK [16]. These extractors and descriptors were designed with multiple goals in mind, such as optimizing for matching accuracy or extraction and matching speeds. In general, they have been

demonstrated to perform well in various applications of computer vision. Furthermore, the literature has seen approaches that learn keypoint detectors [13, 27] and descriptors [3, 6, 24]. We contrast this work by striving to learn both the keypoint detector and descriptor.

In correspondence matching problems, descriptors are used to find geometrical relationships between two or more sets of keypoints, which are then filtered by imposing geometrical constraints through model fitting techniques such as RANSAC [10]. Structure-from-motion solutions, *e.g.* VisualSFM [28, 29], start with correspondence matching, and expand the computed relationships to many images building a global model governing all. In this work, we leverage the compounded effect of geometry on engineered features to provide our supervisory signal.

2.2 Deep-learning and Matching Images

In recent years, the computer vision literature has seen a surge of state-of-the-art results, on all fronts, surfacing from research on Deep Convolutional Neural Networks [11, 15, 25, 26].

In [12, 23, 30], deep architectures were proposed to learn feature descriptors. The siamese architecture [5] forms the basis for these approaches, with the neural networks learning to embed 64×64 patches in a feature space where matching patches are closer to each other than non-matching patches. Their supervisory signal is based on structure-from-motion patches originally used in [6]. However, they do not learn keypoint detection, and do not handle various scales natively. We build on these approaches by showing how to create a model that learns to predict the keypoints and their respective descriptors at various scales.

The detection of salient regions with deep architectures has been mostly discussed within the object detection and recognition literature. In [17], features in later layers were shown to correspond to fine details in the receptive fields covered by those features. One approach to generate salient region proposals are visual attention models, e.g., [2, 20]. There, a recurrent network is trained to examine and propose regions of the image space sequentially. Attention mechanisms typically learn the salient features in an unsupervised manner. One particular approach is the Spatial Transformer Network [14] which describes a region proposal scheme capable of highlighting regions with associated transformations to a canonical pose that is learned automatically. In [8], spatial transformer networks are used to detect a fixed number of probable patch matches. In essence, there the network attempts to detect and match patches simultaneously, with only weak-supervision from match/no-match labels on the image level. Another approach to generate region proposals are Region Proposal Networks [19] that have the sole purpose of identifying regions of the image space that contain objects. Region proposal networks are generally trained in a fully supervised manner. This approach has been recently extended [21] by coupling the proposal network with the classification network for faster performance. We draw inspiration from these works for modeling a network capable of proposing keypoints.

3 Learning Model

The goal of this model is to learn to detect and match keypoints in images. We achieve this by using two models, one for each task. The first is a keypoint detection network, and the second is a keypoint description network.

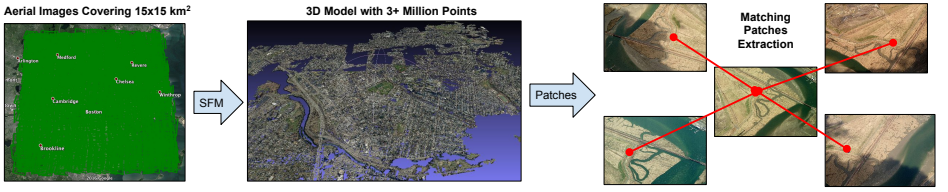

Figure 2: Generating multiscale matching patches using structure-from-motion.

3.1 Training Data

To train the keypoint detection network and the keypoint matching network, we require a large set of patches with high quality keypoints that are also annotated with pairwise match information. Since no such large scale dataset currently exists, a key aspect of our data-driven learning approach is the collected of a large scale training dataset. For that purpose, we follow a self-supervised data collection scheme.

Our data generation approach is similar to that of [6]. We rely on structure-from-motion techniques to identify good keypoints and to generate matching patches. However, our approach differs in that we keep the original patch sizes, without rescaling to a canonical size. This allows us to train a multiscale detector.

We use aerial imagery covering an area of $15 \times 15 km^2$ around the city of Boston, Massachusetts to construct a 3D model using VisualSFM [28, 29]. The model contains 1.3 million 3D points where each is observed from at least two cameras, *i.e.* images. For a single 3D point with k associated keypoints, there are $k(k-1)/2$ unique keypoint pairs that we can extract as matching patch pairs. To generate patches that do not constitute good keypoints, we randomly sample image patches that do not belong to any keypoints. Figure 2 gives an overview of the approach.

The keypoint scales extracted from the 3D model are continuously valued. For our approach, we discretize the scale values into five scales: $S = \{64, 96, 128, 192, 256\}$. We determined the set of scales by clustering the scale ranges in the extracted dataset. As we show later, this discretizion does not limit the model. The fixed scale range directly affects our design, however only in one way: the smallest scale the model handles is 64×64. There is, however, no limit on the largest scale. This is a result of the recursive architecture, which we will discuss in the following subsection.

We denote the generated set of patches P as follows:

$$P = \{p_i : (x_i, s_i, k_i); \quad k \in \{-1, 1\}\} \tag{1}$$

where p_i is a patch with: x_i as the raw pixels, s_i is the scale of the patch, and k_i is the keypoint label. Further, we denote the generated set of matches M as

$$M = \{m_i : (p_j, p_k, y_i); \quad p_{j,k} \in P, \quad y_i \in \{-1, 1\}\} \tag{2}$$

where each match m_i is tuple that references two patches p_j and p_k with y_i being the match true/false label.

3.2 Detection Network

The goal of the detection network is to identify the regions of the input image that constitute good keypoints. Identifying keypoint regions includes both finding the optimal keypoint lo-

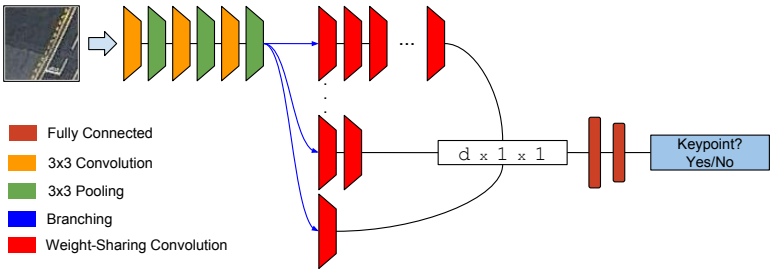

Figure 3: Training architecture for multiscale keypoint detection network. First, patches pass through a set of convolutions and pooling layers. Then, a recursive convolution is applied until the feature map dimension is 1×1. Since a batch can contain patches of different scales, a scale-dependent branch is chosen for each patch determining the number of recursive convolutions. Finally, two fully connected layers lead into a binary keypoint classifier.

cations as well as their scales. In particular, we learn a nonlinear function $f(X)$, from images X into a feature space $\mathbb{R}^{w \times h}$, where high activations correspond to respective image regions that constitute good keypoints. The architecture used for training the detection network differs slightly from the architecture used during inference, since it is trained on image patches, but inference is performed on whole images.

3.2.1 Training Procedure

Figure 3 illustrates the training architecture. The input to the network are batches of patches $\{p_i\} \subset P$ and associated binary labels indicating whether the patches represent good keypoints. In essence, the detection network is a binary classification CNN that learns to decide whether a given patch constitutes a good keypoint or not. As such, it consists of a sequence of convolutional and pooling layers followed by two fully-connected layers for classification.

Keypoints vary largely with respect to their scale and thus the patches come in many different sizes. To address this, we propose a scale-dependent branching mechanism, shown in Figure 3 by blue arrows. There, a scale-dependent branch is chosen for each patch. Within each branch convolutional filters are applied recursively until the output feature is of dimension $(d \times 1 \times 1)$. This allows for encoding keypoints of varying scales in a common feature space of fixed size. This is essential for efficient multiscale inference. All convolutions across all scale-dependent branches share the same weights, allowing for inference over arbitrarily large input images. In essence, the recursive application of the same convolutional filters resembles a rolled-out recurrent neural network for handling multiscale inputs. The d-dimensional output from the recursive branches is then used to determine whether the patch is centered around a good keypoint.

To sample training patches, we use hard-negative mining to improve the performance of the keypoint detector. For each training batch, we randomly sample the dataset searching for patches with high-loss, to construct batches of difficult examples. Each batch is chosen to have a mix of positives and negatives with a 1:1 ratio.

3.2.2 Training Objective

We define a loss-function $\mathcal{L}_{\text{KP}}$ for training the keypoint detection network. The loss function comprises two terms. First, as we model keypoint detection as a binary classification prob-

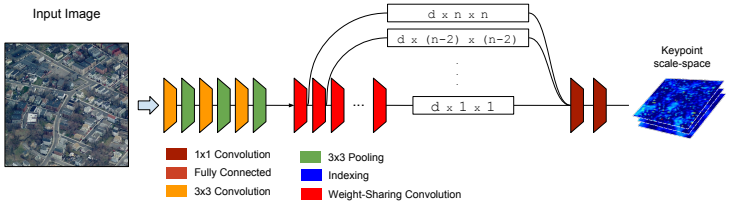

Figure 4: Inference architecture for multiscale keypoint detection network. First, an input image is passed through a set of convolutions and pooling layers. Then, a recursive convolution is applied until the feature map dimension is 1×1. After each recursive convolution we compute the keypoint feature map. Since convolutions later in the network have larger receptive fields the output feature maps resemble a keypoint *scale-space*.

lem, we make use of the hinge-loss to define our first term. Second, we use a squared difference loss to penalize network responses on non-centered patches. This employs a Gaussian-like response around the center of the patch and is inspired by the response shape penalty used in [27]:

$$h_j = e^{-\frac{\|v_j\|_2}{2\sigma^2}} \tag{3}$$

where v_j is the vector from the keypoint towards the center of the patch. During training, non-centered patches are generated by extracting patches jittered around the keypoints. This results in data-augmentation as well as incentivizes maximum responses around the centers of informative regions. Putting the two terms together, the joint loss function is given as

$$\mathcal{L}_{\text{KP}} = \frac{1}{N}\sum_j \left[\lambda \max\left(0, 1 - y_j x_j\right) + (1-\lambda)\left(x_j - h_j\right)^2\right] \tag{4}$$

with x_j as network output, $y_j \in \{-1, 1\}$ as the training label, and λ as mixing-weight.

3.2.3 Inference

Figure 4 depicts the inference architecture, which differs slightly from the training architecture. During inference the network processes whole images, as opposed to patch-sized inputs. However, we assume that input images are at least of size 64×64.

Instead of a single value describing the keypoint quality of a single patch, the network is converted to be fully convolutional as to output a feature map. There, each value corresponds to the keypoint score of a specific image region. In particular, we compute the keypoint feature map after each recursive convolution. As inputs progress deeper into the network, the receptive field of individual neurons increases so that larger patches in the input image are considered. As a result, the output feature maps resemble a keypoint *scale-space*. We illustrate this in Figure 5. This allows us to select the best scale for each patch by finding the scale with the highest keypoint response score. Finally, the best keypoints are extracted with non-maximum suppression.

3.3 Description Network

The goal of the descriptor network is to learn a nonlinear feature embedding $f(p)$, from patches p into a feature space $\mathbb{R}^d$, such that for a pair of patches p_1 and p_2, the Euclidean

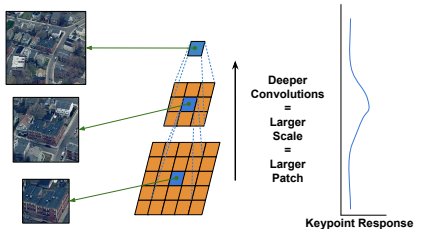

Figure 5: Convolutions later in the detection network correspond to larger patch sizes. The keypoint feature map with the highest response indicates the best keypoint scale.

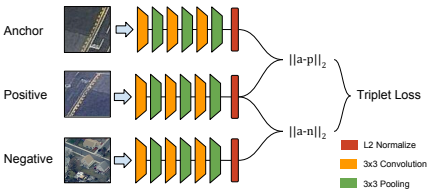

Figure 6: Training architecture of the keypoint description triplet network. Three patches are passed through channels which share weights to rank their euclidean distances in the feature space.

distance between $f(p_1)$ and $f(p_2)$ is small if the patches match and is large if they do not match. The training follows an approach similar to the triplet network proposed in [22]. In particular, the nonlinear embedding should ensure that a patch p_1 (anchor) is closer to all patches depicting the same keypoint p_2 (positive) than it is to any other patch p_3 (negative). Given the feature embedding and a sets of keypoints with respective patches, the best matching keypoint can be found by retrieving nearest neighbors in the embedding space.

3.3.1 Training Procedure

Figure 6 illustrates the training architecture. The input to the network are batches of patch triplets $\{p_1, p_2, p_3\}$. First, each patch is fed through a convolutional neural network to compute its embedding feature vector, where three networks share the same weights. The feature vectors are then normalized to lie on the d-dimensional unit hypersphere. Afterwards, the pairwise Euclidean distances between the feature vectors of anchor and the two other patches are computed. The network is then supervised with a triplet ranking loss shown in Equation 5 to project the matching patches closer in the feature space and the non-matching patches.

The patch triplets are sampled online. The anchor and the positive match are drawn from the match set M. The negative patches are sampled at random. In order to ensure good convergence, it is important to sample triplets that induce loss, i.e., they violate the triplet constraint. To get triplets that violate the constraints, we perform online hard negative mining. In particular, for each matching pair (anchor and positive) in the training batch, we choose the negative patch in the batch that violates the triplet constraint the most. All matching pairs within a batch can choose from the same set of negative patches.

3.3.2 Training Objective

We define a loss-function $\mathcal{L}_\mathrm{T}$ for training the keypoint description network, as follows. Given triplets $T = \{t_j : (p_1^j, p_2^j, p_3^j)\}$ and a scalar margin h, the loss function is given by:

$$\mathcal{L}_\mathrm{T} = \frac{1}{N} \sum_j \left[\max\left(0,\ D(p_1^j, p_2^j) - D(p_1^j, p_3^j) + h\right) \right] \tag{5}$$

where h is chosen as 0.2 and D is a Euclidean distance function defined on the embedding feature vectors, which are computed from the image patches.

$$D(p_a, p_b) = \| f(p_a) - f(p_b) \|_2 \tag{6}$$

Model Component	Structure
Feature Detection	C3/128/2-BN-P3/2-C3/128/1-BN-P3/2-C3/d/1-Repeat{C3/d/1-BN}
Keypoint Scoring	C3/64/1-BN-C1/1/1
Patch Matching	C3/128/2-BN-P3/2-C3/256/1-BN-P3/2-C3/256/1-BN-P3/2-L2Normalize

Table 1: Network structure parameters. *Convolution* is denoted with **C**$k/f/s$, where k is the kernel size, f is the number of filters or outputs, and s is the stride. Similarly, *Max Pooling* is denoted with **P**k/s, batch normalization is **BN**, and fully-connected layers are **FC**. Parameter d denotes the number of filters, in the convolutional layers that vary among experiments.

3.4 Full Model

After both networks are trained, keypoint detection and matching can be performed. The process is similar to the traditional keypoint extraction and description pipeline.

First, a whole image is fed through the fully convolutional detection network. A sample output is shown in Figure 8. From the output feature map, a set of keypoints are extracted by filtering with non-maximum suppression. Then, for each keypoint, we crop a patch according to the detected scale and rescale it to 64×64, the canonical patch size of the description network. Subsequently, the keypoint descriptors are computed with the triplet network. Finally, given the keypoint descriptions for two images, corresponding keypoints are found using nearest neighbor search.

4 Experiments

4.1 Experimental Setup and Model Parameters

We verify the effectiveness of our model by testing on a separate held-out testset, which was formed by removing cameras (images) from the structure-from-motion 3D model prior to training. The held-out set is comprised of about 800K patches of varying scale, with matching information.

The specific parameters of the networks used in the experiments are described in Table 1. All convolutions are without padding. For optimization, we used Stochastic Gradient Descent with a learning rate of 0.01, momentum of 0.9, and a weight decay of 0.005.

4.2 Keypoint Detection

To test the keypoint detector, we run different versions of the keypoint-detection network, and compute precision/recall for each. The networks differ in the number of feature dimensions (referred to with d in Table 1), and hard-negative mining procedure.

Our first network "KP-1" has $d = 256$ and uses hard-negative mining from the first iteration. The second network "KP-2" has $d = 256$ and uses hard-negative mining starting from mid-training with the whole batch comprised of hard-negatives. The last network "KP-3" has $d = 128$ and follows the same hard-negative mining procedure as "KP-1". The precision/recall curves are shown in Figure 7.

The results indicate that using hard-negative mining from early training allows the model to find a better solution as opposed to start using hard-negatives only during mid-training. One explanation could be that the model may have already arrived at a good local-minimum

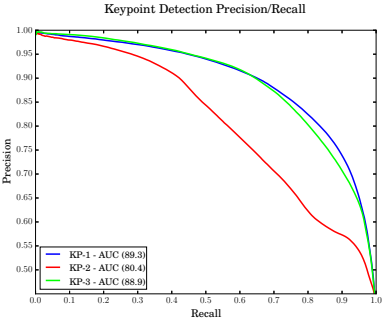

Figure 7: Precision/Recall curves for different variants of our keypoint detector.

Figure 8: Sample keypoint detections on a full-sized image.

for identifying keypoints. The results also show added benefits from additional model parameters. Overall, the model performance well with an area-under-the precision-recall curve of 89.3 on keypoints of varying scales.

4.3 Patch Matching

To evaluate our triplet-based patch matching network, we compare against DeepCompare [30] and MatchNet [12], which both leverage a siamese-based architecture. The evaluation is based on a retrieval framework. For a randomly sampled pair of matching patches in the test set, we use one of the patches as *probe* and the other as *target*. The target is mixed with a set of non matching patches, for a total set of 100 patches. Then, given the probe, the task is to find the matching patch within that set.

In our evaluation we report the retrieval at rank 1 (top-1%) and within ranks [1-5] (top-5%). The test includes two variants of our network. The first variant "Triplet-1" is based on a small number of convolution and pooling layers such as shown in Figure 6. The second variant "Triplet-2" is based on the VGG-16 network [8]. We run each network and rank the matches according to distance or similarity (MatchNet and DeepCompare, both follow a similarity metric). Our testset contains 5K matching pairs, randomly sampled from the test

Method	Top-1%	Top-5%
Triplet-1	**73.8**	**93.4**
Triplet-2	**76.6**	**95.5**
MatchNet [12] - Liberty	57.3	82.3
MatchNet - Yosemite	44.0	73.1
MatchNet - Notredame	52.5	78.6
DeepCompare [30]- 2ch - Liberty	71.1	88.7
DeepCompare - 2ch - Yosemite	70.9	88.6
DeepCompare - 2ch - Notredame	71.9	88.0
DeepCompare - siam - Liberty	67.6	90.0
DeepCompare - siam - Yosemite	70	88.6
DeepCompare - siam - Notredame	70.7	91.0

Table 2: Retrieval at rank 1 (top-1%) and within ranks [1-5] (top-5%) on our test-set.

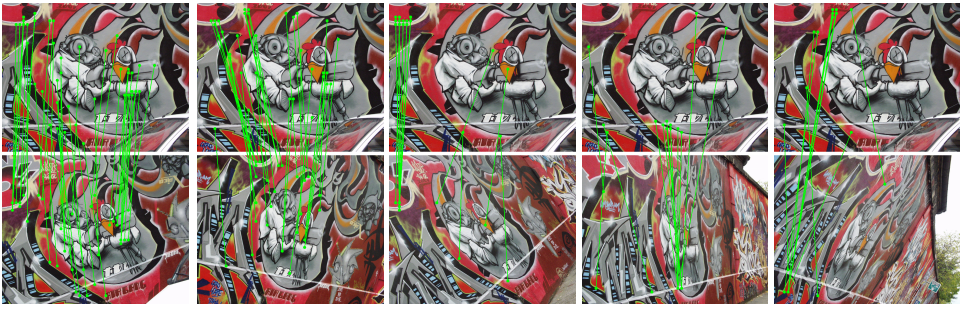

Figure 9: Qualitative evaluation of feature transferability: Keypoint detection and matching results from network trained on aerial imagery and tested on "Wall' image from the Oxford dataset [19]. For the first two images the network successfully retrieves the correct homography. The result on the third is partly correct. The last two image demonstrate failure cases.

set. We perform two runs independently, and report the average in Table 2. For DeepCompare [30], we report results only for the two best variants (out of five) for brevity. The results show that the proposed method outperforms previous results. We believe this is due to the structure of the embedding learned by the triplet loss function which is more suitable for ranking purposes.

4.4 Extending to Other Datasets

To evaluate the generalization of the learned keypoint detector and descriptor, we present qualitative results for our learned models on a dataset with different image statistics. In particular, we applied ours models on the "Wall" sequence from the Oxford dataset [19].

In Figure 9, we show the five image sequence comparing the first image with the rest of the images in the sequence. Our network shows good results in the first two images, retrieving the correct homography. The results on the third image is partly correct, and the last two image demonstrate failure cases. The type of images differs largely between our training dataset and the test images. However, the approach shows promising results indicating good capabilities of extending to other datasets.

5 Conclusion and Future Work

Feature extraction and description is a central problem in computer vision. We presented a novel deep learning architecture capable of multiscale keypoint detection and description. Our approach serves as a step to bring classical approaches closer together with the recent progress in deep learning. We plan to further investigate the models performance on other benchmarks and explore other avenues for multiscale detection and description.

Acknowledgments

We would like to thank Michael Wilber and Tsung-Yi Lin for their valuable input. This work was supported by the KACST Graduate Studies Scholarship.

References

[1] Hani Altwaijry, Eduard Trulls, James Hays, Pascal Fua, and Serge Belongie. Learning to Match Aerial Images with Deep Attentive Architectures. In *CVPR*, 2016.

[2] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[3] Boris Babenko, Piotr Dollár, and Serge Belongie. Task specific local region matching. In *ICCV*, 2007.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *ECCV 2006*.

[5] J. Bromley, I. Guyon, Y. Lecun, E. Sckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994.

[6] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *PAMI*, 2011.

[7] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. BRIEF: Computing a local binary descriptor very fast. *PAMI*, 2012.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR 2005*.

[10] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 1981.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[12] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015.

[13] W. Hartmann, M. Havlena, and K. Schindler. Predicting matchability. In *CVPR*, 2014.

[14] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*. 2012.

[16] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[17] Jonathan L Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *NIPS*, 2014.

[18] David G. Lowe. Object recognition from local scale-invariant features. *ICCV 1999*.

[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] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[23] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015.

[24] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *PAMI*, 2014.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[26] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[27] Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. TILDE: A temporally invariant learned DEtector. In *CVPR*, 2015.

[28] Changchang Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013.

[29] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *CVPR*, 2011.

[30] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *CVPR*, 2015.