Euclidean and Hamming Embedding for Image Patch Description with Convolutional Networks

Zishun Liu Zhenxi Li Juyong Zhang Ligang Liu
University of Science and Technology of China

Abstract

Local feature descriptors represent image patches as floating-point or binary arrays for computer vision tasks. In this paper, we propose to train Euclidean and Hamming embedding for image patch description with triplet convolutional networks. Thanks to the learning ability of deep ConvNets, the trained local feature generation method, which is called Deeply Learned Feature Transform (DELFT), has good distinctiveness and robustness. Evaluated on the UBC benchmark, we get the state-of-the-art results using floating-point and binary features. Also, the learned features can cooperate with existing nearest neighbor search algorithms in Euclidean and Hamming space. In addition, a new benchmark is constructed to facilitate future related research, which contains 40 million image patches, corresponding to 6.7 million 3D points, being 25 times larger than existing dataset. The distinctiveness and robustness of the proposed method are demonstrated in the experimental results.

1. Introduction

Image local features, which describe image patches as floating-point or binary arrays, have been widely used in many computer vision tasks, from low-level to high-level ones. Local features are required to be distinctive to different physical points while robust to scale, orientation, viewpoint, illumination and other variations. Thus it is nontrivial to extract good local features and a lot of research has been done on this problem, including handcrafted [14, 2, 24, 20, 4, 28] approaches and data-driven [3, 23, 25, 26] ones in the last two decades. Nowadays, local feature still plays an irreplaceable role in many problems such as image matching (including structure-from-motion, image-based localization, image stitching, *etc*.) and object instance recognition [9], although Convolutional Networks based methods have been dominant on semantic-level problems, such as object classification [11] and detection [19].

In this paper, we propose to train Euclidean and Hamming embedding for local patch description with deep ConvNets (Convolutional Networks). Utilizing the learning ability of deep ConvNets, the trained local feature generation method has good distinctiveness and robustness. This is done by minimizing contrastive loss or triplet loss with massive training data. The trained floating-point or binary features are extracted directly from raw pixel data, in an end-to-end learning manner, without relying on human experience. Our learned feature embedding is named as Deeply Learned Feature Transform (DELFT).

ConvNets have achieved the state-of-the-art results on many computer vision tasks and recently have been used for patch level feature learning. In Han *et al*. [8], a system named MatchNet is proposed, in which a metric network is used to compute the similarity between two features extracted from different patches. Methods proposed by Zagoruyko and Komodakis [33] have similar network structures while get better results with several techniques such as taking multi-scale input. One of the drawbacks is that a metric network is needed in both works to compute the similarity, for which we have to use linear search method to find the nearest neighbor for some query patch. In this paper, Euclidean and Hamming metric is directly used which make the learned features cooperating with existing nearest neighbor search algorithms, such as those with k-d trees, k-nearest neighbor graph or multi-index hashing[18]. In Simo-Serra *et al*. [22], contrastive loss is used to learn such an embedding. In this work a novel triplet loss is used and get the state-of-the-art result compared with existing works on the same feature length. Moreover, binary features are learned in our work.

In addition, we constructed a local patch correspondence dataset to facilitate future related research. This dataset contains 40 million color image patches, corresponding to 6.7 million 3D points, being 25 times larger than existing dataset.

The main contributions of our approach are summarized in the following.

- A new local patch feature embedding method is proposed which cooperates with fast nearest neighbors

search, getting a better speed-accuracy tradeoff.

- This is the first work that learns compact binary feature with ConvNets and got the state-of-the-art result on standard benchmark.
- A new local image patch correspondence dataset is constructed which is 25 times larger than existing benchmark, in which color information is also included.

The remainder of this paper is structured as follows. Section 2 briefly reviews several related works. In Section 3 our network structure, loss function and optimization method is specified. The experimental results and comparisons with related works are demonstrated in Section 4. Finally, Section 5 presents our conclusions and directions of future work to improve our method.

2. Related Work

Local features should be partially invariant to illumination, 3D projective transforms, and common object variations, while be sufficiently distinctive to identify specific objects among many alternatives. Many works have been done to design such local features, aiming for a better tradeoff between distinctiveness and robustness, including SIFT [14], SURF [2], DAISY [24], ORB [20], BRIEF [4], MIOP [28], *etc*. These floating-point or binary features are based on histogram of gradient, intensity comparison or some other image operations. Designing such hand-crafted feature rely on human experience heavily.

Some other approachess have been proposed with data-driven methods. In Brown *et al*. [3], a set of building blocks is described for constructing descriptors which can be combined together and jointly optimized so as to minimize the error of a nearest-neighbor classifier for local image descriptor learning. In Trzcinski *et al*. [26], boosting is applied to learn complex non-linear local visual feature representation. Trzcinski *et al*. [25] propose a novel framework to learn an extremely compact binary descriptor, each bit of which is computed with a boosted binary hash function, which is efficient to optimize the different hash functions so that they complement each other. In Simonyan *et al*. [23] learning the pooling regions for the descriptor and descriptor dimensionality reduction are formulated as convex optimization problems.

Recently, several works have been done on image patch feature learning using ConvNets. In Han *et al*. [8], a system named MatchNet is proposed, which contains two feature networks with tied parameters and a metric network. The metric network is used to compute the similarity between two features extracted from different patches by the feature networks. Methods proposed by Zagoruyko and Komodakis [33] have similar network structures while get better results with several techniques such as taking multi-scale input. In Simo-Serra *et al*. [22], contrastive loss is used to learn such an embedding. As a novel triplet loss is used in our work, the results are significantly better than that of [22]. In Han *et al*. [8], a lossy quantization method is used to represent the features as binary codes compactly. The compressed features should be decoded in the similarity computation stage, thus cannot be treated as binary features. In contrast, similarity between our binary features can be measured by Hamming distance directly, so that exact sub-linear k-NN search is permitted. In addition, Zbontar and LeCun in [34] proposed a ConvNet-based approach to compare patches for computing cost in small baseline stereo problem and shown the best performance in KITTI dataset. However, the focus of that work was on narrow baseline stereo, which is different from our work.

The metric learning problem [12] is concerned with learning a distance function tuned to a particular task, and has been shown to be useful when used in conjunction with nearest-neighbor methods and other techniques that rely on distances or similarities. LMNN (Large Margin Nearest Neighbor [29]) is one of the representative work in metric learning field, where a triplet loss is used. A similar triplet loss is used in [21] to learn a unified feature embedding for face recognition and clustering. There are also some works attempt to map high-dimensional data to binary codes that preserve semantic similarity. In [17], the difficulty of discontinuous optimization of the discrete mappings is overcome by minimizing a piecewise-smooth upper bound on empirical loss. In [13], a divide-and-encode module is used to learn binary mapping with preserved similarity, by minimizing a triplet ranking loss, in which a piece-wise threshold function is used to encourage binary output.

3. Euclidean and Hamming Embedding

Networks to learn the Euclidean and Hamming embedding are based on the Siamese Network [7]. Basic block of the network is a feature tower which transforms a given image patch into a floating-point or binary array. Each network contains 2 or 3 such feature towers with tied parameters. Given pairs or triplets of training patches, each patch is fed into the feature tower and the loss layer computes the cost value based on their output features. Then, back propagation is used to train these feature towers simultaneously. Overview of the system is illustrated in Figure 1. Architecture of the feature tower is listed in Table 1.

In [22], the contrastive loss [7] is used to learn the feature transform for image patch matching. The embedding is represented by $f(x) \in \mathbb{R}^d$ or $\{0,\ 1\}^d$. It embeds an image patch x into a d-dimensional Euclidean or Hamming space. By minimizing the following loss function, we want to make $\|f(x_1) - f(x_2)\|_2$ small when x_1 and x_2 represent the same 3D point and make it large if they are extracted

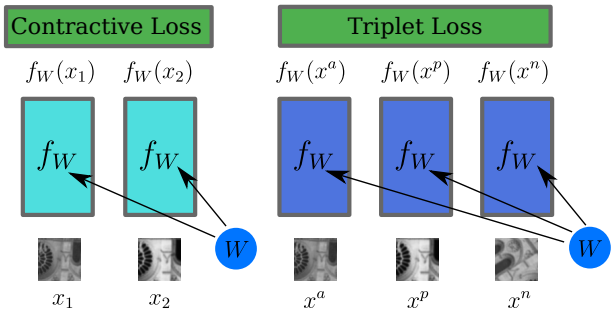

Figure 1. System overview. Basic block of the network is a feature tower f_W which transform a given image patch x into a floating-point or binary array $f_W(x)$. Left: network contains 2 such feature towers with tied parameters. If the input pair correspond to the same 3D point, the network is trained to make the outputs $f(x_1)$ and $f(x_2)$ to be as close as possible. If not, it is trained to make them far from each other. Right: network contains 3 such feature towers with tied parameters, which is trained to make $f(x^a)$ and $f(x^p)$ get close to each other while $f(x^a)$ and $f(x^n)$ get far from each other.

Ind	Type	Filter Size	Num	Stride	Pad
1	Conv	7	48	1	3
2	ReLU	-	-	-	-
3	Max Pool	3	-	2	0
4	Conv	5	64	1	2
5	ReLU	-	-	-	-
6	Max Pool	3	-	2	0
7	Conv	3	128	1	1
8	ReLU	-	-	-	-
9	Conv	3	256	1	1
10	ReLU	-	-	-	-
11	Conv	3	512	1	1
12	ReLU	-	-	-	-
13	Max Pool	3	-	2	0
14	FC	-	1024	-	-
15	ReLU	-	-	-	-
16	FC	-	1024	-	-
17	ReLU	-	-	-	-
18	FC	-	dim(f)	-	-

Table 1: Architecture of the feature tower. FC is for fully-connected layer. It is essential to add a normalization layer on the top of the tower when triplet loss is used. To learn Hamming embedding, a piece-wise threshold layer should be added on the top.

from different 3D points.

$$\sum_{i=1}^{N} \Big[Y_i d_i + (1 - Y_i) \max(\alpha - d_i,\ 0) \Big], \qquad (1)$$

in which

$$d_i = \|f(x_{i1}) - f(x_{i2})\|_2^2,$$

N is the batch size. $Y_i = 1$ when x_{i1} and x_{i2} represent the same 3D point and 0 otherwise.

3.1. Triplet Loss

Compared with the contrastive loss, triplet Loss is a better choice as previous works [21, 29] suggested. We want to ensure that an image patch x_i^a (*anchor*) of a specific 3D point is closer to all other patches x_i^p (*positive*) of the same 3D point than it is to any patches x_i^n (*negative*) of any other 3D points. The loss that is being minimized is then

$$\sum_{i=1}^{N} \Big[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \beta \Big]_+$$

where β is a margin that is enforced between positive and negative pairs. The normalization layer, which normalizes the L2 norm of output layer to be 1, is essential to prevent the norms of output arrays from exploding when triplet loss is used.

To learn the Hamming embedding, the following piece-wise threshold function [13] is used on the top of the feature tower to encourage binary outputs.

$$g_\epsilon(x) = \begin{cases} -1, & x < -\epsilon \\ x, & -\epsilon \le x \le \epsilon \\ 1, & x > \epsilon \end{cases}$$

In the training stage, ϵ is initiated to be 0.5 and decreased by 0.1 after some iterations. During the test stage, 0 is used as the threshold to map floating-point arrays into binary features.

3.2. Training

Mini-batch gradient descent is used to minimize the loss functions. Existing works suggest to initialize network parameters using gaussian random numbers with small variation for classification problems. In our task, it will lead to small d_i and will make the gradient of contrastive loss in Eqn. (1) vanished. The learning procedure will be slow even with a large learning rate. As a consequence, we tuned the standard variation of the gaussian distributions to make the norms of feature layer's outputs near 1 (*i.e.* the margin α we used for the contrastive loss). Once we obtain the model with the contrastive loss, the trained model could be used as the initialization to further improve the performance with the triplet loss. Also, this initialization helps to prevent the network from producing degenerated result, *i.e.* the collapsed feature transform $f(x) = 0$.

We use greedy strategy to select pairs or triplets for training. Similar with that of [21], pairs or triplet that can produce loss are selected while those make the loss functions

output 0, which would lead to slow learning, are dropped. Also, online data augmentation on patch brightness and contrast is used to improve the generalization ability. We used margin $\alpha = 1$ for contrastive loss and margin $\beta = 0.2$ for triplet loss. Some other parameters are batch size 64, weight decay 5.0×10^{-5} and initial learning rate 0.02. The learning rate is dropped by a factor after some iterations. The implementation is based on the open source deep learning framework Caffe [10].

4. Experiments

4.1. Local Patch Correspondence Benchmark

The UBC benchmark [30] consists of 64×64 grayscale image patches sampled form 3D reconstructions of 3 scenes: the Statue of Liberty, Notre Dame and Half Dome in Yosemite. Each of them contains more than 450,000 image patches detected using the Difference of Gaussians detectors. The patches are scale and orientation normalized. Each of the subsets was generated using actual 3D correspondences obtained via multi-view stereo depth maps. These maps were used to produce 500,000 ground-truth feature pairs for each dataset, with equal number of positive (correct) and negative (incorrect) matches.

A larger benchmark is constructed to further improve performance of the feature embedding. The new benchmark is based on the MSR-Object3D-300 dataset [9], which contains 300 objects (mainly toys) and each object is represented by more than 200 images captured from different viewpoints. Patch correspondences are extracted from them, using VisualSFM [31], a structure from motion system which implements multicore bundle adjustment [32]. At last, 40 million color image patches are extracted, corresponding to 6.7 million 3D points. These patches are normalized as that of the UBC benchmark. Sampled images and patch correspondences are shown in Figure 2. As MSR-Object3D-300 is captured under controlled laboratory conditions, data augmentation especially brightness and contrast transformations are essential in the training stage.

We trained the networks on the UBC dataset and our newly constructed dataset. and listed the test error on the UBC benchmark in Table 2. For evaluating our models, we use the evaluation protocol of [3] and test on the subset containing 10,000 patch pairs which is sampled by the authors of [30]. We report the false positive rate at 95% recall (FPR95) on each of the six combinations of training and test sets. From this table, we can see that 128d DELFT trained with triple loss achieves the state-of-the-art result. 128 bits binary DELFT is even superior to 256 bits BinBoost. RootSIFT, which is proposed in [1], is used here as the baseline. It is a simple transformation of SIFT descriptor, whose Euclidean distance is equivalent to the Hellinger distance between SIFT descriptors, which works better than measuring SIFT with Euclidean distance directly. We use the implementation of SIFT in VLFeat [27] for this evaluation.

In Figure 3, we show top-ranking false and true matches predicted by 128 bits binary DELFT. We observe that false matches could be easily mistaken even by human. Some of the false pairs may be mislabeled in the dataset. Also, false positives demonstrate that DELFT is robust against patch rotation to some degree.

Using the configuration of Simo-Serra *et al*. [22], we evaluate DELFT and list the areas under precision-recall curves (PR AUC) in Table 3. On average, we have an improvement of about 10% relatively compared with [22].

Train	Test	Simo-Serra *et al*.	DELFT (triple)
LY+YO	ND	0.667	0.705
LY+ND	YO	0.545	0.639
YO+ND	LY	0.608	0.653

Table 3: PR AUC of Simo-Serra *et al*. [22] and our approach. Both of them are 128 dimensional floating-point descriptors.

4.2. Robustness Evaluation on the Mikolajczyk Dataset

In Mikolajczyk *et al*. [16], a series of images and ground truth homographies are provided to evaluate the descriptors' robustness against image transformations, including rotation, scale change, viewpoint change, image blur, JPEG compression and illumination. We follow the evaluation protocol of [16] and use Harris-Affine detector [15]. Results of Binary DELFT (128 bits), BinBoost (128 bits) and SIFT (128d) are demonstrated in Figure 4. The trained model of BinBoost is provided by the authors of [25]. We can see that binary DELFT is superior to BinBoost with the same length. Also, binary DELFT is at least comparable with SIFT, while the memory cost of SIFT is 8 times larger than that of binary DELFT. In addition, although our model is trained on patches detected by DoG, it also works well on Harris-Affine regions.

4.3. Extraction and Matching Time

Finally, we provide the computational cost in Table 4. The CPU descriptors run on a Intel(R) Core(TM) i5-4570 CPU @ 3.20 GHz multi-threaded. Our GPU variant runs on a NVIDIA GeForce GTX 760 GPU. SIFT rely on VLFeat [27]. DELFT extraction could be speeded up using [6] by about 2 times. In the scenario of object instance recognition [9], given a dataset of target model feature descriptors of size n and a query image with m feature descriptors, the matching complexity of [8] and [33] is $\mathcal{O}(mn)$, where it is necessary to use brute force method. However, only $\mathcal{O}(m \log n)$ operations are required if the

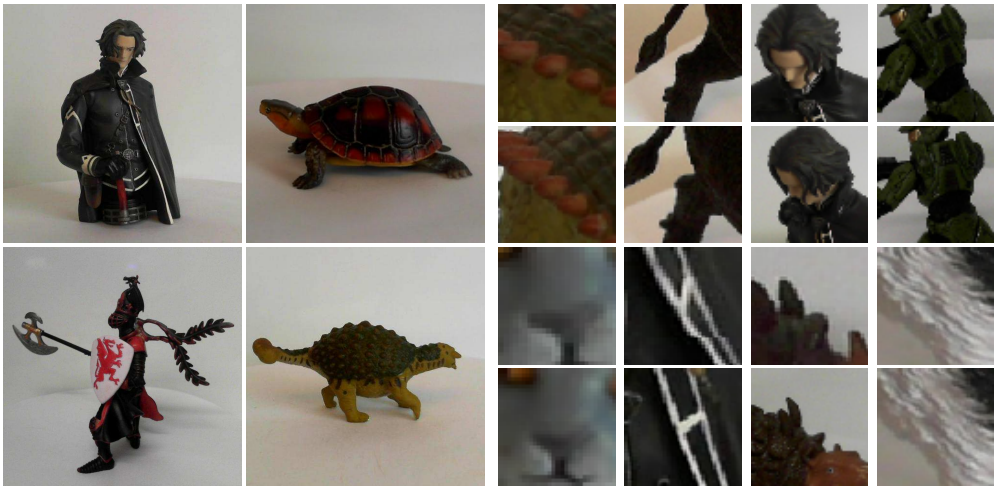

Figure 2. Sampled images MSR-Object3D-300 and patches detected using the Difference of Gaussians operator from it. The patches are scale and orientation normalized. These samples demonstrates viewpoint and original scale (from large body-parts to local textures) variations in the newly constructed benchmark for image patch descriptor learning.

Training		Notredame	Yosemite	Liberty	Yosemite	Liberty	Notredame
Test	Dim	Liberty		Notredame		Yosemite	
RootSIFT [1]	128d	28.76		20.96		26.37	
L-BGM [26]	64d	18.05	21.03	14.15	13.73	19.63	15.86
Brown *et al*. [3] w/ PCA	29d	16.85	18.27	–	11.98	–	13.55
Simonyan *et al*. [23] discrim. proj.	<80d	12.42	14.58	7.22	6.17	11.18	10.08
Simonyan *et al*. [23] discrim. proj.	<64d	12.88	14.82	7.52	7.11	11.63	10.54
MatchNet [8] (F=1024, B=64)	64d	9.82	14.27	5.02	9.15	14.15	13.20
MatchNet [8] (F=512, B=128)	128d	9.48	15.40	5.18	8.27	14.40	12.17
MatchNet [8] (F=512, B=512)	512d	8.84	13.02	4.75	7.70	13.58	11.00
Siamese Network in [33]	256d	8.77	13.48	4.33	5.75	14.89	13.23
DELFT (pair)	128d	8.50	12.57	5.02	6.54	12.25	11.06
DELFT (triple)	128d	7.26	11.83	4.29	5.37	10.67	9.82
DELFT (pair, on new benchmark)	128d	7.54		4.86		10.22	
BGM [26]	64 bits	31.90	33.54	29.60	26.80	38.13	30.58
BinBoost [25]	64 bits	20.49	21.67	16.90	14.54	22.88	18.97
BinBoost [25]	256 bits	21.62	22.18	15.99	14.69	21.11	18.42
Binary DELFT (pair)	64 bits	19.11	20.43	13.59	12.34	19.25	17.36
Binary DELFT (triple)	128 bits	16.27	18.32	9.31	10.56	15.77	13.94

Table 2: FPR95 (false positive rate at 95% recall) on the UBC benchmark.

	DELFT (GPU)	DELFT (CPU)	SIFT
Time (ms)	0.89	5.66	0.14

Table 4: Extraction time (in batch).

floating-point features are index in a k-d tree for matching. The complexity is similar when multi-index hashing is used for binary features. Also, the data structure only need to be constructed once.

5. Conclusion and Future Works

In this paper, we propose to train Euclidean and Hamming embedding for image patch description with convolutional networks. Thanks to the learning ability of deep ConvNets, the trained local feature generation method has good distinctiveness and robustness, which is called Deeply Learned Feature Transform (DELFT). Evaluated on the

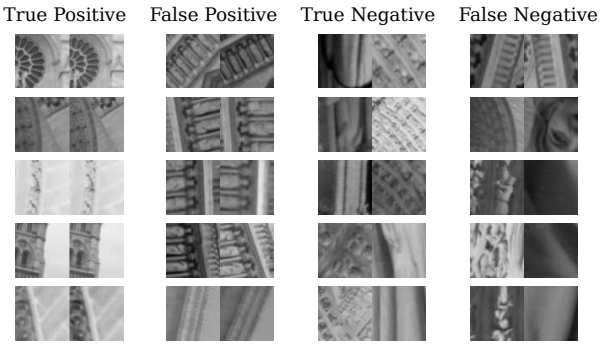

Figure 3. Top-ranking false and true matches predict on Notredame, using the 128 bits binary DELFT trained on Liberty with triplet loss.

UBC benchmark, we get the state-of-the-art results using floating-point and binary features that have the same length with existing works. Also, the learned features can cooperate with existing nearest neighbor search algorithms in Euclidean and Hamming space. In addition, based on the MSR-Object3D-300 dataset, a new benchmark is constructed to facilitate future related research, which contains 40 million color image patches, corresponding to 6.7 million 3D points, being 25 times larger than existing dataset. The distinctiveness and robustness of the proposed method are demonstrated in the experimental results.

On the other hand, our method can still be improved in several directions. First, as truecolor image patch correspondence dataset is available, we can take use of them to train more distinctive feature transforms for color patch description. Secondly, those techniques proposed by [33], such as the 2-stream multi-scale method, should be considered to improve the performance of DELFT. Thirdly, we consider to compress the trained models so that they could be used on mobile devices for related computer vision tasks such as SLAM. For example, HashedNets proposed by [5] could be used to shrink the storage requirements of DELFT substantially while mostly preserving generalization performance. Also, related works such as [6] will help to speed up the feature extraction.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2911–2918, 2012. 4, 5

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. 2006. 1, 2

[3] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):43–57, 2011. 1, 2, 4, 5

[4] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1281–1298, 2012. 1, 2

[5] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. *International Conference on Machine Learning*, 2015. 6

[6] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, pages 1269–1277, 2014. 4, 6

[7] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742, 2006. 2

[8] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. MatchNet: Unifying feature and metric learning for patch-based matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 4, 5

[9] Q. Hao, R. Cai, Z. Li, L. Zhang, Y. Pang, F. Wu, and Y. Rui. Efficient 2d-to-3d correspondence filtering for scalable 3d object recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 1, 4

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, MM '14, 2014. 4

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1

[12] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012. 2

[13] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 3

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1, 2

[15] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004. 4

[16] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005. 4

[17] M. Norouzi, D. M. Blei, and R. R. Salakhutdinov. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, pages 1061–1069, 2012. 2

[18] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3108–3115, 2012. 1

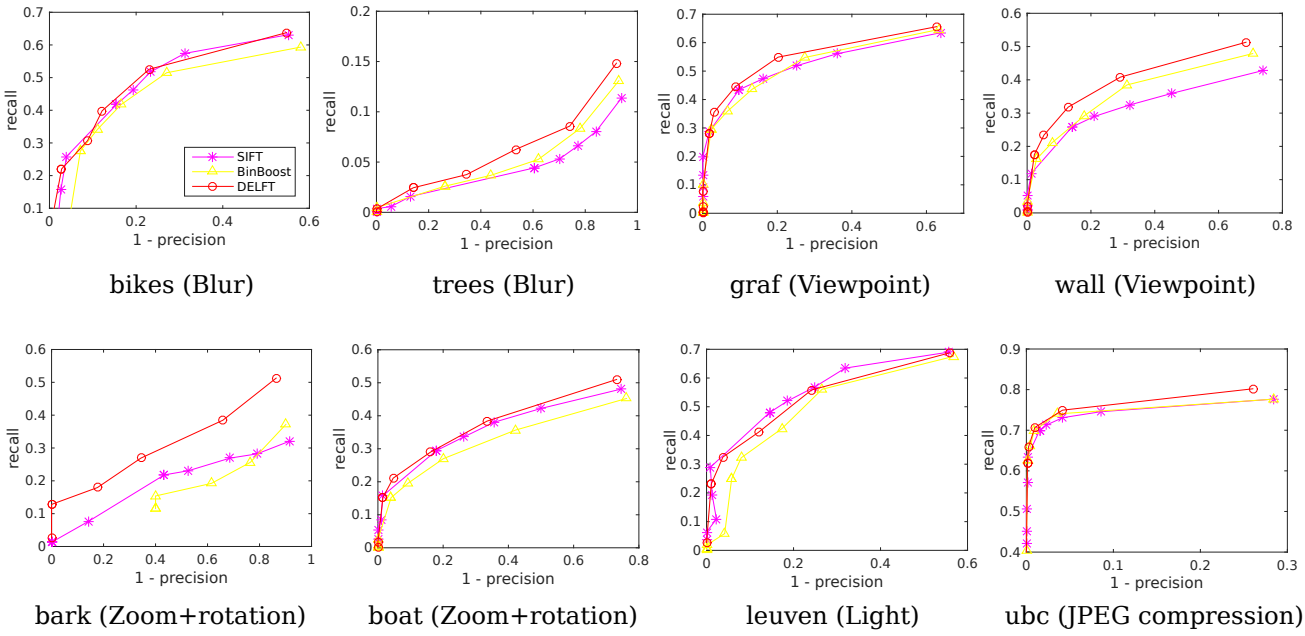

Figure 4. Evaluation of robustness on the Mikolajczyk dataset. We can see that binary DELFT (128 bits) is superior to BinBoost (128 bits) with the same length. Also, binary DELFT is at least comparable with SIFT (128 bytes).

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011. 1, 2

[21] F. Schroff, D. Kalenichenko, and J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 3

[22] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015. 1, 2, 4

[23] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1573–1585, 2014. 1, 2, 5

[24] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. 1, 2

[25] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting binary keypoint descriptors. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2874–2881, 2013. 1, 2, 4, 5

[26] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua. Learning image descriptors with the boosting-trick. In *Advances in Neural Information Processing Systems*, pages 269–277, 2012. 1, 2, 5

[27] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008. 4

[28] Z. Wang, B. Fan, G. Wang, and F. Wu. Exploring local and overall ordinal information for robust feature description. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2016. 1, 2

[29] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems*, pages 1473–1480, 2005. 2, 3

[30] S. Winder, G. Hua, and M. Brown. Picking the best daisy. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 4

[31] C. Wu. VisualSFM: A visual structure from motion system. http://ccwu.me/vsfm/, 2011. 4

[32] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064, 2011. 4

[33] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 2, 4, 5, 6

[34] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2