IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Learnable Line Segment Descriptor for Visual SLAM

ALEXANDER VAKHITOV[1,2] and VICTOR LEMPITSKY[1,2]
[1]Skolkovo Institute of Science and Technology Nobelya Ulitsa 3 Moscow, Russia 121205 (e-mail: alexander.vakhitov@gmail.com, v.lempitsky@skoltech.ru)
[2]Samsung AI Center, Moscow

Corresponding author: Alexander Vakhitov (e-mail: alexander.vakhitov@ gmail.com).

This work was supported by the Russian MES grant RFMEFI61516X0003.

ABSTRACT Traditionally, indirect visual motion estimation and simultaneous localization and mapping (SLAM) systems were based on point features. In recent years, several SLAM systems that use lines as primitives were suggested. Despite the extra robustness and accuracy brought by line segment matching, the line segment descriptors used in such systems were hand-crafted and therefore sub-optimal.

In this work, we suggest to apply descriptor learning to construct line segment descriptors optimized for matching tasks. We show how such descriptors can be constructed on top of a deep yet lightweight fully-convolutional neural network. The coefficients of this network are trained using an automatically collected dataset of matching and non-matching line segments. The use of the fully-convolutional network ensures that the bulk of the computations needed to compute descriptors is shared among multiple line segments in the same image, enabling efficient implementation. We show that learned line segment descriptors outperform previously suggested hand-crafted line segment descriptors both in isolation (i.e. for the subtask of distinguishing matching and non-matching line segments), but also when built into the SLAM system. We construct a new line based SLAM pipeline built upon a state-of-the-art point-only system. We demonstrate generalization of the learned parameters of the descriptor network between two well-known datasets for autonomous driving and indoor micro aerial vehicle navigation.

INDEX TERMS feature descriptors, line segments, robotic perception, simultaneous localization and mapping, SLAM

I. INTRODUCTION

MANY, perhaps most, robots operate in the environments that are created or heavily modified by humans, such as building interiors or city streets. Robust visual navigation and the associated task of visual simultaneous localization and mapping (SLAM) in such environments is therefore a crucial task. Most practically-used SLAM systems these days are based on point features. While point features are well-studied and are relatively easy to handle mathematically and algorithmically, parts of human-modified environments often have insufficient number of point features that are easy to track. At the same time, human-modified environments usually contain visually-distinctive straight line segments that can be detected and matched from multiple viewpoints as the robot moves in the environment. In such cases, the incorporation of line matching is able to boost the accuracy and robustness of motion estimation or SLAM considerably [1]–[4].

To match line segments across frames, a *line segment descriptor*, i.e. a function that maps the appearance of a line segment into a high-dimensional space suitable for distance-based matching is required. Importantly, a good line segment descriptor should be robust to the change of the endpoints, i.e. the descriptors for pairs of the overlapping segments of the same line should be similar. Under such condition, matching line segments can benefit a SLAM system as opposed to relying on point feature matching alone. Previous works use hand-crafted line segment descriptors such as the *Line Band descriptor (LBD)* in [5] or the *scale-invariant mean-standard deviation line segment descriptor (SMLSD)* in [6]. The handcrafted line descriptors are inspired by SIFT and work well in well-textured scenes but may be useless when used for repetitive and low-textured scenes.

Existing line descriptors are based on popular *scale-invariant feature transform (SIFT) point descriptors* [7]. While SIFT is a highly successful technique, its accuracy for point matching has been surpassed by descriptors based on machine learning [8]–[13]. The idea behind them is to

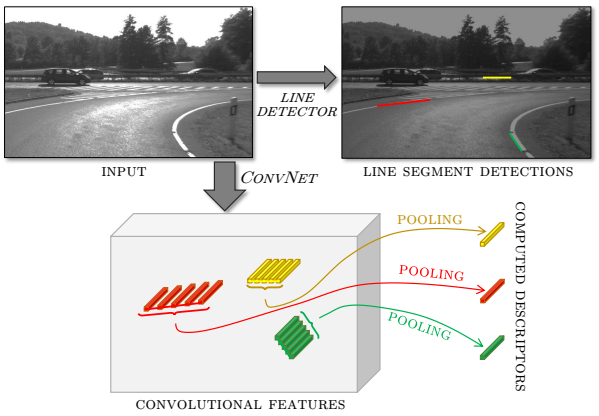

FIGURE 1. The architecture of our descriptor. The input image is passed through a fully-convolutional neural network that computes a high-dimensional descriptor for each pixel using multiple convolutions interleaved with non-linearities. In parallel, a standard line segment detector is applied. Each of the resulting line segments is then described using average pooling of a fixed number of feature vectors corresponding to a number of points sampled along the segment.

collect a dataset of matching and non-matching feature pairs and use such pairs to learn the parameters of the descriptor function in a discriminative way. While initial learning-based architectures were relatively shallow [8]–[10], the recent ones [11], [12] use gradient-based learning to train deeper architectures using Siamese architectures [13] and related loss functions. In general, learning-based descriptors are able to improve the accuracy and robustness of visual matching compared to SIFT and related hand-crafted techniques by better exploiting the statistics of visual data.

In this work, we show that learning-based deep descriptors can advance the accuracy of matching for line segment features in an analogous way to point features. Towards this end we design an architecture that computes discriminative descriptors for a set of line segments found in an image. The architecture is not specific to a particular line detector, as it can adapt to peculiarities of a certain detector in the process of learning.

At test-time, the proposed architecture takes an input image as well as a set of detected line segments, passes the image through a so-called fully-convolutional neural network [14] obtaining a stack of convolutional maps at the resolution comparable to the resolution of the input image. The descriptor of each individual line segment is then obtained using average-pooling of a fixed number of convolutional features located along the segment (Figure 1). Importantly, the input image is passed through a convolutional network only once, and the bulk of the computations are thus shared between multiple line segments, which makes our approach efficient (as opposed to a hypothetical approach that would run a deep network once per each line segment).

The training of the parameters of the convolutional network is performed on an automatically-mined dataset of matching and non-matching line segment pairs. In our implementation, such mining is performed on two popular datasets (KITTI [15], EuRoC MAV [16]). The mining uses the ground truth poses of frames provided with the datasets.

Our evaluation is then performed on the hold-out parts of these datasets (same parameters are used for both datasets, despite the significant difference between them). We evaluate the new descriptor, its variants, and the baselines at the task of distinguishing matching and non-matching line segments, as a component within a RANSAC-based motion estimation module and within a full SLAM system [17].

To summarize, our contributions are as follows:

- We introduce a new efficient deep convolutional architecture that generates learnable deep line segment descriptors.
- We show how to mine learning data for such a system from the dataset with known ground truth camera poses.
- We construct a point-and-line SLAM pipeline based on a state-of-the-art indirect stereo SLAM system using a minimal 3D line parameterization.[1]
- We evaluate the learned line segment descriptor and show its advantage over previously proposed line segment descriptors across several tasks including full SLAM.

The remainder of the paper is organized as follows. After the related work discussion in Section II, we cover the details of our approach, including the architecture of the network, the learning process, the mining process, and our variant of point+line SLAM system in Section III. We then present the evaluation of the learned line segment descriptor in Section IV. We conclude with a short discussion in Section V.

II. RELATED WORK

a: Line segment matching

While line segment matching is a less studied topic than point feature matching, one can still identify three different classes of matching approaches. Historically the first were the approaches which rely on geometric information for the initial matching. They include an early method for local tracking of straight line segments [18]. In [19], Schmid et al. assume known epipolar geometry between images and use line segment endpoints to put segments into correspondence. They match the lines by calculating average cross-correlation between the image neighbourhoods. Geometric verification of line correspondences proposed in [20] assumes known trifocal tensors between image triplets. In general, these methods require significant additional information about intrinsic and extrinsic calibration of cameras, which constrains their applicability.

The second group of approaches match line groups rather than individual segments. Thus, Lourakis et al. [21] suggests"two lines + two point" method for line matching on planar surfaces. In [22], the lines are grouped by spatial proximity. The signatures of groups are then computed and matched. Kim et al. [23] operate on groups of coplanar lines and their intersections, while [24] proposes an iterative

[1]https://github.com/alexander-vakhitov/ORB_SLAM2_LLD

matching process using structural information collected from different line neighbourhoods. Triplets of line matches are used in [25] and [26] proposes a joint line-point matching process. In [27], a shape descriptor that uses relative position of line segments in an image is used to match satellite images.

Finally, most related to ours are the methods that match pairs of line segments across two views. Like in point matching, such methods construct descriptors for individual line segments and match segments based on the distance between these descriptors. Bay et al. [28] use color histograms of lines ignoring the surrounding image texture. In [29] the authors propose a mean-standard deviation line segment descriptor (MSLD), which divides segment's neighborhood into a grid of overlapping cells and describes individual cells using gradient direction histograms (an approach inspired by SIFT-like point descriptors). MSLD was evaluated in conjunction with the Canny edges separated in high curvature points ([28]) in various image modification scenarios (blur, JPEG, rotation or viewpoint change, additive noise, illumination) showing its superiority to the epipolar geometry-based Schmid's method [19]. The multiscale version of MSLD is described in [6]. Zhang et al. [5] propose a line-band descriptor (LBD) that computes gradient histograms over bands that are parallel to the considered line segment.

b: Learnable point-feature descriptors

While a large number of hand-crafted descriptors have been proposed [30] since the original SIFT work [7], in the recent years the focus has been on learnable descriptors. The datasets for learning have been harvested using the Phototourism dataset [31], using which the authors have originally tuned the parameters of the popular hand-crafted DAISY [32] desriptor. An alternative dataset collection involved synthetic images [33].

By leveraging large automatically-generated patch pair datasets, learning of feature point descriptors with lots of tunable parameters becomes possible. Thus, [34] learns two layer descriptors using convex optimization. The works that used non-convex local optimization to tune the parameters of deep networks include [10]–[12], [35]. In [36], a compact binary patch descriptor for image matching and patch retrieval is learned and evaluated. It is worth mentioning, that despite overwhelming success of deep learning for high-level computer vision [37] as well as for some dense matching tasks such as stereo and optical flow [38], the application of deep learning to point-feature descriptors has not improved state-of-the-art radically (yet). Indeed, according to a recent comparison [39], the performance of hand-engineered features is still competitive with that of deep learning-based descriptors in some circumstances.

c: Learnable global descriptors

Finally, we note that our approach to line description is closely related to methods that build global image or region descriptors by pooling convolutional features over the entire image [40] or bounding boxes [41].

d: Point-and-line SLAM

In [2], the authors use a minimal 3D line parameterization [42] and propose a line-only stereo SLAM pipeline. The system however performed poorly on a publicly available dataset due to reliance on availability of a significant number of line features in each video frame. Recently, several studies analyzed the effect of point-line fusion in a modern optimization-based SLAM [4], [43]–[46]. In these works non-minimal line parameterization is used as it allows for easier integration with point-based pose graph optimization. Non-minimal line parameterizations however pose a problem because an infinite set of parameter values can encode the same 3D line, while resulting in different reprojection cost values. As a fix, [45] proposes a 'line-cutting' algorithm essentially restricting the set of valid line parameters and showing improvement in accuracy. In this work, we show that an approach of using a minimal line parameterization can be used with success on standard SLAM benchmarks as well.

III. LEARNABLE LINE DESCRIPTOR

In this section, we provide details of our approach. We start with describing the architecture of our descriptor, and then describe how it can be learned from a dataset of matching and non-matching pairs of line segments. Given the learning-based nature of our work, mining matching and non-matching line segment is an important part of the approach. Our way to integrate the learned line segment descriptor into a visual SLAM system is described in the end of the section. Section IV then contains the evaluation of the descriptor. *Note:* below, we denote images and stereo-images using bold uppercase, 3D objects using uppercase and 2D objects using lowercase, $\|\cdot\|_p$ denotes the l^p norm.

A. DESCRIPTOR ARCHITECTURE

We start with the discussion of the architecture of our descriptor. At test time, a (monocular) grayscale image $\mathbf{I}$ is passed through a convolutional neural network f_θ, where θ denotes parameters learned on the training set. The network has the "fully-convolutional" architecture detailed below, and the result of such processing $\mathcal{F} = f_\theta(\mathbf{I})$ has the same spatial dimensions as the input image $\mathbf{I}$ and 64 channels. Effectively, the mapping f_θ assigns each image pixel (x, y) a $q = 64$-dimensional feature vector $\mathcal{F}(x, y) \in \mathbb{R}^{64}$.

Given the convolutional representation $\mathcal{F}$, the descriptors of individual line segments are then computed using simple pooling operations. In more detail, given a line segment l, we split it uniformly into T subsegments (where T is the parameter of the algorithm). We then choose a center of each subsegment and get a feature vector for it using bilinear interpolation [47]. The line segment descriptor $d^\theta(l)$ is then obtained by averaging T feature vectors (of dimensionality 64) corresponding to the sampled points. As will be seen in the experiments, $T \geq 5$ provides a good choice.

The architecture of the network f_θ is given in Table 1. The architecture is derived from the *L2-Net* proposed by Tian et al. [11] for computing point descriptors. The fol-

lowing modifications to their architecture have been applied. A pair of a convolution and a strided convolution layers have been added. We have also inserted a $8\times$ upsampling layer before the normalization layer, which ensures that the representation $\mathcal{F}$ and the input image $\mathbf{I}$ have the same spatial resolution. The filter size of the last convolution was changed from 8×8 to 7×7. Similarly to L2-Net, our architecture terminates with the normalization layer, which brings the descriptors of each pixel to unit l^2-norm. We call our architecture *LLD-Net* and verify some of the choices behind it experimentally.

In general, an attractive property of our approach is that the convolutional network f_θ is applied only once to the image $\mathbf{I}$ irrespective of the number of line segments detected in $\mathbf{I}$. As a result, it takes only 17 milliseconds to compute our descriptor for an image pair on a 1080Ti GPU card. Further significant speed-ups are possible through such techniques as tensor factorization/separable convolutions [48], low-bit quantization [49], group sparsification [50].

B. LEARNING THE DESCRIPTOR

Our primary goal is to learn a line segment descriptor that works well within a SLAM or a visual odometry pipeline, which mostly considers matching nearby frames in an input video-sequence. We are thus interested in a descriptor that performs best when matching is done across small temporal spans, e.g. within five frames. We therefore form a dataset of mini-sequences consisting of 11 subsequent stereo-images from the training stereo-sequences, so that the temporal distance between the middle frame pair and any other frame pair in a mini-sequence is five or less. The mini-sequence thus contains $2 \times 11 = 22$ monocular images.

The convolutional network f_θ is then trained on mini-batches of image pairs ($\mathbf{I}$, $\mathcal{J}$). Each mini-batch consists of the left image of the middle stereopair of the mini-sequence (denoted in a subsequent derivation as $\mathbf{I}$) and a subset of the remaining 21 images in the mini-sequence. We denote the whole subset as $\mathcal{J}$, and we denote with $\mathbf{J}$ individual images from this subset. We sample $b \leq 21$ images as $\mathcal{J}$ thus turning a mini-sequence into a mini-batch of b image pairs ($\mathbf{I}$, $\mathbf{J} \in \mathcal{J}$).

Next, we assume that a line segment detector has identified a number of candidate line segments in each image. During the learning process, for each 2D line segment l in a sampled image $\mathbf{I}$, we compute a descriptor $d^\theta(l) = AvgPool(f_\theta(\mathbf{I}, l))$ using the current state of the network.

We now describe the process of triplet sampling for line segment candidates, where the first two elements of the triplet correspond to the descriptors computed for the projections of the 3D segments of the same 3D line (see Fig. 2, bottom). Let L be a 3D line. Let l_a be a 2D line segment which is a projection of a segment of L on the image $\mathbf{I}$. On some of the images $\mathbf{J} \in \mathcal{J}$ there can be 2D line segments $l_{+,j}$ that are projections of the segments of L onto $\mathbf{J}$. For each $\mathbf{J} \in \mathcal{J}$ there is also a set $\mathbf{l}_{-_j} = \{l_{-,j,i}\}_i$ containing the 2D line segments that are not the projections of any segment of L.

	Layer block	Spatial size	Num. output maps
1	Conv+BN+Relu	3x3	8
2	Conv+BN+Relu	3x3	8
3	Conv+BN+Relu	3x3, stride=2	16
4	Conv+BN+Relu	3x3	16
5	Conv+BN+Relu	3x3, stride=2	32
6	Conv+BN+Relu	3x3	32
7	Conv+BN+Relu	3x3, stride=2	64
8	Conv+BN+Relu	3x3	64
9	Conv+BN	7x7	64
10	Upsampling	8x8	64
11	l_2-normalization	1x1	64

TABLE 1. Learnable line segment descriptor network (LLD-Net) composition. 'Conv' denotes convolution, 'BN' denotes batch normalization, 'Relu' denotes rectified linear unit. In general, the architecture is inspired by the L2-Net [11] with few modifications. In the third column, we show the spatial kernel size of the convolutional operation, as well as the upsampling kernel size. The fouth column shows the number of output maps after each block.

For each L and $\mathbf{J} \in \mathcal{J}$ we then consider the following triplet objects. First, the *"anchor"* descriptor $d^\theta_a = d^\theta(l_a)$, then a *matching* descriptor $d^\theta_+ = d^\theta(l_{+,j})$, and finally we consider the whole *non-matching* set $\mathbf{d}^\theta_- = \{d^\theta(l_{-,j,i})\}_{l_{-,j,i} \in \mathbf{l}_-}$. The process for finding the matching line segment and identifying the set of non-matching line segments is detailed in Section III-C.

We then compute the hard-negative triplet loss proposed in [51], which in our case is $\mathcal{L}(\theta, \{d^\theta_a, d^\theta_+, \mathbf{d}^\theta_-\}) =$

$$\left[m + \|d^\theta_a - d^\theta_+\|^2 - \min_{d^\theta_- \in \mathbf{d}^\theta_-} \|d^\theta_a - d^\theta_-\|^2 \right]_+, \qquad (1)$$

where $[x]_+ = \max\{0, x\}$. The $\min$ operator ensures that the loss focuses on the hardest negative example from the negative set. The margin parameter m is set to 0.5 in our experiments.

We split KITTI [15] and EuRoC [16] sequences with available ground truth into a training and validation set and a test set as described in the experiments section. To sum up, the training process uses batches of b image pairs, computes the descriptors of all line segments in all chosen frames, and then proceeds by sampling the triplets for which the loss (1) is computed and accumulated. The cumulative loss is then back-propagated so that the parameters θ can be updated.

C. MATCH FILTERING

The learning process described above is the main contribution of this work. However to succeed it requires a large dataset of matching and non-matching pairs. To avoid the costly annotation process, we mine matching and non-matching pairs in each triplet automatically from training sequences, utilizing the ground truth camera positions (see Fig. 2, top). This can be regarded as training with weak supervision.

For each image, we detect straight line segments in the images using the *EDLines* line segment detection algorithm [52] over the image pyramid of four levels with the scale step of $\sqrt{2}$. We reject the 2D line segment detections with the

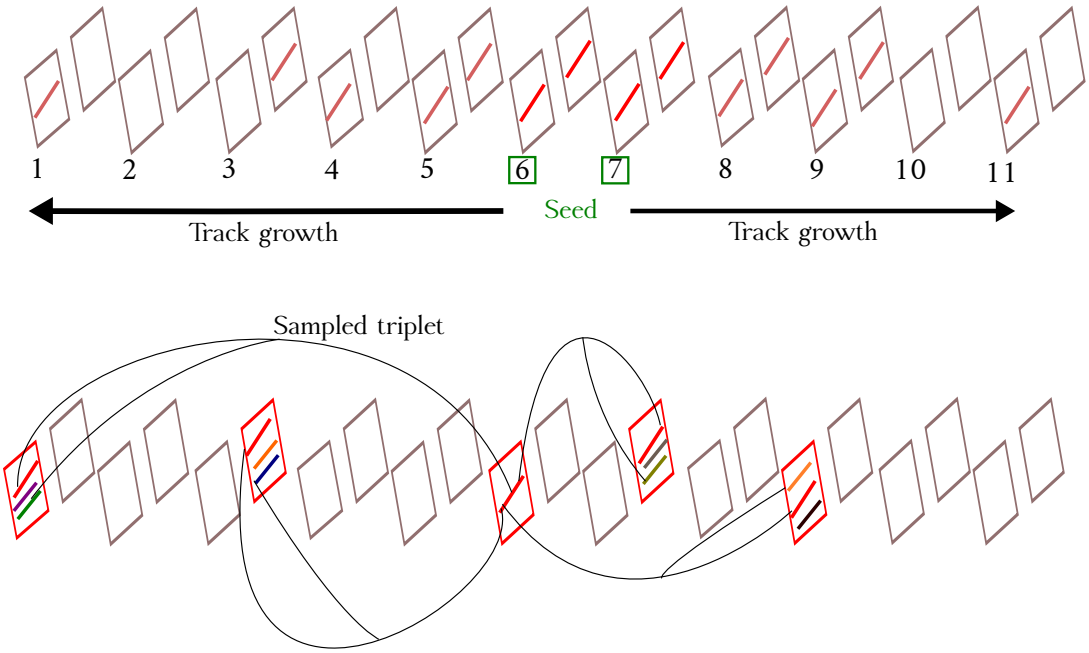

FIGURE 2. **Top:** Mechanism of track formation during dataset generation. For each mini-sequence, we start with finding a "seed" consisting of geometrically consistent quadruplets of line segments. We take the views of the middle stereo pair and the adjacent (the seventh) stereo pair and mine the segments having matches in all the four views. Next, for each track we triangulate the 3D line and project it on the other frames of the mini-sequence and look for the matching segments close enough to the projections ("track growth"). **Bottom:** Mechanism of image and line segment triplet sampling during training. A mini-batch of b images is generated from a single randomly chosen mini-sequence. It always includes an anchor image, which is the left image of the middle (the sixth) stereopair, and $b-1$ other randomly sampled images of the mini-sequence. At the anchor image, an anchor descriptor is formed from a line segment (red). In the mini-batch images, the matching line segments are found (red) as well as non-matching (various colors). Among the non-matching examples, the hardest, i.e. the closest to the positive example, is found. The triplets are formed from the anchor, matching and hardest non-matching line segments.

length less than w pixels (as measured in the original image resolution). Note that our algorithm can work with any line segment detector, and can adjust to its peculiarities during the descriptor learning process.

Below we describe the mining of positive (matching) pairs from a given mini-sequence. Such mining consists of two stages: *seed construction* and *track growth*. Seed construction uses the middle (the sixth) and the adjacent to it (the seventh) stereo pairs. Track growth uses the remaining frames of the mini-sequence. The result of this process is a set of tracks $\mathcal{T} = \{t_i\}_{i=1}^{n}$, where each track is a set of triplets $t_i = \{(\mathbf{I}_j, l_j, \mathbf{l}_\mathbf{j}^-)\}_{j=1}^{n_i}$ (to remind, the line segment $l_j \in \mathbf{I}_j$ is a positive detection and a set $\mathbf{l}_\mathbf{j}^-$ consists of negative matches).

For an image $\mathbf{I}$, consider a 3D line L and a line segment l connecting 2D endpoints a, b. Let the points $X, Y \in L$ be the 3D points on line L and let x, y be their projections onto $\mathbf{I}$ using the ground truth camera P. We choose X and Y such that x and y minimize the distance to a and b in the image coordinates. We then call a 3D segment (X, Y) of the line L *a 3D reprojection* of l onto L. We denote the clockwise angle between l and a positive horizontal axis of image coordinates as $\phi(l)$. We define the 2D-3D *maximal* line distance as $\rho(l, L) = \max\{\|x-a\|, \|y-b\|\}$ and the l^2-distance as $\rho_2(l, L) = (\|x-a\|^2 + \|y-b\|^2)^{1/2}$. The following conditions are checked in the different steps of the dataset mining algorithm.

1) *Cheirality* holds if a 3D reprojection of l onto L is in front of the camera that has been used to detect l by the line detector.
2) *Reprojection* condition holds if $\rho(l, L) < \epsilon$ for a predefined threshold $\epsilon = 3$ pixels in the image pyramid level corresponding to l.
3) *Angle consistency* condition holds if $\|\phi(l) - \phi'\| < \delta$ for a predefined threshold $\delta = 10°$ and angle ϕ'.
4) *3D overlap* condition holds if $|(X, Y) \cup (X', Y')| / \|X - Y\| > \xi$ for a predefined 3D segment (X', Y'), $X', Y' \in L$ and a threshold $\xi = 0.25$.

Failing at least one of the above-listed conditions (criteria) suggests that the match between L and l is not certain and maybe accidental, and therefore should not affect the learning process.

a: Seed construction

We start to construct the dataset from the temporally middle (the sixth) image pair. To check if the set of 2D lines corresponds to a 3D line one needs at least three images. We therefore use one more adjacent stereo pair (the seventh) in this phase. We thus take two sequential stereopairs being the sixth and the seventh pairs within the minibatch. We denote

the corresponding left-right frames as $\mathbf{I}_l, \mathbf{I}_r$ and $\mathbf{J}_l, \mathbf{J}_r$. Given the set of line segment detections in $\mathbf{I}_l$ and $\mathbf{I}_r$ we consider all possible pairwise matchings. For every possible pair (l_l, l_r) we triangulate a 3D line L. We then check the Cheirality for l_l, L and l_r, L, and check Angle Consistency for l_r with $\phi' = \phi(l_l)$. If a pair passes both checks, we seek corresponding line segments from $\mathbf{J}_l$ and $\mathbf{J}_r$. We choose $l_l^* \in \mathbf{J}_l$ and $l_r^* \in \mathbf{J}_r$ with minimal $\rho(l_l^*, L)$ ($\rho(l_r^*, L)$) satisfying the Reprojection and 3D overlap conditions.

We add a new track for each quadruplet (l_l, l_r, l_l^*, l_r^*). We form the negative set as $\mathbf{l}_\mathbf{j}^- = \{l_j \in \mathbf{I}_j : \ \rho(l_j, L) < \psi\}$ for $\psi = 12$ pixels.

The seed construction procedure takes around a second to process a pair of stereo-pairs with few hundred line segment detections in each frame.

b: Track growth

In the beginning of this stage, each track has exactly 4 elements, because at the seed construction stage we created a track only if there were projections of some 3D segments of the same 3D line in every view of the two stereo pairs. We process the frames of minibatch $\mathbf{J}$ not used in a seed construction stage in the order of increasing absolute time difference from the middle (the sixth) frame pair in the following way. For each line L_i we use the Reprojection condition to find possible matching candidates. Then the segment l_j with the lowest $\rho(l_j, L_i)$ among those not belonging to any other track is picked and a triplet $(\mathbf{J}, l_j, \mathbf{l}_\mathbf{j}^-)$ is added to the track ($\mathbf{l}_\mathbf{j}^-$ is formed as before). The 3D line representation L_i is refined by minimizing the sum of $\rho_2^2(l_j, L)$ for the whole track.

The result of the track growth process provides us with positive matches that can then be used to construct triplets and to train the LLD network.

D. ORB-SLAM2 WITH LINE FEATURES

As a part of our contribution, we add line segment features to the ORB-SLAM2 [17] pipeline making it essentially a point+line fusion system. We match the line segment features using a line descriptor. Next we describe the algorithms we have added to the ORB-SLAM2 pipeline. Note that this part of our work is independent of the concrete descriptor (and we use the same procedure for the baseline system that uses LBD descriptors). For line segments l_i, l_j, both for our LLD descriptors and for the real-valued LBD descriptors, we use the standard Euclidean distance $\mu(d^\theta(l_i), d^\theta(l_j)) = \|d^\theta(l_i) - d^\theta(l_j)\|_2$ in the matching process.

a: Segment detection and stereo matching

The incorporation of line features starts with line detection in a stereo-pair containing left and right images $\mathbf{I}_l$ and $\mathbf{I}_r$. As during learning, we filter out lines shorter than w pixels (in the zeroth level of an image pyramid). We form a non-overlapping 50×50 grid in the space of 2D lines where each line is represented by a pair (γ, ν) where γ is the angle between the line and the horizontal image axis, and ν is the distance from the line to the image center. From each cell in the grid we keep only the longest N_g lines. We iterate over the detections from $\mathbf{I}_l$, and match them to the detections from $\mathbf{I}_r$ in a greedy way. For each line segment l_l from $\mathbf{I}_l$ we select the not-yet-matched line segments $\{l_{ri}\}$ from $\mathbf{I}_r$ that fulfill the Cheirality condition and choose among the selected segments the one with the lowest descriptor distance $\mu(d^\theta(l_l), d^\theta(l_{ri}))$. We do not store the detections l_l having no matches in the right frame.

b: Track initialization

Next, we describe the initialization of the line segment tracks ('addition to the map' in the terminology of [17]). Generally, the map contains the geometric features participating in the bundle adjustment and thus affecting the final estimates of the robot location.

As with the point features, we add lines to the map when a new keyframe is created. Each matched pair (l_l, l_r) of line detections from left and right frames is eligible for map addition if these detections do not belong to some other tracks. We triangulate the line L from l_l and l_r. If there exists an additional line segment from a previous frame such that the Reprojection criterion (for $\epsilon = 6$ pixels) and 3D overlap conditions are fulfilled, we initialize the track. By checking these conditions, we essentially use geometric validation to discard outliers.

c: Matching a detection to an existing track

Given a new stereo-image, and a line segment detection l_{li} in the left frame matched to the detection l_{ri} in the right frame, we try to match it to the 3D lines that are already in the map. We reproject each 3D line L corresponding to a track in the map onto both frames and check the Reprojection criterion for (L, l_{li}) and for (L, l_{ri}). In the case when several 3D lines in the map are matched to the detection, we choose the medoid of the set of descriptors for line segments forming a track, d_{med}, and pick the detection with the lowest value of $\mu(d_{med}, d^\theta(l_i))$.

d: Map line culling

The ORB-SLAM2 system maintains the set of active elements of the map that are used for local bundle adjustment procedure. Each line segment element stays active after creation, as discussed above, until removed. At each time moment, we remove the line segment tracks from the active map, if they have less than four matched 2D line segments in three recent key frames (each key frame can have upto two matched line segments, one in each of the two frames). The track is also removed from the active map, if the length of the entire 3D line segment reprojected from the first detection of a track is out of the $(0.2, 50)$ meters range. Thus, line segments that are too short or too long are removed from the active map.

e: Local Bundle adjustment

The elements in the active map (both points and line segments) participate in the local bundle adjustment procedure. Using line segments within the bundle adjustment requires the choice of the line parameterization in 3D. We use the minimal 3D line parameterization inspired by [42].

In particular, consider the parameterization, when a line is parameterized by a 3D points X and a 3D vector Y, so that the line contains the points $\{X + tY \,|\, t \in \mathbb{R}\}$. To fix the gauge freedom in this case, we may impose the conditions $X^T Y = 0$, and $\|Y\| = 1$. Such conditions are however hard to impose during bundle adjustment.

To perform the updates of the line parameters, we therefore construct a new minimal parameterization of L by a rotation matrix and a constant. More precisely, we form a rotation matrix $\mathtt{R}(L) = [Y,\ X/\|X\|,\ Y \times X/\|X\|]$ and compute a constant $\alpha(L) = \|X\|$. The rotation matrix and the constant uniquely define the line. We further parameterize $\mathtt{R}(L)$ with a quaternion. The proposed parameterization has the same benefits of being minimal as described in [42]. It does not require additional constraints or introduce gauge freedoms. At the same time, it leads to a nontrivial mechanism of projecting a line onto an image described next. The new rotation+constant line parameterization is used during optimization within bundle adjustment.

To project L onto an image plane, we decode X and Y from $\mathtt{R}$ and α. We then use the function $\pi_L(L, \beta)$ that computes the projection of the 3D line L into a 2D line l defined in homogeneous 2D coordinates using the camera parameters β:

$$\pi_L(L, \beta) := l_h / \|l_h^{(1:2)}\|, \quad l_h = \pi(X, \beta) \times \pi(X + Y, \beta), \tag{2}$$

where $\pi(\cdot, \beta)$ projects a point onto a camera parameterized by β, $l_h^{(1:2)}$ is a 2-length subvector of l_h starting from the 1^{st} coordinate. For the bundle adjustment we use the following formulas to compute the Jacobians of the parameter updates. First of all, for the quaternions we use the Jacobians provided by the Lie $SO3$ algebra. The Jacobians of X, Y with respect to the rotation matrix and the scalar α are directly obtainable from the parameterization definition. If we denote as $J_{\pi,X}(\cot)$ the Jacobian of the perspective projection with respect to the point parameters, then the Jacobians of l_h with respect to X and Y are given by the following formulas:

$$\frac{\partial l_h}{\partial X} = [\pi(X, \beta)]_\times J_{\pi,X}(X+Y) - [\pi(X+Y, \beta)]_\times J_{\pi,X}(X), \tag{3}$$

$$\frac{\partial l_h}{\partial Y} = [\pi(X, \beta)]_\times J_{\pi,X}(X+Y), \tag{4}$$

The Jacobian of the projection function with respect to the l_h vector is given by:

$$\frac{\partial \pi(L, \beta)}{\partial l_h} = \frac{1}{\|l_h^{(1:2)}\|} E - 2 \frac{1}{\|l_h^{(1:2)}\|^3} l_h [l_h^{(1:2)}\ 0]^T, \tag{5}$$

Finally, the Jacobian of $\pi(L, \beta)$ with respect to the line parameters update can be directly obtained using the chain rule.

In order to obtain the value of our parameterization, we compute Y and then X from a pair of the 2D line segments detected at two different cameras. To compute Y, we note that it is a unit norm direction vector of a 3D line, so $Y = l_i \times l_j / \|l_i \times l_j\|$, where l_i and l_j are 2D line equations in the normalized coordinates. Then we compute X by forming a linear system of the constraints

$$\begin{cases} l_i^T (R_i X + t_i) & = 0, \\ l_j^T (R_j X + t_j) & = 0, \\ Y^T X = 0, & \end{cases} \tag{6}$$

where R_i, R_j are rotation matrices and t_i, t_j are translation vectors for the cameras.

The proposed approach to line parameterization is similar to the one proposed in [42] but differs from it slightly. Whereas they encode $\|X\|$ as a ratio of elements of a 2D rotation matrix, we use the parameter α to encode the length of $\|X\|$ explicitly. Likewise, we use the line direction vector Y, whereas [42] use a normal N to the plane joining O and L. Note that while the endpoint-based parameterization commonly used in the point-line SLAM, e.g. [4], is non-minimal, it leads to simpler projection equations.

In our experiments, we do not modify the point feature processing pipeline from the original point-only ORB-SLAM2. We thus retain the point descriptor, the point triangulation and point projection functions, and other parts of the pipeline related to point feature processing. The bundle adjustment objective contains the same terms as in ORB-SLAM2 and is augmented with line feature reprojection residuals defined as follows. Let $h_\nu(\cdot)$ be a Huber cost with threshold ν, let X_L be the 3×2 matrix with homogeneous coordinates of the detected line endpoints, and let Σ^{-1} be the 2×2 information matrix. We use the following term in the local bundle adjustment for line reprojection error:

$$\rho_h(L, l, \beta, \Sigma^{-1}) = \lambda h_\nu(X_L^T \pi_l(L, \beta) \Sigma^{-1} \pi_l^T(L, \beta) X_L), \tag{7}$$

where λ controls the influence of line segments reprojection error onto the cost function. We can then refine either β (motion-only), L (structure-only) or both β and L. We augment the objectives of the motion estimation procedure and the local bundle adjustment procedure with the term (7) using the information matrix $\Sigma^{-1} = \frac{1}{s^2} I$, (where I is a unit matrix and s is the scale of the detection's pyramid level). Finally, the constant ν in the Huber cost is the same as used for matching 2D points in stereo-images in the original ORB-SLAM2 system [17].

IV. EXPERIMENTS

A. DATASETS AND DETAILS

We evaluate the proposed descriptor on the hold-out set of KITTI [15] and EuRoC MAV [16] datasets. The two datasets represent two different popular usecases for SLAM (autonomous driving and micro-aerial vehicles) that are quite different in statistics of motion (smoother motion in the former, more jerky motion in the latter) and scene types

(outdoor for KITTI, indoor for EuRoC). We train our method on a dataset combined from sequences 0-6 of KITTI and MH01, MH02, MH04, V101, V103, V201, V202, V203 of EuRoC MAV, resulting in a descriptor that is suitable for both cases. The remaining sequences (sequences 8-10 for KITTI and sequences MH01, MH05, V102 of EuRoC MAV) are used for evaluation.

The detector [52] is always used over an image pyramid of four levels with scale step 1.44. We run several epochs of training. After each epoch we check the accuracy of motion estimation with the matched line segments using the relative pose algorithm [53] on the validation sequence KITTI 07, and stop training when the number of inliers is maximal. We learn the network with the ADAM [54] algorithm with learning rate 10^{-4} and use the mini-batches of $b = 6$ images.

The architecture of our descriptor has several meta-parameters and design choices, and we proceed by evaluating some of these choices in the next subsection, while using the SLAM performance as the benchmark.

After picking the variant of our descriptor, we evaluate it on several tasks. We first compare it to the baseline on the task of distinguishing matching and non-matching line segments. Then, we evaluate the descriptor agains the baseline for a more practical task of intra-frame motion estimation.

Finally, we evaluate the new descriptor on the SLAM task by embedding it into the ORB-SLAM2 pipeline. We evaluate the learned desriptor (ORB-SLAM2+LLD) against the same pipeline using the handcrafted LBD descriptor [5] (ORB-SLAM2+LBD) and the original ORB-SLAM2 pipeline that does not use any line features. For all experiments we report a median error over 5 runs. We use a computer with i7-4960X 3.6 GHz CPU and a GT1080 TI GPU. For the new point-and-line SLAM pipeline we use $w = 25$ pixels, $N_g = 5$ for KITTI and $w = 75$ pixels, $N_g = 2$ for EuRoC MAV, and we set $\lambda = 0.5$ for both datasets irrespective of the descriptor used.

B. ARCHITECTURE SEARCH

In this section, we compare several variants of the architecture to justify the proposed variant. As a comparison measure we use an average root mean squared error (RMSE) of SLAM trajectories for the KITTI sequences 08-10 and EuRoC sequences that were not seen during training.

a: Downsampling level

We compare the proposed network that uses the $\times 8$ downsampling network with the $\times 4$ downsampling network (see Table 2 for the description of the network architectures). The trajectory reconstruction errors obtained with descriptors generated by these networks are given in Table 4, column '$\times 4$'. The inference time for the stereo pair of a KITTI sequence (approximately 1MPix images) is given in Table 3. We see that the chosen network with $8\times$ downsampling of an input image is better in terms of trajectory accuracy and faster by 30 %. The advantage in terms of accuracy can be interpreted as the benefit of having larger receptive field when computing the descriptors (in the case of the default architecture).

	Layer block	Spatial size	Num. output maps
1	Conv+BN+Relu	3x3	16
2	Conv+BN+Relu	3x3	16
3	Conv+BN+Relu	3x3, stride=2	32
4	Conv+BN+Relu	3x3	32
6	Conv+BN+Relu	3x3, stride=2	64
7	Conv+BN+Relu	3x3	64
8	Conv+BN	7x7	64
9	Upsampling	8x8	64
10	l_2-normalization	1x1	64

TABLE 2. Learnable line segment descriptor network (LLD-Net) with $4\times$ downsampling, same notation as in Table 1.

Downsampling level	Stereo pair processing time, ms
$2\times$	0.044
$4\times$	0.023
$8\times$	0.017

TABLE 3. Evaluation of the time needed for inference for a stereo pair, in milliseconds, for different levels of downsampling in the deep network. The $8\times$-downsampling network is faster than $4\times$-downsampling network by 30 %.

b: Learnable skip connections

Our default architecture has a 8×8 upsampling layer in the end. One may wonder whether more conventional gradual upsampling via several layers would result in better matching accuracy. Towards this end, we have replaced the 8×8 upsampling layer with a sequence of 2×2 upsampling layers and 3×3 conv + BatchNormalization + ReLu blocks. This sequence of layers is repeated three times resulting in the same output size as in the default LLD-Net architecture. We also add so-called skip-connections to the downsampling layers of the same resolution, as has been popularized by the U-Net architecture [55]. The results ('skip' column in the table 4) show degradation in accuracy. The new variant with skip connections also has markedly increased running time.

c: Loss function

We compare the proposed training approach with the one using a recently proposed local descriptor learning loss [12] with $m = 0.5$. Their loss penalizes the small negative distances in absolute scale rather than the closeness of positive and negative distances. The result (column 'loss_ldl') suggests that the variant of the triplet loss used to train the default system works better for our task.

d: Sampling density

To highlight the role of pooling within the desrciptor computation, we compare the proposed network which samples $T = 5$ points on a line with a baseline that uses just a single middle point of a segment. As expected, the result ($T = 1$ column in table 4) shows the degradation in accuracy, proving importance of line-based pooling of the feature descriptors. Note that the computational cost of the pooling is negligible compared to the cost of computing the descriptors.

	LLD-Net	$\times 4$	skip	loss_ldl	$T = 1$
KITTI-08	**3.17**	3.35	3.20	3.23	3.31
KITTI-09	**2.73**	2.77	2.90	3.07	3.39
KITTI-10	0.99	1.00	0.98	0.95	**0.93**
Mean KITTI	**2.30**	2.37	2.36	2.42	2.54
EuRoC-MH03	0.029	**0.025**	0.038	0.030	0.038
EuRoC-MH05	**0.086**	0.101	0.110	0.136	0.137
EuRoC-V102	**0.046**	0.048	0.056	0.064	0.088
Mean EuRoC	**0.053**	0.058	0.068	0.077	0.088

TABLE 4. Architecture search results. We use different modifications of the basic LLD-Net to compute line segment descriptors and use them inside the SLAM pipeline. We use the KITTI Odometry sequences 08-10 and EuRoC MAV sequences MH03, MH05 and V102 which were held out during training. We report root mean square (RMS) errors in meters. The chosen (default) architecture performs better than other variants that we have tried. See text for the explanation of other tested variants.

	08			09			10			Avg		
	t_{rmse}	t_{rel}	r_{rel}	t_{rmse}	t_{rel}	r_{rel}	t_{rmse}	t_{rel}	r_{rel}	t_{rmse}	t_{rel}	r_{rel}
ORB-SLAM2	3.28	**0.99**	3.05	3.26	0.86	2.77	1.08	0.62	3.07	2.53	0.83	2.96
ORB-SLAM2+LBD	3.61	1.02	3.08	3.05	0.85	2.48	**0.97**	**0.59**	2.95	2.49	0.82	2.83
ORB-SLAM2+LLD	**3.17**	1.00	**3.02**	**2.73**	**0.81**	**2.33**	0.99	0.60	**2.78**	**2.29**	**0.81**	**2.71**

TABLE 5. Trajectory errors on the KITTI Odometry sequences 08-10 for the ORB-SLAM2 system, and its modifications using our LLD and SIFT-inspired LBD for line segment matching. t_{rmse} is a RMS error (m.), t_{rel} and r_{rel} are relative translation (cm.) and rotation errors (deg. $\times 10^{-3}$) measured over parts of the trajectory using the code provided with KITTI. According to the mean errors over the three sequences, the system based on handcrafted line descriptors ORB-SLAM2+LBD outperforms ORB-SLAM2, and the system based on the learnable descriptors ORB-SLAM2+LLD outperforms both baselines.

	MH03	MH05	V102	Avg
ORB-SLAM2	0.044	0.182	0.051	0.092
ORB-SLAM2+LBD	0.050	0.139	**0.041**	0.076
ORB-SLAM2+LLD	**0.029**	**0.086**	0.046	**0.053**

TABLE 6. Root mean square (RMS) trajectory errors (meters) on the EuRoc-MAV dataset for the ORB-SLAM2 system and its modifications using our LLD and SIFT-inspired LBD descriptors for line segment matching. In terms of the mean RMSE over three sequences, the point+line systems outperform the point-only baseline and the system using learnable line descriptor has higher accuracy than the system using the handcrafted one.

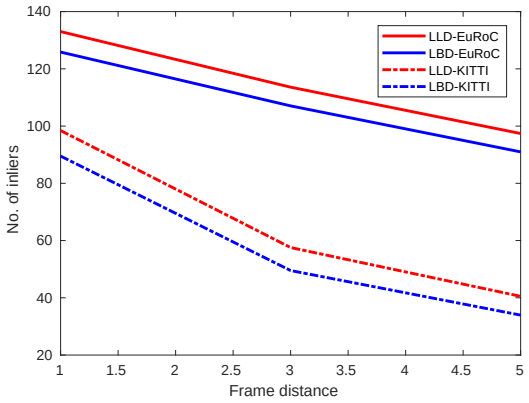

FIGURE 3. Number of inliers in line-based pose estimation inside a RANSAC loop for the test sequences from KITTI and EuRoC datasets between a left and a right frame number n and a left frame number $l + s$ for $s = 1, 3, 5$ and all possible n. LLD-based motion estimation consistently produces two times more inliers than LBD-based one.

C. RETRIEVAL OF LINE SEGMENTS

Having validated the default LLD-Net architecture against several variants, we now compare it against the baseline. We first evaluate the learnable descriptor against the handcrafted LBD (line band descriptor) baseline on the task of distinguishing the matching and non-matching descriptors. We use a dataset of matching and non-matching line segments for the KITTI test sequences 08-10. The dataset is obtained as described in Section III. For all the line segments we compute the 72-dimensional LBD descriptor using the OpenCV implementation and the proposed 64-dimensional LLD descriptor. Once the matching and non-matching pairs are identified, we test the ability of descriptors to distinguish such pairs by comparing the distances between descriptors to a certain threshold τ (we expect that for an appropriate τ the distances between matching line segment descriptors should be less than τ, while the distances between non-matching line segment descriptors should be above τ).

We compare the descriptors both for the original sequences, and in the case when strong artificial noise (Gaussian, $\sigma = 30$) is added to the images during training and when computing descriptors on the test data. The latter experiment tests the ability of our learnable descriptor to adapt to certain sensor characteristics. We show the recall-precision curves obtained for varying matching threshold τ in Figure 5. We observe that our descriptor outperforms the baseline by a considerable margin, and that it shows graceful degradation when noise is added (in fact, in the high-recall area the performance of our learnable descriptor does not degrade at all when noise is added).

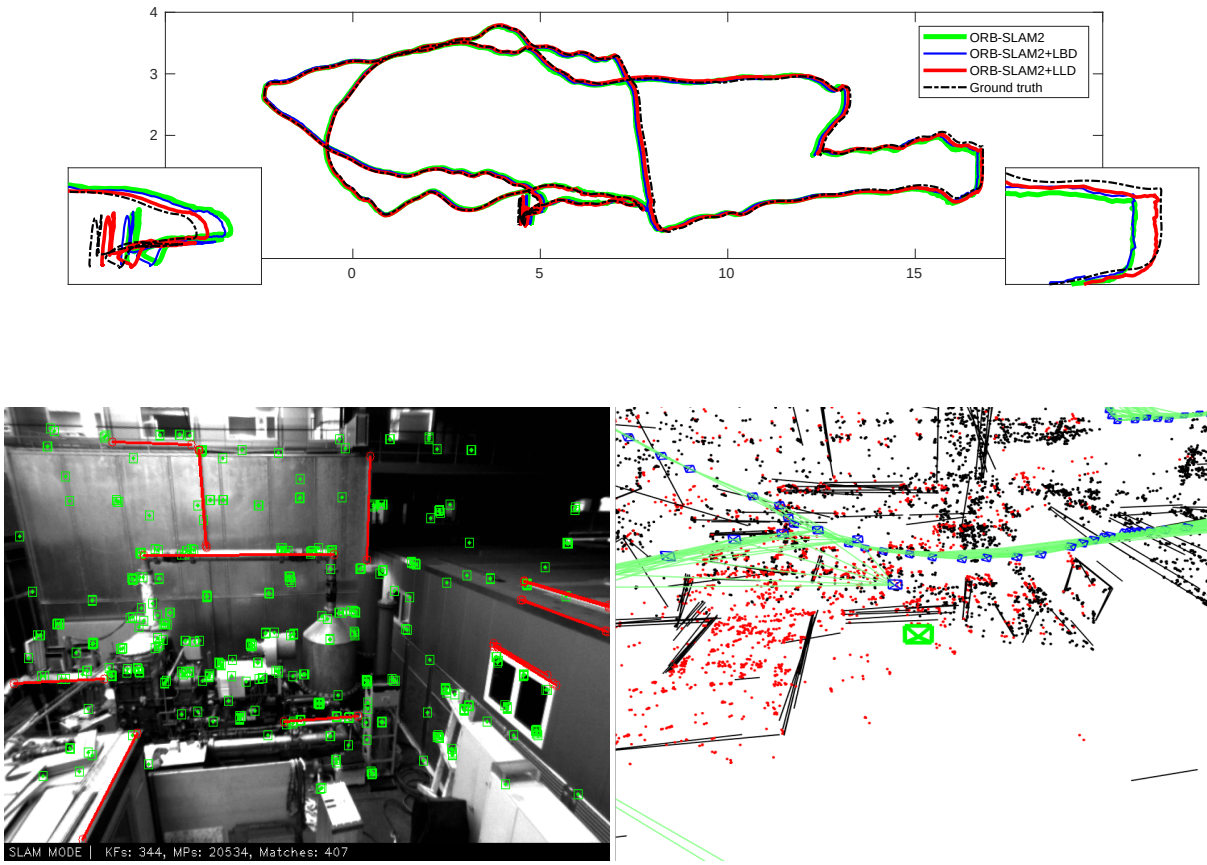

FIGURE 4. *Top*: EuRoC Track MH05, the trajectory ground truth projected on XZ plane, and the reconstructed trajectories by different methods (each trajectory is rigidly aligned with the ground truth). The trajectory reconstructed by the ORB-SLAM2+LLD (red) is closer to the ground truth trajectory (dashed black) demonstrating the advantage of the proposed descriptor. *Bottom-left*: feature tracking visualization in the ORB-SLAM2+LLD system for a single frame of the sequence. *Bottom-right*: the map constructed by ORB-SLAM2+LLD system. The green frustum corresponds to the frame shown in the bottom-left; the blue frustums represent other frames of the reconstructed trajectory. Black lines show segments of 3D lines inserted into the map. Red (black) points correspond to the map points used (not used) in the local bundle adjustment.

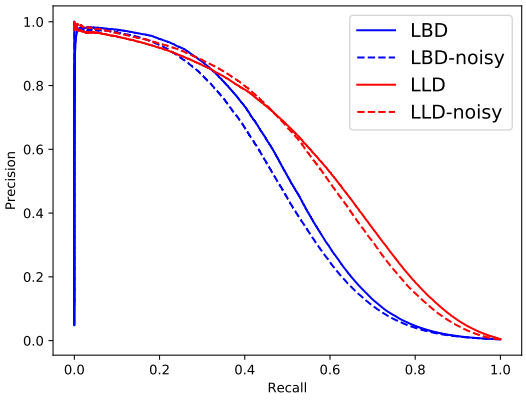

FIGURE 5. Precision-recall curves for the learnable (LLD) and hand-crafted (LBD) descriptors computed on the test sequences of the KITTI dataset (08-10). Solid lines: descriptors computed on the unmodified images. Dashed lines: descriptors computed (and trained, in case of LLD) on the images with additional Gaussian noise with $\sigma = 30$. The learnable descriptor outperforms the handcrafted one in terms of precision and is more robust to additive noise.

D. DESCRIPTOR INVARIANCE TO CAMERA MOTION

We now move from a more artificial task (retrieval of line segments) to a more practical task of inter-frame motion estimation. We therefore compare the invariance of the descriptors to camera motions. More specifically, we do line-based relative camera pose estimation. We match the line segments from left and right frames number n and left frame number $n + s$ for $s = 1, 3, 5$ and all possible n. Once again, we use hold-out KITTI and EuRoC sequences, and we compare LLD and LBD descriptors. For relative pose estimation we use an algorithm [53] inside a RANSAC loop with a threshold of ten pixels. The average numbers of inliers for different s shown at the Figure 3 indicate that LLD has greater invariance for smaller as well as bigger camera motions, so that LLD-based motion estimation consistently produces more inliers than LBD-based one.

E. EVALUATION INSIDE A SLAM PIPELINE

Finally, we evaluate how the new descriptors affect the performance of a SLAM system. We thus compare the original **ORB-SLAM2** pipeline, the modified ORB-SLAM2 pipeline with the non-learned LBD line segment descrip-

tor (**ORB-SLAM2+LBD**), and finally the proposed system (**ORB-SLAM2+LLD**). The results of the comparison for KITTI sequences are shown in Table 5. The proposed ORB-SLAM2+LLD system outperforms the point-only system ORB-SLAM2 in all metrics except relative translation in one sequence out of three. The ORB-SLAM2+LBD outperforms the point-only baseline in term of all metrics on two out of three sequences. According to the mean errors over the three sequences, the system based on handcrafted line descriptors ORB-SLAM2+LBD outperforms ORB-SLAM2, and the system based on the learnable descriptors ORB-SLAM2+LLD outperforms both baselines. While the relative translation error is almost unchanged, the RMSE and the relative rotation errors of the ORB-SLAM2+LLD descrease by 10-20% compared to the point-only system. Due to improvement of rotation estimation accuracy in the line-based systems, the turning pieces of the trajectory are estimated more accurately which leads to the improvement in the alignment of the predicted trajectory with the ground truth one.

Naturally, the advantage is bigger on EuRoC sequences, which are more challenging, and where the performance of ORB-SLAM is less close to perfect. The results on EuRoC MAV are shown in Table 6, where we report RMSE on three test sequences of the dataset. The mean errors are reduced by more than by 40% when using the learnable line desriptor-based system compared to the point-only system, and by more than 20% when using the handcrafted line descriptor-based system. The system based on the handcrafted LBD descriptors outperforms the point-only system ORB-SLAM2 on two out of three sequences, while the system based on the proposed LLD descriptor outperforms the point only baseline on all sequences, and the LBD-based baseline on two out of three sequences. We show the difference in obtained trajectories on one of the EuRoC sequences in Figure 4.

V. CONCLUSION

We have investigated the use of deep learning for the training of line segment descriptors. Along the way, we have shown that adding lines into the popular ORB-SLAM2 system improves its accuracy. Once integrated into ORB-SLAM2, line segment descriptors obtained with our deep learning approach outperform the hand-crafted descriptors.

The advantage of the learned descriptor over the hand-crafted one comes at a computational price, which is currently around 17 milliseconds on a 1080Ti GPU per frame pair. In our system, this computational cost is largely independent on the number of line segments detected in the frame.

Our current system uses an external line segment detector. It can thus be regarded as a first step towards a completely learnable feature pipeline for visual line-based SLAM. The end-to-end learning of a descriptor and a detector could increase the resulting accuracy by co-adaptation of the modules, which are trained separately in the current system. Another prospective direction is to investigate how same convolutional architecture can be shared between the point and line descriptors thus increasing the efficiency. Another possibility that has been left for future work is investigating various ways of speeding up the computation further using low-bit quantization of weights and quantizations, depthwise separable and spatially separable convolutions, which can lead to reducing the number of layers (with potential use of dilated convolution to keep the receptive field large).

REFERENCES

[1] J. Sola, T. Vidal-Calleja, J. Civera, and J. M. M. Montiel, "Impact of landmark parametrization on monocular ekf-slam with points and lines," International journal of computer vision, vol. 97, no. 3, pp. 339–368, 2012.

[2] G. Zhang, J. H. Lee, J. Lim, and I. H. Suh, "Building a 3-d line-based map using stereo slam," IEEE Transactions on Robotics, vol. 31, no. 6, pp. 1364–1377, 2015.

[3] T. Holzmann, F. Fraundorfer, and H. Bischof, "Direct stereo visual odometry based on lines," in Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2016, 2016, pp. 1–11.

[4] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: real-time monocular visual SLAM with points and lines," in Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017, pp. 4503–4508.

[5] 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.

[6] B. Verhagen, R. Timofte, and L. Van Gool, "Scale-invariant line descriptors for wide baseline matching," in Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on. IEEE, 2014, pp. 493–500.

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

[8] S. A. J. Winder, G. Hua, and M. A. Brown, "Picking the best DAISY," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR, 2009, pp. 178–185.

[9] K. Simonyan, A. Vedaldi, and A. Zisserman, "Descriptor learning using convex optimisation," in European Conference on Computer Vision(ECCV), 2012, pp. 243–256.

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

[11] B. F. Y. Tian and F. Wu, "L2-net: Deep learning of discriminative patch descriptor in euclidean space," in Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2017.

[12] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in Advances in Neural Information Processing Systems, 2017, pp. 4829–4840.

[13] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR, 2005, pp. 539–546.

[14] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012, pp. 3354–3361.

[16] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The EuRoC micro aerial vehicle datasets," The International Journal of Robotics Research, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[17] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[18] R. Deriche and O. Faugeras, "Tracking line segments," Image and vision computing, vol. 8, no. 4, pp. 261–270, 1990.

[19] C. Schmid and A. Zisserman, "The geometry and matching of lines and curves over multiple views," International Journal of Computer Vision, vol. 40, no. 3, pp. 199–233, 2000.

[20] R. I. Hartley, "A linear method for reconstruction from lines and points," in Computer Vision, 1995. Proceedings., Fifth International Conference on. IEEE, 1995, pp. 882–887.

[21] M. I. Lourakis, S. T. Halkidis, and S. C. Orphanoudakis, "Matching disparate views of planar surfaces using projective invariants," Image and Vision Computing, vol. 18, no. 9, pp. 673–683, 2000.

[22] 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.

[23] H. Kim and S. Lee, "Simultaneous line matching and epipolar geometry estimation based on the intersection context of coplanar line pairs," Pattern Recognition Letters, vol. 33, no. 10, pp. 1349–1363, 2012.

[24] J. López, R. Santos, X. R. Fdez-Vidal, and X. M. Pardo, "Two-view line matching algorithm based on context and appearance in low-textured images," Pattern Recognition, vol. 48, no. 7, pp. 2164–2184, 2015.

[25] Y. Li and R. L. Stevenson, "Multimodal image registration with line segments by selective search," IEEE transactions on cybernetics, vol. 47, no. 5, pp. 1285–1298, 2017.

[26] Q. Jia, X. Gao, X. Fan, Z. Luo, H. Li, and Z. Chen, "Novel coplanar line-points invariants for robust line matching across views," in European Conference on Computer Vision. Springer, 2016, pp. 599–611.

[27] X. Shi and J. Jiang, "Automatic registration method for optical remote sensing images with large background variations using line segments," Remote Sensing, vol. 8, no. 5, p. 426, 2016.

[28] H. Bay, V. Ferraris, and L. Van Gool, "Wide-baseline stereo matching with line segments," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1. IEEE, 2005, pp. 329–336.

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

[30] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 10, pp. 1615–1630, 2005.

[31] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," IEEE transactions on pattern analysis and machine intelligence, vol. 33, no. 1, pp. 43–57, 2011.

[32] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 5, pp. 815–830, 2010.

[33] B. Kaneva, A. Torralba, and W. T. Freeman, "Evaluation of image features using a photorealistic virtual world," in IEEE International Conference on Computer Vision, ICCV, 2011, pp. 2282–2289.

[34] K. Simonyan, A. Vedaldi, and A. Zisserman, "Learning local feature descriptors using convex optimisation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 36, no. 8, pp. 1573–1585, 2014.

[35] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in European Conference on Computer Vision. Springer, 2016, pp. 467–483.

[36] M. Zieba, P. Semberecki, T. El-Gaaly, and T. Trzcinski, "Bingan: Learning compact binary descriptors with a regularized gan," Advances in neural information processing systems, 2018.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097–1105.

[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2758–2766.

[39] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk, "HPatches: A benchmark and evaluation of handcrafted and learned local descriptors," in Conference on Computer Vision and Pattern Recognition (CVPR), vol. 4, no. 5, 2017, p. 6.

[40] A. Babenko and V. S. Lempitsky, "Aggregating local deep features for image retrieval," in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, 2015, pp. 1269–1277.

[41] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI, 2016, pp. 241–257.

[42] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," Computer vision and image understanding, vol. 100, no. 3, pp. 416–441, 2005.

[43] R. Gomez-Ojeda, D. Zuñiga-Noël, F.-A. Moreno, D. Scaramuzza, and J. Gonzalez-Jimenez, "Pl-slam: a stereo slam system through the combination of points and line segments," arXiv preprint arXiv:1705.09479, 2017.

[44] H. Li, J. Yao, J.-C. Bazin, X. Lu, Y. Xing, and K. Liu, "A monocular slam system leveraging structural regularity in manhattan world," in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 2518–2525.

[45] Y. Zhao and P. A. Vela, "Good line cutting: towards accurate pose tracking of line-assisted VOVSLAM," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 516–531.

[46] J. Zhang, G. Zeng, and H. Zha, "Structure-aware slam with planes and lines in man-made environment," Pattern Recognition Letters, 2018.

[47] M. Jaderberg, K. Simonyan, A. Zisserman, et al., "Spatial transformer networks," in Advances in neural information processing systems, 2015, pp. 2017–2025.

[48] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," arXiv preprint, 2016.

[49] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in European Conference on Computer Vision. Springer, 2016, pp. 525–542.

[50] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on. IEEE, 2016, pp. 2554–2564.

[51] 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.

[52] C. Akinlar and C. Topal, "Edlines: A real-time line segment detector with a false detection control," Pattern Recognition Letters, vol. 32, no. 13, pp. 1633–1642, 2011.

[53] A. Vakhitov, V. Lempitsky, and Y. Zheng, "Stereo relative pose from line and point feature triplets," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 648–663.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[55] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III, 2015, pp. 234–241.