2016 INTERNATIONAL CONFERENCE ON INDOOR POSITIONING AND INDOOR NAVIGATION (IPIN), 4-7 OCTOBER 2016, ALCALÁ DE HENARES, MADRID, SPAIN

A Binary Robust Line Descriptor

Shiwei Zhuang *Danping Zou Ling Pei Di He Peilin Liu
Shanghai Key Laboratory of Navigation and Location-based Services
Shanghai Jiao Tong University

***Abstract*—*Line features can be found extensively in man-made environments and are easier to be extracted than point features in low-texture scenes like indoor environments. Line features have been successfully applied to indoor localization recently. However, a good and efficient descriptor for line features is critical for real-time applications. Current line descriptors are mostly based on image gradients, involving expensive computations. Inspired by the ORB descriptors for point features, we propose a novel line descriptor using binary strings, called BRLD. To compute the binary line descriptor, we introduce a training method to select good pairings for binary tests. BRLD is robust under image transformations and noise. Experimental results show that BRLD's matching performance is close to that of the state-of-the-art line descriptors provided in OpenCV library, but ten times faster in both descriptor composition and matching.*

Keywords*—*Line Feature; Binary Descriptor; Feature Selection

I. INTRODUCTION

Feature matching is the basis of many image processing algorithms and applications. While most researches focus on point and region features [1][2][3], these kinds of features may not be available in low-texture scenes. In such cases, line features may present and can be adopted in vision applications such as scene recognition, visual SLAM[4] and indoor localization. One critical issue is to extract the descriptor for lines. Present descriptors for lines like LBD[5] and MSLD[6] are mostly based on image gradients and can be too costly to be applied to real-time applications, especially on mobile devices. Since binary descriptors are very fast to compute and match, related work has already been done for point features. The most well-known and commonly-used ones are BRIEF[7] and its improved version ORB[8]. Inspired by these two point descriptors, we propose a novel binary descriptor for line features, called binary robust line descriptor(BRLD). BRLD is generated only by binary tests of the local appearance between two positions around a line, without any other constraints or prior information. After that, matching is done by calculating the Hamming distance between two descriptors, which is highly efficient for real-time applications.

Instead of manually selecting those positions for binary tests, we introduce a training procedure to select good positions for binary tests automatically. The training is different from that of ORB because line features are more complicated than point features - line features have different lengths and orientations. To address such problem, we first transform the local region of a line segment into a normalized coordinate system where training process is operated. After offline training process is done, computing BRLD and matching them can be very fast since only binary operations are involved. The matching performance is however as good as the state-of-art line descriptors. Our main contributions are as follows:

1)We map the local region of a line to a normalized coordinate system according to the location and direction of this line, which makes BRLD invariant under image rotation, camera translation and scale changes.

2)We have tested different approaches for building the training data, including manual labeling and automatic generation by geometric transformation. After throughout comparison and analysis, we find out a way to get training data that lead to the best matching performance.

To test BRLD, we conducted experiments to evaluate its performance and compared it with the state-of-the-art line descriptor in OpenCV. The results show the robustness and efficiency of our approach.

II. REALTED WORK

Work related to line descriptors is much less than point ones. In earlier research, geometrical information is usually used for line matching[9][10]. However, such method requires prior geometrical relationship between images. Lourakis et al.[11] utilize projective invariance for line matching, but their approach is restricted to planar scenes. Herbert and Vittorio[12] present a line matching method depending on color histogram. The obvious weakness is that their method relies on color information of the image and may fail in situations where color discrimination is not strong such as gray image.

Wang et al. propose a line descriptor called mean-standard deviation line descriptor (MSLD) [6] . It is generated by a SIFT-like strategy, based on the gradients in pixel support region. Although it only relies on local appearance, MSLD only handles small baseline stereo line matching and it is not scale-invariant. Verhagen et al. solve the scale problem by applying five rules to the original descriptor to confirm the corresponding relationship of the descriptors. Another improved version of MSLD, called line band descriptor (LBD), is proposed by Lilian Zhang and Reinhard Koch [5]. Different from MSLD, both a global and a local Gaussian window are

* Corresponding author.
Email address: dpzou@sjtu.edu.cn

assigned to each row in the line support region, which makes LBD more robust and computationally efficient than MSLD. With the process of detecting lines in scale space, LBD also handles scale changes well. However, both Verhagen's and Zhang's approaches still have computational costs too high to meet real-time requirements, especially for mobile applications. The storage of these float-type descriptors may not be economical enough for some applications where the size of data is growing larger.

By contrast, the approach we propose in this paper is highly efficient to be computed and robust to image transformations. Inspired by BRIEF and ORB, we adopt their main idea to lines. The descriptor is also built by comparing the gray-level of two image positions. However, line features are very different from point features since they have different orientations and lengths. A new way of choosing local region and selecting test points is studied in this work to adapt the binary-test idea to the characteristics of lines.

III. OUR APPROACH

In this section, we present our approach in detail. For line detection, we use line segment detection (LSD)[13] to detect lines in each image. There are two critical steps to compute line descriptors: coordinate transformation and the training process to select good binary-test positions, which are elaborated below.

A. Choosing the local region of a line

The greatest difference between points and lines is that lines have different lengths and directions while points don't. So, to describe a line, the choice of local region of a line cannot be as trivial as that of a point. For points, we can simply choose a rectangular area with fixed size around a key point as local region to calculate the descriptor, like BRIEF[7] and ORB[8]. Since each line has its own length and direction, a normalization is needed to fix the way of generating the descriptor and the dimension of the descriptor. In MSLD[6] and LBD[5], they both choose a rectangular region centered at the line as the local region, or in the other word, pixel support region (PSR). The pixel support region is divided into several sub-regions and a descriptor is calculated for each sub-region by a SIFT-like strategy. The normalization of length is done by constructing the final descriptor with the mean and standard deviation of these sub-region descriptors.

In our approach, we introduce a novel method of choosing the local region and a coordinate transformation to adapt to different lines.

First, we introduce a normalized coordinate system in which all local regions of lines are expressed. All lines are normalized so that they have the unit length in the normalized frame and locate at *x*-axis marked in red as shown in Fig. 1. The line is also divided into *M* segments for sampling. The direction perpendicular to the line is not normalized as we choose a constant window size *W* for all lines.

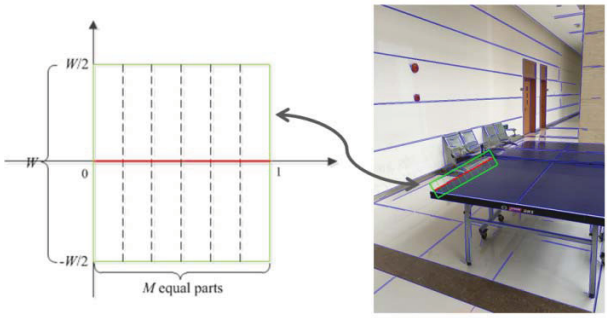

Fig.1 Local region in the normalized system and real image. The red line is the extracted key line. The area within the green lines is the local region.

After normalization, a transformation between the normalized coordinates and pixel positions is established. The transformation is computed as the following. Denote the coordinate of a point in normalized system by (x_0, y_0). We write it in homogeneous form $\mathbf{X}_0 = (x_0, y_0, 1)^{\mathrm{T}}$. Let the coordinates of line endpoints in the image be (x_1, y_1), (x_2, y_2), where $x_1 \leq x_2$. The first step is scaling:

$$\mathbf{X}_1 = \mathbf{T}_1 * \mathbf{X}_0 \tag{1}$$

$$\mathbf{T}_1 = \begin{bmatrix} l & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Here *l* is the length of the line. The second step is rotation:

$$\mathbf{X}_2 = \mathbf{T}_2 * \mathbf{X}_1 \tag{3}$$

$$\mathbf{T}_2 = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\alpha = \arctan(k) \tag{5}$$

where *k* is the slope of the line. The last one is translation:

$$\mathbf{X} = \mathbf{T}_3 * \mathbf{X}_2 \tag{6}$$

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

Let $\mathbf{X} = (u, v, 1)^{\mathrm{T}}$. (u, v) is the actual coordinate of $\mathbf{X}_0$ in the real image. The overall transformation can be written as:

$$\mathbf{X} = \mathbf{T} * \mathbf{X}_0 \tag{8}$$

where $\mathbf{T} = \mathbf{T}_3 * \mathbf{T}_2 * \mathbf{T}_1$ is the transformation matrix. For each detected line, all pixels in the local region are mapped into the normalized coordinate system by above transformation. The binary-test position sampling are all operated on the normalized coordinate system. In the following section, we'll describe the method of sampling good positions for binary tests.

B. Selecting good sampling positions by training

As the ORB descriptor, the feature vector of BRLD consists of many binary values that are obtained from comparing the intensity between two pixels. The key issue is

how to sample those pixel pairs for comparison. We propose to use training to automatically extract those pairs.

To generate the training dataset, we take pictures at different locations and from different viewpoints. In each location we take about five or six pictures. We use line segment detection (LSD) to detect line in the images. Then we manually label those segments belonging to the same line. We label those lines whose local regions have distinctive textures.

It has been shown in [8] that, a good test pair should generate binary features on the training images whose average value is close to 0.5. Apart from that, a test pair that leads to binary features with high variance is more discriminative. Therefore, we search for those test pairs with a mean close to 0.5. In our normalized local region, there are $(W+1)*(M+1)$ points. We do exhaustive search among all possible test pairs, so the number of tests are $C^2_{(W+1)*(M+1)}$.

To avoid noises, instead of using single pixels, we use the average intensity of a sub-window to do the binary test. In our test, the window size is chosen as 5*5. A binary test is done as follows:

- For a test pair $(\mathbf{X}_1, \mathbf{X}_2)$ in the image, a binary test τ is defined by:

$$\tau(\mathbf{X}_1, \mathbf{X}_2) = \begin{cases} 1, \text{if } \mathbf{I}(\mathbf{X}_1) \geq \mathbf{I}(\mathbf{X}_2) \\ 0, \text{if } \mathbf{I}(\mathbf{X}_1) < \mathbf{I}(\mathbf{X}_2) \end{cases} \quad (9)$$

  where $\mathbf{I}(\mathbf{X}_1)$, $\mathbf{I}(\mathbf{X}_2)$ are the average intensities of the sub-windows centered at point $\mathbf{X}_1, \mathbf{X}_2$.

- To speed up, we use an integral image instead of direct computation of the sum. The integral image is a 2-dimentional convolution of the original image and a convolution template matrix **S**. The 5*5 matrix **S** is defined as:

$$\mathbf{S} = \begin{pmatrix} 1/25 & \cdots & 1/25 \\ \vdots & \ddots & \vdots \\ 1/25 & \cdots & 1/25 \end{pmatrix} \quad (10)$$

  Then the average intensity is directly calculated by the gray-level of a point in the integral image.

We do binary tests against all lines in all images in our training set. For each test pair, we get a high-dimensional binary vector. We compute the mean of these vectors and order them by the distance from 0.5. The first 1000 test pairs are selected for the following training.

After obtaining these candidate pairs, we choose *D* pairs from them according to their matching score to form a pair set. *D* also is the dimension of the final descriptor. We first choose randomly from these candidate pairs and repeat this process until 100 pair sets are formed. Each set contains *D* test pairs. Then we use each set to calculate the descriptor for labeled lines in the image. We use different descriptors derived from different pair sets to match lines. The matching is done by calculating the Hamming distance between two descriptors. The one gets the minimum value is considered as a match. As the ground truth is available, the matching performance for each kind of descriptors can be evaluated. The pair set that produces the best matching score is selected to compute the line descriptors.

The training process is time consuming, but it requires to be done only once. Only are the final test pairs after training used for computation of descriptors. The binary tests as well as calculation of Hamming distance can be done very fast in practice. We summarize the training process in the following steps:

1. Set up a normalized coordinate system and establish a mapping between the normalized coordinate system and the rectangular area around the line.

2. Detect lines for all images in the training set with LSD detector and label the segments belong to the same line in different images.

3. Sample possible test pairs in the normalized coordinate system and map them to the original image to compute the binary feature value.

4. Run binary test against all possible test pairs of all lines in all images in the training set. For each test pair, we get a high-dimensional binary vector.

5. Sort these vectors by their distance from a mean of 0.5 and store the first 1000 pairs, forming a 1000*4 matrix **C**.

6. Randomly pick *D* pairs from **C** without repetition to get a candidate set and repeat the picking until 100 candidate sets are found.

7. Evaluate the matching performance by using different descriptors from different pair sets. The one with the best performance is used to compute the line descriptor.

IV. EXPERIMENT

A. Training dataset

The training data are generated from two approaches. The first one is manual labeling (see in Fig. 2). However, it requires huge workload to obtain even a moderately large training set. Therefore, we also adopt the second approach to get the training data. That is, we use the automatic way to generate the training data by geometric transformation. The basic idea is to use homography to transform an image into a new one. The line correspondences can be easily established by the pre-known geometric relationship.

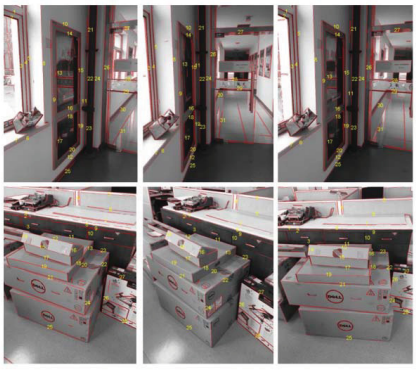

Fig.2 Manually-labeled samples in our training set.

We choose input images from the VOC 2008 dataset[14]. These images mainly capture indoor environments and the appearance of buildings. Therefore these images contain more lines. We choose one of them as a reference, and apply random homography transformations to generate query images. The homography is composited by different distortion, rotation and scale change and defined as:

$$\mathbf{T} = s\mathbf{R}(\varphi)\mathbf{R}^{-1}(\theta)\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}\mathbf{R}(\theta) \tag{11}$$

$$\mathbf{H} = \begin{bmatrix}\mathbf{T} & \mathbf{0}\\ \mathbf{0} & 1\end{bmatrix} \tag{12}$$

The right hand term $\mathbf{R}^{-1}(\theta)\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}\mathbf{R}(\theta)$ represents the distortion, which is done by rotating the image by an angle of θ, changing the scale in the vertical and horizontal direction of the rotated image by λ_1 and λ_2, and rotating it back. $\mathbf{R}(\varphi)$ is the rotation matrix, where φ is the rotation angle. s is the scale factor, which gives the image an entire scale transformation.

In our experiment, the value ranges of these parameters are listed in Table I.

TABLE I. PARAMETER VALUE RANGES

Parameter	Value Range
θ	[0,2pi]
λ_1	[0.8,1.2]
λ_2	[0.8,1.2]
φ	[-pi/12,pi/12]
s	[0.5,2]

Since all homographies are known, the ground truth of line correspondences can be simply established between the reference image and query image. We also add Gaussian noises and intensity changes to the transformed image to mimic real-world situations. The Gaussian noise has a mean of 0 and a variance ranges from 0.001 to 0.002. The average PSNR is 28.7 dB. The overall pixel intensity in the query image changes from 80% to 120% of that in the reference. Fig. 3 shows some examples of our training set.

Fig. 3 Some automatically generated samples in our training set.

The lines extracted by LSD detector will never exactly coincide with the lines by homography transformation in practice. So we perturb the endpoints of each line within a range of five pixels, which makes the line extraction more like practical ones.

In the following sections, we first evaluate how some critical parameters, including dimension of descriptors, window size of local regions and choice of training sets, affect the final matching performance of the proposed approach.

B. *The dimension of the descriptor*

The dimension of the descriptor affects both the matching performance and the efficiency. We test different dimensions of 64, 128, 256 and 512. The sizes of local region are kept the same. This experiment is to evaluate the performance of our algorithm using different dimensions of descriptors.

In our experiment, the dimension is set after we obtain candidate test pairs with a mean value close to 0.5. Then we randomly pick D test pairs from those candidate test pairs until 100 sets are formed, of which D dimension. We use exactly the same training set to get the final result of test pair locations. Finally a same test set is used to evaluate the performance of descriptors with different dimensions. Both the training and test set has around 200 images with ground truth. The results are shown in Fig. 4.

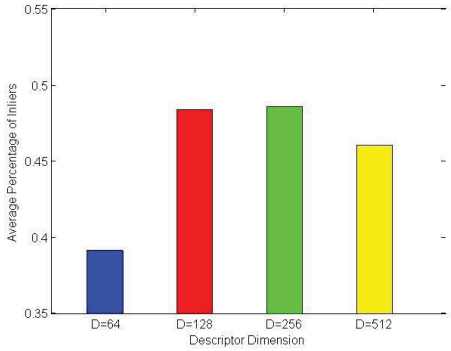

Fig. 4 The matching performance of descriptors with a dimension of 64, 128, 256 and 512.

The average correctness over the test set are 39.16%, 48.35%, 48.57% and 46.05% for descriptors with a dimension of 64, 128, 256 and 512. From the result we can see that 128-bit descriptor is the best one considering the tradeoff between computation cost and matching performance.

C. *Window size of local region and number of equal parts*

We change the window size of local region W in our experiment to study its impact on descriptor performance. For all evaluations, we fix the value of M to be 10 for discretization. We vary W from 20 to 60, and record the average number of correct matches generated by all candidates on 20 images with ground truth. Fig .5 shows our result.

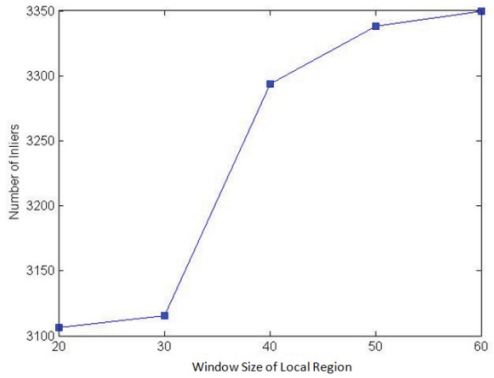

Fig.5 The average number of inliers when the window size of local region changes.

As we can see, when the window size of local region changes from 30 to 40, the number of inliers has a significant raise. However, after W exceeds 50, the increase becomes gentle. The result shows that descriptors generated in a small window is not discriminative enough while a wide local region may carry abundant information. To balance the computational cost and the matching performance, we choose 40 as the window size of local region in our experiment. Furthermore, we study the influence of the number of equal parts M on the descriptor performance. In the experiment, we fix the value of W to be 40 and vary M from 10 to 30. Results are shown in Fig. 6.

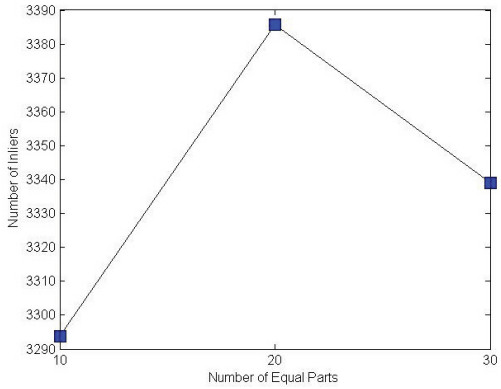

Fig.6 The average number of inliers when the number of equal parts changes.

In Fig. 6, the descriptor performance is the best when M = 20. A value of M either too big or too small leads to a bad performance. Since the length of lines in an image has a wide range, we need to choose a value of M to cover as many pixels in the local region as possible while avoiding too much abundant information. Judging from the results, we choose M to be 20.

D. The choice of training set and the distribution of sampling positions

The training set is important to the performance of our descriptor. In our experiment, we use three different training sets to sample locations for test pairs. The first one is the image set used in Zhang and Koch's work[5]. The images has been used for the evaluation of LBD descriptor. It contains eight groups of images with different transformations including illumination, rotation, compression, blurring, occlusion, low-texture scene, viewpoint changes and scale changes. These images are all planar scenes or taken with fixed camera position. Thus, the image transformations can be computed and the ground truth can be acquired by them. The second training set is our manually-labeled image set. This set contains over 100 images taken from different places with combined transformations. Images are captured mostly in indoor scenes and taken with no limitations. This set is quite challenging for line matching. The last one is a combination of automatically generated images by homography transformations and manually-labeled ones. We select 100 reference images from the VOC 2008 set[14]. Each reference image generates 20 query images. And the number of manually-labeled images in this set is over 200.

We set M = 10, W = 30 and D = 256. Since these training sets are quite different, the distribution of the final sampling locations turns out to differ widely. Fig.7 shows the distributions of these sets of test pairs acquired from training.

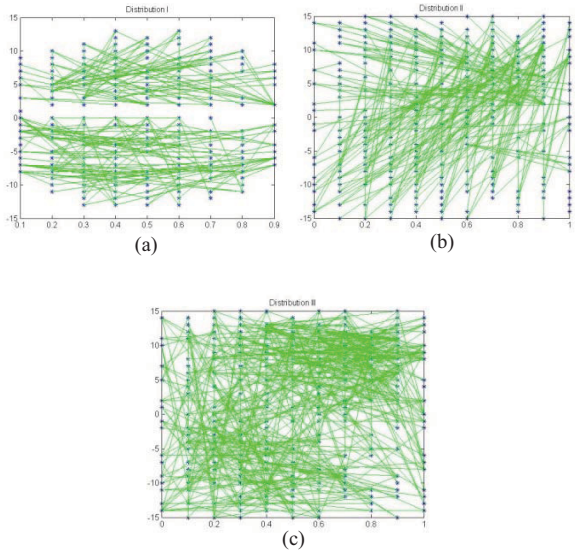

Fig. 7 Distribution of binary tests. (a) The training result of Zhang's dataset. (b)The training result of our hand-labeled dataset. (c)The training result of the combined dataset.

From the result we can see the obvious difference among training results from different datasets. In the result from Zhang's set, the two positions of a test pair are located on the same side of the line. No pixels on different sides are found. In the result from the hand-labeled set, most selected test pairs are pixels on different sides. And in the result from the third set, both pixel pairs on the same side and different sides are selected. We do some matching tests with these results on the same test set as shown in Fig. 8.

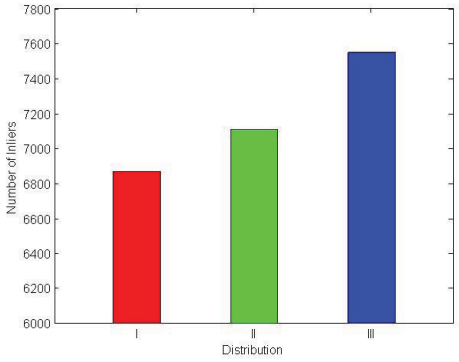

Fig. 8 The number of inliers using different distribution of sample locations. There are in total 15637 lines computed from 138 manually-labeled images.

The result shows that the matching performance is the best when the location of test pairs follows distribution III which contains pixels both on the same side and on the different sides. It also shows that the choice of training set should be carefully collected, otherwise it may leads to a bad result like Distribution I. For instance, images in Zhang's dataset are groups with single transformations. The size of this dataset is too small and the transformations of images in it are too simple for the training. Images with combined transformations are better than those with a single transformation, since the result of our hand-labeled set outperforms that of Zhang's set.

E. Performance of BRLD

In this section, we evaluate the performance of the proposed BRLD descriptor. First, we test the robustness of BRLD against image rotations. The results are shown in Fig. 9.

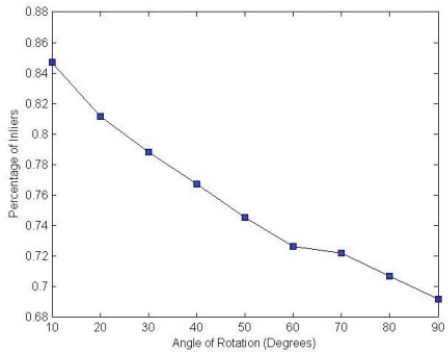

Fig. 9 Matching performance of BRLD under different image rotations.

We use a rotation matrix to generate query images. As we can see, BRLD is robust to image rotations. Only 14% performance degrade has been observed under a 90 degree rotation. The reason is largely due to the transformation between the normalized coordinate system and the real image before sampling the test pairs.

We also test the performance of BRLD under different level of image noise. We add Gaussian noise with variance ranges from 0.001 to 0.02. The PSNR ranges from 29.98 dB to 17.25 dB. The matching performance of BRLD is shown in Fig. 10.

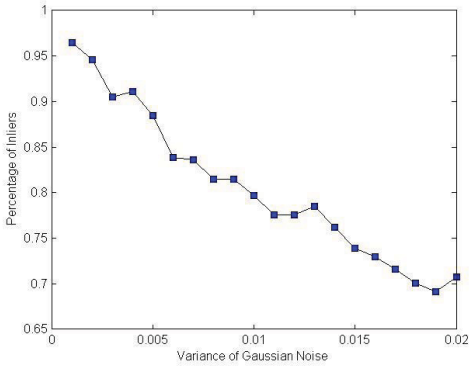

Fig. 10 Matching performance of BRLD under image noise.

The result shows that the performance of BRLD is still promising under even large image noise. Since images in the training set also contains noises and the binary test is done from the integral images, the robustness of BRLD to noise is not out of expectation.

In the following, we compare the proposed approach with the state-of-the-art LBD descriptor that has been already implemented in OpenCV. First, we compare the matching correctness of BRLD and LBD on our test set which contains 216 manually-labeled images. These images are not included in the training set. LBD descriptors are generated using functions with default parameters in OpenCV. We set $M = 20$, $W = 40$ and Dimension = 128 for BRLD. The training set contains 2000 images automatically generated by homographic warp and more than 200 manually-labeled images. The result is shown in Fig. 11.

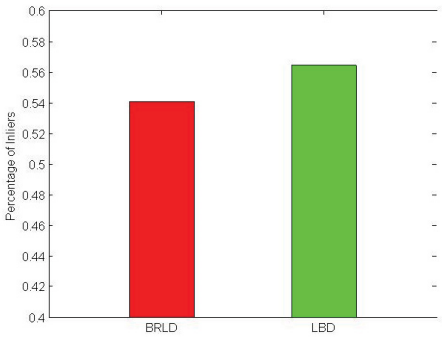

Fig.11 The average matching correctness of BRLD and LBD.

The result shows that the percentage of inliers computed by BRLD is close to that of LBD. The average correctness over 216 testing images is 54.06% for BRLD and 56.42% for LBD. Fig. 12 shows an illustration of the BRLD matching results.

Fig.12 Matching results of BRLD on two pairs of labeled images. The percentage of inliers in the first image pair is 36/53, and that in the second one is 18/31.

The manually-labeled images in our test set are mostly low-texture indoor scenes and lines in these images have high similarity. However, BRLD descriptor still has a relatively good performance without any other constraints.

Moreover, we compare the running-time efficiency of BRLD and LBD. The tests are performed on an Intel i7 2.4 GHz processor. We compute BRLD and LBD on 20 612*816 images from our test set. The descriptor length is 128 for BRLD and 32 for LBD. The results are shown in Table II.

TABLE II. THE AVERAGE TIME OF COMPOSITION AND MATCHING

Descriptor	Descriptor composition time per frame (ms)	Matching time per frame (ms)
BRLD	4.89	1.74
LBD	60.98	10.56

10568 lines are detected by LSD in 20 images in total. Both the time of descriptor extraction and matching are recorded. We can see that BRLD is an order of magnitude faster than LBD.

V. CONCLUSION

In this paper, we have proposed a novel binary line descriptor - BRLD. We compared it with the-state-of-the-art descriptor regarding both matching correctness and running time efficiency. The results show that our method achieves a matching performance close to the descriptors implemented in OpenCV, while is one order of magnitude faster. The computation cost of the proposed descriptor is low enough to meet real-time requirements of mobile device applications.

Two key techniques has been applied in our approach. Firstly, we establish a mapping between a normalized coordinate system and the local patch of a line to normalize different orientations and lengths, which makes the descriptor robust to geometric transformations. Secondly, we use a training approach to sampling positions automatically for binary tests. After conducting an extensive study, we find the parameters and way for collecting training datasets that leads to a descriptor which is the most robust and discriminative.

In our experiments, we just use three kinds of database with ground truth - images automatically generated by homography, images taken with known homography and manually-labeled images. Since the change of training set - both the size of it and the types of images in it - have a great influence on descriptor performance, we will collect more images with ground truth to find a better training set to improve the performance in the future work. Furthermore, the training method also has much space to improve. With enough images with ground truth, some state-of-art deep learning methods such as convolutional neutral network[15] may be utilized. The matching method in our experiment is just using descriptor distance. However, other information could also be combined to improve the matching performance, such as distance and orientation closeness.

ACKNOWLEDGEMENT

This work is funded by Natural Science Foundation of China under Grant No.61402283 and No.61573242, Important National Science and Technology Specific Project of China under Grant No.2016ZX03001022-006, the Shanghai Science and Technology Committee under Grant No.16DZ1100402 and No.15511105100 and the National Science and Technology Major Project under Grant No.GFZX0301010708.

REFERENCES

[1] Lowe D G. Distinctive image features from scale-invariant keypoints[J]. International journal of computer vision, 2004, 60(2): 91-110.

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

[3] Mikolajczyk K, Tuytelaars T, Schmid C, et al. A comparison of affine region detectors[J]. International journal of computer vision, 2005, 65(1-2): 43-72.

[4] Zhou H, Zou D, Pei L, et al. StructSLAM: Visual SLAM with building structure lines[J]. Vehicular Technology, IEEE Transactions on, 2015, 64(4): 1364-1375.

[5] Zhang L, Koch R. An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency[J]. Journal of Visual Communication and Image Representation, 2013, 24(7): 794-805.

[6] Wang Z, Wu F, Hu Z. MSLD: A robust descriptor for line matching[J]. Pattern Recognition, 2009, 42(5): 941-953.

[7] Calonder M, Lepetit V, Strecha C, et al. Brief: Binary robust independent elementary features[J]. Computer Vision–ECCV 2010, 2010: 778-792.

[8] Rublee E, Rabaud V, Konolige K, et al. ORB: an efficient alternative to SIFT or SURF[C]//Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011: 2564-2571.

[9] Schmid C, Zisserman A. Automatic line matching across views[C]//Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on. IEEE, 1997: 666-671.

[10] Schmid C, Zisserman A. The geometry and matching of lines and curves over multiple views[J]. International Journal of Computer Vision, 2000, 40(3): 199-233.

[11] Lourakis M I A, Halkidis S T, Orphanoudakis S C. Matching disparate views of planar surfaces using projective invariants[J]. Image and Vision Computing, 2000, 18(9): 673-683.

[12] Bay H, Ferrari V, Van Gool L. Wide-baseline stereo matching with line segments[C]//Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. IEEE, 2005, 1: 329-336.

[13] von Gioi R G, Jakubowicz J, Morel J M, et al. LSD: A fast line segment detector with a false detection control[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2008 (4): 722-732.

[14] Everingham M, Van Gool L, Williams C K I, et al. The pascal visual object classes (voc) challenge[J]. International journal of computer vision, 2010, 88(2): 303-338.

[15] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.