DEEP LEARNING BASED FEATURE MATCHING AND ITS APPLICATION IN IMAGE ORIENTATION

Lin Chen*, Franz Rottensteiner, Christian Heipke

Institute of Photogrammetry and GeoInformation, Leibniz Universität Hannover, Germany
chen, rottensteiner, heipke@ipi.uni-hannover.de

Commission II, WG II/1

KEY WORDS: Image Matching, Affine Shape Estimation, Descriptor Learning, Feature Orientation, Image Orientation

ABSTRACT:

Matching images containing large viewpoint and viewing direction changes, resulting in large perspective differences, still is a very challenging problem. Affine shape estimation, orientation assignment and feature description algorithms based on detected hand crafted features have shown to be error prone. In this paper, affine shape estimation, orientation assignment and description of local features is achieved through deep learning. Those three modules are trained based on loss functions optimizing the matching performance of input patch pairs. The trained descriptors are first evaluated on the Brown dataset (Brown et al., 2011), a standard descriptor performance benchmark. The whole pipeline is then tested on images of small blocks acquired with an aerial penta camera, to compute image orientation. The results show that learned features perform significantly better than alternatives based on hand crafted features.

1. INTRODUCTION

Feature based image matching aims at finding correspondences among images and is a fundamental research issue in photogrammetry and computer vision. The related pipeline is composed of four steps, namely feature detection, feature orientation, feature description and high dimensional descriptor matching. Distinctive features are obtained during feature detection and localization across scale. After the assignment of a principal direction to each detection, all features are corrected accordingly to remove the rotation difference. Afterwards, a support window surrounding each detected feature with a size proportional to the detected scale is chosen and is used to extract a high dimensional vector, i.e. the feature descriptor, to represent the detected features. Optionally, the affine shape of detected features can also be estimated to compensate potential affine distortions, in particular when facing large changes in viewpoint and viewing direction.

The key challenge of feature based image matching frameworks is to ensure invariance against complex geometric and radiometric changes between images. For instance, when two images are taken from distinctively different viewpoints, the local appearance of image patches surrounding detected features can differ significantly due to the geometric setup. Radiometric changes can result from varying illumination, differences in imaging bands and non-Lambertian reflection properties. As indicated in (AanÃęs et al., 2012) , the invariance of hand crafted detectors and descriptors decreases sharply for images containing 3D scenes when viewpoint and viewing direction changes increase.

However, feature based image matching algorithms can be designed to be invariant against certain geometric and radiometric changes. For instance, the well known SIFT operator (Lowe, 2004) is rotation and scale invariant to a certain degree. This invariance can be extended to a reasonable level of affine transformation between images, see e.g., the Hessian-Affine detector (Mikolajczyk et al., 2005). The feature support window is then mapped to a high dimensional vector to represent the underlying feature. The feature support window is normally corrected according to the estimated orientation and optional affine shape parameters.

Many successful detectors and descriptors based on handcrafted features were developed in the past, but achieved only limited success for large view point changes. Therefore, acquiring local feature descriptors using deep neural models has been attracting more attention recently, because the latter have been shown to have advantages with respect to discriminability, e.g. (Lenc, Vedaldi, 2016, Tian et al., 2017, Mishchuk et al., 2017). In this paper, a new feature based image matching framework based on deep neural networks, including affine shape estimation, feature orientation and description is presented and results for the image orientation of small blocks of oblique aerial images are reported and analysed.

2. RELATED WORK

Feature based image matching has been studied for many decades. The central idea of detecting features is to find points or blobs that are distinctively different from neighbouring pixels. Following this idea, the determinant or trace of the Hessian matrix computed from second order Gaussian smoothed image derivatives is often used to measure how distinctive the underlying feature is. This analysis can be extended to scale space by incorporating images convolved by derivatives of Gaussian kernels of different width. In this way, local extrema in both image x, y and in the scale dimension are detected as features. The trace of the Hessian is approximated by the Difference of Gaussians in SIFT (Lowe, 2004) and the determinant of the Hessian is used in the scale invariant Hessian detector which is a part of the SURF algorithm (Bay et al., 2008).

After features are detected, the rotation of a feature can be estimated by calculating a principal direction using the gradient

* Corresponding author

orientations calculated in a local window surrounding the detected feature. Gradient orientation bins are used to find the principal direction in SIFT (Lowe, 2004), while the Haar wavelet responses in horizontal and vertical direction are used to assign a principal direction to a detected feature in SURF (Bay et al., 2008). In (Moo Yi et al., 2016), the orientation is estimated by a deep convolutional neural network (CNN), showing significantly better performance than the aforementioned methods based on hand crafted features.

Detecting local features in scale space and assigning them an orientation is basically equivalent to normalizing translation, rotation and scaling of local features before description. However, this transformation is not sufficient to model the geometric transformations between local image patches in case of large changes in viewpoint and viewing direction between images. Perspective changes, which for small windows can be compensated by an affine transformation, should also be estimated and taken into account before feature description. Compared to feature detection, fewer works have been published in this direction.

In (Mikolajczyk et al., 2005), the second moment matrix M is used to measure the level of isotropy of a feature. The patch surrounding features is normalized by multiplying the patch with $M^{-1/2}$ (restricting the largest eigenvalue of $M^{-1/2}$ to 1) and then, the second moment matrix for the normalized patch is iteratively calculated and normalized with $M^{-1/2}$, until the two eigenvalues of M for the normalized patch are close enough to each other. After each iteration, the spatial localization of the maximum value of the feature response function is re-detected. As a result, the affine transformation between two image patches is removed and only a rotation remains. However, this algorithm is not stable when the tilt between patches is large. Instead, affine shape estimation based on a deep neural network is proposed in (Mishkin et al., 2018), where it is estimated by minimizing the distance between the matched descriptors. In addition, ASIFT (Morel, Yu, 2009) simulates the input image with different versions of affine transformations and then the DoG features and SIFT descriptors detected in each transformed image are combined for descriptor matching; this means ASIFT is computationally expensive.

Once orientation and affine shape are assigned to a detected feature, a small patch surrounding the feature is corrected to obtain the feature support window, which is then fed into a description module to obtain the descriptor. Although the pixel grey values in the feature support window can be used as a simple form of descriptor, this is often too sensitive to remaining local deformations in the patch and not discriminative enough. As discussed in (Brown et al., 2011), a feature descriptor is a composition of transformation, aggregation, normalization and optional dimension reduction. The transformation step magnifies some signal, e.g., gradients in SIFT (Lowe, 2004) and Haar wavelet response in SURF (Bay et al., 2008). Afterwards, the transformation response is aggregated by means of computing the mean value over a grid or the histogram of local response. After that, features are normalized to improve their invariance against radiometric transformations. Also, dimension reduction algorithms, e.g. Principal Component Analysis, can be used to further decrease the dimension of the output descriptor. Based on these steps, a descriptor learning method is proposed in (Winder, Brown, 2007, Brown et al., 2011) to optimize the configuration of different steps in building feature descriptors, resulting in a notable performance improvement compared to hand-crafted descriptors.

Descriptor learning methods use image patch pairs as input, derive descriptions using some initial model, and then obtain a similarity score for the descriptors of the patch pairs. The model is then optimized using a loss function to increase the similarity of matched feature pairs and decrease the similarity of unmatched pairs. Following this approach, boosting is used to learn weak features that can best discriminate matched and unmatched pairs in (Trzcinski et al., 2012, Trzcinski et al., 2015, Chen et al., 2014). More recently, many researchers extract the descriptor by CNN, e.g., (Zagoruyko, Komodakis, 2015, Han et al., 2015, Simo-Serra et al., 2015, Chen et al., 2016, Kumar et al., 2016, Balntas et al., 2016, Tian et al., 2017, Mishchuk et al., 2017). Also, additional constraints for regularization (Zhang et al., 2017, Luo et al., 2018) were added for descriptor learning. Not surprisingly, the deliberate choice to enlarge the amount of training data has also increased the matching performance of learned descriptors, as suggested in (Mitra et al., 2018).

In this paper we suggest a feature matching pipeline based on CNNs in order to derive image orientation parameters for blocks of aerial penta cameras. These images typically have viewing directions with differences amounting to 45 degrees (nadir vs. oblique view) and 90 degrees (one oblique view vs. the other). The work most related to ours is (Mishkin et al., 2018). In that work the affine shape of local features is estimated by deep learning through optimizing the similarity of pairs of input patches simulated by affine transformations. We use the same approach and the same network architecture, but train the different modules from scratch. (Mishkin et al., 2018) differs from our work in mainly two further aspects: 1) the affine shape estimation part is trained based on a different form of loss; and 2) our algorithm is applied and tested on real image orientation tasks, instead of only on image matching benchmarks. The second aspect forms the main contribution of this paper.

3. THE FEATURE MATCHING PIPELINE

In our method the three steps of affine shape estimation, orientation assignment and description of local image patches are all learned based on a CNN architecture with detected feature pairs as input, which serve as training data (for details on the detection see section 3.4). Subsequently, features of individual images are detected using classical methods, followed by applying the trained CNN modules to obtain descriptors. Finally matching is carried out. An overview of the training phase is illustrated in Fig 1. The training data is a series of image patches with known matching relationship ("matched" or "unmatched"), thus matched and unmatched pairs can be sampled from the training data. These training patches only have small orientation and affine shape differences. First, the descriptor part is trained based on the sampled dataset. Then, the affine shape and orientation modules are trained separately (and independently of each other) based on the descriptor learned in the previous step and on sampled pairs of patch data which are now rotated and distorted based on simulation (note that in contrast to training, during inference, the sequence of first applying the affine correction and then carrying out orientation assignment does matter, see section 3.4 and figure 6 for details). The affine shape and orientation are not solved simultaneously, because preliminary experiments of solving the two parts in one step did not converge.

3.1 Descriptor Module

To train descriptors, matched and unmatched patch pairs are used. The patches are support windows of features. After ap-

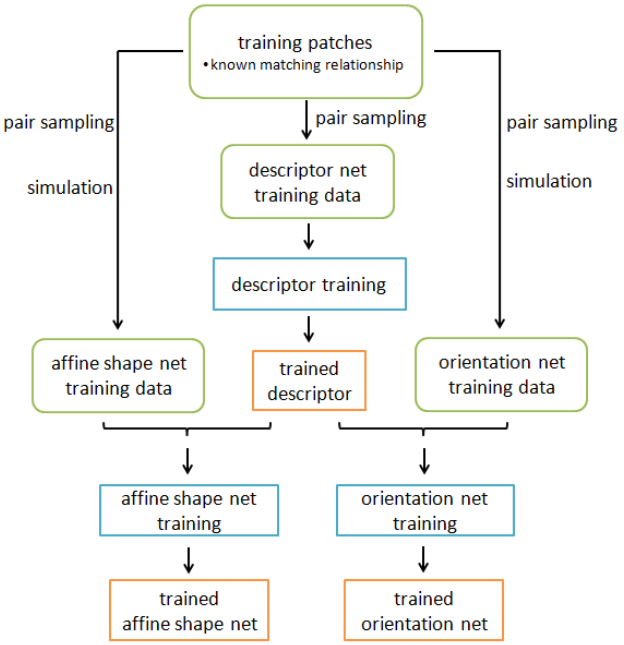

Figure 1. Overview of the method.

plying a CNN to those patches, related descriptors are obtained. As the pair of patches is either matched or unmatched, the corresponding descriptor distances are used to build the loss function. In this section, the framework for descriptor learning is discussed first, followed by the generation of training pairs and the design of the loss function. Additionally, the employed hardest mining strategy (Mishchuk et al., 2017) and data augmentation are discussed.

3.1.1 Descriptor Training Architecture: The descriptor is trained based on a Siamese CNN, which is illustrated in figure 2. The two branches of the CNN share the same weights and each branch of the descriptor network is used to extract descriptors from an input image patch.

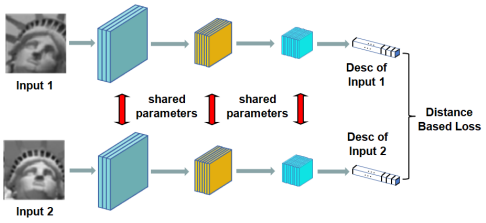

Figure 2. Siamese CNN used for descriptor learning.

Details of the descriptor network used to generate descriptors are provided in table 1. This network is identical to the one used in (Mishchuk et al., 2017) and (Mishkin et al., 2018), and was originally proposed by (Tian et al., 2017). Through a series of convolution layers, a 32×32 pixel single channel image patch is transformed and compressed into a 128 dimensional descriptor, which is then scaled to unit length.

3.1.2 Generation of training pairs: Following (Mishkin et al., 2018), the training data is composed of image patches. For each patch it is known which other patches are correct matches (this is realised via a 3D point index for each patch, thus matched patches are characterized by the same 3D point index). First for each mini-batch, N 3D points are sampled without replacement

Layer	Filter	#In-Out	Stride	Activation	BN
Desc-1	3x3	1-32	1	ReLU	Yes
Desc-2	3x3	32-32	1	ReLU	Yes
Desc-3	3x3	32-64	2	ReLU	Yes
Desc-4	3x3	64-64	1	ReLU	Yes
Desc-5	3x3	64-128	2	ReLU	Yes
Desc-6	3x3	128-128	1	ReLU	Yes
Desc-7		Dropout with rate=0.1			
Desc-8	8x8	128-128	1	ReLU	Yes

Table 1. Descriptor Network architecture. #In-Out: number of input and output channels. ReLU: Rectified Linear Unit, which works as $g(z) = max(0, z)$ for an input z. BN: batch normalization.

from the training data and then, two different patches associated with the same 3D point index are randomly selected to form a pair of positive patches for each selected 3D point.

During training we also need counter examples, i.e. non-matching pairs: for a patch p_1, patches associated with a different 3D point index belong to the unmatched patches (see Figure 3). The use of the 3D index thus ensures that these pairs, when sampled randomly from the training data, do not by chance contain correct matches. Obviously, the number of possible unmatched pairs is much higher than that of the matched pairs. For properly training our network, we need an equal number of matched and unmatched pairs. We thus have to select the unmatched pairs to be used in training from the larger set. For this selection step we employ the hardest mining strategy (Mishchuk et al., 2017).

Hardest mining of unmatched pairs: According to this strategy, the unmatched pairs are required to be the most difficult ones for each pair of patches. In this way the network best learns how to differentiate between matched and unmatched samples. The corresponding Euclidian distance $A_i^{hardest}$ is defined as:

$$A_i^{hardest} = min(min_{j \neq i} A(d_1^i, d_1^j), min_{j \neq i} A(d_1^i, d_2^j)) \quad (1)$$

where d_1^i = the descriptor for the ith patch in the patch set 1, i.e. the set passed through branch 1
d_2^j = the descriptor for the jth patch in the patch set 2, i.e. the set passed through branch 2
$A(:,:)$ = Euclidean distance of two descriptors

$minA(d_1^i, d_1^j)$ and $minA(d_1^i, d_2^j)$ compute the hardest negative samples by finding the unmatched pair with the smallest distance. The selection of the hardest unmatched pair is illustrated in figure 3. Through seeking the hardest samples anew in each training epoch, the network "sees" many more unmatched training pairs than matched ones, which corresponds to the fact that matching is a problem where a much larger number of negative pairs than positive ones is typically compared for real matching applications. After mining, a triplet containing a pair of matched patches and a "most difficult" negative patch is obtained for each training patch passed to the first branch of the network.

3.1.3 Loss function: The triplet margin based loss function (Hoffer, Ailon, 2015) is used as the loss function for descriptor training. For N sampled triplets, the loss is defined as:

$$L = \sum_{i=1}^{N} max(0, A_i(d_1, d_2) + \beta - A_i^{hardest}) \quad (2)$$

where $A_i(d_1, d_2)$ is the Euclidian distance between the ith matched pair and $A_i^{hardest}$ is the hardest distance computed

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-25-2020 | © Authors 2020. CC BY 4.0 License.

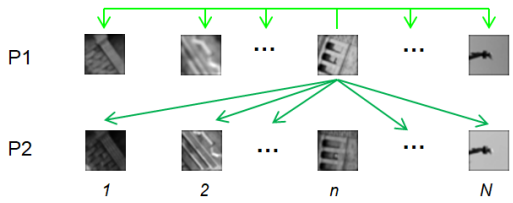

Figure 3. Hardest mining. The patch p_1^n is compared with all the patches with a different 3D index j from the patch sets $P1$ and $P2$. The pair with the smallest distance is picked for calculating the loss for unmatched pairs. Arrows: comparison between patches.

for the i^{th} triplet as described above. β is a pre-defined margin between the distance of matched and unmatched pairs. Considering all the descriptors are normalized, the maximum distance between any two descriptors is 2. In this paper, β is set to be 1, as suggested in (Mishchuk et al., 2017).

3.1.4 Data augmentation: In order to increase the number of matched pairs during training, the available ones are augmented by flipping or rotating them by a value randomly chosen from the set $[90°, 180°, 270°]$.

3.2 Affine Shape Estimation Module

In this section, the learning architecture for affine shape estimation, affine shape parametrization and the corresponding loss function are discussed.

3.2.1 Training Architecture: Similar to the descriptor network, the affine estimation network has two branches and shared weights to handle patch pairs, which are sampled according to the method described in section 3.1.2. However, here the input patch pairs are first distorted by simulation using an affine transformation and are then fed into the affine estimation network. Note that both patches of a pair are distorted, and the shape correction is then computed for both of them. Through the affine estimation network the underlying affine transformation parameters are estimated and the patches are then re-sampled using the inverse transformation. Then, the resampled patches are fed into the descriptor network to obtain descriptors and to calculate the descriptor distance based loss. The whole architecture is illustrated in figure 4.

Affine Shape Parametrization: Similar to (Mishkin et al., 2018) and (Perd'och et al., 2009), the affine transformation applied to each patch individually is decomposed into the following form

$$M = \lambda \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} a'_{11}+1 & 0 \\ a'_{21} & a'_{22}+1 \end{bmatrix} \quad (3)$$

where λ = detected feature scale, kept constant during estimation of affine shape parameters
$a'_{11}, a'_{21}, a'_{22}$ = residual form of affine shape parameters, computed during affine shape estimation
ψ = feature rotation angle, also kept constant during estimation of affine shape parameters

The rotation matrix with angle ψ will be discussed in section 3.3; the other matrix contains the affine shape parameters. Setting $a_{12} = 0$ enables the affine shape estimation to preserve the direction orthogonal to x-axis for a image patch, because the affine shape matrix always has one eigenvector equal to $(0, 1)^T$.

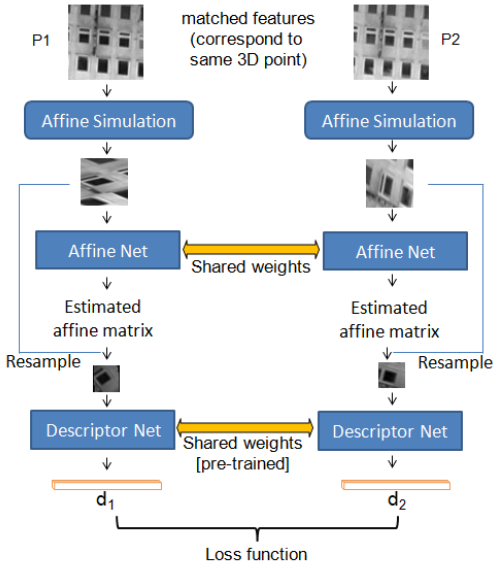

Figure 4. Affine shape estimation architecture. After affine distortion of the patches, the affine shape matrix elements are estimated and the patches are resampled. The corrected patches are then introduced to the pre-trained descriptor network to obtain descriptors.

The affine shape estimation network (we again use the same network as (Mishkin et al., 2018)) is used to estimate the affine matrix elements for each input patch. This network has a similar structure as the descriptor network, delivering residual affine shape parameters $a'_{11}, a'_{21}, a'_{22}$. To fix the overall scale of features we subsequently divide them by $det(A) = (a'_{11}+1)*(a'_{22}+1)$. During training, all affine shape parameters are randomly sampled according to a uniform distribution, and for a matching pair ψ is set to the same angle for both patches.

3.2.2 Loss function: The k hardest loss is used for the estimation of the affine shape matrix elements.

$$L = \sum_{i=1}^{N} max(0, A_i(d_1, d_2) + \beta - 1/k \sum_{j=1}^{k} A_i^{hardestk_j}) \quad (4)$$

The k hardest unmatched pairs are used for each matched pair (here $k > 1$; for descriptor learning $k = 1$). β is the margin defined in equation (2). This larger value of k reduces the risk that the hardest unmatched sample lies between the matched features in descriptor space, while still relying on difficult samples. Based on experimental evaluation we set k to 3 in this paper. This mining procedure is different from (Mishkin et al., 2018), who set $k = 1$ in the loss function and eliminate the effect of the hardest unmatched samples during back-propagation by setting the respective gradients to zero.

3.3 Orientation Assignment Module

As feature pairs can not only exhibit affine distortion but also be rotated in an arbitrary way, there still exists the (unknown) rotation angle ψ for each patch. In this section we describe how we estimate ψ.

3.3.1 Training Architecture: The feature orientation network is again similar to the descriptor network and has two

branches with shared weights to handle patch pairs, which are sampled according to the method described in section 3.1.2. Here, both input patches are first rotated by simulation and then fed into the network. Both patches of a pair are rotated by an angle sampled independently of each other , and the orientation angle is then computed for both of them. The architecture for orientation learning is shown in figure 5.

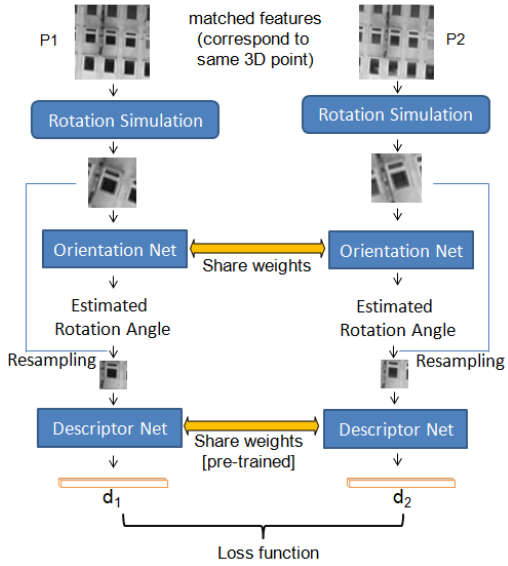

Figure 5. Orientation estimation network architecture. After rotation simulation, the rotation angle of simulated patches are estimated and then the patches are resampled using the estimated rotation angle. The resampled patches are subsequently used as input to the pre-trained descriptor network to obtain descriptors d_1, d_2 and the loss.

3.3.2 Data Augmentation: A uniformly distributed random rotation in a range of $[0, 2\pi)$ is applied to each patch of the input patch pair. Also, a random translation in a range of $[-2, 2]$ pixels and random scaling in a range of $[0.9, 1.1]$ is applied to simulate the localization and scale determination noise in the feature detection stage, as suggested in (Mishkin et al., 2018).

3.3.3 Training Loss: In this case, the loss is only based on the distance of matched feature pairs. It is defined as

$$L = \sum_{i=1}^{N} min(0, A_i(d_1, d_2)) \tag{5}$$

where $A_i(d_1, d_2)$ is distance of the i^{th} pair. When patches are rotated, the contents of the feature support window remain the same, and assuming there exists no rotated repetitive texture, incorporating hardest negative samples is not necessary.

3.4 Feature Description using Trained Models

Once the three aforementioned modules are learned, they are integrated into a feature and descriptor extraction pipeline which outputs detected features and their descriptors for an input image. The whole process is shown in Fig. 6. First, the Hessian matrix determinants of each pixel in the input image are calculated for each sample scale of the input image in scale-space. Then, the local extrema of the Hessian determinant are detected in scale-space, followed by the refinement of image coordinates and characteristic scale. Subsequently, a patch is re-sampled to 32×32 pixels around the detected feature position with a range proportional to the characteristic scale. This patch is regarded as input for the affine shape network to predict its estimated affine shape, which is used in the next step to compensate affine distortion of local patches. In the following step, the orientation network is applied to the patch corrected for affine distortion in order to estimate rotation. The patch is then further corrected by the estimated rotation angle, and the related patch forms the feature support window for the local feature. The trained descriptor network is then applied to this support window and a descriptor for the underlying local feature is derived. Note that whenever resampling is necessary, all related parameters are combined first and only a single resampling step is carried out.

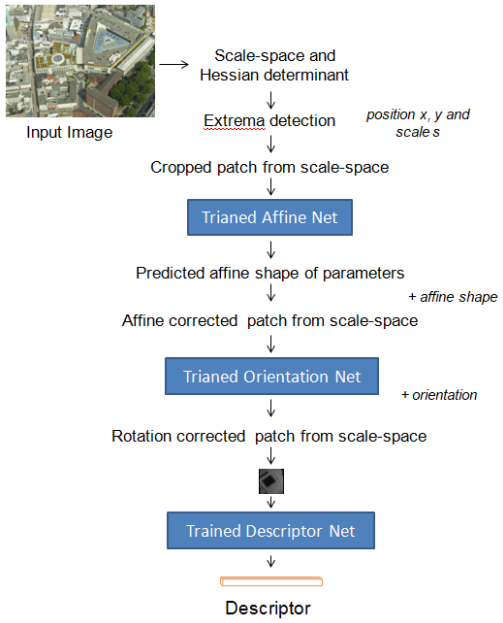

Figure 6. Feature detection and descriptor generation.

4. EXPERIMENTS AND RESULTS

First, the learned descriptor is evaluated using the Brown dataset (Brown et al., 2011) to reflect its performance for classification of patch pairs. Second, the whole pipeline is used to extract features and descriptors for small image blocks taken from an aerial penta-camera system including nadir and oblique images with significant changes in viewing direction. The quality of image orientation parameters and the computed 3D points after bundle adjustment using the matched features as input are used to assess the results.

4.1 Experimental Datasets

Brown Dataset: The Brown dataset (Brown et al., 2011) is used to train the descriptor network and to test its performance. This dataset was generated from a multi-view image collection containing a large number of images of community photo collections (Goesele et al., 2007). Through structure from motion and dense multi-view stereo matching (Snavely et al., 2008), matches were retrieved, which are considered as ground truth in the following[1]. The dataset is composed of three different

[1] For a feature f_L in image I_L, a small grid surrounding f_L in I_L is extracted and transferred to image I_R through the depth map estimated between stereo image pair I_L, I_R. The transferred grid is then used

subsets: *Notre Dame*, *Liberty* and *Yosemite*. In each subset, a test set containing equal numbers of matched and unmatched pairs is also provided.

Dataset for Image Orientation Five different image blocks, each containing a mix of nadir and oblique images, are used in this experiment. Blocks 1 to 3 were acquired using a penta system with Cannon EOS-1DS Mark II cameras, while blocks 4 and 5 are part of the ISPRS/EuroSDR benchmark for multi-platform photogrammetry (Nex et al., 2015)[2], in which the IGI penta camera system was used. The image dimension is 4994 x 3328 pixels for blocks 1 to 3 and 8176 x 6132 pixels for blocks 4 and 5. Details of the blocks are given in table 2.

Block	#images	#N	#F	#R	#B	#L	Scene contents
1	18	5	3	3	4	3	buildings, roads, forest, lake
2	17	5	3	2	3	4	buildings, forest
3	19	5	3	4	3	4	buildings, lakes, road
4	17	3	3	3	4	4	building, trees
5	20	4	4	4	4	4	buildings, trees

Table 2. Details of the image blocks used for determining image orientation. #N #F #R #B #L: the number of images from nadir, front, right, back and left camera, respectively

4.2 Training of Network

All three modules were trained using a mini-batch size of $N = 1024$ with 20 epochs, where one epoch represents the number of iterations in which all training samples are used once. For the descriptor, the pairs were regenerated after each training epoch. The employed optimizer was standard gradient descent with momentum, the learning rate α was set to 0.001 and decayed linearly with a step of $\alpha/\#steps$, where $\#steps$ is the number iteration steps. The training of the descriptor for the Brown dataset used 10 million pairs. The networks of descriptor, affine shape and orientation estimation for image orientation were all trained based on the complete Brown dataset. For descriptor training, 30 million pairs were used, while for the other two modules, 10 million pairs were employed. Note that there is a domain gap between the training data and the image orientation task data, because the Brown dataset is composed of close-range terrestrial images and the orientation task uses aerial images.

4.3 Evaluation Protocols

4.3.1 Brown Dataset: The descriptor network is trained using one subset and then tested on the other two (in all permutations), therefore six different combinations of training-test subsets are obtained. The distances of descriptors for pre-defined patch pairs, containing an equal amount of matched and unmatched pairs, are computed and a threshold is applied to obtain the False Positive Rate (FPR) - True Positive Rate (TPR) curve. FPR (in %) at 95% TPR is reported as our evaluation criterion.

to estimated the scale and pixel localization for the transferred feature point in I_R. If the difference of estimated scale and pixel localization for the transferred features point is close to the scale and localization of a feature point f_R in I_R, then f_R and f_L are judged as a ground truth match.

[2] http://www2.isprs.org/commissions/comm1/icwg15b/benchmark_main.html

4.3.2 Image Orientation: After feature detection and matching a bundle adjustment is carried out for the five blocks to obtain image orientation parameters. A number of quality measures are recorded as evaluation criteria (see below for details). In this experiment, three different combinations of affine shape estimation, orientation and feature description algorithms are compared in order to evaluate the contribution of the three different modules in comparison to published work. Note that for all three combinations the same set of detected features is provided as input. The three variants formed by different combinations are:

- *hbss:* the variant uses Baumberg iteration (Baumberg, 2000) for affine shape, gradient statistical orientation assignment as in SIFT (Lowe, 2004) and the SIFT descriptor (Lowe, 2004).
- *hbsl:* same affine shape and orientation solution as for *hbss*, and the descriptor learnt as explained in this paper.
- *hlll:* the method explained in this paper, i.e. all affine shape, orientation and descriptor are learned.

Determination of image orientation For this task the following steps are conducted.

- Detection of features for all images, see section 3.4. In this step, a fixed number of Hessian features with potentially high repeatability against viewpoint changes is detected in scale space. We use 5000 features per image in blocks 1 to 3 and 12000 features per image for blocks 4 and 5[3].
- Estimation of the affine shape and orientation of local features and computation the descriptors using the trained network, as explained in section 3.4.
- Nearest neighbour threshold matching (Lowe, 2004) to obtain initial matches for each pair of images in a block. The maximum ratio allowed between nearest and second nearest matching features is set to 0.67.
- Structure from motion (SfM) software COLMAP[4] (Schönberger, Frahm, 2016) to obtain initial orientation parameters. Note that we ignore matching points that only appear on two images in this step.
- Transformation of the initial orientation results obtained from the different pipelines into a common coordinate system for further comparison[5]. To achieve this goal, one image is selected as origin of the common coordinate system, setting both its projection centre and rotation angles to zero; a second image is selected to define the scale between the two projection centres, then the length of the baseline is set equal to 1. The selected first and second image are identical for each image block processed by the different variations, i.e. *hbss, hbsl, hlll*.
- Robust bundle adjustment. The bundle adjustment delivers the final orientation parameters. Robust estimation is used and observations with residuals larger than 3.2 times the standard deviation are considered as outliers and are excluded from further iterations. According to the prior camera information different sets of additional parameters are used in the bundle adjustment: For blocks 1 to 3, radial and tangential distortion parameters are estimated in both, the SfM pipeline and the bundle adjustment. For blocks

[3] Features for blocks 4 and 5 are extracted in four separate non-overlapping tiles of the original image and are subsequently combined.

[4] https://github.com/colmap/colmap

[5] As matches obtained from the different variants vary, so do the orientation parameters, especially when different image pairs are used as initial pairs in Structure-from-Motion (SfM). Also, not all images can be registered with all variants.

4 and 5, distortion parameters are not used because the cameras only have a negligible level of distortion.

Evaluation Criteria For the evaluation of the results, the following criteria are reported:

- Number of registered images, i.e. the number of images for which the orientation parameters could be determined.
- Number of reconstructed 3D points.
- Average number of observations per image, namely the mean number of matching points of each image used in bundle adjustment.
- Number of intersecting rays per 3D point. For each 3D point, depending on which images these rays come from, four cases are distinguished:
  - A) from only one viewing direction (nadir or oblique) (*only_nad_or_obl*);
  - B) from the nadir and one oblique camera (*nad_obl*);
  - C) from the nadir and at least two different oblique cameras (*obl_nad_obl*);
  - D) from different oblique cameras (*obl_obl*).

  In general, the difficulty of matching increases from level A to level in D, as the change of viewpoint and viewing direction becomes larger accordingly.
- Precision of the 3D point coordinates obtained from error propagation. A higher precision indicates a better quality of the whole image orientation pipeline.

4.4 Evaluation results

4.4.1 Brown Dataset: As mentioend before, we tested six different combinations of the three subsets: in all combinations one subset was used for training, the other two subsets as test data. The results are shown in Table 3. The comparison to the work of (Mishchuk et al., 2017), called HardNet, and to SIFT (Lowe, 2004) is also reported.

For the descriptor evaluation on the Brown dataset our model achieves a quality comparable to HardNet, which is the basis of our work, and which can be considered as state-of-the-art in the field: for three cases our result is better, for three cases is is worse. This is a good pre-requisite for the determination of image orientation parameters reported in the next section.

4.4.2 Results for image orientation: The result for the image orientation experiment are reported in table 4 for the three different variants *hbss, hbsl* and *hlll*. The table contains the details of the bundle adjustment results. The distribution of the number of rays intersecting at the 3D points are illustrated in figure 7. The image orientation results indicate that the use of a learned descriptor improves the performance compared to hand crafted features (with SIFT as the example), see table 4. The numbers for *hbsl* are significantly better than those for *hbss*. In particular, the learned descriptor leads to a more complete image block, more matches and thus more 3D points, more observations per image, and a better 3D coordinate precision, while the mean reprojection error stays approximately constant.

The incorporation of learned affine and orientation parameters further improves the results. From the distribution of the number of views per 3D point in figure 7 it can be observed that our completely learned variant *hlll* results in a larger number of multiple ray 3D points. Also, the number of 3D points in the cases B(*nad_obl*), C(*obl_nad_obl*) and D(*obl_obl*) is higher for our pipeline than for the other two. Partly due to the fact that 3D points in our pipeline are observed in a larger number of views, the precision of reconstructed 3D points is also higher, which is more obvious in block4 and block5.

Based on the above observations, the learned affine shape, orientation and descriptor modules provide more complete and more accurate results for image orientation than the other tested pipelines for the challenging task of dealing with images containing large viewpoint and large viewing direction changes.

5. CONCLUSION AND FUTURE WORK

In this paper, a feature based image matching framework making use of deep learning is proposed. The affine shape estimation, orientation assignment and description of local features are all learned using CNN. The method provides state-of-the-art feature descriptors. Tests for image orientation of small blocks of penta cameras reveal that the proposed method achieves a better matching performance than more traditional methods.

Currently, we do not use canonical descriptions for the image patches, such as referring the orientation to the main gradient direction as is the case in SIFT. Consequently, multiple solutions exist for image matching in terms of orientation and affine transformation, and our solution contains an over-parametrisation of the feature correspondence problem. While we still use very good results, we plan to introduce constraints for the predicted transformation parameters to make the estimation more stable in the future. Also, we plan to use larger photogrammetric blocks in our evaluation. Furthermore, we plan to explore the possibility to integrate the three steps into one single network and to evaluate the limitations of learning our network on one dataset and then transferring the results to different sets of images.

ACKNOWLEDGEMENTS

The authors would like to thank NVIDIA Corp. for donating the GPU used in this research through its GPU grant program.

REFERENCES

AanÃęs, H., Dahl, A., Steenstrup Pedersen, K., 2012. Interesting Interest Points. *International Journal of Computer Vision*, 97(1), 18-35.

Balntas, V., Riba, E., Ponsa, D., Mikolajczyk, K., 2016. Learning local feature descriptors with triplets and shallow convolutional neural networks. *Proceedings of the British Machine Vision Conference*, 1–11.

Baumberg, A., 2000. Reliable feature matching across widely separated views. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1, 774–781.

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3), 346-359.

Brown, M., Hua, G., Winder, S., 2011. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1), 43–57.

Chen, L., Rottensteiner, F., Heipke, C., 2014. Learning image descriptors for matching based on Haar features. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3, 61-68.

Train	Feature	Lib.		Notre.		Yose.		
Test	Dim	Notre.	Yose	Lib.	Yose	Lib.	Notre.	mean
SIFT (Lowe, 2004)	128	22.53	27.29	29.84	27.29	29.84	22.53	26.55
HardNet (Mishchuk et al., 2017)	128	0.53	1.96	1.49	1.84	2.51	0.78	1.51
OURS	128	1.37	2.42	0.48	0.72	1.94	1.51	1.41

Table 3. Descriptor evaluation result (FPR at 95%TPR, unit %) on Brown dataset (Brown et al., 2011). Notre. = *Notre Dame*; Lib.= *Liberty*; Yose.= *Yosemite*. OURS represents the solution presented in this paper, which is a refinement of HardNet.

Block Index	Variant of Methods	# reg. img	# rec. pts	m. rpj. error [pixel]	#. gross errors	#m. obs. per img.	RMS value of X,Y,Z
Block-1	*hbss*	13/18	716	0.80	5	153.1	0.42, 0.81, 0.83
	hbsl	**17/18**	1670	**0.76**	**2**	270.7	**0.33**, 0.44, **0.58**
	hlll	**17/18**	**1834**	0.79	4	**347.1**	0.34, **0.42, 0.58**
Block-2	*hbss*	15/17	523	**0.65**	**0**	100.8	0.94, 1.05, 1.65
	hbsl	16/17	1444	0.72	2	290.9	0.34, 0.34, 0.54
	hlll	**17/17**	**1870**	0.74	**0**	**364.2**	**0.33, 0.32, 0.53**
Block-3	*hbss*	16/19	977	**0.68**	**0**	186.9	0.45, 0.50, 0.71
	hbsl	**19/19**	1708	0.70	1	302.8	0.37, 0.39, 0.63
	hlll	**19/19**	**2288**	0.74	2	**426.3**	**0.36, 0.35, 0.59**
Block-4	*hbss*	10/17	569	0.65	**7**	165.0	0.46, 0.70, 0.87
	hbsl	**17/17**	2471	0.62	13	457.5	0.26, 0.20, 0.37
	hlll	**17/17**	**3375**	**0.61**	23	**676.9**	**0.18, 0.17, 0.30**
Block-5	*hbss*	12/20	995	0.63	**12**	257.4	**0.19**, 0.33, 0.39
	hbsl	**20/20**	2714	0.61	16	455.3	0.31, 0.25, 0.38
	hlll	**20/20**	**4076**	**0.60**	41	**733.4**	0.20, **0.22, 0.28**

Table 4. The result for all blocks after robust bundle adjustment. #reg. img: number of registered images over available number of images; # rec. pts: number of reconstructed 3D points; m. rpj error [pixel]: mean reprojection error in [pixel]; #gross errors: number of gross errors, which are the matching points have residuals larger than 3.2 times of the standard deviation of adjusted matching points; #m. obs. per img.: number of mean observation per image; RMS values of X,Y,Z refer to the scaling of the block and are multiplied by 10^3. The best value in each column is shown in bold font.

Chen, L., Rottensteiner, F., Heipke, C., 2016. Invariant descriptor learning using a Siamese convolutional neural network. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3, 11-18.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., Seitz, S. M., 2007. Multi-view stereo for community photo collections. *Proceedings of International Conference on Computer Vision*, 1–8.

Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A. C., 2015. Matchnet: Unifying feature and metric learning for patch-based matching. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3279–3286.

Hoffer, E., Ailon, N., 2015. Deep metric learning using triplet network. *Proceedings of the International Workshop on Similarity-Based Pattern Recognition*, 84–92.

Kumar, B., Carneiro, G., Reid, I. et al., 2016. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5385–5394.

Lenc, K., Vedaldi, A., 2016. Learning covariant feature detectors. *Proceedings of the European Conference on Computer Vision Workshops*, Springer, 100–117.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91-110.

Luo, Z., Shen, T., Zhou, L., Zhu, S., Zhang, R., Yao, Y., Fang, T., Quan, L., 2018. Geodesc: Learning local descriptors by integrating geometry constraints. *Proceedings of the European Conference on Computer Vision*, 168–183.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L., 2005. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2), 43–72.

Mishchuk, A., Mishkin, D., Radenovic, F., Matas, J., 2017. Working hard to know your neighbor's margins: Local descriptor learning loss. *Proceedings of the Advances in Neural Information Processing Systems*, 4826–4837.

Mishkin, D., Radenovic, F., Matas, J., 2018. Repeatability is not enough: Learning affine regions via discriminability. *Proceedings of the European Conference on Computer Vision (ECCV)*, 284–300.

Mitra, R., Doiphode, N., Gautam, U., Narayan, S., Ahmed, S., Chandran, S., Jain, A., 2018. A large dataset for improving patch matching. *arXiv:1801.01466v3*.

Moo Yi, K., Verdie, Y., Fua, P., Lepetit, V., 2016. Learning to assign orientations to feature points. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 107–116.

Morel, J.-M., Yu, G., 2009. ASIFT: A new framework for fully affine invariant image comparison. *SIAM journal on imaging sciences*, 2(2), 438–469.

Nex, F., Remondino, F., Gerke, M., Przybilla, H.-J., Bäumker, M., Zurhorst, A., 2015. ISPRS Benchmark for Multi-platform Photogrammetry. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-3/W4, 135–142.

Perd'och, M., Chum, O., Matas, J., 2009. Efficient representation of local geometry for large scale object retrieval. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9–16.

Schönberger, J. L., Frahm, J.-M., 2016. Structure-from-motion re-

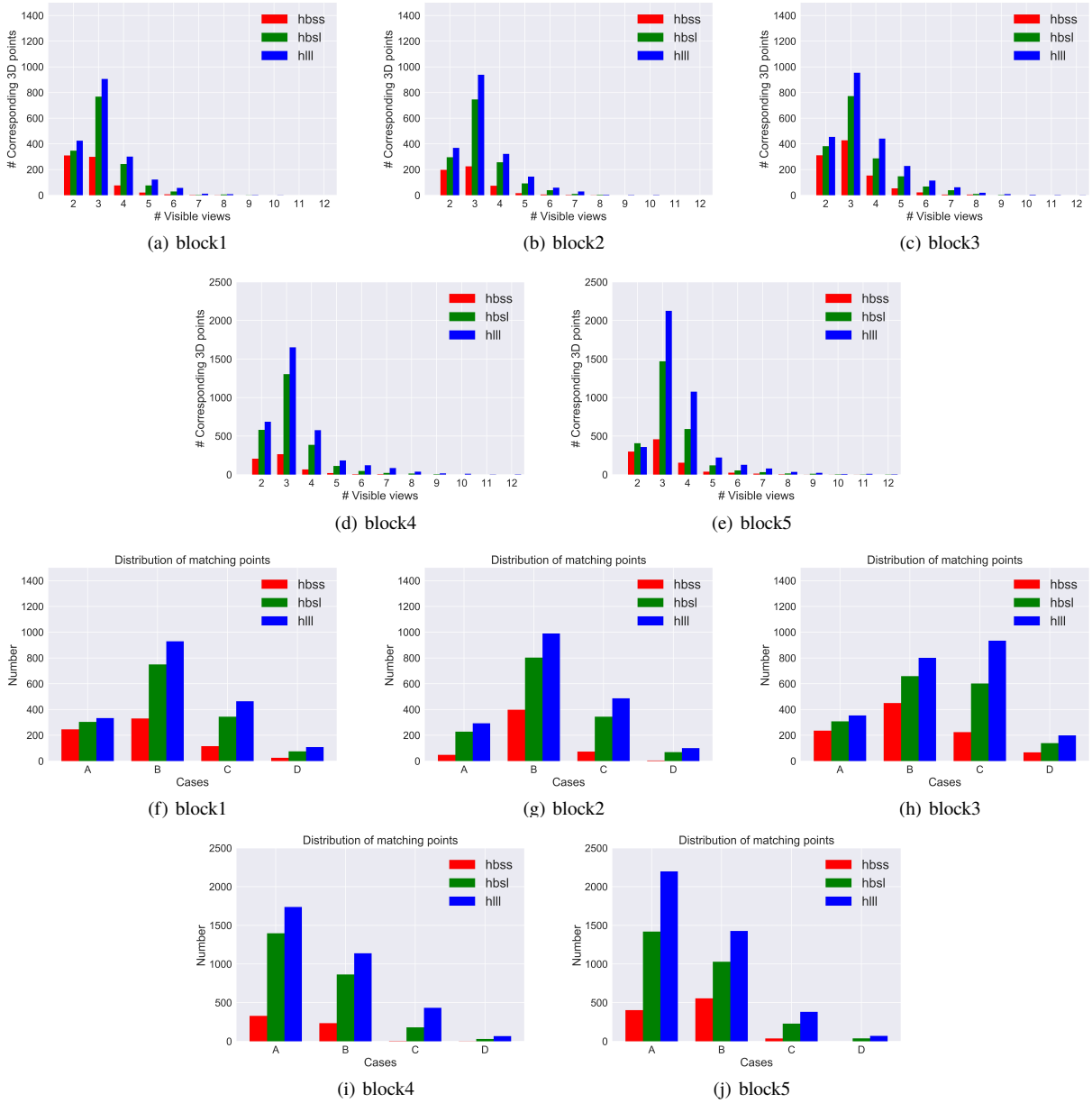

Figure 7. Resuls for the three pipeleins (see text for explanation of the abbreviations). Figure (a) to (e) indicate the distribution of the number of multiple rays per object point for blocks 1 to 5. Figure (f) to (j) indicates the distribution of involved cameras for the five blocks, where A="*only_nad_or_obl*", B="*nad_obl*", C="*obl_nad_obl*" and D="*obl_obl*".

visited. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4104–4113.

Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F., 2015. Discriminative learning of deep convolutional feature point descriptors. *Proceedings of the IEEE International Conference on Computer Vision*, 118–126.

Snavely, N., Seitz, S. M., Szeliski, R., 2008. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2), 189–210.

Tian, Y., Fan, B., Wu, F., 2017. L2-net: Deep learning of discriminative patch descriptor in euclidean space. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 661–669.

Trzcinski, T., Christoudias, M., Lepetit, V., 2015. Learning image descriptors with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3), 597–610.

Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P., 2012. Learning image descriptors with the boosting-trick. *Proceedings of the Advances in Neural Information Processing Systems*, 269–277.

Winder, S. A., Brown, M., 2007. Learning local image descriptors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 1–8.

Zagoruyko, S., Komodakis, N., 2015. Learning to compare image patches via convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4353–4361.

Zhang, X., Felix, X. Y., Kumar, S., Chang, S.-F., 2017. Learning spread-out local feature descriptors. *Proceedings of the IEEE International Conference on Computer Vision*, 4605–4613.