Vision, Modeling, and Visualization (2020)
J. Krüger, M. Niessner, and J. Stückler (Eds.)

WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching

Manuel Lange[1], Claudio Raisch[1] and Andreas Schilling[1]

[1]Visual Computing, University of Tübingen, Germany

Abstract

We present a machine learning based and wavelet enhanced line feature descriptor for line feature matching. Therefor we trained a neural network to compute a descriptor for a line, given preprocessed information from the image area around the line. In the preprocessing step we utilize wavelets to extract meaningful information from the image for the descriptor. This process is inspired by the human vision system. We used the Unreal Engine 4 and multiple different freely available scenes to create our training data. We conducted the evaluation on ground truth labeled images of our own and from the Middlebury Stereo Dataset. To show the advancement of our method in terms of matching quality, we compare it to the Line Band Descriptor (LBD), to the Deep Learning Based Line Descriptor (DLD), which we used as a starting point for this work, and to the Learnable Line Segment Descriptor for Visual SLAM (LLD). We publish the project on github to support the community: https://github.com/manuellange/WLD

CCS Concepts

• ***Computing methodologies*** → ***Matching;*** *Computer vision; Neural networks; Machine learning;*

1. Introduction

Many computer vision related problems are based, or, at least, supported by visual features, for example: Object recognition [Low99], Multi-View Stereo [GSC*07], 3D reconstruction [MVGV09] [FLG14], Structure from Motion (SfM) [SSS06], Visual Odometry (VO) [NNB04], Visual Simultaneous Localization and Mapping [DRMS07], place recognition [Mil13] and terrain classification [FB12]. All of the afore mentioned works deal with point features to address their problem. But points are not always available in sufficient quantities. They can be hard to find in blurred or noisy images, or in images with many homogeneous areas like in man made environments. Buildings, for example, are hardly textured, but they yield structural information through the edges formed by its silhouette, or by features like doors and windows.

1.1. Line Features

In recent years, line features received more attention and made it possible to improve the robustness, and accuracy for solutions of computer vision challenges mentioned earlier. Line features are well suited to represent structure [HMB14]. They helped improve Structure from Motion [ZK14], Visual Odometry [LRS19], Visual Simultaneous Localization and Mapping [GOZNM*17], and place recognition using a bag of words technique [DWLK19].

A method for line detection that was published in the early seventies is based on the Hough transform [DH71]. A disadvantage is its relatively high computational cost. Faster algorithms were released in the last decade. Famous ones are the EDLine detector from Cuneyt Akinlar and Cihan Topal [AT11], the LSD Line Segment Detector by Rafael Grompone von Gioi et al. [VGJMR12], and a work of Jin Han Lee et al. [LLZ*14], which was coined Fast Line Detector.

The matching of features is fundamental when using them to accomplish computer vision tasks. Geometric, and descriptor based methods are most common. If features shall be tracked in a video stream with small displacements between frames, matching, based on predictions and simple geometry, is often used as it is fast [NPVCL08] [GOGJ18] [LRS19]. However, if there are no predictions, and the distance between images is larger, tracking typically fails, and more complex geometric approaches are required.

A recent approach of Wang et al. forms groups of lines to create 'line signatures' in order to match them over wide baselines [WNY09]. Kim and Lee focused on matching images based on coplanar line intersections while they designed their work "for dealing with poorly textured and/or non-planar structured scenes" [KL10]. Li et al. [LYL*16] developed a hierarchical method, which is based on 'Line-Junction-Line' structures, that are each formed of two adjacent lines incorporating their intersection. A mix using geometric ideas and the computation of a descriptor is presented by Kwon et al. [KKKM16]. They construct a descriptor out of multiple line segments. Their intention is to tackle the difficulty that the endpoints of lines may change between different images, e.g. on curved objects, or due to occlusions. But their descriptor is computation-

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

DOI: 10.2312/vmv.20201186

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

ally expensive. The calculation of their descriptors takes 7 seconds, and matching another 6 seconds (on 700×500 image resolution). Lilian Zhang goes into more detail about geometric methods in his dissertation [Zha13].

Geometric approaches are in general computationally expensive compared to descriptor based methods. This disadvantage also holds for patch matching [ZK17] where one could see similarities to line matching when the region around the line is seen as a patch. Furthermore, lines have specific characteristics like a distinct direction and a left and right side which can lie on different surfaces if the line lies on the edge of an object. We argue that a network which always gets the lines vertically and centered in the middle is more likely to specialize on the characteristics of a line. Additionally, descriptors have the advantage that, once they are computed, distance calculation and matching is fast, and descriptors can be reused which is important when matching features between more than two images, e.g. for SLAM, or for reconstruction purposes. Because of these reasons, we focus on descriptor based line matching. An early line descriptor was proposed by Wang et al. [WLW09]. They place small regions along the line, calculate a Harris matrix, and compute a descriptor from this information (HLD). Liu et al. [LWD10] proposed to place point descriptors along the lines in order to describe them while Hirose and Saito used Histograms [HS12]. The mean-standard deviation line descriptor (MSLD) by Wang et al. [WWH09] is similar to the HLD, and also works on regions along, and next to the line. It computes the mean and standard deviation vectors in those regions to create a descriptor. The MSLD has often been compared to by earlier works; also Lilian Zhang evaluated his Line Band Descriptor (LBD) and the MSLD in his dissertation [Zha13]. He showed that his band wise approach is superior to the region based method. For this reason, we chose the LBD as one of the works to compare to and to go more into detail about it in our Results 3. There we also explain more about the Learnable Line Segment Descriptor for Visual SLAM by Vakhitov and Lempitsky [VL19], which is a recent learning based method we also included in the benchmarks.

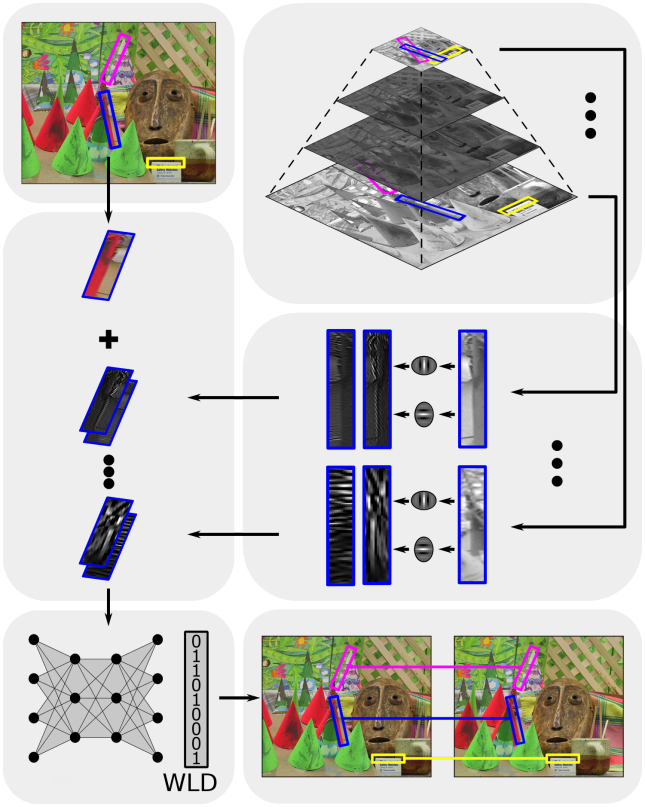

Figure 1: *Top left shows the original image and top right shows the multi-scale pyramid of it, both with some line segments highlighted for visualization. Middle right depicts the Wavelet convolution. The results of all layers are stacked together with the original rgb cutout, as shown in the middle left image. The cutout stack is the input to the ResNet which calculates a descriptor (bottom left). This descriptor can be compared to other descriptors for matching (bottom right).*

1.2. Wavelets

In the eighties the term wavelets was coined by Alex Grossman and Jean Morlet [GM84]. While researching square integrable real-valued functions, they found that these can be represented by creating a family of functions using shifts and dilations on a single function, which they called wavelet. There are different types of wavelets, and some have already been found before the eighties. Famous ones are the Haar wavelet, the Cohen-Daubechies-Feauveau-Wavelet and the Gabor wavelet. The Haar wavelet has been named after Alfréd Haar, who proposed the Haar sequence in 1909, long before wavelets became a field of research on their own. It is widely used in image processing and pattern recognition due to its low computing requirements and its simplicity [PL04]. The Cohen-Daubechies-Feauveau-Wavelet (CDF-Wavelet) is used for image compression in the JPEG 2000 format. There are different variants of it: the CDF-5/3-Wavelet (developed by D. Le Gall and Ali J. Tabatabai) is for lossless compression, and the CDF-9/7-Wavelet (by Ingrid Daubechies) is used in the lossy compression algorithm. Gabor wavelets, which are similar to Morlet wavelets, have been named after Dennis Gabor, who researched communication theory [Gab46]. They minimize the uncertainty of the information that they carry by optimizing the trade off between spatial and frequency resolution [Dau85]. Gabor wavelets are used in all kinds of perception and vision related tasks as, for example, in motion estimation [BP02], object representation [KS01], visual attention analysis [BSV05], and texture classification [GPK02]. They are even represented in the mammalian visual cortex as they support efficient processing of visual information [Dau85] [Mal89] [Lee96]. Therefore, we chose the Gabor wavelet to extract important information from our images as a preprocessing step in our method.

As a starting point of this work we used the Deep Learning based Line Descriptor (DLD) from our group [LSS19]. We extended this approach by incorporating larger surroundings of the lines for the computation of each line descriptor. However, this would overly increase the size of the used neural network and, thereby, slow down processing. To avoid this, we use a multiscale image pyramid. Additionally, we introduce wavelets to preprocess the image parts of the pyramid. In this preprocessing we use Gabor wavelets to extract the important information from the image area, which is biologi-

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

cally motivated, as mentioned before. An overview of the descriptor creation is given in Fig. 1.
The rest of this paper is structured as follows: The next Section 2 explains our approach in detail. After that, we present the results including comparisons to other methods in Section 3. Finally, Section 4 concludes this paper.

2. Our Method

In this section we give a detailed explanation of our method, including the data generation for the learning process, the preprocessing using gabor wavelets, and the training itself with details about the network and the loss function.

2.1. Data Generation

In order to generate large amounts of learning data with ground truth information, we needed a suitable simulation environment. It should also yield quite realistic image data so that it worked well with real world data. Therefore, we chose the Unreal Engine 4 [Epi], which is able to visualize very realistic scenes, as a simulation environment for our data creation. There are various sceneries which are freely available. We chose four different ones with varying environments: *Epic Zen Garden*, which shows an outdoor garden area including a pond, and an angular concrete house with big glass windows, *Infinity Blade: Grass Lands* is an earthy and rocky area including an old citadel reaching out of the surrounding water, *Scifi Hallway*, showing a futuristic hallway which could be part of a space station, and *Living Room*, which was used as tech demo for the Engine. We created splines in each of these scenes to move the camera along them. The camera is moved stepwise, and images are taken at every step. In total the scenes contain 37689 labeled stereo images, one for each step (12068 in *Epic Zen Garden*, 17875 in *Infinity Blade: Grass Lands*, 5328 in *Scifi Hallway* and 2418 steps in *Living Room*). The image-data includes the color information, the depth, and, additionally, 3D coordinates for each pixel in the image. It is used to detect the lines visible from the current camera position and also to calculate the ground truth matching information of the lines. Altogether we collected about 11.9 Million line segment matches for our training.

2.2. Preprocessing

We call the image area around a line segment a cutout. The cutout of a line segment is rotated by the same angle as the line segment, and contains the line segment itself in its center. Depending on its size it includes more or less of the area surrounding the line segment. A line segment's neighborhood is very helpful when matching lines, but there are cases where the broader neighborhood is especially important. For example, the line segments on a bigger building with many similar windows are likely to look alike. This makes matching difficult when only the closer surrounding is in focus. But if more of the neighboring area is regarded, matching gets more reliable. For example, if the information is included that the line segment on the window frame is close to a downspout, or that it is on the third window counted from the right edge of the building. Therefore, this should be part of the information when creating a descriptor for a line segment.

Wavelet Integration: We wanted to include a broader area around the line without using large input layers in the network. For that reason we chose to shrink the cutout area and, additionally, precompute it using Gabor wavelets. These wavelets are especially well suited for image processing as they optimize the trade off between spatial and frequency resolution [Dau85], and they have been found in the human vision system [Dau85] [Mal89] [Lee96] (see also 1.2). In order to increase the receptive field to include a wider area around the line, we use an image pyramid. The pyramid has 5 levels and the width and height are reduced by a factor of 2 from level to level. The lowest level is the original image with three channels for red, green, and blue. From this level, we copy a rectangular region around the line segment including all three color channels for the cutout. We will not consider the color for the remaining levels. Thus, for the next octave, we convert the image to grey values and scale it down by a factor of two. Then we convolve it with the Gabor wavelet kernels. The 2D Gabor wavelet is given as:

$$\psi(x,y;\sigma,\theta,\lambda,\phi,\gamma) = e^{-\frac{x'^2+\gamma^2y'^2}{2\sigma^2}} \cdot e^{i(2\pi\frac{x'}{\lambda}+\phi)}, \tag{1}$$

where x, y are the coordinates of the center location, σ is the standard deviation of the Gaussian window function, θ is the wavelet orientation angle, λ is the wavelength of the sinusoidal factor, ϕ is the phase shift and γ is a factor defining the spatial aspect ratio. x' and y' are given as:

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta, \\ y' &= -x\sin\theta + y\cos\theta. \end{aligned} \tag{2}$$

We get an image for each kernel from which we copy another rectangular region around the line segment with the line segment in its center. These rectangular regions are of the same size as the ones from the previous level. But they contain more of the surrounding of the line segment as the image is scaled down, compared to the previous level. Also we did not extract the color information here but the result of the convolution with the Gabor wavelet kernels. We get one rectangular region for each kernel we convolved with and stack them on top of the already extracted regions. We continue the process of scaling, convolving and extracting the region for the next three levels. But for them, we only keep scaling the width of the region (perpendicular to the line segment's direction) as the length of the region mostly is already far beyond the extent of the line segment after scaling it once by a factor of 2. An overview is given in Table 1, to efficiently compute this in our implementation, we start by extracting the largest region around the line segment. This is determined by the top most level in the pyramid. After the extraction, we rotate it to the same angle as the line segment. This rotated image patch is the source for all the other levels. This way we only have to rotate once per line segment. An alternative option would be to precalculate different levels and wavelet convolution directions of the whole image, and extract the patches from them. We want the Gabor wavelet kernel direction to be adjusted to the line segment direction. Thus precomputing the convolutions

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

on the whole image would mean to precompute the convolutions in many different angles, so that there would be a pre-convolved image ready for every line segment angle. If we accepted an angle difference tolerance of up to one degree, we would need 360 precomputed convolution directions of the whole image for every level. This would result in more calculations than the firstly described process which we chose. This is because a usual Full HD image only contains a couple hundred lines.

Here we want to note that at first a fixed size for the rectangular region around a line segment does not seem to be a good choice. One could think that lines will grow drastically if the image size is increased. But while this can be the case for a few lines, the majority of lines will be around a similar length in pixels. Lines are more likely to get split in multiple parts if the resolution increases. This is because small curvatures reach over more pixels in higher resolution images; also other influences can get more prominent. Both may cause the detector to finish a line segment earlier. Usually with higher resolutions more lines are found. One reason is that additional lines are split in multiple parts, while another reason is that some lines will be detected which would not have been detected in lower resolutions as they are too small or the gradient is not sharp enough. The size of the line mostly depends on the line detector and its settings. The region size we chose works very well with the EDLine detector [AT11] and with the LSD [VGJMR12] using standard settings. While training works better with fixed region sizes, variable line lengths can also be used in normal operation due to the pooling layers in the network.

Table 1: *Network input - Layers of the cutout stack*

	Layer Description	Level	Size	Receptive field
Color	Red Channel	0	100 x 27	100 x 27
	Green Channel	0	100 x 27	100 x 27
	Blue Channel	0	100 x 27	100 x 27
Gabor Wavelet Direction	Vertical	1	100 x 27	200 x 54
	Horizontal	1	100 x 27	200 x 54
	Vertical	2	100 x 27	200 x 108
	Horizontal	2	100 x 27	200 x 108
	Vertical	3	100 x 27	200 x 216
	Horizontal	3	100 x 27	200 x 216
	Vertical	4	100 x 27	200 x 432
	Horizontal	4	100 x 27	200 x 432

2.3. Training

The ResNet architecture set a new state of the art when it was released [HZRS16]. In general, one would think that a more complex neural network will at least perform as well as a less complex version of it, which has fewer layers, but, besides that, is the same network. This is not guaranteed and until there was the ResNet architecture so called exploding and vanishing gradients posed a difficult problem. The reason is that in a larger network the gradients, which are propagated back through the network, pass through more matrix multiplications, and are, thereby, more likely to shrink until they vanish, or increase till they are overly huge. To address this problem, Kaiming He et al. introduced skip connections. Those connections forward the input of a layer to its output. When the weights are initialized at zero and no training has adjusted them, the values before and after a layer are the same. This means that the layers start with identity mapping and their weights are only changed if it is beneficial to the networks performance. If there are unnecessarily many layers, a lot of them do not change weights which results in identity mapping. Theoretically a network could also learn an identity mapping for several layers. But in practice this does not work as well, likely due to the interplay between layers. Due to this advancement, we chose a ResNet architecture for our problem.

The triplet loss: is well suited [WS09] [SKP15] as error function for our problem as it has been shown that it is applicable for Nearest-Neighbour-Classification and computer vision tasks [HBL17]. Our goal is the computation of a line segment descriptor which is very different to other line segments' descriptors, but similar to descriptors computed of the same line segment from other images. The triplet loss is ideal for our problem as it rewards similarity of descriptors of the same line segment and penalizes similarity of other line segments' descriptors at the same time. In the terminology of the triplet loss, our line segment descriptors are called embeddings. The triplet loss $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = max\left[\left(dist(e_i^a, e_i^+) - dist(e_i^a, e_i^-) + \delta\right), 0\right], \tag{3}$$

where e_i^a is the anchor embedding, e_i^+ is the positive match and e_i^+ the negative match respectively. The positive embedding is computed from the same origin as the anchor, but of another sample of it (which would be a descriptor computed from the same line segment, but of another image in our case). The negative embedding is from another origin (from another line segment in our case). The anchor is compared to the other two embeddings using a distance measure $dist(\cdot)$, which, in our case, is the squared euclidean distance between the descriptors. During training, the triplet loss function aims to achieve a small distance between anchors and the corresponding positive embeddings, and a larger distance between anchors and negative embeddings. For a faster convergence of the training, negative embeddings, that are closer in distance to the anchor, are favored (hard negatives) in the triplet selection process, over those which already have a larger distance. But to make sure that not only the most difficult ones are selected, which are possibly outliers, a margin δ is used. This margin promotes a certain minimum distance difference between the distance of the anchor and the negative embedding, and the distance of the anchor and the positive embedding, and, therefore, defines which triplets are difficult and which are not.

3. Results

Parameters: We created the wavelets using the following parameters: $\sigma = 1$, $\theta = \{0, \frac{\pi}{2}\}$, $\lambda = 2 \cdot \sigma$, $\phi = \frac{\pi}{2}$, $\gamma = 1$. The kernel is set to a size of only 5×5 resulting in a faster convolution compared to larger sizes. The results show that this choice is not too coarse.

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

We use two wavelet kernels, one with $\theta = 0$ and the other with $\theta = \frac{\pi}{2}$. They are oriented 90° to each other and thereby respond differently to vertical/horizontal information in the image. This is also depicted in Fig. 1. For each line segment the layers from the multi-scale pyramid are convolved with each of the two kernels, as listed in Table 1 and stacked on the cutout stack. Table 2 shows the values of the Gabor Wavelet kernel with $\theta = 0$. The kernel with $\theta = \frac{\pi}{2}$ is the same, just transposed. In our implementation we scale the kernel down by factor 2π to ensure staying in data type range.

Table 2: *Kernel of the Gabor Wavelet with* $\theta = 0$.

0.0183	0.0821	0.1353	0.0821	0.0183
-0.0821	-0.3679	-0.6065	-0.3679	-0.0821
0.1353	0.6065	1	0.6065	0.1353
-0.0821	-0.3679	-0.6065	-0.3679	-0.0821
0.0183	0.0821	0.1353	0.0821	0.0183

Setup: We conducted our experiments on a Core i7-4700MQ mobile CPU. The preprocessing during testing took about 1 *ms*, and the descriptor calculation by the network also took about 1 *ms* per line, both on the CPU. The calculation time could be decreased by integrating the preprocessing directly into the framework of the network, and running it on a GPU. The training was conducted on a cluster using one GeForce GTX 1080 where we trained for about a week. The bottleneck during training was not the graphics card's performance, but the loading of the data, which could be accelerated by more prefetching if training time was a concern. The ResNet we used is similar to the one in [LSS19], it is like a 10-layer ResNet variant. We achieved the best training results with batch normalization deactivated. But this also contributed to a slower convergence. The network with the best results was set to a descriptor size of 512 Bit. A fixed line length of 100 pixels worked well during training. In normal usage, the line size does not have to be fixed, but it is beneficial during training to help the network converge. Additional explanation is given at the end of Section 2.2. The EDLine detector's [AT11] OpenCV [Bra00] implementation was used for the line detection. To examine our line descriptors matching performance, we compare it against the results of three other methods which we will elaborate in more detail now.

LBD: The Line Band Descriptor from Lilian Zhang and Reinhard Koch [ZK13] is an analytical approach in which they first crop out a rectangular area which is centered around the line. Then they use differently parametrized, one dimensional Gaussian filters and convolve along the line, with each of them defining a so called *Band*. The values of these filtered *Bands* are then averaged and used to construct a descriptor. A disadvantage of this method is that structural information is lost as the values are averaged along the line. Nevertheless, it outperformed all other methods when it was released, and, up to today, it is still among the best line descriptors, and the only one implemented in the OpenCV library [Bra00].

DLD: The Deep Learning Based Line Descriptor for Line Feature Matching is a method published in 2019 by Lange et al. [LSS19]. Like the LBD it crops out a rectangular area which is centered around the line. The size of the area was set to the same size as for the LBD. One goal was to find out if one could do better than the LBD having the same input and also the same (output) descriptor size. It turned out that a better matching performance can be achieved by using the DLD.

LLD: The Learnable Line Segment Descriptor for Visual SLAM from Alexander Vakhitov and Victor Lempitsky [VL19] is also a machine learning based approach. In contrast to the previously described methods their first processing step directly runs the whole image through the network. They receive a matrix of the same width and height as the input image, but with a depth of 64 channels from originally one (gray) image channel. This means that the network calculates some kind of region, or image descriptor. On that basis, they build line descriptors by splitting each line segment into multiple segments, calculating a feature vector for each subsegment using bilinear interpolation, and obtaining the final descriptor by averaging over the feature vectors. To calculate one large descriptor on the whole image all at once can be faster than processing one descriptor for each line segment successively, depending on the number of lines in the image and also depending on the used hardware. Another advantage of this approach is the possibly large receptive field it provides for each line feature, with the actual size depending on the choice of layers. However, one question is how well the network is able to specify on the feature type of line segments, when trained that way. Lines have a distinct direction and also a left and right side. Especially lines that lie on the edge of an object have one side on the object itself and the other side from the background. It is unclear if the network will be able to make use of this information when it receives the lines all at once in one image and oriented in many different directions. Most of the network's layers are convolutional layers which have proven to work well on images [LB*95] [LBBH98] [SSP*03] and can cope with translations. But rotations are usually more difficult. Also, averaging over multiple sub feature vectors incorporates the danger of loosing previously contained structural information.

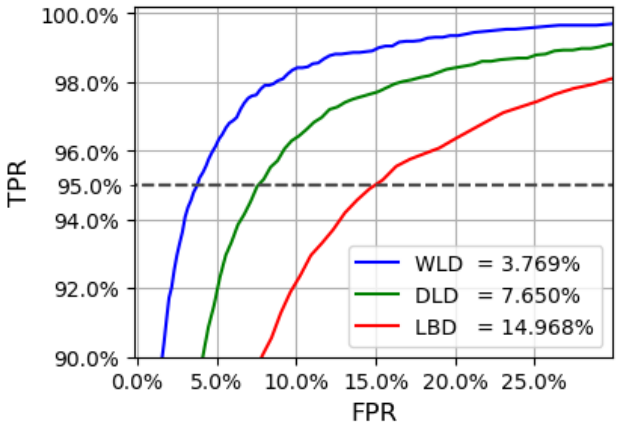

Figure 2: *This plot shows the receiver operating characteristic curves of our method (WLD), vs the DLD, vs the LBD over all of the eight labeled Middlebury Images [SHK*14]. The legend shows the False Positive Rate (FPR) at the True Positive Rate (TPR) of 95% (dashed line).*

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

Table 3: *Correct first best matches on images of the Middlebury Stereo Dataset (2014) [SHK*14].*

Method	Number of correct first best matches for each scene							
	Adirondack	Backpack	Bicycle	Classroom	Couch	Flowers	Piano	Vintage
WLD (Ours)	**280**	**953**	**839**	**427**	**232**	**102**	**329**	**439**
LLD	265	684	723	338	211	90	255	354
DLD	244	539	547	366	217	80	281	313
LBD	203	455	524	307	192	47	240	299
Labeled Matches:	318	1216	944	500	296	143	353	517

3.1. Performance on the Middlebury Stereo Dataset:

We selected eight images from the Middlebury Stereo Dataset (2014) [SHK*14] that we found to be interesting and challenging to benchmark our and other line descriptors' matching performance on them. They show different indoor scenes such as a classroom, or a computer lab, and also rooms with objects like a couch, a piano or a bicycle. To obtain the ground truth line matches of the stereo images, we first used the provided depth images and our automatic matching procedure, which we also used in the training, to get a set of initial line matches. Then we checked all these matches and added additional ones by hand. Overall the eight stereo images contain 4287 line matches.

Figure 2 shows the receiver operating characteristic curves of our method (WLD) compared to the DLD and the LBD. These resulting curves show the characteristic over the eight labeled Middlebury Images [SHK*14]. The closer to the left and the top one curve is, the better the performance of the method. An advancement of our WLD is clearly visible.
For methods like SfM, VO, etc. the number of correct first best matches is especially relevant. For this reason we list the first best matches of the compared methods for each scene in Table 3. The numbers indicate the amount of correct first best matches for all the lines of the left image matched with the lines of the right image. The bottom row lists the total number of labeled ground truth matches for each scene. The highest number of correct matches for each scene is shown in bold. We can see that the LBD is outperformed in every scene by the other three methods that are all machine learning based. The DLD approach scores second best in three of the scenes. The descriptor of Vakhitov and Lempitsky (LLD) yields the second best number of matches on five of the scenes. Our method outperforms all the other methods with the highest number of correct first matches in all of the eight scenes. While scoring only about 5% better than the second best on the *Adirondack* and the *Couch* scenes, relative to the total number of labeled matches, we achieve over 10% more of the possible matches on each of the other six scenes. Our WLD even matches more than twice as many lines than the LBD on the *Backpack* and the *Flowers* scenes.

3.2. Performance under difficult conditions:

In order to test the methods under difficult, but not unrealistic conditions, we took several photos of different scenarios and additionally augmented some of them to benchmark the methods on them. The results are shown in Table 4. The *Blurring*, *Noise* and *Jpeg Compression* scenarios show the same corridor scene which was augmented using Gimp [GIM] to test how well each of the methods can cope with blurring, noise and Jpeg compression artifacts respectively. We can see that the LBD is far behind on all three scenes. The LLD and DLD perform similarly well with the LLD doing somewhat better on all three augmented images. Our WLD achieves the highest number of matches in all three, matching even more than twice as many lines correctly when the image is augmented by blurring. This is probably due to the wavelets which are used in multiple scales. The higher scales will be altered by fewer extent from the blurring as they are already less sharp. The DLD performs best in the *Scale Change* scenario. It weights the near surrounding more than the WLD which seems to be slightly

Table 4: *Correct first best matches under difficult conditions.*

Method	Number of correct first best matches for each scenario							
	Blurring	Noise	Jpeg Compression	Scale Change	Viewpoint Change	Rotation	Tiles	Repeating Patterns
WLD (Ours)	**235**	**320**	**342**	10	**69**	**106**	**219**	**110**
LLD	113	203	300	6	33	6	190	77
DLD	99	168	273	**12**	44	100	210	49
LBD	58	111	194	3	25	81	152	49
Labeled Matches:	351	351	351	35	122	106	219	180

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

beneficial here, but the distance is not as large as it is to the LLD or the LBD. Our WLD matches all lines correctly in the *Rotation* scenario with the DLD being close and the LBD gets about four out of five lines correct. These three approaches make use of the directional information of lines, whereby they cope well with rotations. For the LLD they appear to be problematic. It is not supplied with images showing large rotations during training as they focused on "nearby frames in an input video-sequence" [VL19]. However, we think that the main reason for the lack of ability to deal with rotations comes from the point feature alike composition of the descriptor that does not make use of the directional information a line provides. The *Tiles* scene mostly shows a floor that is fully tiled. The intention here was to challenge the approaches with a scene that contains many similar lines, where making use of the structural information of the farther surrounding would be necessary to perform well. Here we can see that our WLD approach matches all lines correctly. But the other approaches, besides the LBD, also manage to match most lines correctly. Since this scene did not turn out to be as challenging as intended, we went outside to take some images of an Institute Building for a *Repeating Patterns* scenery. This scene is full of similarities like similar windows, rainwater pipes and roof endings. It is more challenging than the *Tiles* scene as we can see in the matching results. Our WLD approach was able to match 110 lines correctly, followed with some distance by the LLD successfully matching 77 lines. The DLD and the LBD only manage to score a bit more than a quarter of the labeled matches. This indicates that it is especially beneficial in scenes with repeating patterns like the *Repeating Patterns* scene to incorporate the farther surroundings as it is done by the LLD and by our WLD. Besides being close to the LBD in the *Scale Change* scenario, we can see that overall our WLD can handle difficult scenes better than the other methods.

4. Conclusion

In this paper we presented a new line feature descriptor which uses wavelets and machine learning for robust line descriptor matching. By using a multi-scale scheme, the receptive field around the line which contributes to the descriptor calculation is quite large. Inspired by the mammalian visual cortex, we used Gabor wavelets to preprocess this information in a meaningful way. A ResNet type neural network is trained to compute the line descriptor using a triplet loss function. In the results, we compared our method to other recent line descriptors on varying scenes and under difficult conditions. It showed that we were not able to improve the results for the scale change scenario compared to the DLD, on which this method is build upon. However, we were able to do better than the DLD and the other methods in all other scenarios, showing a great improvement on images with blurring, noise, viewpoint changes and repeating patterns which are typical difficult scenarios for feature matching. The improvement was also apparent on all of the Middlebury scenes. This proves that using a larger receptive field combined with Gabor wavelets is quite beneficial for descriptor based line feature matching. We already know that Gabor Wavelets, as they appear in the mammalian visual cortex, extract important information for detecting and describing structure and texture. While convolutional layers could learn this on their own, it is unknown how much data would be required to achieve similarly well generalized results. In this work Gabor Wavelets act like pre-trained filters, which favours convergence and generalization. Future work could investigate this further. The project will be published at: *https://github.com/manuellange/WLD*

References

[AT11] AKINLAR C., TOPAL C.: EDLines: Real-time line segment detection by edge drawing (ED). In *Image Processing (ICIP), 2011 18th IEEE International Conference on* (2011), IEEE, pp. 2837–2840. 1, 4, 5

[BP02] BRUNO E., PELLERIN D.: Robust motion estimation using spatial gabor-like filters. *Signal Processing 82*, 2 (2002), 297–309. 2

[Bra00] BRADSKI G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000). 5

[BSV05] BERNARDINO A., SANTOS-VICTOR J.: A real-time gabor primal sketch for visual attention. In *Iberian Conference on Pattern Recognition and Image Analysis* (2005), Springer, pp. 335–342. 2

[Dau85] DAUGMAN J. G.: Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A 2*, 7 (1985), 1160–1169. 2, 3

[DH71] DUDA R. O., HART P. E.: *Use of the Hough transformation to detect lines and curves in pictures*. Tech. rep., SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1971. 1

[DRMS07] DAVISON A. J., REID I. D., MOLTON N. D., STASSE O.: Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence 29*, 6 (2007), 1052–1067. 1

[DWLK19] DONG R., WEI Z.-G., LIU C., KAN J.: A novel loop closure detection method using line features. *IEEE Access 7* (2019), 111245–111256. 1

[Epi] EPIC GAMES: Unreal engine. URL: https://www.unrealengine.com. 3

[FB12] FILITCHKIN P., BYL K.: Feature-based terrain classification for littledog. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 1387–1392. 1

[FLG14] FUHRMANN S., LANGGUTH F., GOESELE M.: Mve-a multi-view reconstruction environment. In *GCH* (2014), Citeseer, pp. 11–18. 1

[Gab46] GABOR D.: Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering 93*, 26 (1946), 429–441. 2

[GIM] GIMP DEVELOPMENT TEAM: Gimp. URL: https://www.gimp.org. 6

[GM84] GROSSMANN A., MORLET J.: Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM journal on mathematical analysis 15*, 4 (1984), 723–736. 2

[GOGJ18] GOMEZ-OJEDA R., GONZALEZ-JIMENEZ J.: Geometric-based line segment tracking for hdr stereo sequences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), IEEE, pp. 69–74. 1

[GOZNM*17] GOMEZ-OJEDA R., ZUÃŚIGA-NOÃŃL D., MORENO F.-A., SCARAMUZZA D., GONZALEZ-JIMENEZ J.: PL-SLAM: a Stereo SLAM System through the Combination of Points and Line Segments. *arXiv preprint arXiv:1705.09479* (2017). 1

[GPK02] GRIGORESCU S. E., PETKOV N., KRUIZINGA P.: Comparison of texture features based on gabor filters. *IEEE Transactions on Image processing 11*, 10 (2002), 1160–1167. 2

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S. M.: Multi-view stereo for community photo collections. In *2007 IEEE 11th International Conference on Computer Vision* (2007), IEEE, pp. 1–8. 1

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

[HBL17] HERMANS A., BEYER L., LEIBE B.: In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017). 4

[HMB14] HOFER M., MAURER M., BISCHOF H.: Improving sparse 3d models for man-made environments using line-based 3d reconstruction. In *2014 2nd International Conference on 3D Vision* (2014), vol. 1, IEEE, pp. 535–542. 1

[HS12] HIROSE K., SAITO H.: Fast line description for line-based slam. In *BMVC* (2012), pp. 1–11. 2

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778. 4

[KKKM16] KWON Y. P., KIM H., KONJEVOD G., MCMAINS S.: Dude (duality descriptor): A robust descriptor for disparate images using line segment duality. In *2016 IEEE International Conference on Image Processing (ICIP)* (2016), IEEE, pp. 310–314. 1

[KL10] KIM H., LEE S.: Wide-baseline image matching based on coplanar line intersections. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), IEEE, pp. 1157–1164. 1

[KS01] KRÜGER V., SOMMER G.: Gabor wavelet networks for object representation. In *Multi-Image Analysis*. Springer, 2001, pp. 115–128. 2

[LB*95] LECUN Y., BENGIO Y., ET AL.: Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 10 (1995), 1995. 5

[LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324. 5

[Lee96] LEE T. S.: Image representation using 2d gabor wavelets. *IEEE Transactions on pattern analysis and machine intelligence 18*, 10 (1996), 959–971. 2, 3

[LLZ*14] LEE J. H., LEE S., ZHANG G., LIM J., CHUNG W. K., SUH I. H.: Outdoor place recognition in urban environments using straight lines. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (2014), IEEE, pp. 5550–5557. 1

[Low99] LOWE D. G.: Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision* (1999), vol. 2, Ieee, pp. 1150–1157. 1

[LRS19] LANGE M., RAISCH C., SCHILLING A.: LVO: Line only stereo Visual Odometry. In *2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (2019), IEEE. 1

[LSS19] LANGE M., SCHWEINFURTH F., SCHILLING A.: DLD: A Deep Learning Based Line Descriptor for Line Feature Matching. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), IEEE, pp. 5910–5915. 2, 5

[LWD10] LIU H.-M., WANG Z.-H., DENG C.: Extend point descriptors for line, curve and region matching. In *Machine Learning and Cybernetics (ICMLC), 2010 International Conference on* (2010), vol. 1, IEEE, pp. 214–219. 2

[LYL*16] LI K., YAO J., LU X., LI L., ZHANG Z.: Hierarchical Line Matching Based on Line–Junction–Line Structure Descriptor and Local Homography Estimation. *Neurocomputing 184* (2016), 207–220. 1

[Mal89] MALLAT S. G.: Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing 37*, 12 (1989), 2091–2110. 2, 3

[Mil13] MILFORD M.: Vision-based place recognition: how low can you go? *The International Journal of Robotics Research 32*, 7 (2013), 766–789. 1

[MVGV09] MOONS T., VAN GOOL L., VERGAUWEN M.: *3D Reconstruction from Multiple Images: Principles*. Now Publishers Inc, 2009. 1

[NNB04] NISTÉR D., NARODITSKY O., BERGEN J.: Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* (2004), vol. 1, Ieee, pp. I–I. 1

[NPVCL08] NEUBERT P., PROTZEL P., VIDAL-CALLEJA T., LACROIX S.: A fast visual line segment tracker. In *2008 IEEE International Conference on Emerging Technologies and Factory Automation* (2008), IEEE, pp. 353–360. 1

[PL04] PORWIK P., LISOWSKA A.: The haar-wavelet transform in digital image processing: its status and achievements. *Machine graphics and vision 13*, 1/2 (2004), 79–98. 2

[SHK*14] SCHARSTEIN D., HIRSCHMÜLLER H., KITAJIMA Y., KRATHWOHL G., NEŠIĆ N., WANG X., WESTLING P.: High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition* (2014), Springer, pp. 31–42. 5, 6

[SKP15] SCHROFF F., KALENICHENKO D., PHILBIN J.: 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. 4

[SSP*03] SIMARD P. Y., STEINKRAUS D., PLATT J. C., ET AL.: Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (2003), vol. 3. 5

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: exploring photo collections in 3d. In *ACM Siggraph 2006 Papers*. 2006, pp. 835–846. 1

[VGJMR12] VON GIOI R. G., JAKUBOWICZ J., MOREL J.-M., RANDALL G.: LSD: a Line Segment Detector. *Image Processing On Line 2* (2012), 35–55. 1, 4

[VL19] VAKHITOV A., LEMPITSKY V.: Learnable line segment descriptor for visual slam. *IEEE Access 7* (2019), 39923–39934. doi:10.1109/ACCESS.2019.2901584. 2, 5, 7

[WLW09] WANG Z., LIU H., WU F.: HLD: A robust descriptor for line matching. *2009 11th IEEE International Conference on Computer-Aided Design and Computer Graphics* (2009), 128–133. 2

[WNY09] WANG L., NEUMANN U., YOU S.: Wide-baseline image matching using line signatures. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 1311–1318. 1

[WS09] WEINBERGER K. Q., SAUL L. K.: Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research 10*, Feb (2009), 207–244. 4

[WWH09] WANG Z., WU F., HU Z.: MSLD: A robust descriptor for line matching. *Pattern Recognition 42*, 5 (2009), 941–953. 2

[Zha13] ZHANG L.: *Line primitives and their applications in geometric computer vision*. Universitätsbibliothek Kiel, 2013. 2

[ZK13] ZHANG L., KOCH R.: An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency. *Journal of Visual Communication and Image Representation 24*, 7 (2013), 794–805. 5

[ZK14] ZHANG L., KOCH R.: Structure and motion from line correspondences: Representation, projection, initialization and sparse bundle adjustment. *Journal of Visual Communication and Image Representation 25*, 5 (2014), 904–915. 1

[ZK17] ZAGORUYKO S., KOMODAKIS N.: Deep compare: A study on using convolutional neural networks to compare image patches. *Computer Vision and Image Understanding 164* (2017), 38–55. 2

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.