2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Macau, China, November 4-8, 2019

DLD: A Deep Learning Based Line Descriptor for Line Feature Matching

Manuel Lange, Fabian Schweinfurth and Andreas Schilling

***Abstract*— In this paper, we present an appearance based line descriptor which was developed with the help of machine learning. Our descriptor uses a ResNet which we modified in its size to improve the performance. We utilized the Unreal Engine and multiple scenes provided for it to create training data. The training was performed using a triplet loss, where the loss of the network is calculated with triplets each consisting of three lines including a matching pair and another non-matching line. During learning, the goal of the minimization function is to calculate descriptors with minimal descriptor distance to matching lines' descriptors and maximal descriptor distance to other lines' descriptors. We evaluate the performance of our descriptor on our synthetic datasets, on real-world stereo images from the Middlebury Stereo Dataset and on a benchmark for line segment matching. The results show that in comparison to state-of-the-art line descriptors our method achieves a greatly improved line matching accuracy.**

I. INTRODUCTION

Descriptors in general are used in various applications. For instance Terrain Classification, Stereo and Feature Matching, Place Recognition and Face Recognition. One important utilization is to rerecognize features. There are various detectors and descriptors for point features [1] [2] [3]. Learning based point descriptors have shown an improved performance compared with the established analytic descriptors [4][5][6].

Line features came into focus just a couple of years ago. They require more processing power to be extracted than most point features do. There are multiple line detection algorithms by now. The Hough Line Transform [7], the Line Segment Detector (LSD) [8], the EDLine detector (EDL) [9] and the fast line detector (FLD) [10] are well established line detection algorithms.

Early line descriptors placed point descriptors along the line [11], or used histogram based point descriptor alike descriptors along and around the line [12] [13]. Clearly, this does not adapt well to the characteristics of lines, which we will elaborate soon. There are also methods that incorporate geometric constraints for line matching: Kwon et al. [14] used line/point duality and a randomized strategy for matching. Wang et al. [15] clustered line segments into groups, calling them 'line signatures', and matched them. Li et al. [16] implemented a line segment pair based hierarchical method for line segment matching. Those geometric methods have been evaluated by Zhang [17]. A drawback is their high requirement of calculation power making them very slow. Also, they are not just descriptors, but rather special line matching algorithms.

We will focus on appearance based line descriptors from here on. The mean standard-deviation line descriptor (MSLD) proposed by Wang et al. [18] is one of the most

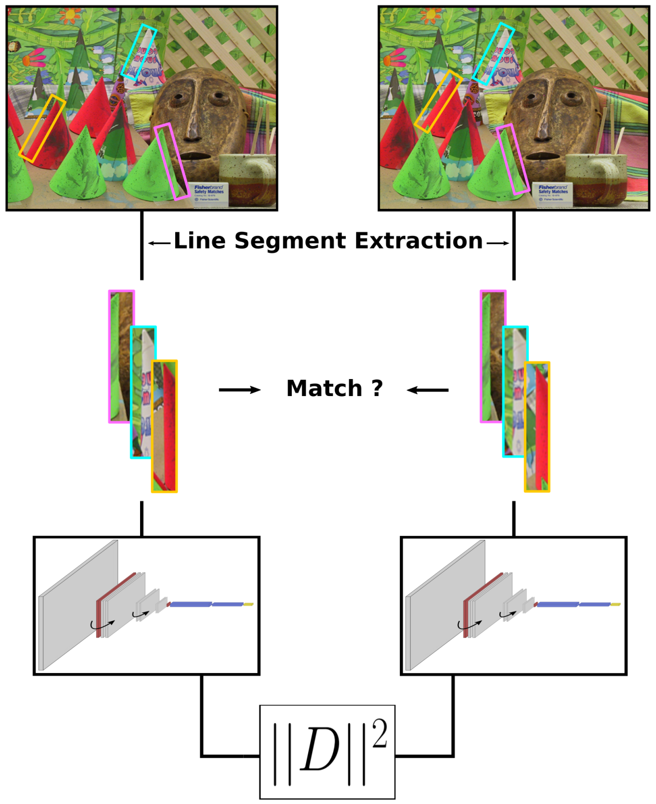


Fig. 1. Basic line segment matching procedure. First, line segments are detected on the input images. Then, descriptors are computed by our DLD network. Finally, these descriptors can be matched by calculating the squared euclidean distance.

compared to, and cited appearance based line descriptors. Lilian Zhang evaluated the MSLD and his Line Band Descriptor (LBD)[19] in his dissertation [17] showing that the LBD performs faster with more correct and less false matches. Inspired by the ORB point descriptor Zhuang et al. developed *A Binary Robust Line Descriptor*[20]. They achieved to be faster than the LBD. Yet, they were not as accurate. The LBD is the only line descriptor that made it into OpenCV [21]. We chose the LBD in our experiments to compare our results with, as to our knowledge up to now no other line descriptor yields better results in terms of matching accuracy.

A drawback of descriptor computing methods like the MSLD and the LBD is that they lose much of the structural information around the line by taking (weighted) means along the line. Due to recent years findings, we argue that this is a task for which deep neural networks are well suited.

Patch matching can also be noted here; it has been researched thoroughly [22][23][24][25][26]. A survey of image matching is given in [27]. One could think that appearance based line matching is similar to patch matching. But in fact, lines are a certain type of feature with special characteristics. A line can lay on the edge of an object with the object itself being on one side of the line and something else being on the other side of the line. In this specific case it's important that the latter lies in the background. Since this line, depending on the perspective, separates the object and the background, we will call it an edge line. Such edge line stays the same line even if the side showing the background changes. Additionally, areas closer to the line can be more important than those further away. One should keep in mind that a line extends in one direction which differs for regular patches. Those special characteristics distinguish lines from patches. Furthermore, most patch matching methods use the whole image patch, whereas we reduce the line to a small 256-Bit descriptor, similar as it is done for the LBD.

For the sake of completeness we want to mention a descriptor that is based on learning like our descriptor [28] which was published shortly after our first submission. Although one of the reviewers made us aware of this publication, unfortunately it was not possible to include it in our comparison due to time constraints.

The Unreal Engine was utilized to create training data. We chose to use four different scenes to obtain data from different environments: *Scifi Hallway*, *Epic Zen Garden*, *Infinity Blade: Grass Lands* and a *living room* from the unreal engine showcase realistic rendering example. From these scenes, we took 18842 stereo images including the 3D-Map for each image. Based on the 3D coordinate information, we matched the line segments from the left and right images. To deal with the uncertainty of the line detection, we employed a RANSAC process to improve the quality of our ground truth matching data.

We chose to train a ResNet [29], a well trainable standard architecture, on our matching problem. Regular ResNet comes in different depths, starting with size 18 upto size 152. After some tests, we found that the smallest ResNet with size 18 works best. As this is the smallest size, we wanted to know if an even smaller net would also work. Thus, we created a ResNet with size 10. We achieved a result which is as accurate as the size 18 ResNet but faster due to the reduced size. We also tried a DenseNet [30] but did not achieve a similarly well performance using it.

Our error function uses a triplet loss which was originally developed by Weinberg and Saul [31] and got popular through Schroff et al. [32]. Each triplet consists of three lines, the first line is the one which the other two lines are compared to by descriptor distance. The net trains to calculate the descriptor in a way that minimizes the distance between the matching pair and maximises the distance of the first line's descriptor to the false line's descriptor. Our main contributions are as follows:

- We present a new learning based line descriptor which we compare to the state-of-the-art, showing an advancement especially in terms of the matching accuracy.
- We elaborate how to generate huge amounts of learning data for line segment matching and how to use it for training a network with triplet loss.

An overview of the matching procedure is drawn in Fig. 1. Section II gives a detailed explanation of our data generation process. In Section III we describe how the learning was conducted and in IV the results are presented. In Section V we draw a conclusion of our proposed method.

II. Data Generation

Machine Learning in general requires a lot of data to learn from. A difficulty when training a network for line matching is the lack of suitable datasets. Established machine learning datasets for computer vision as the Cityscapes Dataset [33] and the KITTI Vision Benchmark Suite[34] do not come with the necessary information. Regular images are not enough, as the correspondences between lines have to be known. Thus, we chose to create our own data with the necessary information utilizing the Unreal Engine.

A. Unreal Engine 4 - Ground truth data generation

The Unreal Engine 4 (UE) comes with a realistic rendering engine and many freely available sceneries. From those scenes, we chose four quite different ones:
- The *Epic Zen Garden (EZG)* shows an outdoor modern garden area including some trees, a pool and multiple structured things like outdoor furniture, tiles and the house.
- The *Infinity Blade: Grass Lands (IB:GL)* is a very rocky wide open scenery including a path mostly made of stone with the see beneath it.
- The *Scifi Hallway (ScHw)* shows a science fiction hallway with many details like pipes and other tech things on the walls and standing around.
- The *living room (LvR)* from the unreal engine showcase realistic rendering example shows a basic living room with a table, a couch, vases, a plant and pictures as well as a TV on the wall.
We chose those diverse scenes to provide the learning process with a wide variety of different examples. To extract the images and the corresponding 3D-Coordinate-Maps, we defined long splines through the scenes as a path for the camera. Following this path we stepped slowly through the scenes recording data at each step. In total 37689 steps were recorded (12068 *(EZG)*, 17875 *(IB:GL)*, 5328 *(ScHw)*, 2418 *(LvR)*) with about 11.9 Million lines overall.

Our problem is to acquire large quantities of data with ground truth information for our training on line segment matching. To solve this, we extract image pairs from the UE that contain a lot of matching lines, visible in both images. However it is difficult to identify the ground truth matches. We therefore use the Z-Buffer information from the UE for this identification.

First, we start by detecting 2D line segments with the EDLine detector. Then, we use the extracted 3D-Map to unproject the detected 2D line segments to 3D line segments in the observed scene. Having this 3D information, we

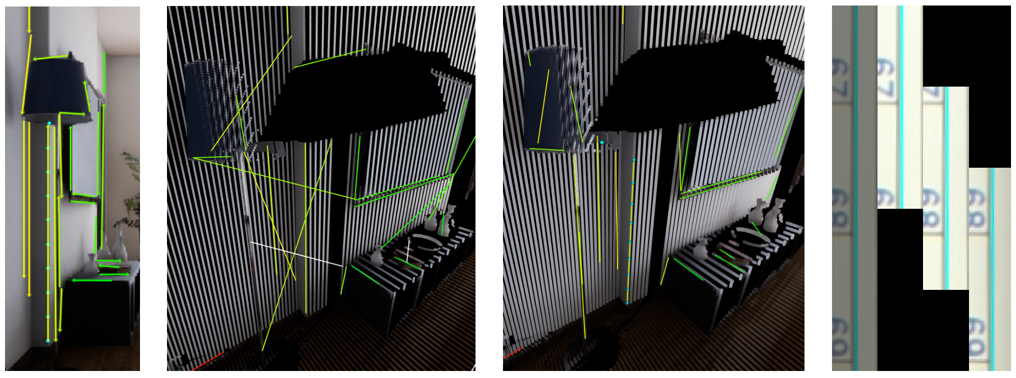


Fig. 2. Figure (a) shows the camera view from which it is not possible to see if the lines' depth is correct. Figure (b) and (c) show the 3D scene from the side. One can see that in Figure (b) the 3D line segments are quite skewed. Figure (c) shows the same scene but with the 3D line segments corrected by the RANSAC scheme. Figure (d) shows a visual illustration of a line segment which is split in three parts (cutouts).

project the 3D line segments to 2D line segments into the second stereo image in which they are matched. There, the matching is done in 2D. To be a valid match, two lines have to have a similar angle, have to overlap in their length sideways (in respect to their direction) and have to be close to each other.

Unfortunately, when unprojecting a 2D line segment the resulting 3D line segment is occasionally inaccurate. The problem here is that the 2D segments from the line detection are not always exact. When the segment is an edge line (see Section I), it happens that part of it reaches over the edge. In this case the underlying 3D coordinates are wrong as they come from the background. This results in an inaccurate 3D segment sometimes pointing deep into the scene. To deal with such problems, we employed a RANSAC [35] scheme: We sample ten points evenly distributed on the 2D line segment. Of those points, we get the 3D coordinates from the map to obtain 3D points. Then pairs of two are selected to calculate an infinite 3D line model and the other points are used to validate the line model. If enough of the other points are close, then the model is accepted. The distance of the other points is also used as a measure of quality of the model. Using this scheme improves the 3D lines and, thereby, the quality of the line matches. The improvement is depicted in Fig. 2.

III. Learning

ResNet has demonstrated a great performance on various benchmarks [29]. It also solved the problem of deeper networks performing worse than their smaller counterparts by introducing so called *identity shortcut connections*. For these reasons, we chose this architecture for our line matching problem.

During our first experiments with ResNets in different sizes, we quickly discovered that the network does not have to be as deep as 18-layers to solve our problem. The smallest ResNet described by Kaiming He et al. is an 18-layer version. We created a 10-layer version by halving the number of the 3×3 convolutions in the middle of the network which still performed similarly well.

To enable the network to train a solution on our line matching problem, we had to adjust the data to make it suitable to be provided to the network. As lines are defined by gradients and gradients have a fixed direction going from dark to bright, the line detectors use this gradient direction to assign lines with a fixed direction. We use this direction to rotate the lines facing upwards and as the direction of a line is consistent from one image to another, the lines always face the same direction. While training the network we found out, that the training error reduces to a considerably lower value and the results improve, when we use line segments of a fixed length. In the experiments, we show that it still performs the matching well on line segments with variable length; even when trained on fixed length line segments. To obtain fixed length segments, one option is to scale the line segments. But that usually means interpolation, which implies a loss in information when scaling down. Thus, we chose a different approach which splits the line segment in up to three parts, depending on its length.

We call the parts cutouts and determine their number according to the following scheme:

$$\#\text{cutouts} = \begin{cases} 1 & \text{if } \frac{l}{t} \leq 1.2 \\ 3 & \text{if } \frac{l}{t} \geq 1.4 \\ 2 & \text{else} \end{cases} \tag{1}$$

where l is the length of the line segment and t is the desired fixed length (in most of our experiments, we set t to $120\,px$). One has to keep in mind that the matching is appearance based. Thus we made sure, that even with three cutouts the two which are furthest apart from each other still have a slight overlap, as it can be seen in Fig. 2(d). The cutouts have labels which assign them to the line segment which they originate from. This information is used to recover which lines are matched to each other.

One more thing to note is the setup of the line extraction. Line extractors usually have multiple parameters which influence the line length directly, e.g. the minimum line length,

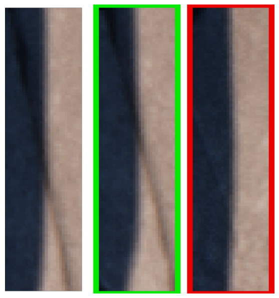


Fig. 3. Visual illustration of a triplet matching sample.

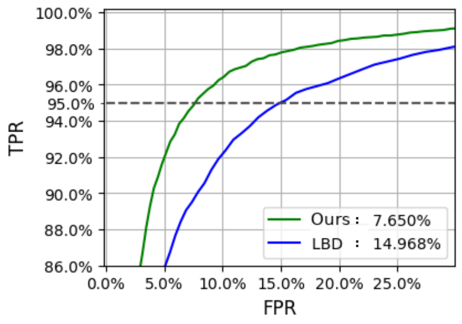


Fig. 4. This plot shows the receiver operating characteristic curves (ROC-Curve) of our method vs the LBD over all of our eight labeled Middlebury Images [36]. The legend shows the False Positive Rate (FPR) at the True Positive Rate (TPR) of 95% (dashed line). An advancement of our method is clearly visible.

or indirectly, e.g. the curvature setting which determines at which curvature a segment is split in two segments. At first glance, it might seem that when the same image is used at a higher resolution, the same lines would be detected and reach over more pixels, but this is often not the case. The additional pixels lead to different detections; as for example curvatures now include more pixels than before and some lines may be detected in the surrounding, which also have an influence on the detection (due to the often used near field analysis which suppresses too many lines being detected in noisy or gradient rich areas, e.g. on a tree). Therefore, the resolution does not influence the line segment length and matching very much; the main factor is the choice of the line segment detector and its parameter setup.

To avoid that the network specializes on one scene, it is important that the examples are provided randomly during the training phase.

A. *Triplet loss*

We chose to use the triplet loss for our descriptor learning, as this technique yielded good results for *Nearest-Neighbour-Classification* [31][32]. The idea using the triplet loss on line segments is as follows: For a given line segment $s_i^a \in \mathbb{S}$ two line segments s_i^+ (positive) and s_i^- (negative) are selected. For their labels $l \in \mathbb{L}$, that hold the information which lines match, applies $l_i^a = l_i^+$ and $l_i^a \neq l_i^-$, which means that one segment is a match and the other is not (see Fig. 3). When calculating the descriptors for each segment, (called *embedding* here) using the function $e(s)$, their descriptor distance is given by the squared euclidean distance. Additionally we enforce a margin δ in the descriptor distance between matching segments and non-matching ones which leads us to:

$$dist\big(e(s_i^a),\ e(s_i^+)\big) + \delta < dist\big(e(s_i^a),\ e(s_i^-)\big) \tag{2}$$

This triplet loss function has the advantage of not only trying to bring close embeddings together, but it pushes non-matching embeddings far apart from each other at the same time.

Many triplets will fulfil Eq. 2 and not encourage a fast convergence. Therefore it is beneficial to select triplets that are difficult, i.e. those which violate Eq. 2. Thus, we use the margin δ to make sure that we get difficult triplets, which are still representative. Therefore the loss function L to be minimized can be written as:

$$\begin{aligned} L = max([&dist\big(e(s_i^a),\ e(s_i^+)\big) \\ &- dist\big(e(s_i^a),\ e(s_i^-)\big) + \delta],\ 0) \end{aligned} \tag{3}$$

Triplets can be generated online or offline. Generating them offline, e.g. based on the most recent network checkpoint would have the advantage to select from a greater amount of segments. But it also has the disadvantage that the triplets are not always based on the most recent weights in the network. Also Schroff et al.[32] found it inconclusive whether doing it offline is beneficial. We chose the online generation which yields triplets based on the current mini-batch using the most recent weights.

IV. Experimental Results

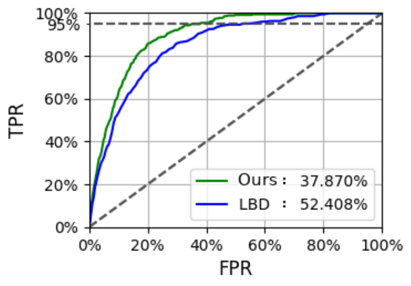
In this section, we evaluate the performance of our descriptor by comparison to the state-of-the-art LBD [19] descriptor. To validate that our descriptor yields good results with real world data, even while trained on synthetic data, we included some real world images from the Middlebury Stereo Dataset [36] and some self shot images. Our results on the Middlebury images are discussed in the next paragraph and are shown in Fig. 4 and in Table I. Furthermore we wanted to evaluate our method with arbitrary camera poses and under difficult conditions. Therefore we took more images, for which the results are shown in Fig. 5.

The EDLine detector [9] was used for the line detection with its standard parameter settings. For a fair comparison, the size of the segments was the same for our method as for the LBD. During training we used stochastic gradient descent as optimizer and a δ margin value of 5 in the triplet loss. The training took approximately one day on a GeForce GTX 1080. During testing, the calculation of one descriptor on the CPU (Core i7 3.4 GHz) takes about one millisecond.

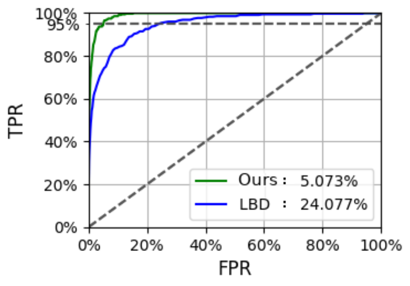
1) Middlebury evaluation: We took the stereo image pairs from eight scenes of the Middlebury Dataset, ran the line segment detection and labeled the line segments to obtain ground truth matches. The receiver operating

TABLE I
PERFORMANCE EVALUATION ON IMAGES OF THE MIDDLEBURY STEREO DATASET (2014) [36]

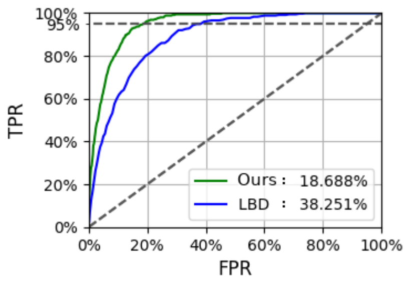
Method	DLD vs LBD: False Positive Rate at a True Positive Rate of 95%; less is better							
	Adirondack	Backpack	Bicycle	Classroom	Couch	Flowers	Piano	Vintage
DLD (Ours)	**4.948%**	**7.653%**	**5.785%**	**4.815%**	**7.531%**	**11.771%**	**4.552%**	**5.456%**
LBD	18.869%	15.198%	14.157%	11.165%	20.478%	62.963%	10.39%	19.739%
Labeled Matches:	318	1216	944	500	296	143	353	517



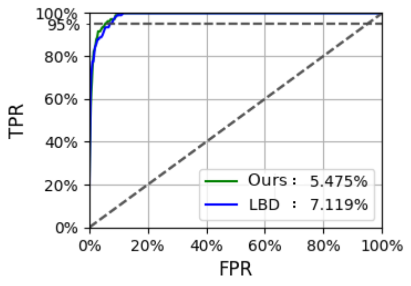
(a)



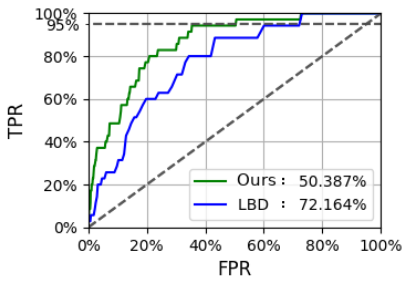
(b)



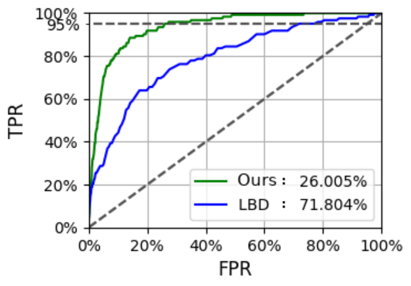
(c)



(d)



(e)



(f)

Fig. 5. The plots (a) to (f) show the ROC-Curves for the following difficult conditions: (a) strong blurring (351 Ground Truth (GT) matches), (b) high jpeg compression (351 GT matches), (c) intense noise (351 GT matches), (d) about 45° rotation (106 GT matches), (e) scale change (35 GT matches) and (f) viewpoint change (122 GT matches). TPR is the true positive rate and FPR is the false positive rate. The greater the area under the curve, the better.

characteristic curves (ROC-Curves) including all matches from all eight images are plotted in Fig. 4. We can see that our method is a great advancement in comparison to the LBD. We did not plot the ROC curves to all eight labeled scenes, as they are all similar to the overall ROC curves. But the percentage of false matches at 95% of true matches is listed in Table I. This table also lists the number of matches per scene. We can also see here, that our method does better in all scenes.

2) Difficult condition scenes: We hand labeled some real world images showing difficult situations for further evaluation of our method compared to the LBD. The results are depicted in multiple ROC curves in Fig. 5. All of the subplots contain a legend which shows the methods false positive rate (FPR) at the true positive rate (TPR) of 95% (less is better). The first plot, Fig. 5(a), demonstrates the performance when strong blur is on one of the two images. We can see that it is difficult for both methods, but ours can handle it better. In Fig. 5(b) a high jpeg compression rate is chosen for one image which is not very difficult for our method, but affects the LBD to a considerable amount. Intense noise is added to the image related to in Fig. 5(c) which impacts both methods negatively, but our method is affected only about half as much as the LBD. We rotated one image about 45° which barely has an effect on both methods, see Fig. 5(d). That was expected, as both methods use a region along the line, making them independent to global rotations. Fig. 5(e) shows the ROC curves when a strong scale change took place. Both methods are quite affected. We did not augment our training data especially for scale change, still our method is somewhat less affected. The results to a viewpoint change of over a couple of meters and simultaneously more than a 45° change in the viewing direction are plotted in Fig. 5(f). We can see that our approach performs more than twice as well as the LBD. This is an important finding, as wide baseline matching is a difficult yet important task for applications like 3D reconstruction from arbitrary poses or relocalization for a lost robot.

V. CONCLUSIONS

This paper presents a new deep learning based line segment matching approach. We elaborated our ground truth data generation using the Unreal Engine 4 and multiple different scenarios thereof to obtain a wide variety of samples. Using these samples, we trained a ResNet and explained how to use the triplet loss to train line segment descriptors for descriptor matching. In the results, we demonstrated that our method is an advancement compared to the state-of-the-art LBD descriptor. We showed this on our artificial datasets as well as on images from the Middlebury Dataset and on additional images showing blurring, jpeg compression, noise, rotation, scale change and viewpoint change.

We plan to set up a website on which one can upload an image pair and have a line detection and our matching run on it. Also the network will be downloadable for everyone's usage to support the community[1].

In the future, we want to tackle the problem of training on line segments with variable length. Therefore we are planning to test more hyperparameter settings and try other architectures, especially ResNeXt [37] as it contains multiple branches which help dealing with features of various sizes because it uses multiple different filters in parallel.

References

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision (ECCV)*. Springer, 2006, pp. 404–417.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571.

[4] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT," *arXiv preprint arXiv:1405.5769*, 2014.

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

[6] K. Simonyan, A. Vedaldi, and A. Zisserman, "Learning local feature descriptors using convex optimisation." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1573–1585, 2014.

[7] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, Tech. Rep., 1971.

[8] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: a Line Segment Detector," *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[9] C. Akinlar and C. Topal, "EDLines: Real-time line segment detection by edge drawing (ED)," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 2837–2840.

[10] J. H. Lee, S. Lee, G. Zhang, J. Lim, W. K. Chung, and I. H. Suh, "Outdoor place recognition in urban environments using straight lines," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5550–5557.

[11] H.-M. Liu, Z.-H. Wang, and C. Deng, "Extend point descriptors for line, curve and region matching," in *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on*, vol. 1. IEEE, 2010, pp. 214–219.

[12] Z. Wang, H. Liu, and F. Wu, "HLD: A robust descriptor for line matching," *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics*, pp. 128–133, 2009.

[13] K. Hirose and H. Saito, "Fast line description for line-based slam." in *BMVC*, 2012, pp. 1–11.

[14] Y. P. Kwon, H. Kim, G. Konjevod, and S. McMains, "Dude (duality descriptor): A robust descriptor for disparate images using line segment duality," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 310–314.

[15] L. Wang, U. Neumann, and S. You, "Wide-baseline image matching using line signatures," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1311–1318.

[16] K. Li, J. Yao, X. Lu, L. Li, and Z. Zhang, "Hierarchical Line Matching Based on Line–Junction–Line Structure Descriptor and Local Homography Estimation," *Neurocomputing*, vol. 184, pp. 207–220, 2016.

[17] L. Zhang, *Line primitives and their applications in geometric computer vision*. Universitätsbibliothek Kiel, 2013.

[18] Z. Wang, F. Wu, and Z. Hu, "MSLD: A robust descriptor for line matching," *Pattern Recognition*, vol. 42, no. 5, pp. 941–953, 2009.

[19] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *Journal of Visual Communication and Image Representation*, vol. 24, no. 7, pp. 794–805, 2013.

[20] S. Zhuang, D. Zou, L. Pei, and D. H. P. Liu, "A binary robust line descriptor," in *International Conference on Indoor Positioning and Indoor Navigation, IPIN 2016, Alcala de Henares, Spain, October 4-7, 2016*.

[21] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

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

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

[24] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

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

[26] S. Zagoruyko and N. Komodakis, "Deep compare: A study on using convolutional neural networks to compare image patches," *Computer Vision and Image Understanding*, vol. 164, pp. 38–55, 2017.

[27] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei, and L. He, "Local feature descriptor for image matching: A survey," *IEEE Access*, vol. 7, pp. 6424–6434, 2019.

[28] A. Vakhitov and V. Lempitsky, "Learnable line segment descriptor for visual slam," *IEEE Access*, vol. 7, pp. 39 923–39 934, 2019.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[31] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[32] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[36] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *German conference on pattern recognition*. Springer, 2014, pp. 31–42.

[37] S. Xie, R. Girshick, P. Dollr, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[1] *https://github.com/manuellange/DLD*