SConE: Siamese Constellation Embedding Descriptor for Image Matching

Tomasz Trzcinski[1,2], Jacek Komorowski[1(✉)], Lukasz Dabala[1], Konrad Czarnota[1], Grzegorz Kurzejamski[1], and Simon Lynen[3]

[1] Warsaw University of Technology, Warsaw, Poland
jacek.komorowski@pw.edu.pl
[2] Tooploox, Wrocław, Poland
[3] Google, Mountain View, USA

Abstract. Numerous computer vision applications rely on local feature descriptors, such as SIFT, SURF or FREAK, for image matching. Although their local character makes image matching processes more robust to occlusions, it often leads to geometrically inconsistent keypoint matches that need to be filtered out, e.g. using RANSAC. In this paper we propose a novel, more discriminative, descriptor that includes not only local feature representation, but also information about the geometric layout of neighbouring keypoints. To that end, we use a Siamese architecture that learns a low-dimensional feature embedding of keypoint constellation by maximizing the distances between non-corresponding pairs of matched image patches, while minimizing it for correct matches. The 48-dimensional floating point descriptor that we train is built on top of the state-of-the-art FREAK descriptor achieves significant performance improvement over the competitors on a challenging TUM dataset.

Keywords: Feature descriptor · Image matching · Siamese networks

1 Introduction

Matching images with local feature descriptors is a fundamental part of many computer vision applications, including 3D reconstruction [1], panorama stitching [2] and monocular Simultaneous Localization and Mapping [3]. This topic has therefore gained significant attention from the research community [4–7]. While traditional approaches rely on hand-crafted features [4–7], more recent descriptors use machine learning techniques such as boosting [8] or deep learning [9,10] to train discriminative transformation-invariant representations. Although using local feature descriptors proposed in the literature increases robustness of image matching methods to partial occlusions, it often leads to incorrect descriptor matches, as presented in the upper right part of Fig. 1. In this paper, we propose a more discriminative feature descriptor by encoding information about constellation of keypoints, as shown in Fig. 1. To that end, we use a Siamese neural network that learns low-dimensional feature embeddings by minimizing distance

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11129, pp. 401–413, 2019.
https://doi.org/10.1007/978-3-030-11009-3_24

Fig. 1. Our proposed SConE descriptor uses information about neighbouring keypoints (left figure) to construct a discriminative low-dimensional embedding of a keypoint constellation. This way matching images with SConE reduces the number of incorrect matches found with respect to those found using a standard feature descriptor FREAK (coloured red in the top right figure) and increases the quality of resulting matches (bottom right figure). (Color figure online)

between similar keypoint constellations, while maximizing it for non-matching pairs. Instead of relying on a local intensity patch around the detected keypoint, we construct our embedding by feeding into the neural network information on a central keypoint and its nearest neighbourhood keypoints on the image. The resulting 48-dimensional Siamese Constellation Embedding descriptor, dubbed SConE for simplicity, is built using FREAK [5] as a base descriptor, however our framework is agnostic to descriptor types and can be generalized to other descriptors. Evaluation of our descriptor on the challenging TUM dataset [11] shows that despite its compact nature, SConE outperforms its competitors, while decreasing the computational cost of matching by eliminating the need for a geometrical verification step.

In the remainder of this paper, we first discuss related work in Sect. 2. We then describe the details of our method in Sect. 3. Finally, in Sect. 4 we show that our descriptor is able to outperform the state-of-the-art descriptors on a real-life dataset and we conclude the paper in Sect. 5.

2 Related Work

Due to the role of local features descriptors in many computer vision tasks, significant amount of work has been focused on building those representations effectively and efficiently [4–6,8,9]. Floating-point descriptors, such as SIFT [4] or SURF [6], typically offer better performance at a higher computational cost. Their binary competitors, such as FREAK [5] or ORB [7] approximate many operations and simplify the resulting representation to a binary output. Our proposed method is built on top of the binary FREAK descriptor, which offers an efficient yet powerful alternative to floating-point competitors. Nevertheless, the framework proposed in this paper is general enough to be applicable also to other descriptors.

Recently, due to their success in other domains, deep neural networks have also been used to train feature descriptors [9,10]. For instance, LIFT [10] uses a neural network architecture to handle full pipeline of feature extraction and computation. Another method called MatchNet uses Siamese neural network to jointly learn feature representation and matching procedure [12]. In [13], they use a triplet loss function coupled with convolutional neural network that aims at training context-augmented descriptors based on FREAK. In our work, we use a Siamese architecture to learn low-dimensional feature embeddings based on FREAK descriptors. But instead of using an image patch around detected keypoints, we incorporate data on the neighbourhood keypoints.

[14,15] apply convolutional neural networks to graph data to learn useful features. However our input data does not have a graph structure defined by an adjacency matrix. Spatial positions of neighbourhood keypoints and its attributes (binary descriptor, position and orientation) are important, not the structural relationships between keypoints.

Once keypoint descriptors are extracted, they are typically matched with each other to find correspondences between image regions. Depending on the final application, the matching can be done using brute-force or approximate nearest neighbour (ANN) search methods [16,17]. Heuristic techniques (e.g. two nearest neighbour ratio test [4]) are used to filter out outliers. In the final stage, putative matches are typically subject to geometric validation using epipolar constraint with a robust parameter estimation method, such as RANSAC [18] or its extension USAC [19]. A recently proposed approach called GMS (Grid-based Motion Statistics) [20] also aims at filtering out geometrically incorrect matches using a simple heuristic based on the number of matches in the keypoint neighbourhood. Although often effective, above methods require additional computational cost. In our method we propose to embed the geometrical information useful for filtering the matches within the descriptor itself. This way we can avoid the unnecessary post-processing step and increase the efficiency of the image matching pipeline.

3 Method

Our method aims at improving precision of the descriptor matching step. Instead of matching raw descriptors (e.g. 512-bit FREAK descriptors), we compare more discriminative representations of keypoint constellations. We define a *constellation* as a set of nearby keypoints in an image. It consists of a *central keypoint* and its k nearest, in Euclidean distance sense, keypoints detected on the same image. An exemplary constellation is visualized on Fig. 1. The following information is taken into account when constructing a *constellation*: binary descriptor, scale and orientation of a central keypoint; and binary descriptors, relative position (with respect to the central keypoint), scale and orientation of each of its k nearest neighbours. In this work, based on an initial experiments, we set $k = 20$.

Dimensionality of the data constituting a constellation is rather high. We find low dimensional constellation embeddings by training the Siamese neural

network [21]. This produces low dimensional, real valued, embeddings that can be efficiently stored and processed. High-level architecture of our Siamese neural network is depicted on Fig. 2. The network consists of two identical Siamese modules (same network architecture with shared weights) that compute low dimensional constellation embeddings. Representations computed by Siamese modules can be matched using standard Euclidean distance between them.

The network is trained by presenting mini-batches consisting of pairs of similar and dissimilar constellations. We consider two constellations similar if their central keypoint is a projection of the same 3D scene point (landmark). Otherwise constellations are dissimilar. We use a contrastive loss function, as formulated in [22]. Let X_1, X_2 be a pair of constellations in the training set and Y a binary label assigned to this pair. $Y = 0$ if constellations X_1 and X_2 are similar, and $Y = 1$ if they are dissimilar. D_W is a parametrized distance function between constellations X_1 and X_2, defined as an Euclidean distance between learned constellation embeddings G_W.

$$D_W\left(X_1, X_2\right) = ||G_W\left(X_1\right) - G_W\left(X_2\right)||_2 \tag{1}$$

The loss $\mathcal{L}$ function minimized during the training is defined as:

$$\mathcal{L} = \sum_{i=1}^{P} L\left(W, \left(Y, X_1, X_2\right)^i\right), \tag{2}$$

with

$$L\left(W, \left(Y, X_1, X_2\right)^i\right) = \left(1 - Y\right) L_S\left(D_W^i\right) + Y L_D\left(D_W^i\right) \tag{3}$$

where $\left(Y, X_1, X_2\right)^i$ is the i-th sample composed of a pair of constellations X_1, X_2 and a binary label. $L_s = \frac{1}{2}\left(D_W\right)^2$ is a partial loss function for a pair of similar constellations and $L_D = \frac{1}{2}\left(\max\left\{0, margin - D_W\right\}\right)^2$ is a partial loss function for a pair of dissimilar constellations.

As shown in Fig. 2, a constellation is fed to the *constellation embedding module* as a high dimensional vector. We process binary descriptor (512 dimensions for FREAK), scale and orientation (2 dimensions) of the central descriptor. For each neighbourhood keypoint, we use its binary feature descriptor (512 dimensions) and relative position, scale and orientation (4 dimensions). For a constellation consisting of the central descriptor and its 20 nearest neighbours, this gives $514 + 20 \times 516 = 10834$ dimensions. We designed the *constellation embedding module* (see bottom of Fig. 2) in a modular fashion. The design was based on an extensive series of experiments to help us identify the best architecture of each component. The best performing architecture is described below. The twin *constellation embedding module* first computes k 32-dimensional embeddings of k neighbour binary descriptors. The resulting embeddings are concatenated with k neighbours relative position, relative orientation and relative scale with respect to the central keypoint. This gives a sequence of k 36-dimensional vectors which are further processed by RNN (recurrent neural network) module producing 32-dimensional neighbourhood representation. Then neighbourhood representation

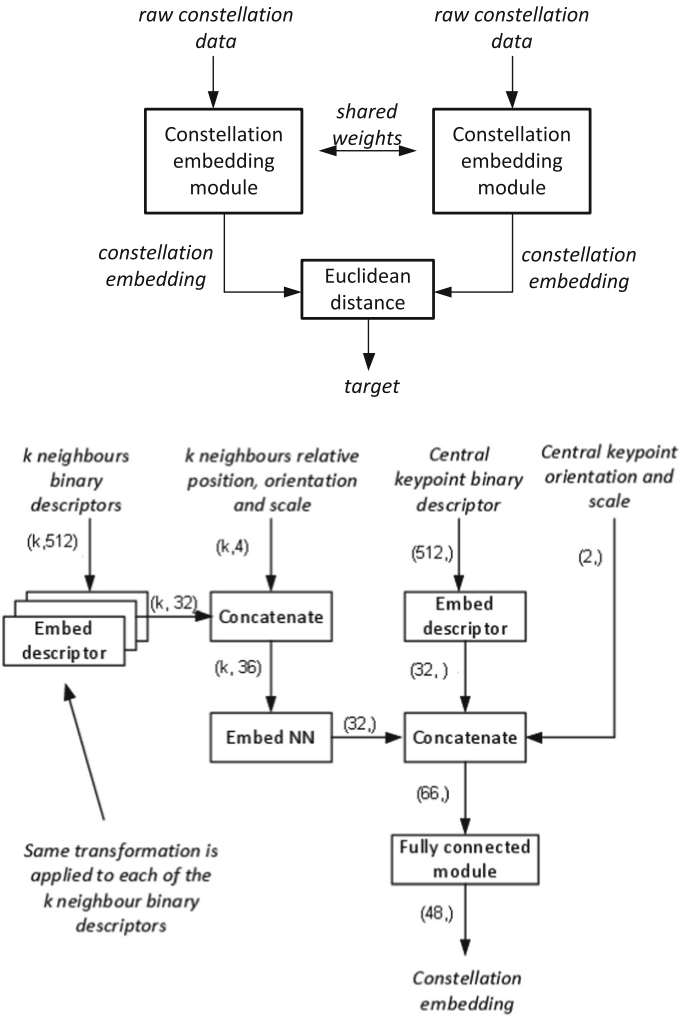

Fig. 2. (Top) High level architecture of the Siamese neural network computing constellation embeddings. (Bottom) Architecture of a constellation embedding module of a Siamese neural network. We feed into the network information on central keypoint and its nearest, in Euclidean distance sense, neighbourhood keypoints on the image. Central keypoint FREAK descriptor, orientation and scale are used along with nearest neighbours, in Euclidean distance sense, informations. That include FREAK descriptors, their relative positions, orientations and scales with respect to the central keypoint. Hence, our resulting SConE descriptor of the central keypoint offers better performance of descriptor matching at a lower computational cost than competing descriptors.

is concatenated with the central keypoint binary descriptor, orientation and scale resulting in 66-dimensional vector. This is processed by a final fully connected module resulting in the 48-dimensional constellation embedding called SConE

Table 1. Components of a twin constellation embedding module from Fig. 2. Scaled Exponential Liner Unit (SELU) [23] activation is used after each fully-connected layer.

Component name	Details
Embed descriptor	3 fully-connected layers with 512/256/32 units
Embed NN	2 layer bidirectional LSTM [24] with 32/32 units followed by 3 fully-connected layers with 64/64/32 units
Fully-connected module	3 fully-connected layers with 64/64/48 units

for Siamese Constellation Embedding descriptor. Details of each component are given in Table 1. The size of the final embedding (48 real values) was chosen as a compromise between the descriptor discriminative power and the storage requirements.

Siamese neural network training is conducted using data acquired with structure-from-motion solution embedded in a Google Tango tablet. The device produces datasets containing keypoints and feature descriptors detected on the recorded video sequences. Camera poses and scene structure, in the form of sparse 3D point sets, are reconstructed using reliable structure-from-motion techniques. The training set was constructed by concatenating samples from multiple video sequences. It consists of almost 10 thousand keyframes with over 4 million FREAK descriptors linked with 259 thousand landmarks. The validation sequence, used to measure the performance of the trained networks, contains almost 5 thousand keyframes with over 2 million feature descriptors linked with 120 thousand landmarks. In both sequences, almost half of the feature descriptors are linked with reconstructed 3D scene points (landmarks) whereas the rest of them is not linked with any landmark.

We experimentally choose the number of neighbours used to form the constellation, that produces the most discriminative SConE descriptor. This is done by training the Siamese network multiple times using constellations of various size and evaluating the performance of the trained network. We use nearest neighbour search precision on the embeddings of the validation set as the performance measure. The precision is calculated as follows. First, embeddings of validation set elements are calculated using the constellation embedding module of the trained Siamese network. Then 10 thousand embeddings is randomly chosen from the validation set. For each sampled embedding, its nearest neighbour in embedding space (that is in Euclidean space, as embeddings are real-valued vectors) is found. If the nearest neighbour is linked with the same 3D scene point (landmark) as the sampled element we declare a match. *Precision* is calculated as the percentage of correct matches. The results are depicted on Fig. 3. As the number of neighbours increases the precision grows, to reach a maximum for 20 neighbours. Compared to using raw FREAK descriptors we get increase of nearest neighbour search precision from 0.807 to 0.851 on our validation set.

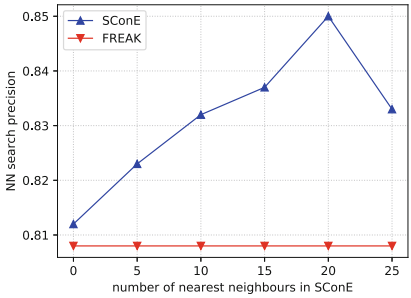

Fig. 3. The influence of k nearest neighbours on the nearest neighbour search precision in the validation set. The best results are achieved when 20 neighbourhood keypoints are used to form a constellation.

4 Evaluation

This section describes evaluation procedure and its results. The evaluation dataset is described in Sect. 4.1. In Sect. 4.2 we present our evaluation protocol. Finally, in Sect. 4.3 we show the results of our evaluations.

4.1 Dataset

For the evaluation procedure, we use a challenging TUM dataset [11], often used in other works to compare descriptors' performance [20]. TUM is a large dataset with sequences recorded using Microsoft Kinect sensor and we choose seven of them for evaluation: `fr1/plant`, `fr2/dishes`, `fr2/metallic_sphere2`, `ft3/cabinet`, `fr3/large_cabinet`, `fr2/flowerbouquet` and `fr3/teddy`. Sample sequences can be seen in top of Fig. 4. In addition to images, sensor ground-truth trajectory and depth-maps are provided. We divide each sequence into 100 long subsequences. First frame is treated as a reference and 99 others are used for matching. Bottom of Fig. 4 presents four different frames from a sequence. They differ significantly, the last one being rotated almost 360°. It makes it hard for matchers to find any correct matches between first and last frame in such scenario.

Due to a lot of blurred images and changes in lighting, the TUM dataset is considered to be rather challenging. Additional difficulty comes from the fact that color images, depth maps and camera positions are not perfectly consistent. They were collected in different moments of time, so timestamps cannot be perfectly aligned and to address this problem we approximate them to minimize the time gaps between frames. Furthermore, due to the limitation of the capturing device, a large portion of depth maps does not provide correct depth values, especially on the edges of objects where a large portion of keypoints is detected. Hence we use an epipolar geometry condition and ground truth camera poses to verify correctness of a match.

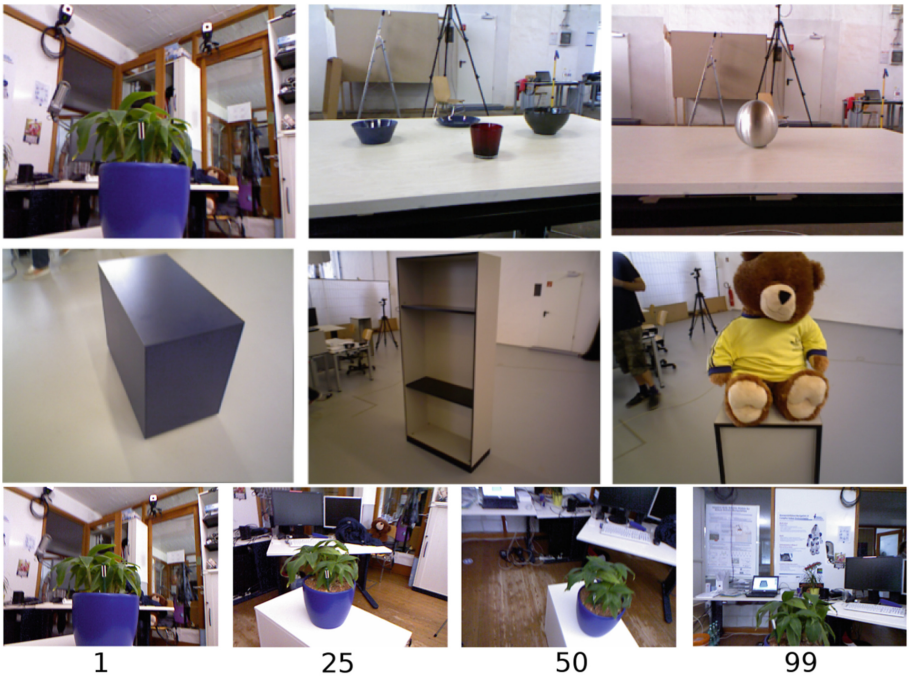

Fig. 4. (Top) Exemplary images from TUM [11] video sequences used in evaluation of our method. (Bottom) 1st, 25th, 50th and 99th keyframes from one test sequence (`fr1/plant`). There's a large viewpoint variation in frames forming one sequence.

4.2 Evaluation Procedure

We test our descriptor in a demanding scenario of a real application, strictly connected with SLAM and Structure from Motion pipelines. We use TUM's ground truth presented as a trajectory and calibration data for each camera in the set. The TUM dataset is specifically designed to evaluate Structure from Motion algorithms and therefore its frame resolution is low and graphical content is often lacking the details necessary to track dense feature sets. Nevertheless, such characteristics create a demanding benchmark for camera pose estimation and we therefore use it in our evaluation.

We compare our method to the state-of-the-art methods for image matching: FREAK [5], SURF [6], SIFT [4], ORB [7] (implementations comes from OpenCV [25] package), GMS [20] and embeddings calculated by our custom artificial neural network (SConE). For SConE, we first use FAST [26] key point detection algorithm. Then we compute FREAK feature descriptors at detected keypoints. SConE descriptors are calculated using the constellation embedding module of the trained Siamese network. The network training is performed as described in Sect. 3. For each 100 frame subsequence from the evaluated TUM sequence, we compute matches between the first frame and all others, resulting in 99 image

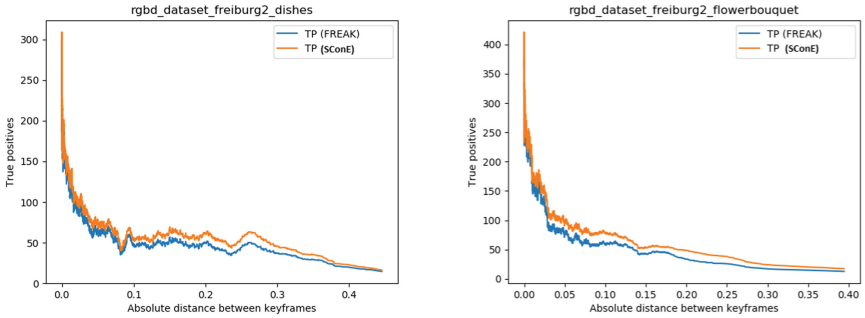

Fig. 5. Number of true positives in matching feature descriptors between a pair of images as a function of an angular distance between keyframes. Results on `fr2/dishes` (left) `fr2/flowerbouquet` (right) sequences from TUM [11] dataset are presented. SConE consistently yields better results than FREAK descriptor due to encoding additional information about the neighbourhood keypoints.

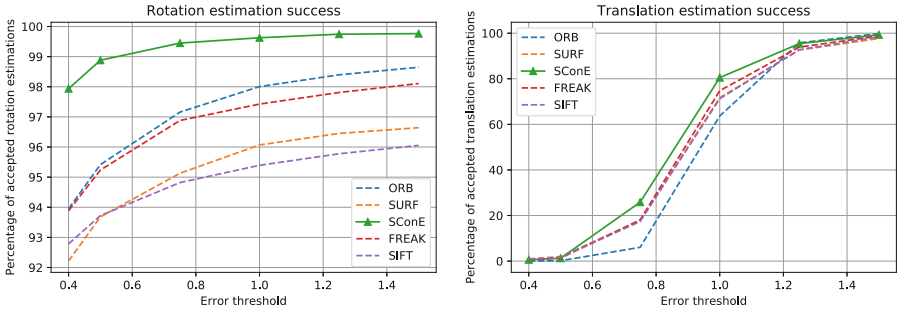

Fig. 6. Pose estimation errors on TUM [11] dataset for SConE and competing descriptors. SConE outperforms the state-of-the-art descriptors, including SIFT and SURF, across all error thresholds.

pair matches. For each image pair we find pairs of corresponding features with a brute-force approach. Matches are then filtered using standard ratio-test.

We compute essential matrix from the key point correspondences for each image pair. From them we estimate relative camera pose for each pair of images in form of a rotation matrix and translation vector. We use the OpenCV [25] implementation with RANSAC for this purpose.

RANSAC is needed in the process because of two reasons. The first is its ability to filter out matches considered good given a 3D model, but giving perturbations in affine transformation estimation. This situation happens when the scene contains moving or deforming objects. The second factor is connected with a level of locality in SConE. SConE makes use of constellation, incorporating structural data of a bigger area than the base descriptor itself, but still is considered as a local descriptor. If duplicate elements of the scene appear, SConE is prone to generate bad matches. Feature duplicates can be seen in various real

case scenarios where textures contain patterns or multiple features of the same appearance, for instance in windows or buildings' facades.

We compare the relative pose recovered using abovementioned procedure against ground truth and calculate qualitative metric for each image pair. The metric is presented as an error in translation and rotation estimation, calculated according to the procedure described in the Odometry Development Kit from KITTI benchmark [27]. KITTI benchmark defines a method of error calculation for 3D tracks with six degrees of freedom with asynchronous sampling. In our case the data is synchronized, so the formulas are straightforward. Error in translation is calculated as a translation vector difference in 3D. Error for rotation is calculated from relative 3×3 rotation matrix dR according to the formula:

$$d = \frac{tr\left(dR\right) - 1.0}{2} \tag{4}$$

$$R_{err} = acos\left(\max\left(\min\left(d, 1.0\right), -1.0\right)\right) \tag{5}$$

In addition to GMS and basic matchers, we use DeepMatching (DM [28]), Bilateral Functions Matching (BM [29]) and Bounded Distortion (BD [30]) as state-of-the-art image matchers. We use original implementations of its authors, so its computational efficiency may be considered as far from optimal. Where possible, we use only one computing thread for better comparison.

4.3 Results

We analyse our results using pose estimation errors obtained for various descriptors. Figure 6 shows the results of this experiment. SConE outperforms all basic features with ratio tests. The performance gap is substantial, and shows gain over basic FREAK descriptor. SConE uses FREAK keypoints as a base descriptor for learning, thus it has the same keypoints pool before filtering stage. This characteristic lets us build simple comparison based solely on true positives after ratio test. Figure 5 shows number of true positives on FREAK keypoint locations, using both FREAK descriptors and SConE embeddings. The X-axis contains absolute distance between frames calculated as the difference between quaternion rotations for each camera position.

SConE gives very good results in comparison with advanced matching methods, as shown in Fig. 7. All of the descriptors give very similar results in translation estimation. Rotation estimation is much more prone to keypoint localization perturbations, thus shows more variance between methods. Our approach outperforms GMS with its default keypoints pool (10 000 ORB keypoints).

Furthermore, we evaluate the computational complexity of our proposed SConE descriptor-based matching and compare it with the competing methods. We measure both descriptor extraction and matching times. For raw descriptors we use brute force matcher. Figure 8 shows the results of this comparison. SConE adds very little overhead to FREAK computation, which is used as base. It's much faster then GMS, while obtaining better results.

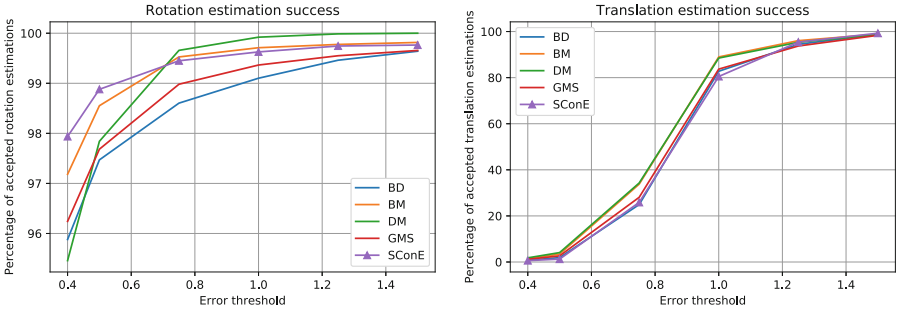

Fig. 7. Pose estimation errors on TUM dataset for SConE and significantly more complex matching procedures. Performance of SConE is au pair with much more advanced keypoint matching procedures.

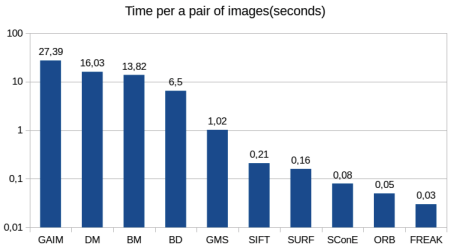

Fig. 8. Descriptor extraction and matching time for matching descriptors between a pair of images. SConE offers very competitive performance compared to more sophisticated matching methods.

5 Conclusions

In this paper, we propose a novel low-dimensional feature descriptor that incorporates geometrical information about the layout of neighbouring keypoints. This way we are able to reduce the importance of additional post-processing step that typically aims at filtering out incorrect matches. To train our descriptor we use Siamese neural network architecture and feed it with central keypoint descriptor, as well as neighbouring keypoints and their descriptors, relative position, orientation and scale. Although our framework is agnostic to descriptor type, we use as our base descriptor FREAK and show that the SConE descriptor generated by our neural network outperforms competitors on a challenging TUM dataset.

Acknowledgement. This research was supported by Google Sponsor Research Agreement under the project "Efficient visual localization on mobile devices".

The Titan X Pascal used for this research was donated by the NVIDIA Corporation.

References

1. Agarwal, S., et al.: Building Rome in a day. Commun. ACM **54**(10), 105–112 (2011)
2. Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. IJCV **74**(1), 59–73 (2007)
3. Lynen, S., Sattler, T., Bosse, M., Hesch, J.A., Pollefeys, M., Siegwart, R.: Get out of my lab: large-scale, real-time visual-inertial localization. In: Robotics: Science and Systems (2015)
4. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
5. Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: fast retina keypoint. In: CVPR (2012)
6. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
7. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to sift or surf. In: ICCV (2011)
8. Trzcinski, T., Christoudias, M., Lepetit, V., Fua, P.: Boosting binary keypoint descriptors. In: CVPR (2013)
9. Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: ICCV (2015)
10. Yi, K.M., Trulls, E., Lepetit, V., Fua, P.: LIFT: learned invariant feature transform. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 467–483. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_28
11. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IROS (2012)
12. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: MatchNet: unifying feature and metric learning for patch-based matching. In: CVPR (2015)
13. Loquercio, A., Dymczyk, M., Zeisl, B., Lynen, S., Gilitschenski, I., Siegwart, R.: Efficient descriptor learning for large scale localization. In: ICRA (2017)
14. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML 2016, vol. 48, pp. 2014–2023. JMLR.org (2016)
15. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS 2016, USA, pp. 3844–3852. Curran Associates, Inc. (2016)
16. Muja, M., Lowe, D.G.: Fast matching of binary features. In: Computer and Robot Vision (CRV) (2012)
17. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. TPAMI **36**(11), 2227–2240 (2014)
18. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)
19. Raguram, R., Chum, O., Pollefeys, M., Matas, J., Frahm, J.M.: USAC: a universal framework for random sample consensus. TPAMI **35**(8), 2022–2038 (2013)

20. Bian, J., Lin, W.Y., Matsushita, Y., Yeung, S.K., Nguyen, T.D., Cheng, M.M.: GMS: grid-based motion statistics for fast, ultra-robust feature correspondence. In: CVPR (2017)
21. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: NIPS (1993)
22. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: CVPR (2006)
23. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30, pp. 971–980. Curran Associates, Inc. (2017)
24. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
25. https://opencv.org/
26. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 430–443. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_34
27. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: CVPR (2012)
28. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: DeepFlow: large displacement optical flow with deep matching. In: ICCV (2013)
29. Lin, W.-Y.D., Cheng, M.-M., Lu, J., Yang, H., Do, M.N., Torr, P.: Bilateral functions for global motion modeling. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 341–356. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_23
30. Lipman, Y., Yagev, S., Poranne, R., Jacobs, D.W., Basri, R.: Feature matching with bounded distortion. ACM Trans. Graph. **33**(3), 26:1–26:14 (2014)