Visual Odometry Using Line Features and Machine Learning Enhanced Line Description

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Dipl.-Inform. Manuel Valentin Lange

aus Filderstadt

Tübingen

2022

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	04.11.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Andreas Schilling
2. Berichterstatter:	Prof. Dr. Hanspeter Mallot

On the shoulders of giants and dwarfs.

Abstract

The research on 2D lines in images has increased strongly in the last decade; on the one hand, due to more computing power available, on the other hand, due to an increased interest in odometry methods and autonomous systems. Line features have some advantages over the more thoroughly researched point features. Lines are detected on gradients, they do not need texture to be found. Thus, as long as there are gradients between homogeneous regions, they can cope with difficult situations in which mostly homogeneous areas are present. By being detected on gradients, they are also well suited to represent structure. Furthermore, lines have a very high accuracy orthogonal to their direction, as they consist of numerous points which all lie on the gradient contributing to this locational accuracy.

First, we introduce a visual odometry approach which achieves real-time performance and runs solely using lines features, it does not require point features. We developed a heuristic filter algorithm which takes neighbouring line features into account and thereby improves tracking of lines and matching of lines in images taken from arbitrary camera locations. This increases the number of tracked lines and is especially beneficial in difficult scenes where it is hard to match lines by tracking them. Additionally, we employed the Cayley representation for 3D lines to avoid overparameterization in the optimization. To show the advancement of the method, it is benchmarked on commonly used datasets and compared to other state of the art approaches.

Second, we developed a machine learning based line feature descriptor for line matching. This descriptor can be used to match lines from arbitrary camera locations. The training data was created synthetically using the Unreal Engine 4. We trained a model based on the ResNet architecture using a triplet loss. We evaluated the descriptor on real world scenes and show its improvement over the famous Line Band Descriptor.

Third, we build upon our previous descriptor to create an improved version. Therefor, we added an image pyramid, gabor wavelets and increased the descriptor size. The evaluation of the new descriptor additionally contains competing new approaches which are also machine learning based. It shows that our improved approach outperforms them.

Finally, we provide an extended evaluation of our descriptor which shows the influences of different settings and processing steps. And we present an analysis of settings for practical usage scenarios. The influence of a maximum descriptor distance threshold, of a Left-Right consistency check and of a descriptor distance ratio threshold between the first and second best match were investigated. It turns out that, for the ratio of true to false matches, it is almost always better to use a descriptor distance ratio threshold than a maximum descriptor distance threshold.

Kurzfassung

Die Erforschung von 2D-Linien in Bildern hat im letzten Jahrzehnt stark zugenommen; zum einen aufgrund der gestiegenen verfügbaren Rechenleistung und zum anderen durch ein gesteigertes Interesse an Odometriemethoden und autonomen Systemen. Linienmerkmale haben einige Vorteile gegenüber den besser erforschten Punktmerkmalen. Linien werden an Gradienten erkannt - sie brauchen keine Textur, um detektiert zu werden. Dadurch eignen sie sich auch gut in Situationen mit überwiegend homogenen Bereichen, solange es Gradienten zwischen den homogenen Regionen gibt. Darüber hinaus sind sie gut geeignet, um Strukturen darzustellen und sie bieten orthogonal zu ihrer Richtung eine hohe Genauigkeit, da sie aus zahlreichen Punkten bestehen, die alle auf dem Gradienten liegen und zu dieser Ortungsgenauigkeit beitragen.

Zunächst stellen wir einen visuellen Odometrie-Ansatz vor, der Echtzeit-Performance erreicht und allein mit Linienmerkmalen auskommt. Wir haben einen heuristischen Filteralgorithmus entwickelt, der benachbarte Linienmerkmale berücksichtigt und dadurch die Verfolgung von Linien und das Matching von Linien aus beliebigen Positionen verbessert. Zusätzlich verwendeten wir die Cayley-Repräsentation für 3D-Linien, um eine Überparametrisierung bei der Optimierung zu vermeiden. Die Methode wurde auf bekannten Datensätzen getestet und mit anderen State-of-the-Art Ansätzen verglichen; dabei zeigt sich ihr Fortschritt.

Zweitens entwickelten wir einen auf maschinellem Lernen basierenden Linienmerkmalsdeskriptor für das Linien-Matching. Dieser Deskriptor kann verwendet werden, um detektierte Linien aus unterschiedlichen Kamerapositionen einander zuzuordnen. Die Trainingsdaten werden synthetisch mit der Unreal Engine 4 erzeugt. Wir haben ein Modell basierend auf der ResNet-Architektur unter Verwendung eines Triplett-Loss trainiert und den Deskriptor an realen Szenen evaluiert. Dabei zeigt sich seine Verbesserung gegenüber dem bekannten Line-Band-Deskriptor.

Drittens bauten wir auf unserem vorherigen Deskriptor auf, um eine verbesserte Version zu erstellen. Dazu haben wir eine Bildpyramide und Gabor-Wavelets implementiert und die Deskriptorgröße erhöht. Die Auswertung des neuen Deskriptors enthält zusätzlich neue Ansätze, die ebenfalls auf maschinellem Lernen basieren. Es zeigt sich, dass unser verbesserter Ansatz diese übertrifft.

Schließlich folgt eine Darstellung der erweiterten Evaluierung unseres Deskriptors, die die Einflüsse verschiedener Einstellungen und Verarbeitungsschritte zeigt. Abschließend untersuchen wir den Einfluss von Distanzschwellen und einer Konsistenzbedingung auf das Verhältnis von wahren zu falschen Korrespondenzen.

Acknowledgments

First of all, I would like to thank Prof. Zell, who motivated me to take a doctorate degree after I submitted my diploma thesis to him. And who made it possible for me to do so by initiating a mini-graduate programme.

I want to thank my co-supervisor Prof. Mallot and Gerrit Ecke from the Mini-GRK for the interesting interdisciplinary discussions we had. I also want to thank my other colleagues from the Mini-GRK for their beneficial contributions: Anjan-Kumar Guttahalli-Krishna, Thomas Linkugel, Yuyi Liu, Marcin Odelga, Sebastian Scherer, Ya Wang and especially Radouane Ait-Jellal with whom I had many fruitful discussions and a joint paper. Another co-author of the just mentioned paper is Benjamin Wassermann, who already had quite some experience and supported me in multiple ways when I started my PhD.

I will not forget the good times in numerous lunch breaks I spent together with my colleagues from the Computer Graphics group, namely Mark Boss, Raphael Braun, Dennis Bukenberger, Jieen Chen, Christian Fuchs, Fabian Groh, Sebastian Herholz, Prof. Hendrik Lensch, Arijit Mallick, Benjamin Resch, Katharina Schwarz and Patrick Wieschollek.

Our PhD seminars were always great gatherings for interesting and horizon broadening chats and presentations (sometimes even with Pizza) for which I want to thank my colleagues Marcel Frueh, Yapeng Gao, Thomas Gulde, Jonas Haeling, Marvin Haeussermann, Andreas Karge, Kevin Laube, Florian Liefers, Dennis Ludl, Maximus Mutschler, Andreas Ray and Johannes Theodoridis.

I will very positively remember the exploratory meetings of our machine learning reading group with Tobias Lang, Mathias Schickel, Andreas Schilling and with Holger Heidrich as educating visitor.

I am highly greatful to Robin Niebergall, Fabian Schweinfurth and Claudio Raisch for assisting me in various tasks in the past years.

I am really thankful to have had Andreas as my doctorate supervisor with his warm personality and inspiring ideas, especially in times when I thought I had reached an impasse. Besides his ideas I am also thankful for his continuous encouragement.

Finally, I would like to express my utmost appreciation to my friends, family and Anna for their endless support and patience.

Contents

Chapter 1

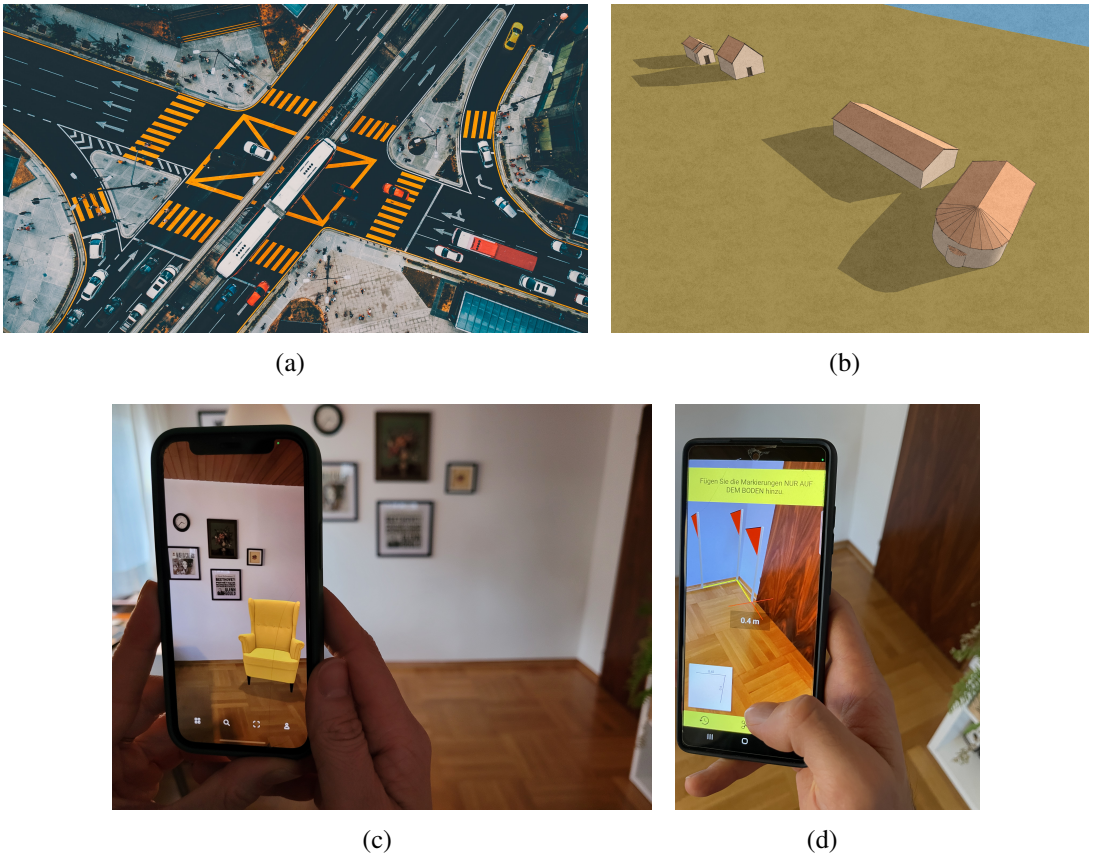
Introduction

1.1 Lines

The perception of the world and ones surroundings is important to living beings as well as, with the rising use of assistive technologies, to information processing systems. We humans perceive vast amounts of our environment using our eyes. Cameras are comparatively cheap sensors which can perceive the surroundings in a similar way. But, unlike more expensive sensors, such as laser scanners, cameras do not directly yield structural information like distance measurements. Therefore, more processing is required to acquire such knowledge of the environment. Techniques that use landmarks are widely utilized. A common type for a landmark is a 2D point feature. Points are detected in the images by specific algorithms on salient locations like corners. Another interesting feature type is a 2D line. In contrast to points, lines have a direction and an elongation in 2D, as well as in 3D space. Furthermore, a line is straight, which yields additional structural information. This can be an advantage, especially when reconstructing man made environments like houses, where the roof ridge and many edges are straight. But lines are also advantageous in other cases, for example, in scenarios where there is barely texture available, but only structure. Point features are hard to detect in those scenarios, but lines can be found as they lie on gradients. Lines can be utilized in all kinds of tasks in which point features are common: in robotic applications like visual odometry or simultaneous localization and mapping, in augmented reality scenarios, for reconstruction tasks, to perform measuring and so forth.

The focus of this work is on lines, more specifically, on 2D lines which are detected in camera images. They have not been researched as thoroughly as point features and with the ever-increasing calculation power, their detection and processing have become more and more real-time feasible.

1.1.1 Application scenarios



(a) (b)

(c) (d)

Figure 1.1: The images display examples of problems which can benefit from using line features. Top left (a) shows a large junction with many lines on the streets which can be used by self driving cars to improve localization, ©Photo by Deva Darshan on Unsplash. The top right illustration (b) depicts an example of a 3D reconstruction, by ©Roland Meinecke © Free Art License 1.3[Myl22]. One can see that buildings often have straight lines. This can be utilized in 3D reconstruction pipelines to improve the quality of the 3D models. In the bottom left image (c) we see an augmented reality example showing how a certain armchair would look at a specific location in the room. (The app used on the smartphone is the *IKEA Place* app, a promotional video is found here: [IKE21].). The image at the bottom right (d) displays the use of a smartphone app to measure distances in a room. The shown app is called *CamToPlan - Maßband messen / Lineal / Zollstock* [Tas21].

Figure 1.1 depicts tasks where lines can be the basis of, or contribute to a reliable solution. Image 1.1(a) depicts a difficult scene for autonomous driving. Utilizing lines can provide multiple benefits for the underlying simultaneous localization and mapping

module. Tracking the lines on the road lanes and on many other structures like pillars and street lights supports the accuracy. Many lines can still be detected in a night drive, as a gradient is sufficient for a line to be detected. Figure 1.1(b) depicts 3D models of buildings. Straight lines are very common on buildings. A 3D reconstruction pipeline can use lines to improve the quality of reconstructions. The detection of a line on a gradient, which is formed not only by one or two, but by numerous points lying on an edge, leads to a high accuracy of its location orthogonal to its direction. Pierre-Alain Langlois et al. [LBM19] utilize this property of lines to create a 3D model with accurate straight edges out of a few camera pictures. Image 1.1(c) shows the augmented reality application *IKEA Place* [IKE21] which augments a piece of furniture at a certain location into ones room, to see if it would suit. To be able to move around and see the augmented object in 3D, visual odometry has to be performed. The scene visible on the smartphone shows a parquet floor which is somewhat textured, and two walls which are mostly homogeneous areas besides the pictures on them. The pictures and floor will be able to provide some point features, but if the smartphone is rotated upwards in such a way that most of the floor and the pictures are outside the field of view, the tracking is likely to fail if it solely relies on point features since homogeneous areas usually lack point features. Whereas lines could still be found on the edge between the walls and on the edge towards the floor or ceiling, which would stabilize the visual odometry task. Figure 1.1(d) displays the use of a smartphone measuring application. Distances are mostly measured along straight lines and often on edges. Lines can accurately lock in on edges and while one moves around, the app simultaneously performs odometry. These are just a few examples of application scenarios which benefit from utilizing lines.

In the following, we will state our contributions to the research on lines.

1.2 Main Contributions

In this thesis we add to the research on lines by multiple contributions. First, we establish a line feature only real-time visual odometry system. After that, we tackle the problem of matching line features from arbitrary camera positions through developing a machine learning featured line descriptor. Then, we refine this descriptor to improve its reliability and robustness. And, finally, we provide an extensive evaluation of its performance. In more detail, the contributions are as follows:

- We present a line-only visual odometry approach. It achieves real-time performance and runs stable in textureless environments. Our approach can compete well with existing point- and line-based methods and outperforms existing line-only methods. We developed a tracking method for line-segments that works well without descriptors. Additionally, we propose a heuristic filter that improves the tracking and arbitrary matching results with lines. Furthermore, an optimization using the Cayley representation for 3D lines is used to reduce the line parameters from usually six to four. We demonstrate our performance on the EuRoC

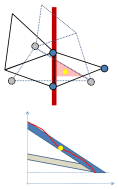
benchmark and on a synthetic scene, which we built to test the algorithms under particularly difficult conditions. [LRS19].

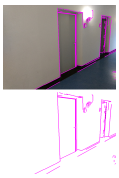
- Additionally, we present an appearance based line descriptor which was developed with the help of machine learning. Our descriptor uses a ResNet which we modified in its size to improve the performance. We utilized the Unreal Engine 4 and multiple scenes provided for it to create training data. The training was performed using a triplet loss, where the loss of the network is calculated with triplets, each consisting of three lines including a matching pair and another non-matching line. During learning, the goal of the minimization function is to calculate descriptors with minimal descriptor distance to matching lines' descriptors and maximal descriptor distance to other lines' descriptors. We evaluated the performance of our descriptor on our synthetic datasets, on real-world stereo images from the Middlebury Stereo Dataset and on a benchmark for line segment matching. The results show that in comparison to state-of-the-art line descriptors our method achieves a greatly improved line matching accuracy. [LSS19].

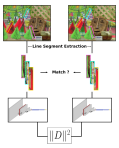
- Furthermore, we present a wavelet enhanced version of our feature descriptor for line feature matching. This time, we trained a neural network to compute a descriptor for a line, providing the network with preprocessed information from the image area around the line. In the preprocessing step we utilize wavelets to extract meaningful information from the image for the descriptor. This process is inspired by the human vision system. We, again, used the Unreal Engine 4 and multiple different freely available scenes to create our training data. We conducted the evaluation on ground truth labelled 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 our original Deep Learning Based Line Descriptor (DLD), and to the Learnable Line Segment Descriptor for Visual SLAM (LLD). [LRS20].

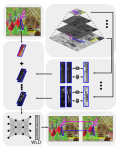
- We analyse the influence of processing steps and settings on our descriptors matching performance. Then, we provide an evaluation of certain settings for practical usage scenarios. We examine the benefit of using a Left-Right consistency check, a maximum descriptor distance threshold and a descriptor distance ratio threshold between the first and second best match. The results show that in most cases it is better to filter matches using a descriptor distance ratio threshold than by setting a maximum descriptor distance threshold.

The following publications were written during the work on this thesis. The bold faced ones are covered in this thesis. Major parts of them were transferred to this thesis without further modification.









1.3 Structure of this thesis

The structure of this thesis is as follows: next, we will explain some fundamentals in chapter 2 which provides background knowledge about lines, their representation and error functions that are used to evaluate the similarity between two lines. Then, in chapter 3, we present our line based visual odometry approach in detail and compare it to other methods. Chapter 4 introduces our machine learning based line descriptor for line feature matching. And in chapter 5 we improve this descriptor by building a new version which features gabor wavelets and other enhancements leading to a significant advancement in terms of matching accuracy. We provide an extended evaluation of our descriptors in chapter 6 which compares the different models we trained and evaluates the practical usage performance for varying configurations. Finally, chapter 7 draws a conclusion of this work.

Chapter 2

Fundamentals

2.1 2D line representations

2D lines, like 2D points, are usually considered as primitives in geometry [Fab83]. 2D points are located at one specific location in 2D space. They are commonly represented by two coordinates $x, y \in \mathbb{R}$. For 2D lines and 2D line-segments, on the other hand, there is not only one commonly used representation. For this reason, we will give an overview of the established representations for lines and line-segments in this chapter. We want to note that whenever mentioning lines we refer to straight lines.

2.1.1 Infinite 2D line representations

Slope-intercept form

The slope-intercept form represents a line by a slope m and the intersect location with the y-axis b, given the independent variable x:

$$y = mx + b. \tag{2.1}$$

A drawback of this form is that it cannot represent vertical lines.

Standard form

The standard form (sometimes also called *general form*) is a linear equation capable of representing any line:

$$ax + by = c \tag{2.2}$$

with a, b and c being constants for which it holds that a and b must not both be zero. If a line is expressed in any other representation, it can also be transformed into this representation. There are many other representations for infinite 2D lines and we will not present all of them. But the normal form is used in chapter 3 and therefore it is important to be mentioned here.

Normal form

The normal form, or Hesse normal form (after Ludwig Otto Hesse), represents a line by a slope θ of its normal, originating from the coordinate origin, and by the length d where the line crosses the normal:

$$y\,sin\theta + x\,cos\theta - d = 0. \tag{2.3}$$

Written in vector notation, the equation is as follows:

$$\overrightarrow{r} \cdot \overrightarrow{n}_0 - d = 0. \tag{2.4}$$

It is satisfied for all $\overrightarrow{r}$ where $\overrightarrow{r}$ points to a point that lies on the line. $\overrightarrow{n}_0$ is the unit normal vector pointing to the line, and d is the distance of the line to the origin.

Parametric form representation

The Parametric form has the advantage that it can represent a line in any dimension. At the same time it has the disadvantage that the number of parameters is not necessarily optimal. Also this representation, unlike the previously described ones, does not formulate a condition which has to be fulfilled by the coordinates in order to lie on the line, but the line's points are represented in dependence on a parameter:

$$\overrightarrow{x} = \overrightarrow{p} + t\,\overrightarrow{v}. \tag{2.5}$$

Here $\overrightarrow{p}$ is the position vector and $\overrightarrow{v}$ the direction vector. Every value of the parameter $t \in \mathbb{R}$ corresponds to a point $\overrightarrow{x}$ on the line. Representing a 2D line this way requires four parameters and thereby two more than minimal representations use.

2.1.2 2D line-segment representations

The previous representations all represented infinite 2D lines using two parameters only. For equations capable of representing any line-segment in $\mathbb{R}^2$, four parameters are necessary.

Two Point representation

An obvious representation for 2D line-segments is the two point representation. Here, a line-segment is defined by two distinct points $A, B \in \mathbb{R}$. This representation helps to picture the line-segment before the inner eye. But calculating an intersection with another line is not straightforward. For this purpose, a representation based on the normal form is more suitable.

Normal form based line-segment representation

A line-segment can also be represented based on the *normal form* of a line. For this we also use a distance d and the unit normal vector $\overrightarrow{n}_0$, as in equation 2.4. In order to define the endpoints, the unit normal vector is used. $s, e \in \mathbb{R}$ define the distance of the start and end point, from the point where the normal crosses the infinite line, by multiples of the length of the unit normal vector. The start point $A \in \mathbb{R}^2$, for example, can be calculated as follows:

$$\begin{aligned} A_x &= d \cdot n_x + s \cdot n_y \\ A_y &= d \cdot n_y - s \cdot n_x \end{aligned} \tag{2.6}$$

With $n_x, n_y \in \mathbb{R}$ representing the x- and y-component of $\overrightarrow{n}_0$.

2.2 3D line representations

2.2.1 Two point representation

Two apparent 3D line representations are the same as for 2D lines, but with additional parameters for the third dimension. The *two point representation* for a 3D line and a 3D line-segment is analogous to the one for 2D lines (see paragraph 2.1.2). It uses six parameters and, therefore, is a minimal form of representation for a 3D line-segment.

2.2.2 Parametric form

The *parametric form* is also similar to its corresponding 2D version (see equation 2.5), but of course the vectors consist of three components each. Unlike the *two point representation* it is not able to represent a 3D line-segment with the minimal number of six parameters. Two additional parameters are required to define the endpoints' location on the 3D line, similarly to the way it is done in 2D (see equation 2.6).

2.2.3 Plücker line representation

A Plücker line is defined by a moment m and a displacement d. The moment is a vector which is orthogonal to the plane containing the line and the origin. The moment can be computed by calculating the crossproduct of two distinct points on the 3D line: $m = x \times y$ where $x, y \in \mathbb{R}^3$. This calculation also results in the magnitude of m being equal to twice the area that is contained in the triangle formed by x, y and the origin of the coordinate system. The displacement is the vector from x to y. This is visualized in figure 2.1.

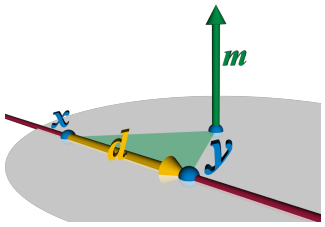


Figure 2.1: Visualization of the Plücker line representation, source: [KSm20]. © KSmrq © Creative Commons Attribution-Share Alike 2.5 Generic

The Plücker coordinates have more properties, but that would go beyond our scope here; detailed information can be found in [HZ03]. We use the Plücker coordinates as an intermediate step towards the Cayley line representation.

2.2.4 Cayley representation

The advantage of representing a 3D line using the Cayley representation is its requirement of only four parameters. The previously described representations required six parameters to define an infinite 3D line. If we picture a 3D line segment before the inner eye, imagine removing the endpoints and figure not having a rotation around the line's own axis (as this does not change the line itself) then we can imagine an infinite 3D line with only four degrees of freedom.
The Cayley representation uses its own object frame for the 3D line denoted as Γ in figure 2.2. The object frame is given by l, which is the direction vector of the line, by the moment m and by their crossproduct $l \times m$. The origin of the object frame is located at P_0 of which the distance to the origin of the reference frame O_r is given by ω. The Cayley line is defined by $v = (\omega, s)$ where s defines the rotation angle θ as well as the rotation axis $\bar{s}$ between the object frame $OXYZ$ and the reference frame $O_rX_rY_rZ_r$:

$$\begin{aligned} \theta &= 2\ arctan(||s||), \\ \bar{s} &= \frac{s}{||s||}. \end{aligned} \tag{2.7}$$

Transferring to and from the Plücker representation is explained in paragraph 3.3.6 and details about the derivation of the formulas are elaborated in [ZK14].

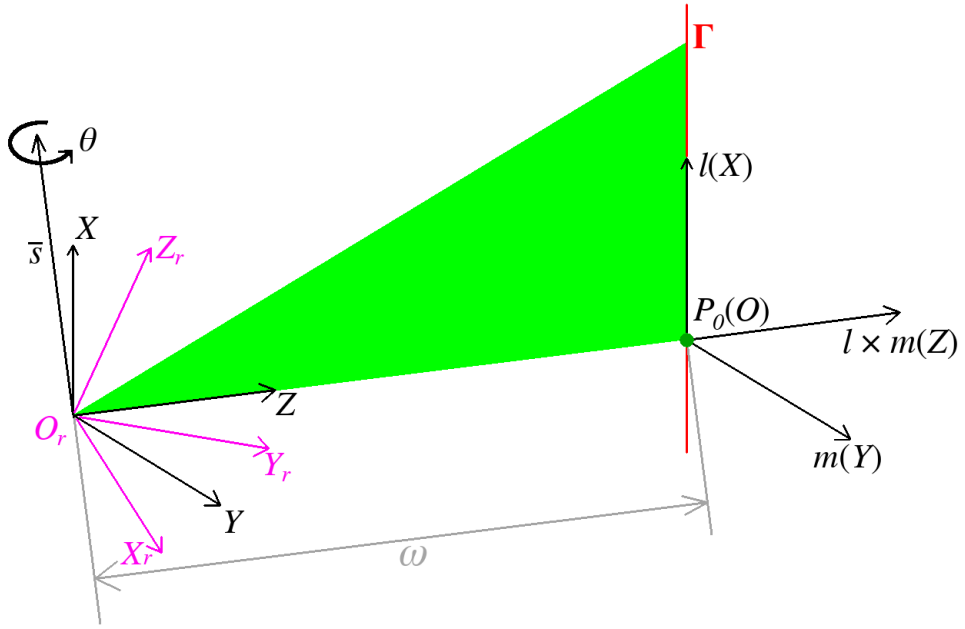


Figure 2.2: Visualization of the Cayley line representation.

The Cayley representation is advantageous when 3D lines are part of an optimization like a bundle adjustment. It avoids over-parametrization and internal gauge freedoms and thereby improves the efficiency of the optimization process.

2.3 Error measures for 2D line-segment similarity

When working with line-segments, one often wants to know how similar two line-segments are, to obtain correspondences between them. Matches between line-segments are, for example, necessary when performing visual odometry, reconstruction or measuring based on lines. Matching lines is usually either performed based on a geometrical distance, or based on a descriptor distance. Using a descriptor yields the advantage to enable matching from arbitrary camera positions without prior knowledge. But it is generally more computationally expensive than simple line-segment to line-segment distance functions. The same holds for matching based on more complex geometrical matching approaches like using 'line signatures' [WNY09], coplanar line intersections [KL10], or 'Line-Junction-Line' structures [LYL*16], just to name a few.
If prior knowledge about the expected location of a line-segment is available, for example from tracking lines, then simpler and quicker calculable functions, which compute a measure of similarity between two line-segments, can be used. In this section we provide an overview of 2D line-segment distance functions which can be used to determine the geometric similarity of two 2D line-segments.
We start with a short note about the geometrical distance of 2D points. A simple 2D

point is characterized by its location. It does not change in size and has no degree of freedom for rotation. Thus, if a point is projected into an image and compared to a point in this image, usually the euclidean distance, or the squared euclidean distance is used:

$$d = \sqrt{(x_{proj} - x_{obs})^2 + (y_{proj} - y_{obs})^2} \tag{2.8}$$

A 2D line, on the other hand, is characterized by its location and rotation. In the case of a 2D line-segment, it also has a certain extent. Thus, a comparison of the similarity of two line-segments is not as simple as in the case of points. Therefore, multiple line-segment similarity measures have been developed, each with different properties.

2.3.1 Overview and definition of line-segment distance criteria

In the following we will elaborate multiple line-segment distance functions in more detail.

The Hausdorff metric or Pompeiu-Hausdorff distance [BT05] measures the distance between two point sets. There are many variants of it [DJ94].
There is a variant of the Hausdorff metric to determine line-segment similarity from Helmut Alt, Bernd Behrends and Johannes Blömer [ABB95], which is depicted in figure 2.3 and defined by the equation 2.11.

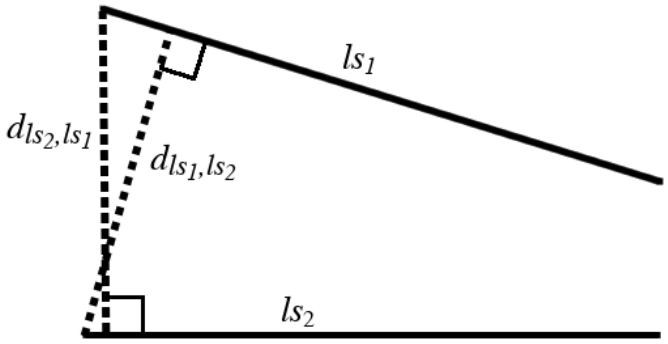


Figure 2.3: Visualization of the Hausdorff Distance variant by [ABB95].

$$\widetilde{\delta}_{ls1,ls2}(ls_1, ls_2) = \max_{p \in ls_1} \min_{q \in ls_2} ||p - q||, \tag{2.9}$$

$$\widetilde{\delta}_{ls2,ls1}(ls_2, ls_1) = \max_{p \in ls_2} \min_{q \in ls_1} ||p - q||, \tag{2.10}$$

$$\delta_H(ls_1, ls_2) = \max\,(\widetilde{\delta}_{ls1,ls2},\ \widetilde{\delta}_{ls2,ls1}). \tag{2.11}$$

Emanuele Trucco and Alessandro Verri introduced a grouping algorithm for line-segments

in their book *Introductory Techniques for 3-D Computer Vision* [TV98]. Their main error metric for proximity is depicted in figure 2.4 and defined by equation 2.14.

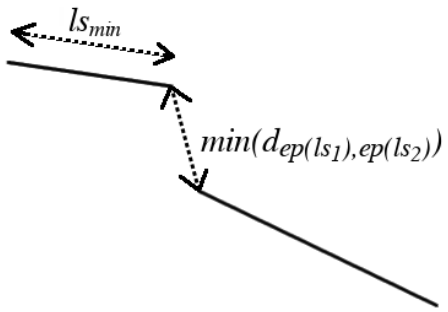


Figure 2.4: Proximity error metric by [TV98].

$$ls_{min} = \min(||ls_1||,\ ||ls_2||), \tag{2.12}$$

$$\begin{aligned}\min(d_{ep(ls_1),ep(ls_2)}) = \min(&||p_{1,ls_1} - p_{1,ls_2}||,\ ||p_{1,ls_1} - p_{2,ls_2}||,\\ &||p_{2,ls_1} - p_{1,ls_2}||,\ ||p_{2,ls_1} - p_{2,ls_2}||),\end{aligned} \tag{2.13}$$

$$\delta_T(ls_1,\ ls_2) = (\frac{ls_{min}}{\min(d_{ep(ls_1),ep(ls_2)})})^2. \tag{2.14}$$

A variant of the Hausdorff distance for line-segments, named *modified line-segment Hausdorff distance*, which includes angle information, was proposed by Jingying Chen, Maylor K. Leung and Yongsheng Gao [CLG03]. They demonstrated that it can be utilized to detect noisy logos. The *modified line-segment Hausdorff distance* is depicted in figure 2.5 and defined by equation 2.15.

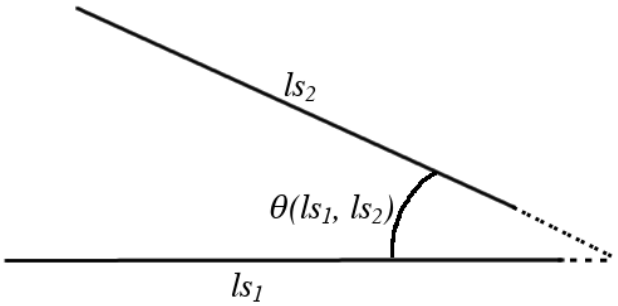


Figure 2.5: Modified line-segment Hausdorff distance by [CLG03].

$$\delta_{MH}(ls_1,\ ls_2) = \min(||ls_1||,\ ||ls_2||)\ sin(\theta(ls_1,\ ls_2)). \tag{2.15}$$

Also, with a contribution of Maylor K. Leung and Yongsheng Gao, Yu Xiaozhou proposed a *modified perpendicular line-segment Hausdorff distance* [YLG04]. It is depicted in figure 2.6 and defined by equation 2.18.

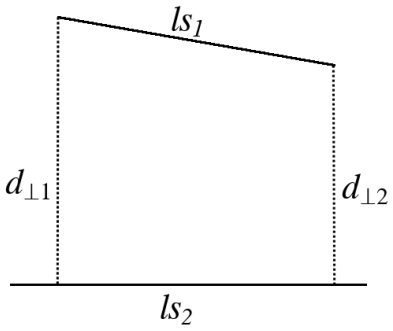


Figure 2.6: Modified perpendicular line-segment Hausdorff distance by [YLG04].

An endpoint of a line-segment is defined by $ls_{x\,p_y}$ where $x \in \{1,2\}$ defines which of the two line-segments is selected and $y \in \{1,2\}$ defines which of the two endpoints of the line-segment is selected.

$$\begin{aligned} d_{\perp 1} = \min(&\min(d_\perp(ls_1, ls_{2p_1}), (ls_1, ls_{2p_2})),\\ &\min(d_\perp(ls_2, ls_{1p_1}), (ls_2, ls_{1p_2}))) \end{aligned}$$

$$\begin{aligned} d_{\perp 2} = \min(&\max(d_\perp(ls_1, ls_{2p_1}), (ls_1, ls_{2p_2})),\\ &\max(d_\perp(ls_2, ls_{1p_1}), (ls_2, ls_{1p_2}))) \end{aligned} \tag{2.16}$$

$$w_i = \frac{d_{\perp i}}{d_{\perp 1} + d_{\perp 2}}\ ,\ with\ i = \{1,\ 2\} \tag{2.17}$$

$$\delta_{PMH}(ls_1,\ ls_2) = \frac{1}{2}(w_1\, d_{\perp 1} + w_2\, d_{\perp 2}) \tag{2.18}$$

A simple error metric is the *mid- and endpoint line-segment distance*. The distances of the endpoints are added to the distance of the midpoints while the distances may be weighted. The midpoint of a line-segment is given by $m_{ls_i} = (ls_{i\,p_1} - ls_{i\,p_2})/2$. The metric is depicted in figure 2.7 and defined by equation 2.19.

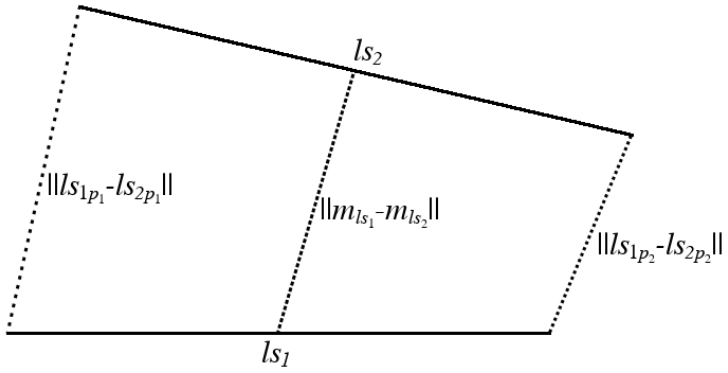


Figure 2.7: Mid- and endpoint line-segment distance.

$$\delta_{ME}(ls_1,\ ls_2) = ||ls_{1\,p_1} - ls_{2\,p_1}|| + ||ls_{1\,p_2} - ls_{2\,p_2}|| + 3\ ||m_{ls_1} - m_{ls_2}|| \tag{2.19}$$

Another error measure between line-segments is the *closest point line-segment distance*. As the name says, it is simply the shortest distance between two line-segments as depicted in figure 2.8 and defined by equation 2.20.

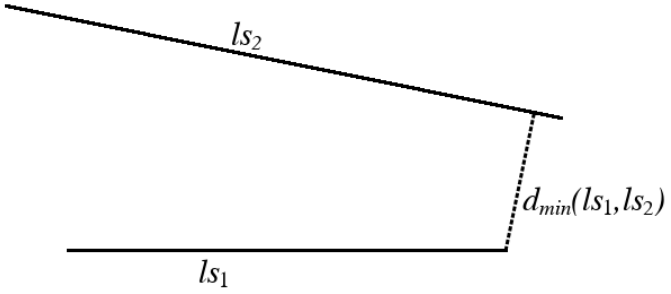


Figure 2.8: Closest point line-segment distance.

$$\delta_C(ls_1,\ ls_2) = d_{min}(ls_1, ls_2) \tag{2.20}$$

Stefan Wirtz and Dietrich Paulus proposed the *straight line distance* in [WP16]. They combined multiple measures to include angular, translational and scale differences in their error measure. This is defined by equation 2.24.

$$d_1 = ||ls_{1\,p_1} - ls_{2\,p_1}||,\ d_2 = ||ls_{1\,p_1} - ls_{2\,p_2}||,$$
$$d_3 = ||ls_{1\,p_2} - ls_{2\,p_1}||,\ d_4 = ||ls_{1\,p_2} - ls_{2\,p_2}|| \tag{2.21}$$

$$d_{translation}(ls_1,\ ls_2) = \frac{d_1 + d_2 + d_3 + d_4}{4} - \frac{||ls_1|| + ||ls_2||}{4} \tag{2.22}$$

$$d_{ST}(ls_1,\ ls_2) = \delta_C + \frac{1}{4}\delta_{MH}(ls_1,\ ls_2) + d_{translation} \tag{2.23}$$

$$\delta_{straightLine}(ls_1,\ ls_2) = \min(d_{ST}(ls_1,\ ls_2),\ d_{ST}(ls_2,\ ls_1)) \tag{2.24}$$

Aleš Jelínek and Luděk Žalud propose a line-segment similarity criterion [JŽ17] which is based on the area between two line-segments. The first part of the formula calculates the area of the parallelogram which is formed by the two line segments. This is depicted in figure 2.9 and defined by equation 2.25.

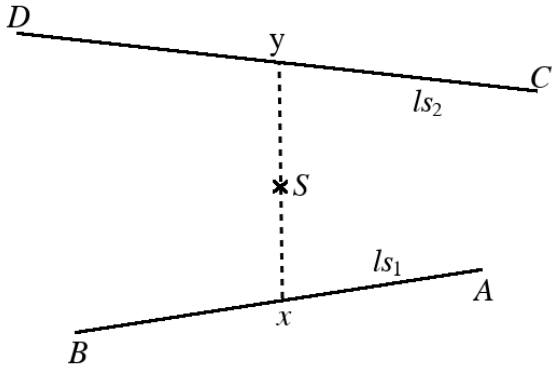


Figure 2.9: Line-segment similarity criterion by [JŽ17].

$$d^2_{ABCD}(ls_1, ls_2) = ||(B-A)\times(D-C)||^2 = ||x\times y||^2 \tag{2.25}$$

The second part of the similarity measure calculates the area of a triangle which is formed by one of the lines and the center point S which is in the middle of the four line-segment endpoints:

$$d^2_{ABS}(ls_1, ls_2) = \frac{1}{4}||x\times(S-A)||^2 \tag{2.26}$$

The final equation 2.27 adds those two measures and weights the triangular area 2.26 by a factor of 64 to balance the two parts of the equation. The choice of the prefactor of 64

is reasoned in their paper [JŽ17].

$$\delta_{area}(ls_1, ls_2) = d_{ABCD}^2 + 64\, d_{ABS}^2 \tag{2.27}$$

An efficient way to compute the *squared endpoint segment to line distance* was presented by Lilian Zhang and Reinhard Koch. They used it in a line-segment based bundle adjustment [ZK14] in which they utilized the Cayley representation of spatial lines (see 2.2.4) to reduce the degrees of freedom of an infinite 3D line to four. This line-segment distance criterion uses the endpoints of a line-segment and projects them onto an infinite line given by the bundle adjustment, or by extension of another line-segment. The projections of the line-segments' endpoints are calculated perpendicularly from the infinite line to obtain the shortest distance. These distances are then squared and summed up. The squaring has the effect that two line segments, that are rotated relative to each other, yield a smaller error when they cross at the center of the line-segment whose endpoints are projected, than when they cross at any other point (given that the angle is held constant). Thus, using the squares in equation 2.28 still pulls the line-segments closer above each other. That is already the case when they touch, since these two line-segments are considered closer when they cross in their centers than if they just touch. This distance criterion is used in chapter 3 to perform the pose optimization described in paragraph 3.3.5 and the local bundle adjustment described in paragraph 3.3.6. It is depicted in figure 3.5 and defined by the following equation 2.28. Let ls_i be the infinite line and ls_j the line-segment, then the equation is as follows:

$$\delta_{SQline} = d_{\perp}(ls_i, ls_{jp_1})^2 + d_{\perp}(ls_i, ls_{jp_2})^2 \tag{2.28}$$

The *endpoint and overlap closeness distance* criterion is similar to the previous *squared endpoint segment to line distance* criterion, but it adds a closeness term. Therefore, the endpoints of both line-segments have to be available. This closeness term is the squared minimum distance between the two line-segments. The term is only added, when there is no lateral overlap between the line-segments. A lateral overlap is examined by projecting rectangular rays from the endpoints of one line-segment and checking if at least one of the endpoints of the other line-segment lies between the rays, or touches one of them. This has to be tested for all four line-segment endpoints using the respective other line-segment. If an overlap is given, then the error is calculated in the same way as for the *squared endpoint segment to line distance* criterion. Otherwise the squared minimum distance is added and causes the error to rise rapidly with an increase of the distance. We have to note here, that this overlap dependency causes the error value to jump, when it is just barely fulfilled in one situation and not fulfilled anymore when the scenario changes slightly. This could be avoided by constantly adding the closeness term without using the overlap dependency. It depends on the task which variant is more suitable. This distance

criterion is used to measure the line-segment similarity in chapter 3 to accomplish the motion tracking task described in paragraph 3.3.3. It is depicted in figure 3.4 and defined by equation 2.29.

$$\delta_{EPoC}(l_i, l_j) := \begin{cases} d_\perp(ls_i, ls_{jp_1})^2 + d_\perp(ls_i, ls_{jp_2})^2 & \text{if } l_i \text{ and } l_j \text{ overlap} \\ d_\perp(ls_i, ls_{jp_1})^2 + d_\perp(ls_i, ls_{jp_2})^2 + d_{min}(l_i, l_j)^2 & \text{else.} \end{cases} \tag{2.29}$$

In the following, we will evaluate the presented line-segment distance criteria.

2.3.2 Evaluation of line-segment distance criteria

Wirtz and Paulus show a comparison of multiple methods in their paper [WP16]. They provide an analytical evaluation in which they created two line-segments and transformed one of them to change the geometrical relation between those two. The transformations they used include rotation, translation and scaling; figure 2.10 visualizes this. The rotation is performed on ls_2 with the point p_1 being the center of the rotation. The translation is performed along the direction of ls_2 and the scaling is performed on ls_2, also along the line-segment direction. It is executed in such a way that the point p_1 stays at the same position.

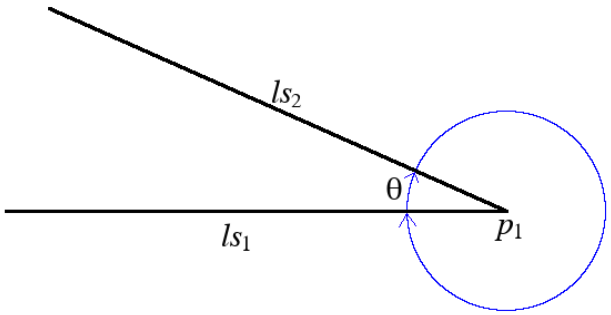


Figure 2.10: This figure depicts two line-segments and the transformations applied on ls_2, which are used in the evaluation scenarios.

The evaluation shows the behaviour of the error metrics in relation to different transformations such as translation, rotation and scaling (change of the line-segment length). The plots in the following figures show the error space in 3D to better visualize the error. Not all error spaces are plotted for every distance criteria, as for some of the measures the error space was zero in certain scenarios.

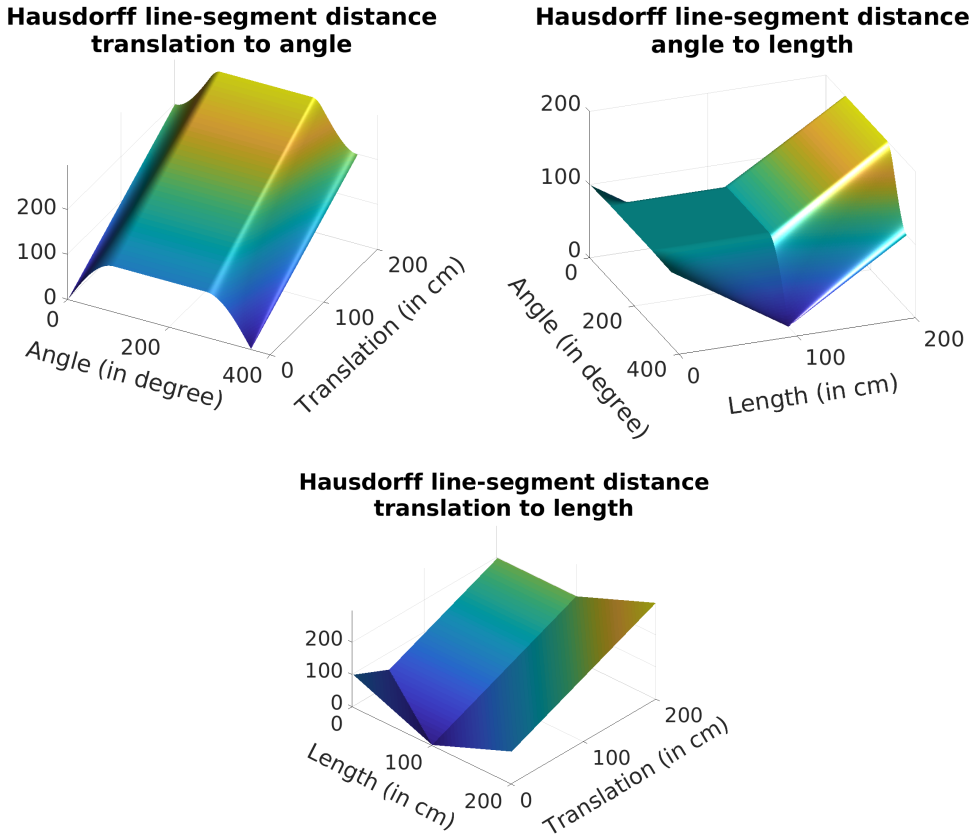


Figure 2.11: *Top left:* The plot shows the error space of the *Hausdorff line-segment distance* criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom:* The error space is shown for the scenario in which the line-segments lie on the same infinite line, and translational and length changes are applied.

The *Hausdorff line-segment distance* is affected by translation and by rotation, which can be seen in the top left plot of figure 2.11. The error of the rotation is exceeded by the error induced from the translation in the areas in which the rotation is small. And due to the δ_H's *max* function the influence of the rotational component is zero in those areas. Thereby, the function is not monotonically increasing. The translation, on the other hand, always affects the error measure because it influences the distance between the line-segments, also when they are rotated. The plots at the top right and at the bottom show that the error increases up to a certain degree, when the line-segments are of different lengths. While changing the angle additionally adds up to the error (see top right plot of figure 2.11), a translation can reduce the error, which is induced from different line-segment lengths (bottom plot). If the translation is larger than the difference in length, then it outweighs the length difference and only the translational error determines the overall error. The error space plots of the *Hausdorff line-segment distance* metric shown

in the paper [WP16] can be reconstructed by regarding the four distances between the endpoints only, instead of searching for the maximum orthogonal distances. We do not know why they have deviated from the formulas when creating their plots.

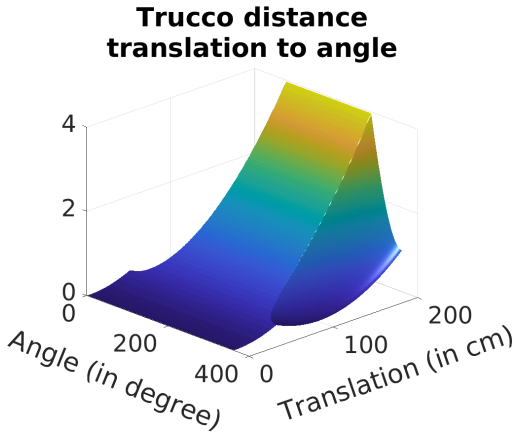


Figure 2.12: The plot shows the error space of the inverted *Trucco distance* metric criterion in relation to translational and angular changes.

The *Trucco distance* takes into account the size of the smaller line-segment and the minimum of the four possible endpoint distances. The magnitude of the error is by far the smallest in comparison to the other error measures. The angle only has an influence while the rotation is small and the translation is at least half of the line-segment's length. The function is not monotonically increasing, as most of the error space is only affected by the translation. Thus, the function is not suitable to calculate a dissimilarity score if the rotation between the line-segments is important. The plot in figure 2.12 shows the inverse of the *Trucco distance*. This better visualizes the error measure. The authors wanted a significance measure of proximity, that yields high values if line-segments are similar. For this reason, they placed the minimum endpoint distance in the denominator of the fraction. The error space plot of the *Trucco distance* metric shown in the paper [WP16] can be reconstructed by removing the squaring operation from the equation 2.14 and by also inverting it. We mention this here to avoid any confusion, as Wirtz and Paulus do not mention it in their paper.

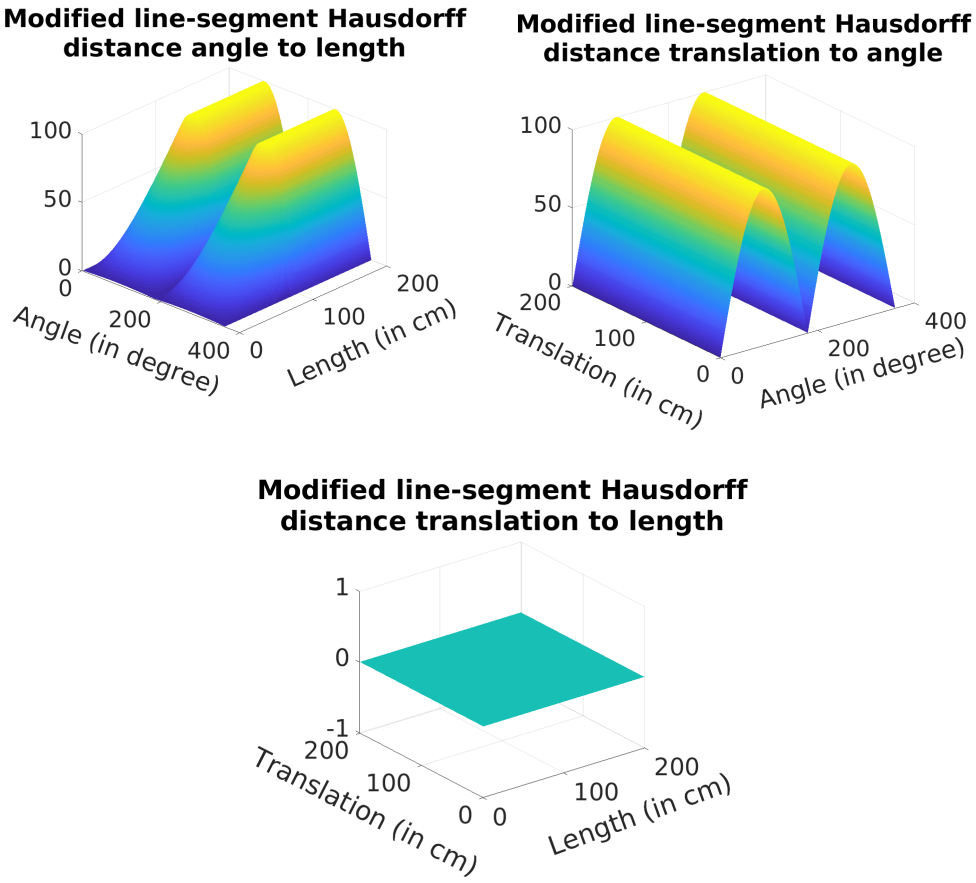


Figure 2.13: *Top left:* Here we see the error space of the *modified line-segment Hausdorff distance* criterion when the length and the angle of a line-segment are changed. *Top right:* The plot shows the error space of the same criterion in relation to translational and angular changes. *Bottom:* The error space is shown for the scenario in which the line-segments lie on the same infinite line and translational and length changes are applied.

The *modified line-segment Hausdorff distance* does not take the translation of the line-segment into account. This is clearly visible in the plots of figure 2.13. Also, if two line segments are parallel, then the error is zero. Otherwise, the function weights the size of the smaller line-segment by the sinus of the angle between the elongated line-segments. This error measure is quite simple and not a good choice for tasks such as tracking or optimization. It can be a good choice in specific applications in which rotations account for a major part of the dissimilarity between line-segments.

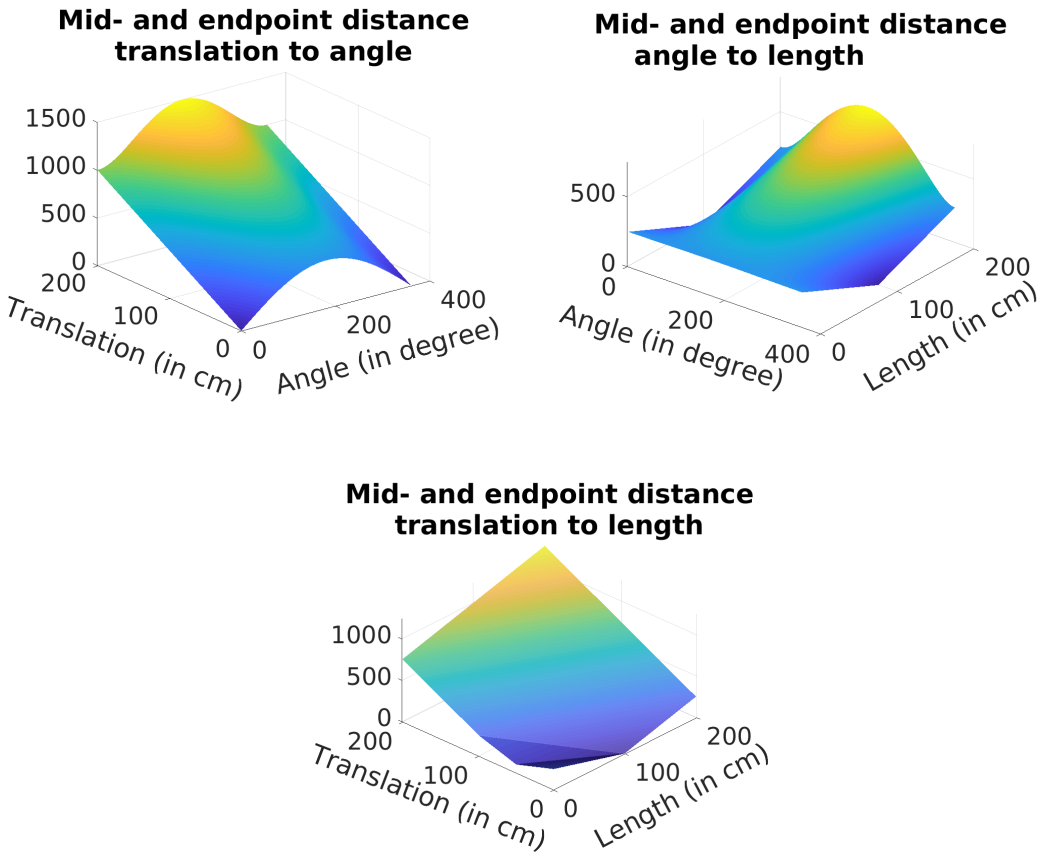


Figure 2.14: *Top left:* The plot shows the error space of the *mid- and endpoint line-segment distance* criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom:* The error space is shown for the scenario where the line-segments lie on the same infinite line and translational and length changes are applied.

The *mid- and endpoint line-segment distance* has a clearly vaulted error space when the angle between line-segments is changed, which can be seen in the top plots of figure 2.14. It increases linearly with an increased translation and is proportional to the rotation angle. The maximum angular error is induced in the case of a 180° rotation. This is different from other error measures in which the angular error is the highest at 90° and lower again at 180°, where the line-segments lie on the same infinite 2D line. But one has to keep in mind, that the error measure does not implicitly penalize an angle difference. If two line-segments were next to each other, a rotation, which brought the line-segments closer together on one end and farther apart on the other end, would hardly affect the error.

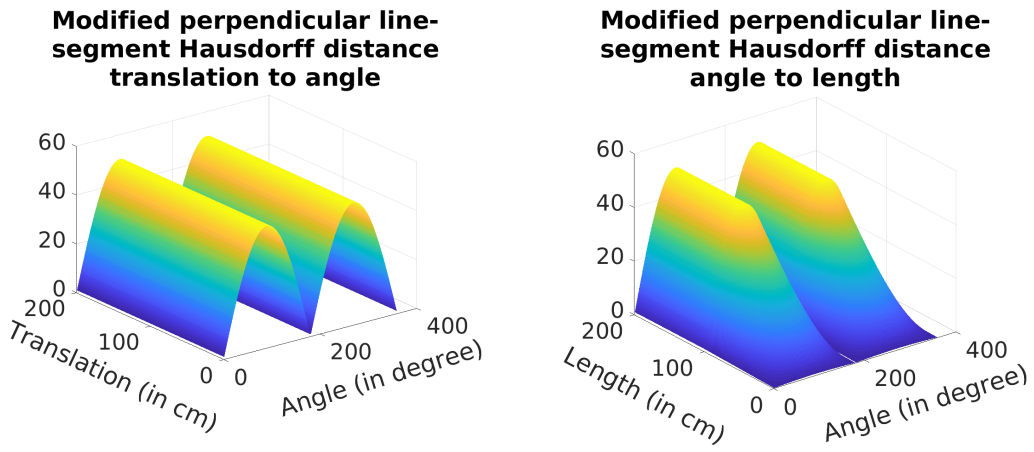


Figure 2.15: *Top left:* The plot shows the error space of the *modified perpendicular line-segment Hausdorff distance* criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed.

The *modified perpendicular line-segment Hausdorff distance* criterion yields the highest angular error when the line-segments are perpendicular to each other. This can be observed in both plots of figure 2.15. The angular error reduces again to zero when the line-segments lie on the same infinite 2D line. The translation does not have an effect in this case. However, the translation does contribute to a greater error, when the line-segments are rotated relative to each other. The *modified line-segment Hausdorff-distance* only takes the angle between the line-segments and their length into account. Opposing to that, this version uses the perpendicular endpoint distances as seen in equation 2.16. The angular distance is not involved in the formula, but it implicitly contributes to the error by influencing the perpendicular distances. The error space is not monotonically increasing. The plots shown in the paper [WP16] can be reconstructed using the following equation, instead of equation 2.16:

$$d_{\perp 1} = \min(\min(d_{\perp}(ls_1, ls_{2p_1}), (ls_1, ls_{2p_2})), \min(d_{\perp}(ls_2, ls_{1p_1}), (ls_2, ls_{1p_2})))$$

$$d_{\perp 2} = \max(\max(d_{\perp}(ls_1, ls_{2p_1}), (ls_1, ls_{2p_2})), \max(d_{\perp}(ls_2, ls_{1p_1}), (ls_2, ls_{1p_2})))$$

Note that this equation is different from the *modified perpendicular line-segment Hausdorff distance* by Maylor K. Leung and Yongsheng Gao, Yu Xiaozhou [YLG04].

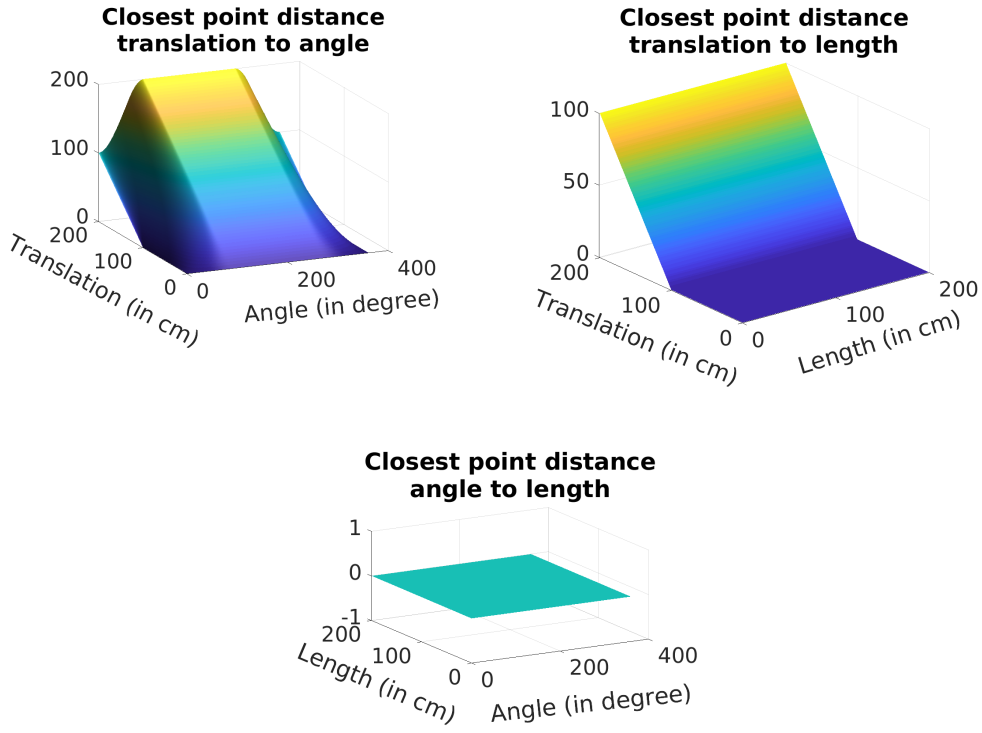


Figure 2.16: *Top left:* The plot shows the error space of the *closest point line-segment distance* criterion in relation to translational and angular changes. *Top right:* The error space of the same criterion is shown for the scenario, in which the line-segments lie on the same infinite line and translational and length changes are applied. *Bottom:* Here we see the error space when the length and the angle of a line-segment are changed.

Independent from the rotation angle, the *closest point line-segment distance* yields zero error, as long as the line-segments touch each other. If there is a translation, then a smaller angle reduces the error, as long as the rotation is below 90°. This can be seen in the top left plot of figure 2.16. At a rotation below 90° the closest distance between the two line-segments is perpendicular from one line-segment to the closer endpoint of the other line-segment. In the case that the rotation is above 90°, the closest connection between the line-segments is simply between the two closest endpoints. In that case, the error only depends on the translation, which is clearly visible in the error space. The error space is not monotonically increasing, as can be seen in all three plots. This line-segment error criterion is very basic and cannot distinguish two identical line-segments from two line-segments that touch or cross, but are perpendicular to one another.

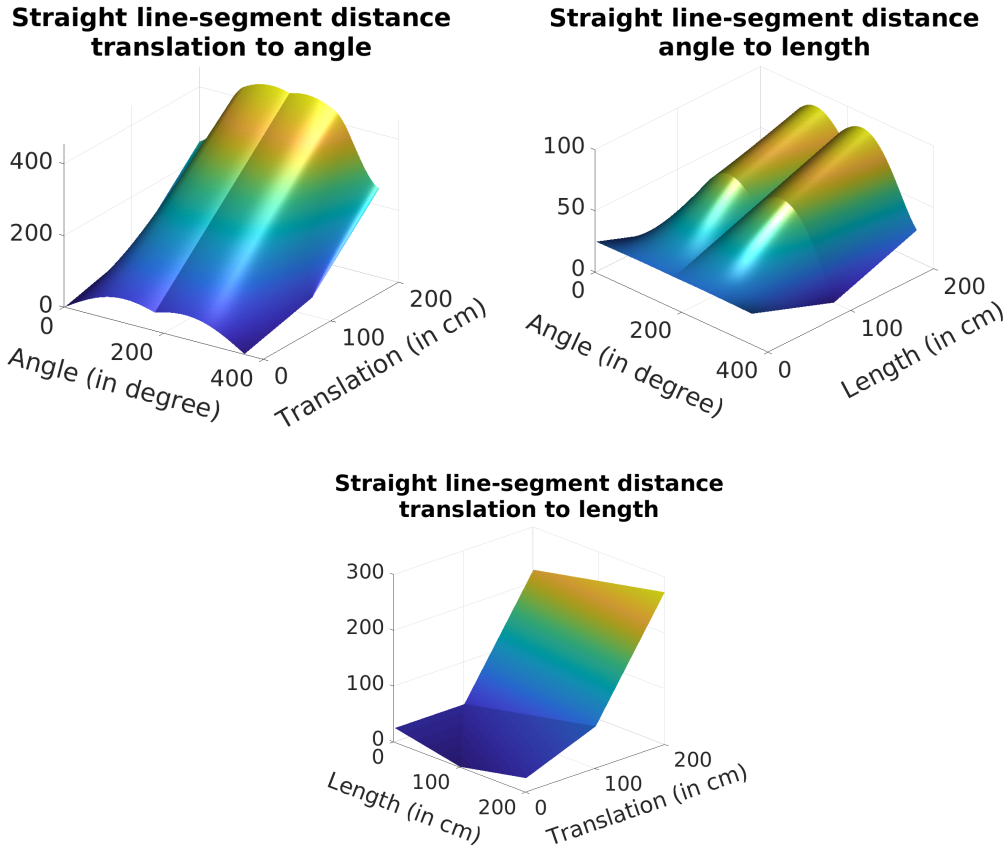


Figure 2.17: *Top left:* The plot shows the error space of the straight line-segment distance criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom:* The error space is shown for the scenario in which the line-segments lie on the same infinite line and translational and length changes are applied.

The *straight line distance* combines the *closest point line-segment distance* with the *modified line-segment Hausdorff distance* and with a measure for translation which is based on the endpoint distances and the length of the line-segments. The plots in figure 2.17 show that an angular difference between the line-segments always leads to an error above zero. The maximum angular error is induced when the line-segments are perpendicular. The error reduces when the rotation proceeds to that extent, that the line-segments lie on the same infinite 2D line. A larger translation continuously leads to an increased error. By using the formulas provided by the authors, we were not able to reconstruct the exact same plots as shown in [WP16] for this error measure.

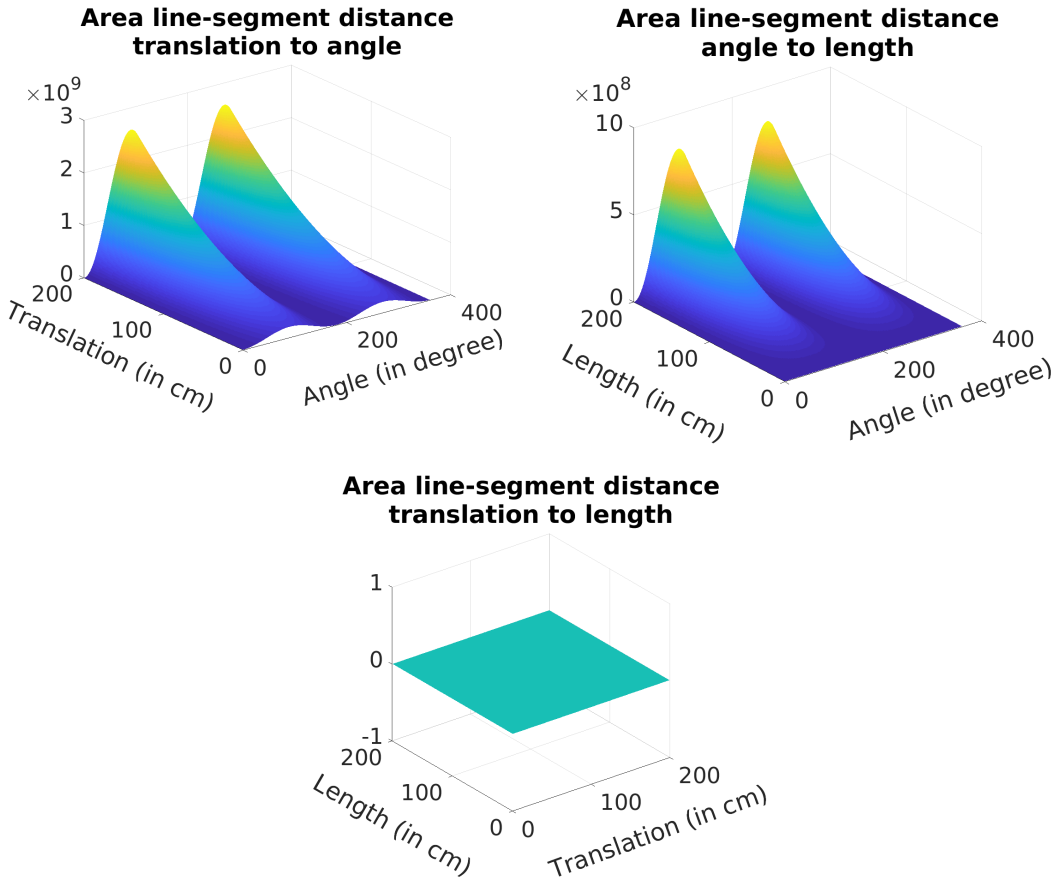


Figure 2.18: *Top left:* The plot shows the error space of the area based line-segment distance criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom:* The error space is shown for the scenario in which the line-segments lie on the same infinite line and translational and length changes are applied.

The area based line-segment similarity criterion from Aleš Jelínek and Luděk Žalud [JŽ17] yields higher errors when the angle, length or translation of one line-segment is increased. This can be seen in the plots of figure 2.18. The angular error is at its maximum when the angle is 90° or 270°. There is no error at a 180° rotation of the line-segments, as they lie on the same infinite line. The bottom plot of figure 2.18 also shows that there is no error, as long as the line-segments lie on the same infinite line. Of course there would be an error if the line-segments were parallel, but displaced to each other as this would span an area between them. The error space is monotonically increasing. In comparison to other methods the magnitude of the error value is quite high.

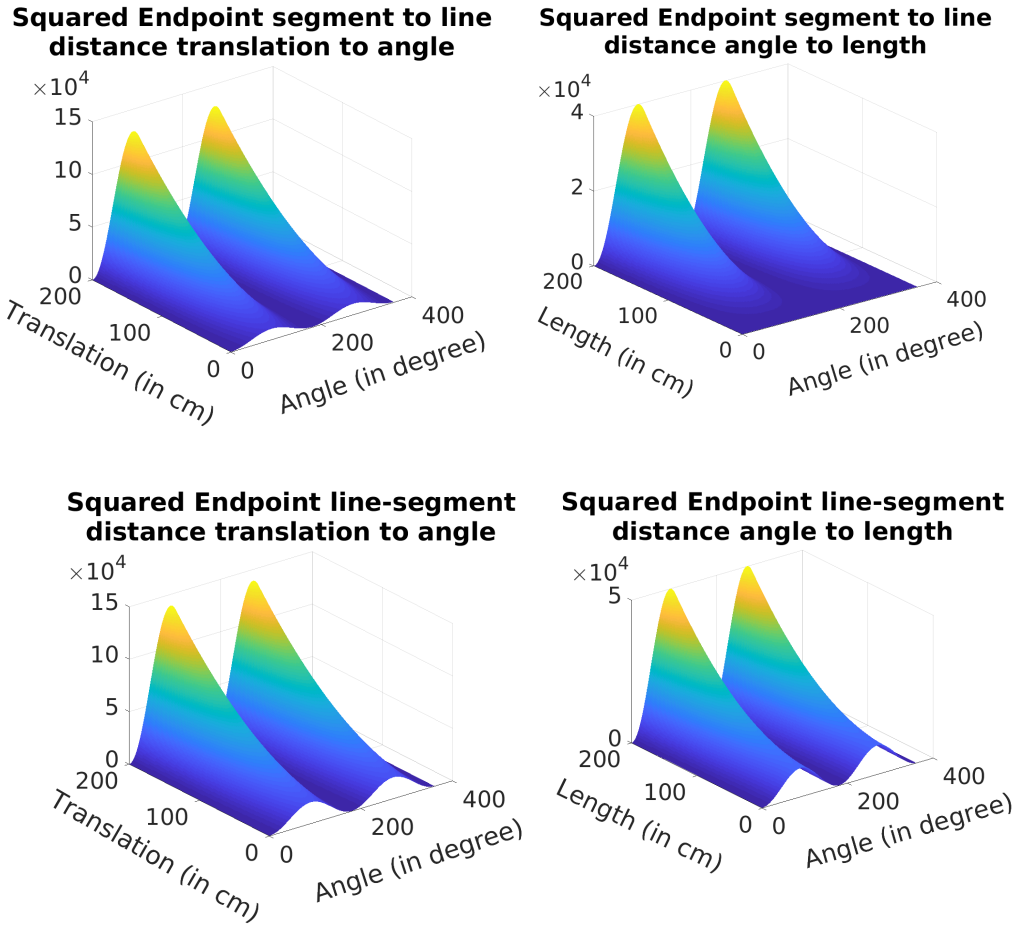


Figure 2.19: *Top left:* The plot shows the error space of the squared endpoint line-segment to infinite line distance criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom left:* The error space is shown for the scenario in which the endpoints of both line-segments are projected respectively to the infinite line of the other line-segment, here for the relation to translational and angular changes. *Bottom right:* The plot shows the same scenario as the left plot, but in terms of the angle and length changes.

The error space of the squared endpoint line-segment to infinite line distance criterion is shown in the top plots of figure 2.19. The line which is not transformed is assumed to be the infinite line here. We can see that the error space is monotonically increasing when the angle, length or translation is changed. The criterion does not take into account, in which direction the line-segments are facing. Thus, the angle dependent error is low at 0° as well as at 180° rotation. The criterion is a good choice if only the endpoints of one

of the line-segments are known. If the endpoints of both line-segments are known, the criterion can easily be changed in such a way that the endpoints of both line-segments are projected to the corresponding infinite line of the respective other line-segment. This makes the error independent of the choice of the end points of the line-segment. The error spaces for this version are shown in the bottom plots of figure 2.19. If the lines lie on the same infinite line, then the error is zero. This is not plotted here, as the plot looks identical to the corresponding one in previous scenarios such as the area line-segment distance (see the bottom plot in figure 2.18).

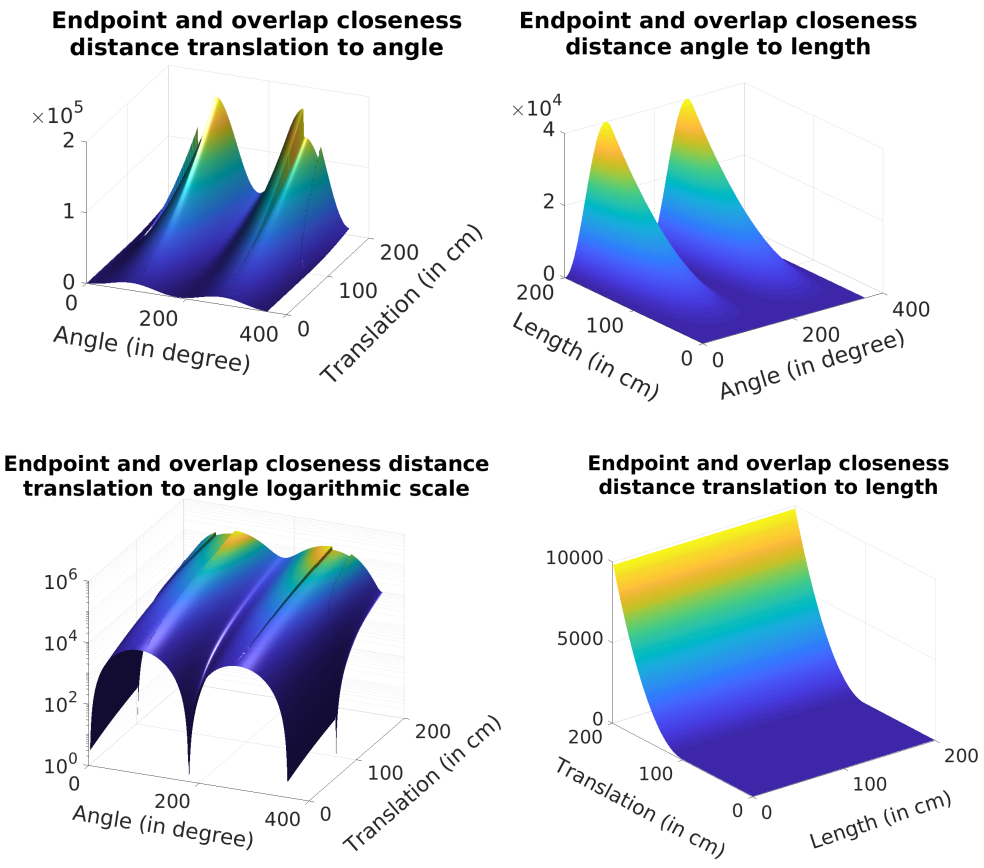


Figure 2.20: *Top left:* The plot shows the error space of the *endpoint and overlap closeness distance* criterion in relation to translational and angular changes. *Top right:* Here we see the error space of the same criterion when the length and the angle of a line-segment are changed. *Bottom left:* The error space is shown for the same scenario as in the top left plot, but with a logarithmic scale on the z-axis. *Bottom right:* The plot shows the error space for the scenario in which the line-segments lie on the same infinite line and translational as well as length changes are applied.

The plots in figure 2.20 visualize the error space for the *endpoint and overlap closeness distance* criterion. The plot in the top right shows the characteristics for length and angle changes. In that scenario, the two line-segments always touch each other at the origin of the coordinate system. Thus, the closeness distance term is never applied. This stands in contrast to the scenario shown in the top left plot, in which translational and angular changes are applied. There, the line-segments do not constantly overlap each other and the closeness term gets applied. This becomes quite apparent at discontinuous jumps of the error value which occur, for example, when the angle is 90° or 270°. There, the error jumps when the line-segments stop to have an overlap. The larger the translation, the higher the jump of the error value. This also happens when the transformed line-segment is translated so far that the line-segments do not touch, but the rotation causes them to overlap again. These jumps of the error value form edge contours in the error space that look quite large. They get less significant if we look at them in the logarithmic scale, which is visualized at the bottom left plot of figure 2.20. Due to the squared error terms, the error itself quickly rises when the distance of the line-segments increases. It depends on the application which areas of the error space are more relevant and if one wants to strongly penalize certain conditions such as a missing overlap. If one wants to avoid the edge contours, one could simply constantly apply the closeness distance term instead of applying it depending on an overlap criterion. Another important difference to the previous squared endpoint line-segment to infinite line distance criterion is the error space for the translation to length scenario, in which both line-segments lie on the same line. In this scenario, the error space is not zero everywhere, as it is the case with the previous distance criterion. As soon as the line-segments stop touching each other, the error rises. This is advantageous when the distance criterion is utilized in a line-segment tracking scenario, as it is done in chapter 3.

2.3.3 Summary

The answer to the question which line-segment distance criterion should be used, depends on the task it will be used for. If the usage scenario is an optimization used in a SLAM or SfM system, for example, then one would want a distance function that is monotonically increasing. In other words one that has no jumps or planar areas in the relevant error space. For these scenarios, the *mid- and endpoint distance*, the *modified perpendicular line-segment Hausdorff distance*, the *straight line-segment distance*, the *area line-segment distance* and the *squared endpoint segment to line distance* are appropriate choices. If the endpoints are not available for both lines, but only for one of them, then the *squared endpoint segment to line distance* is a good choice. Should the error be non-zero when the line-segments are rotated but lie on the same infinite line, then the *mid- and endpoint distance* and the *straight line-segment distance* are good options. Except for the magnitude of the error value, the error spaces of the *squared endpoint segment to line distance* and the *area line-segment distance* are similarly shaped. If one wants to use the similarity measure to perform line-segment tracking, then jumps

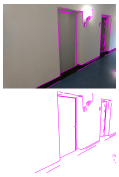
or planar regions in the error space may not be a problem and one can give priority to certain conditions like a perpendicular overlap between the line-segments. In this case, the *endpoint and overlap closeness distance* criterion would be the favoured selection. The *closest point line-segment distance* can be used if it is mainly important whether two line-segments touch. To summarize, the choice of the line-segment error function mainly depends on the problem which is to be solved.

In the next chapter, we will elaborate our *Line only stereo Visual Odometry* system which utilizes some of the presented line representations and line distance measure criteria to track, optimize and finally perform visual odometry using lines.

Chapter 3

Line only stereo Visual Odometry

This chapter presents our *Line only stereo Visual Odometry* system. The chapter is mostly cited from the paper:



3.1 Chapter Overview

The research on visual odometry (VO) [NNB04] as well as on visual simultaneous localization and mapping (V-SLAM) started many years ago, but both are still actively researched. They are required in upcoming applications such as autonomously driving cars, unmanned aerial vehicles (UAVs) and augmented reality.

The classical approach to VO starts with searching for correspondences between consecutive images. Most times, a certain type of feature detector is used to find points in areas that are rich in information. These areas are often found in textured regions of the image. In a following step, those points are extracted and matched to points found in the other image, using a certain type of descriptor [RRKB11][BTVG06]. Reprojecting the points and minimizing the errors between reprojected and observed points leads to the camera transformation. These point features lie on spots of the image that are as distinctive as possible. Such spots often are corners or blobs. A point feature contains little structural information compared to a line which can represent numerous points at once, laying on a structure such as an edge. As a consequence, in an image which is mostly structured and barely textured, point feature matching as well as the VO based on it deteriorate, whereas line features are a good way to represent the structure (see figure 3.1).

Line features in general are not researched as long and thoroughly as point features. Reasons therefore are the higher computational cost to extract them from an image

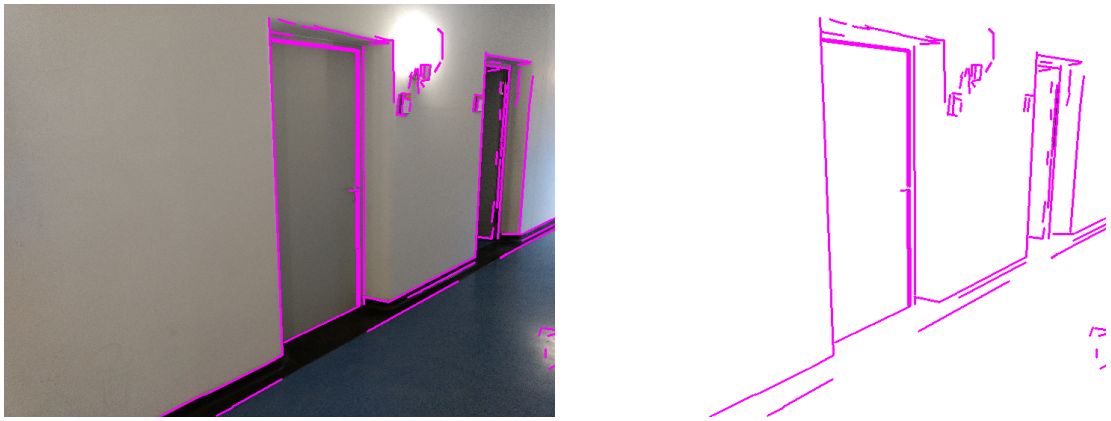


Figure 3.1: On the left we see a scene in a corridor that is rarely textured. Point features can hardly be found, whereas enough lines are available to do visual odometry. Also the structure is well recognizable with just a few lines as we see in the right image. [LRS19] ©2019 IEEE

[VGJMR12] and partial occlusions that complicate the matching process. There are multiple methods that use line features to sample points along the line and then use the sampled points in existing methods made for point features. Gomez-Ojeda et al. [GOBGJ16] detect lines and sample points along them to pass them to [FPS14] which is designed for point features. Pumarola et al. [PVA*17] detect lines and use their endpoints to treat them as if they were points and re-use the ORB-SLAM architecture [MAMT15]. This, too, results in a non-minimal line parametrization in the bundle adjustment (BA). Falling back to representing lines through points or overparameterising lines does not utilize their full potential, thus, we avoid doing so where possible.

The main contributions of our work are as follows:

- A visual odometry approach is proposed that yields accurate and stable results solely using lines. We show the improvement by comparing it to other state of the art methods.
- We propose a novel heuristic filter for line-segment matching that improves line-matching results for tracking and for arbitrary image to image matching of lines.
- A line tracking method is described that is faster than descriptor-based methods as it does not require descriptors. In sequences without high accelerations, it is capable to do all necessary line-segment tracking solely.
- We present the first VO approach using the Cayley representation for 3D lines, avoiding overparameterization in the optimization.

To demonstrate the performance of this algorithm, it is evaluated on the EuRoC dataset [BNG*16]. A comparison against state of the art methods shows that we can keep up

with, and also do better than these methods in terms of accuracy and stability. We want to emphasize that we solve all EuRoC sequences more accurately than the other existing line only methods. Additionally the methods are evaluated on a difficult synthetic scene to test their stability. There we push the algorithms to their limits and can demonstrate a higher stability of our version even when StVO-PL [GOGJ16] uses points and lines while we only use lines. The next Section gives a view onto related works. In Section 3.3 we describe our system and in 3.4 the results are presented. In Section 3.5 we draw a conclusion of our proposed method.

3.2 Related Work

In this section we discuss related work on VO as well as that on V-SLAM since V-SLAM systems usually build upon VO systems. It is common to group VO approaches in direct, and feature-based methods.

Direct methods: These methods use the *photometric error* of the images working directly on the pixel intensity values [IA99]. On one side, they save computational time on feature extraction and matching. But on the other side, they have to optimize the six degrees of freedom motion by *image-to-model alignment* which can be computationally costly, depending on their model and alignment process. When the whole image is taken into account, the model becomes very detailed. Yet, it needs a powerful GPU to be processed in realtime [NLD11]. Thus, many direct methods are sparse [EKC18] or semi-dense [ESC14] [FPS14] [ESC13] to save processing time. They reduce the focus to regions which they choose as being informative, which are mostly regions with an intensity gradient. This, again, brings them closer to feature-based methods on which we will lay our focus from here on.

Point feature based methods: These methods extract point features from an image and match them to the next image. They do some outlier handling and compute the camera motion between the images. An early VO approach was [NNB06] where the authors implemented an iterative procedure to estimate their stereo camera motion. They used some form of the Harris corner detector to detect features and matched them comparing their normalized correlation. Geiger et al. [GZS11] proposed a method which uses a sparse feature matcher to achieve an accurate and fast VO estimation besides a scene reconstruction. The first SLAM systems that showed real-time performance were MonoSLAM [DRMS07], which still had to use quite sparse landmarks due to computational limitations, and Parallel Tracking and Mapping [KM07] that took advantage of the increased processing power, using parallel threads for tracking and for mapping. ORB-SLAM [MAMT15] further combined multiple existing techniques and fine-tuned them which set a new standard concerning the robustness, the accuracy of the pose estimation and the runtime of the algorithm. Still, these algorithms rely on point features and, thus, they depend on textural information in the image. To overcome this problem, line features came into focus.

Line feature based Methods: Unlike point features, line features do not rely on texture, structure is sufficient. One of the first approaches working with lines were [SRD06] where the authors built an Extended Kalman Filter V-SLAM system that uses lines which connect several close keypoints. Eade and Drummond [ED06] proposed the use of parts of lines and called them *edgelets*. Some methods ([LRL*18][PVA*17]) build upon existing point feature methods and extend them by placing points on extracted lines and making more use of that information by giving them a higher weight in their optimization.

Only few methods have been proposed which showed that they can work reliably on line features only and do not need points (and consequently no texture). Yang and Scherer [YS17] extend a direct method and refit each line in every iteration by using RANSAC and a maximum likelihood estimation. The method StVO-PL [GOGJ16] uses a probabilistic approach for VO with points and line-segments. With PL-SLAM [GOZNM*17] they later on extended their method to a V-SLAM system. PL-SVO [GOBGJ16] extends the well known SVO [FPS14] by supporting line features in addition to points.

To show our contribution in improving stability and robustness when relying on lines only we chose StVO-PL [GOGJ16], PL-SLAM [GOZNM*17], PL-SVO [GOBGJ16] for comparison since they are the most advanced approaches incorporating lines. We also benchmark against ORB-SLAM 2 [MAT17] as it is a well known state of the art method.

3.3 LVO: Line only stereo Visual Odometry

Figure 3.2 gives a compact visualization of the proposed framework that we denote as Line only Visual Odometry (LVO). Basically, we track line-segments over consecutive stereo frames in order to determine pairs of 3D and 2D features. Therefore, we rectify the images, detect line-segments, extract descriptors and compute stereo matches in the frame preparation step (see figure 3.2). 3D line-segments are triangulated at the end of each frame processing step. General information about the line-segment triangulation, detection and feature description is given in section 3.3.2.

The main difficulty when tracking line-segments is to become robust to outliers during the matching procedure. In section 3.3.1 we propose a heuristic filter for reliable matching of line-segments. This heuristic is applied after the tracking and it noticeably increases the number of matches.

For line-segment tracking we propose two methods:

1) A geometrical tracking approach in which uniform camera motion is assumed. Since we are tracking line-segments instead of points, we achieved the matching of the features based on geometry only, without using feature-descriptors. We denote this method as *Motion Tracking*; it is treated in section 3.3.3.

2) For the rare cases in which the motion assumption is violated, we provide a tracking procedure based on feature-descriptor matching, additionally using the heuristic filter to increase the reliability of the matches. We refer to this method as *Descriptor Tracking* and it is explained in section 3.3.4.

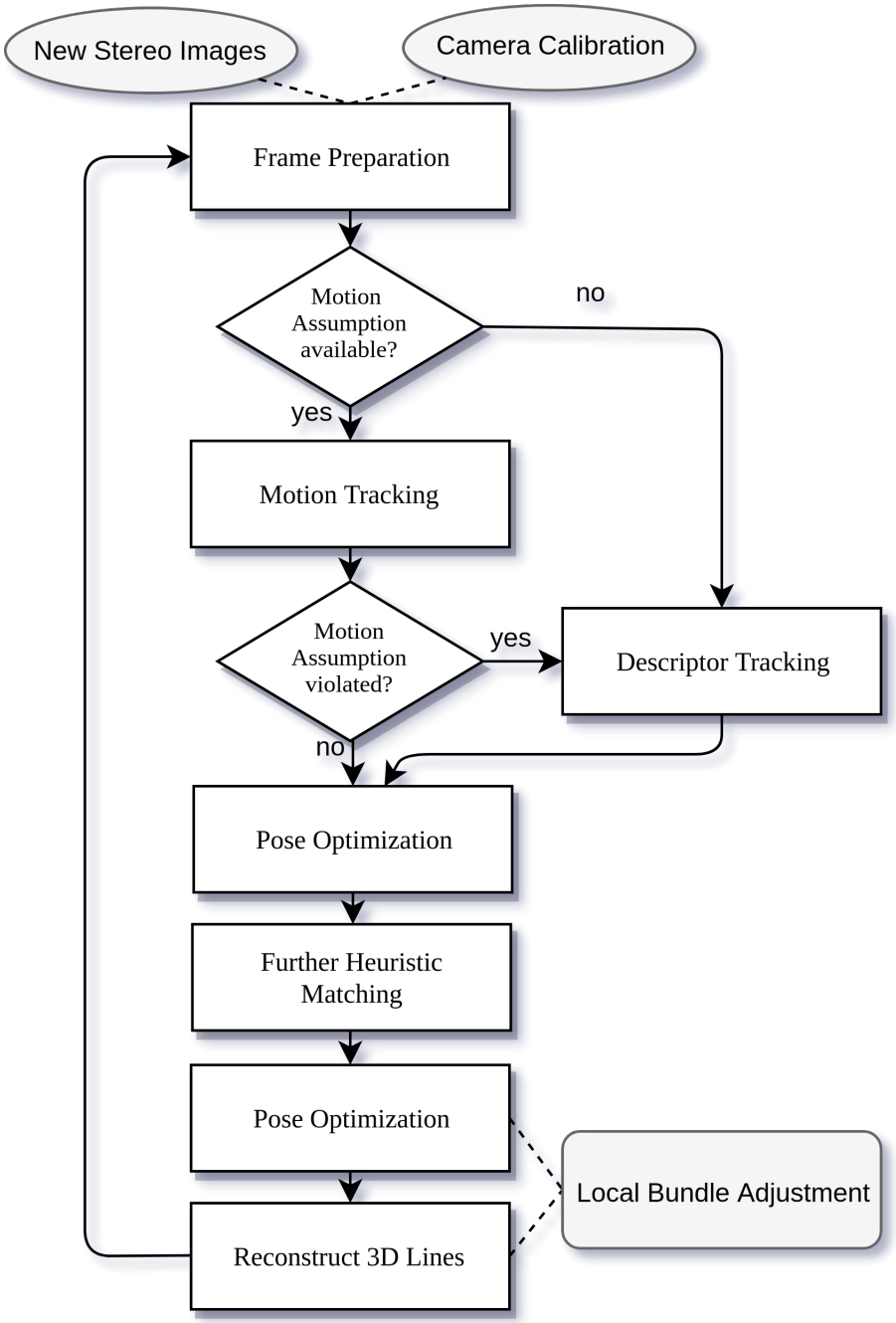


Figure 3.2: Basic framework of LVO. Line-segments and stereo matches are determined in *frame preparation*. After Tracking the segments over consecutive frames via *motion-* or *descriptor Tracking*, further line-matches are gathered applying the heuristic descriptor matching. The *motion assumption* is based on the previous relative motion and will be violated if the current relative motion deviates clearly from the latter mentioned. Given the line correspondences the exact relative motion is optimized in the *pose estimation*. Finally, new 3D line-segments are triangulated. In the *local bundle adjustment* thread adjacent poses are further refined to compensate for drift. [LRS19] ©2019 IEEE

The gathered feature correspondences are then used to estimate the motion by optimizing an incremental pose. More detail on the pose optimization is given in section 3.3.5. When the camera is moving fast it becomes hard to track the features using a motion model. Therefore we gather further correspondences by matching the line descriptors, applying the heuristic from 3.3.1 and again refine the pose. The correspondences and poses of the last N frames are stored and then used to further optimize the local trajectory by a threaded windowed bundle adjustment. More details on that are given in section 3.3.6.

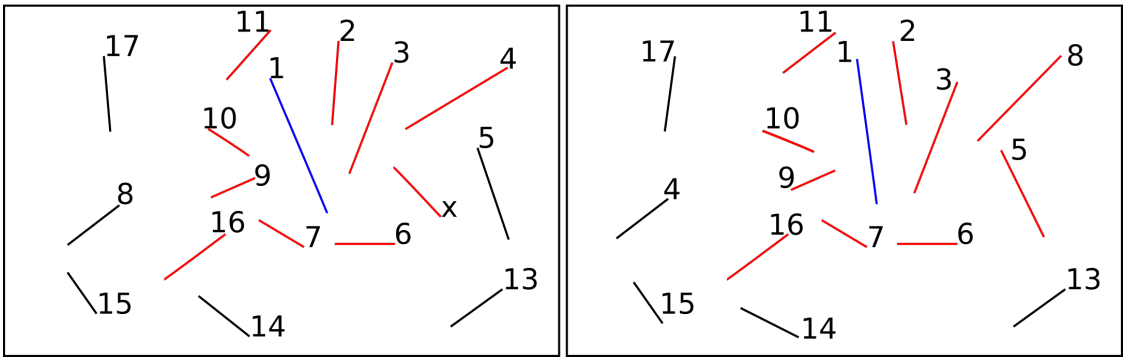


Figure 3.3: This figure describes our heuristic matching approach. Two images with detected line-segments are shown. The numbers on the segments show the descriptor matching results. Here our heuristic checks if the blue segment is a match. The red segments are its ten closest neighbours. Our heuristic would correctly declare a match as eight out of the ten neighbours are in both neighbour groups. (We used a ratio of atleast six out of ten.) [LRS19] ©2019 IEEE

3.3.1 Heuristic on reliable Line-Segment Matching

At sudden movements of the camera, we can not rely on motion predictions. To still be able to track features we have to use descriptor methods. For our purposes ordinary feature matching yields too many outliers. Therefore we developed a heuristic to improve matching quality on line-segments, by considering not only the single features, but additionally taking into account the surrounding features at the matching procedure. Consider two candidate line segment matches $\mathbf{l}_1, \mathbf{l}_2 \in \mathbb{R}^4$ parametrised by their endpoints and $\Omega(\mathbf{l}_1)$, $\Omega(\mathbf{l}_2)$ contain the $n \in \mathbb{N}$ line-segments located closest to $\mathbf{l}_1$ and $\mathbf{l}_2$, respectively.

Then we can simply determine $\Omega(\mathbf{l}_1) \cap \Omega(\mathbf{l}_2)$ containing all corresponding line-segments which are in both neighbourhoods, given a similarity metric on line-segments e.g. descriptor distances. In fact, regional properties on line-segments like relative locations, distances and angles are not invariant to perspective projection. But on non-rigid objects we can assume that for small local neighbourhoods the features in the neighbourhood still remain there. Besides, we need to care about effects like view-dependent occlusion

and the fact that features have different disparities and, therefore, can move out of, or into the neighbouring region at different views. Consequently, we are forced to formulate the heuristic not too strict and simply count the number of surrounding features that match validly according to their feature descriptors, such that $|\Omega(\mathbf{l}_1) \cap \Omega(\mathbf{l}_2)| = k_{th} \in \mathbb{N}$. With k_{th} defining the minimum number of line-segments that must be in both neighbourhoods to declare $\mathbf{l}_1, \mathbf{l}_2$ as a match (see figure 3.3 as illustration).

By applying this heuristic we were able to reduce the number of outliers and noticeably increased the number of matches by 30% on average (over all EuRoC [BNG*16] sequences, see figure 3.6).

3.3.2 Line-Segment Features

For detecting line-segments on the input images the Line Segment Detector (LSD) from von Gioi et al. [VGJMR12] is used. For the heuristic filter proposed in 3.3.1 and at motion tracking in 3.3.3, we need to find line-segments lying in a certain region as efficiently as possible. Therefore, we additionally build a multi-dimensional grid structure on each frame, assigning every line-segment to the grid-cells which it runs into/through. Furthermore, the line-segments are grouped according to their polar angle allowing a more precise matching.

In order to match features based on their local appearance the line band descriptor (LBD) [ZK13] is applied. The LBD is used for detecting stereo features (see 3.3.2), and in the heuristic filter in 3.3.1.

Line-Segment Stereo-Matching

We first compute a regular descriptor matching on the line-segments, which are detected in both images, to find suitable stereo pairs using the LBD [ZK13] descriptor and the *heuristic filter* from 3.3.1. Since the images are rectified right after the new frames have been fetched, according to the epipolar-constraint, a point in one image must lie on the same vertical offset in the other stereo image [HZ03]. Therefore, we exploit the epipolar-constraint by checking if the candidate pair of line-segments overlaps sufficiently in the vertical image direction. Due to disparate occlusion among the stereo views we cannot simply check for the alignment of both lines.

Line-Segment Triangulation

We associate both corresponding line-segments to a line of infinite extent while keeping the endpoints. The 3D line is then triangulated from the line correspondences by intersecting both of their back projected planes, as it is proposed in [HZ03]. For a precise tracking of line segments it is necessary to determine the 3D endpoints of the triangulated line. To obtain these, we triangulate the endpoints from the kept 2D endpoints and determine the closest intersections of them with the triangulated line. The drawback of

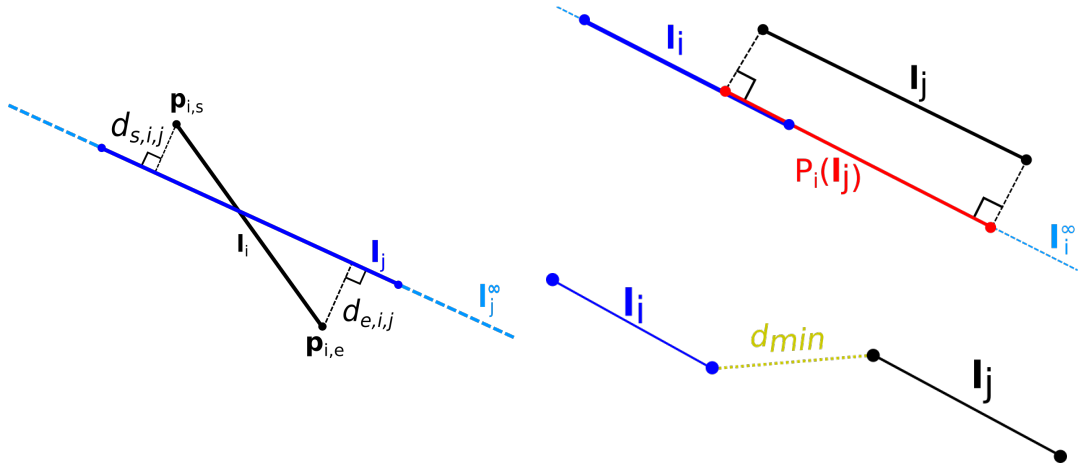


Figure 3.4: *Left:* Visualization of metric $m(\cdot)$ in equation 3.1. The perpendicular distances $d_{s,i,j}$ and $d_{e,i,j}$ from the endpoints of the line-segment $\mathbf{l}_i = (\mathbf{p}_{i,s}, \mathbf{p}_{i,e})^T$ to the infinite expansion of line segment $\mathbf{l}_j^\infty$ are squared and summed up for the error metric $m(\cdot)$. *Top right:* The red line visualizes the line-segment $P_i(\mathbf{l}_j)$ obtained by orthogonally projecting the endpoints of $\mathbf{l}_j$ to $\mathbf{l}_i^\infty$. In the case that $\mathbf{l}_i$ and $P_i(\mathbf{l}_j)$ overlap, the sum of the endpoint distances is a sufficient metric. *Bottom right:* If $\mathbf{l}_i$ and $P_i(\mathbf{l}_j)$ do not overlap, we additionally need to consider the minimum distance $d_{min}(\mathbf{l}_i, \mathbf{l}_j)$ between both line-segments (gold, dashed line). Without considering the minimum distance, two segments which are almost collinear but not overlapping would yield a too small distance error, thus the metric would not be sufficient. [LRS19] ©2019 IEEE

this method is that 3D lines which are close to intersecting the camera baseline cannot be triangulated with high certainty and lines which intersect the baseline cannot be reconstructed at all [HZ03]. But we argue that those lines are likely to be reconstructed from later views and, therefore, we will not lose most of the interesting lines in the long term.

3.3.3 Motion Tracking

Since matching by relying solely on feature-descriptors would yield too many false correspondences, we need to increase the reliability by considering an additional prior on the camera motion. We observed that vehicles are often situated in steady motion, even holonomic moving robots, like quadrotors, are most of the time moving with a smooth trajectory. Thus, a simple prior that incorporates smooth motion is to assume constant velocity as it is proposed in [MAMT15]. During the matching the 3D line-segments are projected into the image plane using the pose, which is at this point roughly predicted. For each projected segment we then collect all detected line-segments which are located nearby and select the closest one as 2D correspondent if the distance measure is below an empirically selected threshold. The following metric $m(\cdot)$ is used as distance measure

for the two line segments:

$$m(\mathbf{l}_i,\mathbf{l}_j) := \begin{cases} d_{proj}(\mathbf{l}_i,\mathbf{l}_j) & \text{if } \mathbf{l}_i \text{ and } P_i(\mathbf{l}_j) \text{ overlap} \\ d_{proj}(\mathbf{l}_i,\mathbf{l}_j) + d_{min}(\mathbf{l}_i,\mathbf{l}_j)^2 & \text{else.} \end{cases} \tag{3.1}$$

Here $\mathbf{l}_i,\mathbf{l}_j \in \mathbb{R}^4$, $i,j \in \mathbb{N}$ are the two line-segments parameterized by their endpoints, $P_i(\mathbf{l}_j)$ is the line-segment given by perpendicularly projecting the endpoints of $\mathbf{l}_j$ onto $\mathbf{l}_i$ and $d_{proj}(\cdot)$ is defined as:

$$d_{proj}(\mathbf{l}_i,\mathbf{l}_j) := d_{s,i,j}^2 + d_{e,i,j}^2, \tag{3.2}$$

where $d_{\{e,s\},i,j}$ denotes the perpendicular distance from the start and endpoint of line segment $\mathbf{l}_i$ to the infinite expansion of $\mathbf{l}_j$. Those distances are computed the same way as in section 3.3.5 and in [ZK14], whereas for the usage in metric $m(\cdot)$ line segment $\mathbf{l}_j$ is treated as infinite line $\mathbf{l}_j^\infty$. In figure 3.4 we provide a visualization of the metric. Note, when both line-segments are almost collinear but do not overlap, we additionally have to take into account the shortest distance $d_{min}(\cdot)$ between both segments. It turns out that using $m(\cdot)$ alone is sufficient to obtain robust tracking results in a way that no additional descriptor distance needs to be taken into account. With the motion assumption we implicitly reduce the number of possible matches for each line-segment which results in an overall decreased matching time, compared to standard descriptor matching. Moreover, the matches become more robust to outliers due to the limited number of candidate matches. If the number of matches exceeds a threshold, we treat the motion model as valid and further refine the guessed pose at the pose optimization step in section 3.3.5.

3.3.4 Descriptor Tracking

In our tests the motion tracking approach in section 3.3.3 most of the time yielded valid motion models. But on sudden movements the motion remained unpredictable for our method. Thus, if no motion on the camera movement exists we attempt to track the line-segments using a descriptor-matching method. Therefore, each line-segment is described based on its local appearance, which makes feature-descriptors less prone to camera motion and imaging effects. As descriptor we use the LBD [ZK13] which is explained in section 3.3.2. To become more robust to outliers we apply our heuristic filter introduced in section 3.3.1. The pose optimization in section 3.3.5 further uses the Huber loss function [H*64] to reduce the impact of the remaining outliers.

3.3.5 Pose optimization

In this section we describe the optimization of the pose $\theta \in SE(3)$ (where $SE(3)$ denotes the special Euclidean group of $\mathbb{R}^3$) given the tracked 2D-3D correspondences. The 3D line-segments $\Gamma \in \mathbb{R}^6$, as well as the appropriate 2D correspondences $l_d \in \mathbb{R}^4$ are parameterized by their endpoints. We project each 3D line model with the projection function

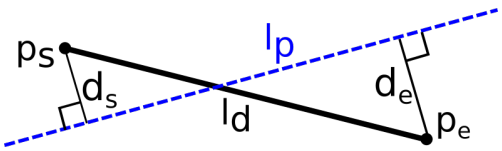


Figure 3.5: Error calculation in the pose optimization. The dashed blue line l_p is the infinite line projected from the Cayley 3D line model. The black line segment l_d is the detected line segment corresponding to the 3D line model. p_s and p_e are l_d's endpoints which are projected onto l_p with the projected distances d_1 and d_2 that are used for the error calculation. [LRS19] ©2019 IEEE

$\pi\colon SE(3)\times\mathbb{R}^6 \to \mathbb{R}^2$ into the image plane of the current frame using the estimated pose, where the projected line now is represented as infinite 2D line. This is depicted in figure 3.5 where l_p denotes the projected infinite 2D line and l_d denotes the observed line segment. We calculate endpoint distances between each segment l_d and the corresponding line l_p by projecting the start point p_s and the end point p_e of l_d onto l_p giving us the distances d_s and d_e respectively. We want to minimize these distances to optimize the pose. Therefore, we formulate a cost function to minimize:

$$\theta^* = arg\,\min_{\theta} \sum_{i\in M} \rho_h(e_{s,i} + e_{e,i}) \tag{3.3}$$

where M is the set of our 2D to 3D matches, ρ_h is the Huber loss function [H*64] and $e_{s,i}$ and $e_{e,i}$ are our errors from projection as proposed in [ZK14]:

$$\begin{aligned} e_s &= d_s^2 = (\pi(\theta,\Gamma)_\perp \cdot p_s)^2 \\ e_e &= d_e^2 = (\pi(\theta,\Gamma)_\perp \cdot p_e)^2, \end{aligned} \tag{3.4}$$

with $(\cdot)_\perp$ denoting the normal of the implicitly parameterized projected line. We use the Ceres Solver optimization library [AMO20] for the minimization.

3.3.6 Well Parameterized Local Bundle Adjustment on Lines

We perform a local bundle adjustment over the recent frames to improve the accuracy. However, unlike most other VO/VSLAM approaches, we do not use 3D line endpoints in our bundle adjustment. Endpoints are often unstable in their exact position along the line due to noise and partial occlusions. Furthermore using them in the optimization is not a good idea because they add unnecessary degrees of freedom to it which leads to an over-parametrization resulting in an inefficient optimization. As we need line-segments for motion tracking, we recreate them after the bundle adjustment. Outside the bundle adjustment, we represent our 3D lines $\Gamma \in \mathbb{R}^6$ with a normalized direction $l \in \mathbb{R}^3$ through a point $p \in \mathbb{R}^3$ and the endpoints are given by scalar values.

Inside the bundle adjustment, we follow the procedure supposed in [ZK14] and start by representing our lines in Plücker coordinates:

$$\Gamma = (\mathbf{m}, \mathbf{l}) \text{ with } \mathbf{m} = \mathbf{p} \times \mathbf{l}, \tag{3.5}$$

now m is called the line momentum and represents the normal to the plane through the line and the coordinate origin. The magnitude is equal to the distance from the line to the origin. There are two constraints that have to be enforced when working with Plücker lines:

$$||\mathbf{l}|| = 1 \textit{ and } \mathbf{l}^T\mathbf{m} = 0. \tag{3.6}$$

Fortunately, these constraints are guaranteed at initialization time. Unfortunately, it is inconvenient to enforce these constraints during the bundle adjustment optimization. Therefore, [ZK14] developed a Cayley representation of spatial lines which automatically guarantees that the constraints are satisfied during optimization: $v = (\omega, s)$ where

$$\begin{gathered}\omega = ||\mathbf{m}||,\ [\mathbf{s}]_x = (\mathbf{Q}-\mathbf{I})(\mathbf{Q}+\mathbf{I})^{-1},\ \text{ with} \\ \mathbf{Q} = \begin{cases} [\mathbf{l}, \mathbf{e}_1, \mathbf{e}_2] & \text{if } ||\mathbf{m}|| \leq \tau; \\ [\mathbf{l}, \frac{\mathbf{m}}{||\mathbf{m}||}, \frac{\mathbf{l}\times\mathbf{m}}{||\mathbf{l}\times\mathbf{m}||}] & \text{else.} \end{cases}\end{gathered} \tag{3.7}$$

τ is a small number close to zero (e.g.10^{-7}). It is used to overcome a singularity when $||\mathbf{m}|| = 0$ as $\mathbf{Q}$ would then degenerate to [**l**,**0**,**0**]. To avoid that, $\mathbf{Q}$ can be set to $[\mathbf{l}, \mathbf{e}_1, \mathbf{e}_2]$ with $\mathbf{e}_1$ and $\mathbf{e}_2$ being one pair of the orthogonal base of the nullspace formed by the linear system $\mathbf{l}^T\mathbf{x} = 0$. To get Plücker coordinates from the Cayley representation one simply calculates $\mathbf{l} = \mathbf{q}_1$ and $\mathbf{m} = \omega\mathbf{q}_2$ with:

$$\begin{aligned}\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3] &= (\mathbf{l} - [\mathbf{s}]_x)^{-1}(\mathbf{l} + [\mathbf{s}]_x) \\ &= \frac{(1 - ||\mathbf{s}||^2)\mathbf{l} + 2[\mathbf{s}]_x + 2\mathbf{s}\mathbf{s}^T}{1 + ||\mathbf{s}||^2}\end{aligned} \tag{3.8}$$

To calculate the error during the optimization, the Cayley line is converted to Plücker coordinates and then projected using the cofactor matrix of the camera calibration matrix $cof(\mathbf{K})$ and the camera pose represented by $\mathbf{R}$ and $\mathbf{t}$:

$$\begin{gathered}\ell \sim cof(\mathbf{K})\mathbf{R}[\mathbf{m} - \mathbf{t} \times \mathbf{l}],\ \text{with} \\ cof(\mathbf{K}) = \begin{bmatrix} f_y & 0 & 0 \\ 0 & f_x & 0 \\ -f_y x_0 & -f_x y_0 & f_x f_y \end{bmatrix}\end{gathered} \tag{3.9}$$

This concludes the formulas, necessary to perform an efficient optimization with 3D lines, avoiding over-parametrization and internal gauge freedoms. More details and their derivation can be found in [ZK14].

The optimization is similar as in the previous section, but additionally the Cayley lines and the poses from the previous frames, denoted in the set $\mathcal{K}$, are refined $\mathcal{X} = \{\theta_k, v_i\}$:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_{k \in \mathcal{K}} \sum_{i \in M_k} \rho_h(e_{s,i} + e_{e,i}) \tag{3.10}$$

The error function is similar to equation 3.4 except that here the Cayley line v has to be projected. Therefore it is transformed into Plücker coordinates via equation 3.8 and then projected using equation 3.9.

Now to retrieve endpoints for the infinite 3D lines, we project each of their 2D observations' endpoints onto them and determine the closest intersection point on the infinite 3D line. Then, we choose the median of all projected start point positions and of all projected end point positions as position for the new start and for the new end point respectively. These endpoints from the 3D lines are only used in our motion tracking, they are not required in the bundle adjustment.

A note on the choice of the error function: We tested multiple different error functions of which some directly included the angle between the lines or the cosine of it, the closest distance between the lines and or the length of the lines. The chosen function yielded the best accuracy while the results did not vary greatly. Also it is a simple but powerful function as it incorporates the angle indirectly, the distance between the lines and also the length of the line observation all by itself. Due to its nonlinearity, it automatically pulls the lines closer to each other even when the lines already overlap. Additionally, when increasing the length of the line and keeping the angular distance constant, and greater than zero, the error increases. This is good as longer lines contain more information (more pixels are involved) and are therefore more precise. It is an advantage to avoid the usage of endpoints of 3D line models in the optimization because the location information of lines is very precise orthogonal to the line direction. This is because lines can accurately be fitted on an edge. But the beginning and ending of a line are more prone to vary depending on the chosen line detector and on occlusions. Thus the locational precision along the line direction, and therefore the usage of endpoints, is not beneficial in the optimization. In contrast to bundle adjustment, where correspondences already exist, endpoints are beneficial in the tracking phase to find the correct correspondences, even when the endpoint location is not very precise along the line direction. Therefore we use a different error function in our tracking as described in section 3.3.3.

3.4 Results

All our experiments were carried out on a Notebook Computer equipped with 16 GB RAM and an i7 4700MQ, 2.4 GHz (3.4 GHz turbo), 6 MB cache fourth generation Intel-Core-i-Series Mobile Processor. We saved our trajectories in the TUM format[1] and used

[1] https://vision.in.tum.de/data/datasets/rgbd-dataset/file_formats

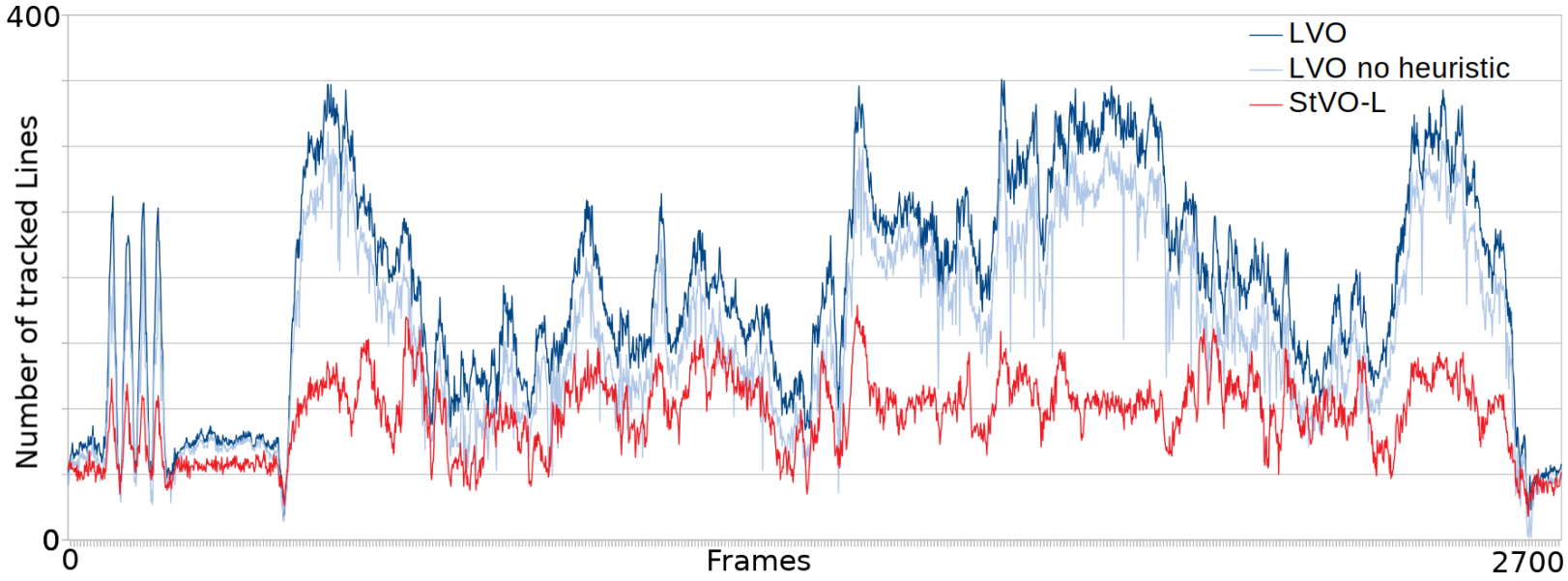


Figure 3.6: This figure shows the number of tracked lines in each individual frame over the MH-03 sequence of EuRoC for LVO in blue and for StVO-L in red. For LVO two numbers are given: the number before our heuristic filter matching is drawn in light blue and the number of lines after the heuristic is in dark blue. We can see two things: one, the heuristic constantly increases the number of tracked lines and, two, we are able to track much more lines than StVO-L, often even more than twice the amount of lines. [LRS19] ©2019 IEEE

the evo[2] tool to evaluate them.

3.4.1 EuRoC Dataset

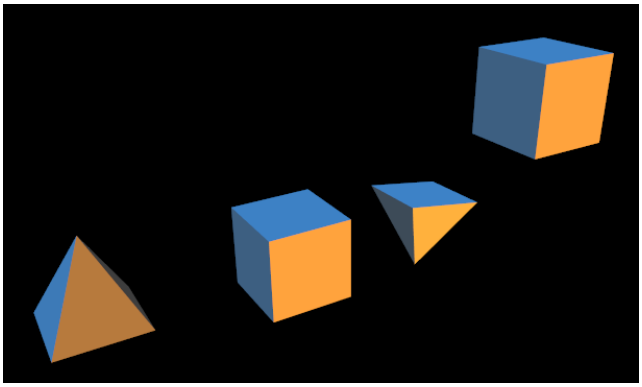


Figure 3.7: This figure shows the synthetic scene that we used to evaluate the state-of-the-art line based visual odometry systems. The scene is difficult because it is barely textured and only shows a few simple objects. But it provides enough structure on the objects for tracking. [LRS19] ©2019 IEEE

The EuRoC datasets [BNG*16] were originally recorded for the European Robotics Challenges[3]. They contain sequences which were recorded in a Machine Hall and in a Vicon Room. The images in the datasets were recorded with a stereo camera setup having a baseline of $11cm$ and at 752×480 resolution. Ground truth data is available for all of them. We neglected the sequence *V2-03-dif* as it has very fast motions with blurred images which were not manageable by any of the algorithms without a relocalization on an existing map which is not done in VO.

Table 3.1 compares our trajectories obtained on the EuRoC Dataset with the ones of current state-of-the-art approaches ([GOBGJ16],[GOGJ16],[GOZNM*17],[MAT17]). We compare the root mean squared errors (RMSE) of the relative poses, which reveals the drift since it compares locally. The relative pose error $\mathbf{E}$ over the aligned trajectories is defined as: $\mathbf{E}_i := (\mathbf{A}_i^{-1}\mathbf{A}_{i+1})^{-1}(\mathbf{B}_i^{-1}\mathbf{B}_{i+1})$, where $\mathbf{A},\mathbf{B} \in SE(3)$ are the corresponding poses from the two compared trajectories. It's split in two parts: the left side shows the algorithms which are able to work on lines only whereas the right side shows point and line algorithms. Our LVO is shown in both parts for comparison. StVO-L / StVO-PL denote the method in [GOGJ16] with lines only and with points and lines combined. The method [GOZNM*17] is similarly denoted with L-SLAM (lines only) and PL-SLAM (points and lines). PL-SVO only works with points and lines which is [GOBGJ16]. And ORB-SLAM2 [MAT17] only uses points. We see that our LVO leads to better results in

[2] https://github.com/MichaelGrupp/evo
[3] http://www.euroc-project.eu/

Table 3.1: Relative RMSE Errors in meters on the EuRoC MAV dataset[BNG*16]. A Dash indicates that the experiment failed.

Sequence	LVO	StVO-L	L-SLAM[4]	LVO	StVO-PL	PL-SVO	PL-SLAM[4]	ORB-SLAM2
MH-01-easy	**0.025819**	0.027521	0.036868	**0.025819**	0.026795	0.029605	0.029501	0.027925
MH-02-easy	**0.026433**	0.027623	0.028157	**0.026433**	0.027045	0.027079	0.027830	0.027806
MH-03-med	**0.062903**	0.070525	0.071057	**0.062903**	0.067128	0.080691	0.070371	0.071537
MH-04-dif	**0.067538**	0.070173	0.070694	0.067538	**0.066897**	-	0.069155	0.070160
MH-05-dif	**0.061425**	0.061746	0.062369	0.061425	**0.060157**	-	0.061497	0.061889
V1-01-easy	**0.021664**	0.025016	0.031586	**0.021664**	0.023799	0.044090	0.026117	0.025989
V1-02-med	**0.044657**	0.049645	0.051612	**0.044657**	0.049232	-	0.051587	0.053453
V1-03-dif	**0.039449**	0.046453	-	**0.039449**	0.042907	-	0.044849	0.050133
V2-01-easy	**0.016388**	0.040475	-	**0.016388**	0.019728	0.077382	0.019749	0.020788
V2-02-med	**0.035671**	0.039842	0.041524	**0.035671**	0.039950	-	0.040811	0.043187

Table 3.2: Average runtimes of particular parts of LVO.

	Machine Hall	Vicon Room
Total Time	78.76916 ms	51.52926 ms
Motion Tracking	0.976468 ms	0.740836 ms
Pose Estimation	5.696434 ms	4.491312 ms
Heuristic Filter	24.85896 ms	12.83857 ms
Frame Preparation	40.27929 ms	27.43780 ms
Triangulation	0.799423 ms	0.712026 ms
Local BA	196.1276 ms	169.189 ms

[LRS19] ©2019 IEEE

all sequences when compared to the other line only methods which demonstrates that its a great advancement in line only VO. In *V2-01-easy* for example it produces less than half the error than *StVO-L* does. We argue that this is because we keep and use more lines than StVO (see figure 3.6) due to our motion tracking (section 3.3.3) plus our heuristic filter matching (section 3.3.1). Even when compared to other methods using both, points and lines, it performs better in most sequences while only using lines.

Table 3.2 shows average runtimes of particular parts of LVO. We see that it runs faster in the Vicon Room. This is because that room provides a less complex scenery in which fewer line-segments are detected than in the Machine Hall. Motion tracking, which creates most of the matches, takes less than a millisecond for computation. The local bundle adjustment runs in a parallel thread.

3.4.2 Low Textured (Simulated) Dataset

We built an OpenGL[5] based simulation environment to create a synthetic scene with corresponding ground truth data. The scene just contains a few quad and pyramid objects that are coloured but not textured. This can be seen in figure 3.7. We designed it to test the algorithms under particularly difficult conditions. The lack of texture makes it hard for point feature based algorithms to find features. Structure is available but it is rare as the scene does not contain many objects. We again tested the methods from the previous section on the two sequences.

In table 3.3 we can see that LVO manages both sequences, while all others fail in at least one sequence. An experiment is marked as failed if the method does not manage to

[4]We changed the standard configuration of the system to maximize the accuracy to achieve these results. Thereby the runtime increased tenfold.

[5]https://www.opengl.org/

Table 3.3: Synthetic dataset results. A Dash indicates that the experiment failed.

Sequence	LVO	StVO-PL	PL-SVO	PL-SLAM	ORB-SLAM2
Scene1-short	✓	✓	-	✓	-
Scene2-long	✓	-	✓	-	-

keep track of the features smoothly or if it does not manage to initialize at all.

One particular reason why the other tested methods are less reliable on these scenes is that their feature tracking mainly relies on visual descriptor techniques. Visual feature descriptors are computed from the variation in pixel intensities within image patches. But the low amount of texture in the artificial scenes do generally not allow the computation of distinctive descriptors. Tracking line segments instead of points enables us to match features relying solely on the scenes geometry, without the need of descriptors (see section 3.3.3). Additionaly we apply our heuristic matching filter on the remaining untracked lines to gather further line matches, where the LBD is only utilized as a prior (see section 3.3.1).

3.5 Summary

In this chapter we have proposed LVO, a visual odometry algorithm that uses lines only. We have shown that by using line features over points, visual odometry becomes more reliable in situations where texture is rare but structure is still available. This is often the case in man-made, indoor environments. Using line segments over points furthermore empowers feature matching without the need of feature descriptors. Instead it can be based solely on geometric constraints.

We proposed a matching filter that improves our matching results by taking into account the local neighborhood of the features. This is a novel approach to increase the robustness of matches and an alternative to the often used practice of mutually best matches. It is also easily applicable in other situations since it does not require a distance measure, but just a general matching estimation as prior. We have also implemented an optimization which uses a minimal four parameter representation for 3D lines avoiding overparametrization, which has not been used for line based VO/VSLAM methods yet and showed good results. We evaluated our system on the EuRoC [BNG*16] benchmark which showed that our algorithm is an improvement in indoor environments w.r.t. competing state of the art methods and especially that it is a good advancement in terms of approaches which use lines only.

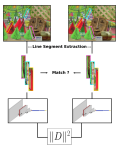
In this approach almost all lines were matched by tracking and our heuristic filter. It is still a difficult task to calculate an accurate and distinctive descriptor for lines. Therefore, we chose to address this problem in the next chapter. Instead of approaching this challenge with an analytical solution, as others have done before [WLW09] [WWH09]

[LWD10] [ZK13], we chose to address it using machine learning techniques. This way, we do not have to try many different ways on how to compose the descriptor, because the network itself figures out which areas around the line are important and how it should weight them for the best result.

Chapter 4

A Deep Learning Based Line Descriptor for Line Feature Matching

This chapter presents our *Deep Learning Based Line Descriptor for Line Feature Matching*. The chapter is mostly cited from the paper:



4.1 Related Work

Many computer vision related problems are based on, 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. There are various detectors and descriptors for point features [Low04] [BTVG06] [RRKB11]. Learning based point descriptors have shown an improved performance compared to the established analytic descriptors [FDB14][SSTF*15][SVZ14]. 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 their edged and features, such as doors and windows, yield structural information.

Line features came into focus just a couple of years ago. They require more processing power to be extracted than most point features do. But they made it possible to improve the robustness and accuracy of solutions to computer vision challenges mentioned earlier. Line features are well suited to represent structure [HMB14]. They helped to 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].

There are multiple line detection algorithms by now. A method for line detection that was published in the early seventies is based on the Hough transform [DH72]. 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 constructed a descriptor out of multiple line segments. Their intention was 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 computationally expensive. The calculation of their descriptors takes seven seconds and matching another six seconds (at 700×500 image resolution).

We will focus on appearance based line descriptors from here on. Early line descriptors placed point descriptors along the line [LWD10], or used descriptors which are similar to histogram based point descriptors along and around the line [WLW09] [HS12]. Clearly, this does not adapt well to the characteristics of lines. The mean standard-deviation line descriptor (MSLD) proposed by Wang et al. [WWH09] is one of the most compared to, and cited appearance based line descriptors. Lilian Zhang evaluated the MSLD and his Line Band Descriptor (LBD)[ZK13] in his dissertation [Zha13] showing that the LBD performs faster with more correct and less false matches. Inspired by the ORB point descriptor Zhuang et al. developed *A Binary Robust Line Descriptor*[ZZPL16]. They achieved to be faster than the LBD. Yet, they were not as accurate. The LBD is the only line descriptor that made it into OpenCV [Bra00]. For these reasons, we chose the LBD as one of the works to compare to. We will go more into detail about it in our results in sections 4.4 and 5.3. A drawback of descriptor computing methods like the MSLD and the LBD is that they lose much of the structural information around the line by taking (weighted) means along the line. Due to recent years findings, we argue that this is a task for which deep neural networks are well suited. We also give an insight

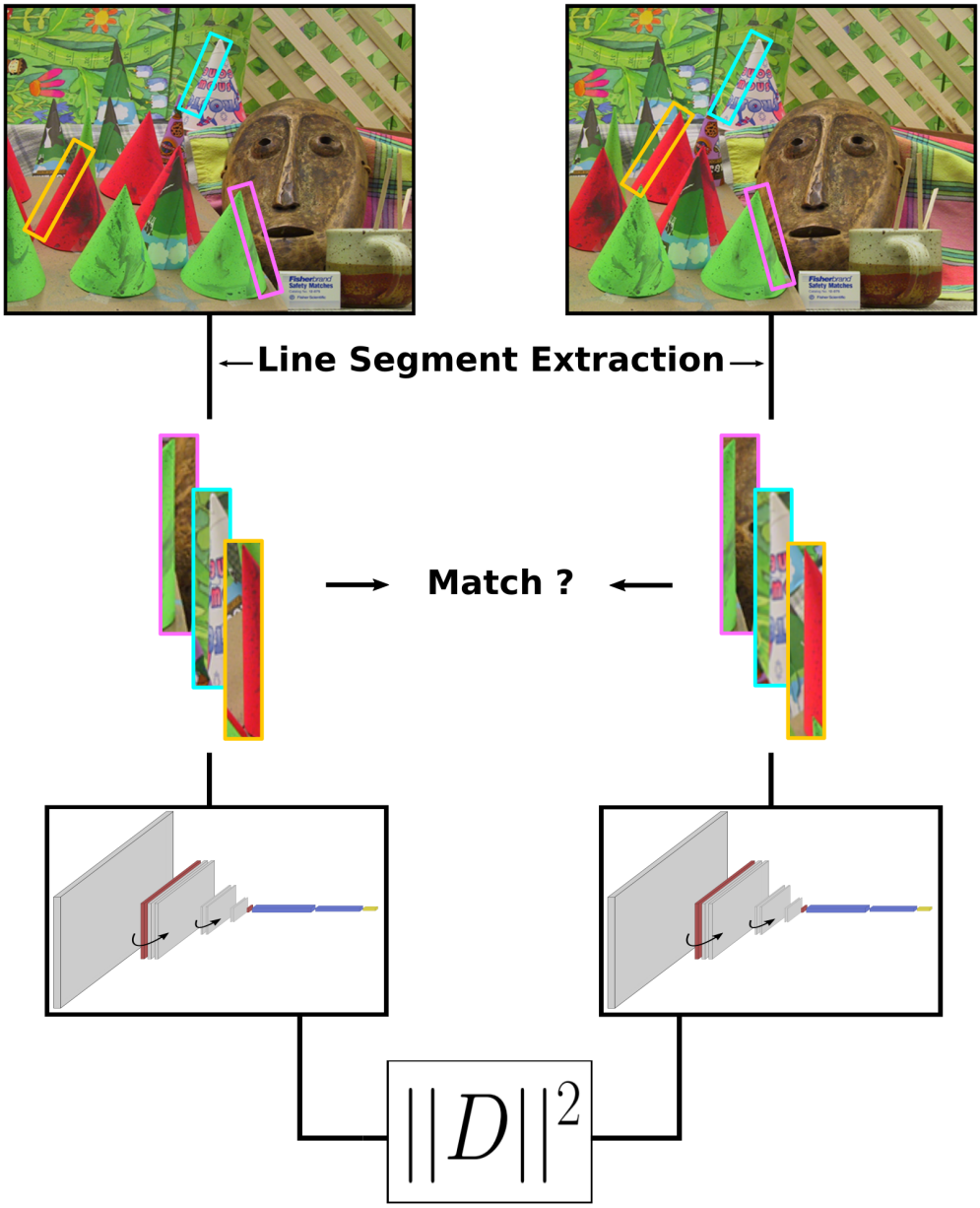


Figure 4.1: Basic line segment matching procedure. First, line segments are detected on the input images. Then, descriptors are computed by our DLD network. Finally, these descriptors can be matched by calculating the squared euclidean distance. [LSS19] ©2019 IEEE

into details of the Learnable Line Segment Descriptor for Visual SLAM by Vakhitov and Lempitsky [VL19], which is a recent learning based method that we included in the benchmarks in the next chapter 5.3.

Patch matching can also be noted here; it has been researched thoroughly [ZK15] [TFW17] [ZL16] [HLJ*15] [ZK17]. A survey of image matching is given in [LZL*19]. One might think that appearance based line matching is similar to patch matching. But in fact, lines are a certain type of feature with special characteristics. A line can lay on the edge of an object with the object itself being on one side of the line and something else being on the other side of the line. In this specific case it's important that the latter lies in the background. Since this line, depending on the perspective, separates the object and the background, we will call it an edge line. Such an edge line stays the same line even if the side showing the background changes. Additionally, areas closer to the line can be more important than those further away. One should also keep in mind that a line extends in one direction which differs for regular patches. Those special characteristics distinguish lines from patches. Furthermore, most patch matching methods use the whole image patch, whereas we reduce the line to a small 256-Bit descriptor, similarly as it is done for the LBD.

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

We chose to train a ResNet [HZRS16], a well trainable standard architecture, on our matching problem. Regular ResNet comes in different depths, starting with size 18 upto size 152. After some tests, we found that the smallest ResNet with size 18 works best. As this is the smallest size, we wanted to know if an even smaller net would also work. Thus, we created a ResNet with size 10. We achieved a result which is as accurate as the size 18 ResNet, but faster due to the reduced size. We also tried a DenseNet [HLVDMW17], but did not achieve a similarly good performance using it.

Our error function uses a triplet loss which was originally developed by Weinberg and Saul [WS09] and got popular through Schroff et al. [SKP15]. Each triplet consists of three lines, the first line is the one which the other two lines are compared to by descriptor distance. The net trains to calculate the descriptor in a way that minimizes the distance between the matching pair and maximises the distance of the first line's descriptor to the false line's descriptor. Our main contributions are as follows:

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

An overview of the matching procedure is drawn in figure 4.1. Section 4.2 gives a detailed explanation of our data generation process. In Section 4.3 we describe how the learning was conducted and in 4.4 the results are presented. In Section 4.5 we draw a conclusion of our proposed method.

4.2 Data Generation

Machine Learning in general requires a lot of data to learn from. A difficulty when training a network for line matching is the lack of suitable datasets. Established machine learning datasets for computer vision as the Cityscapes Dataset [COR*16] and the KITTI Vision Benchmark Suite[GLSU13] do not come with the necessary information. Regular images are not enough, as the correspondences between lines have to be known. Thus, we chose to create our own data with the necessary information utilizing the Unreal Engine.

4.2.1 Unreal Engine 4 - Ground truth data generation

The Unreal Engine 4 (UE) comes with a realistic rendering engine and many freely available sceneries. From those scenes, we chose four quite different ones:
- The *Epic Zen Garden (EZG)* shows an outdoor modern garden area including some trees, a pool and multiple structured things like outdoor furniture, tiles and the house.
- The *Infinity Blade: Grass Lands (IB:GL)* is a very rocky wide open scenery including a path mostly made of stone with the see beneath it.
- The *Scifi Hallway (ScHw)* shows a science fiction hallway with many details like pipes and other tech things on the walls and standing around.
- The *living room (LvR)* from the unreal engine showcase realistic rendering example shows a basic living room with a table, a couch, vases, a plant and pictures as well as a TV on the wall.

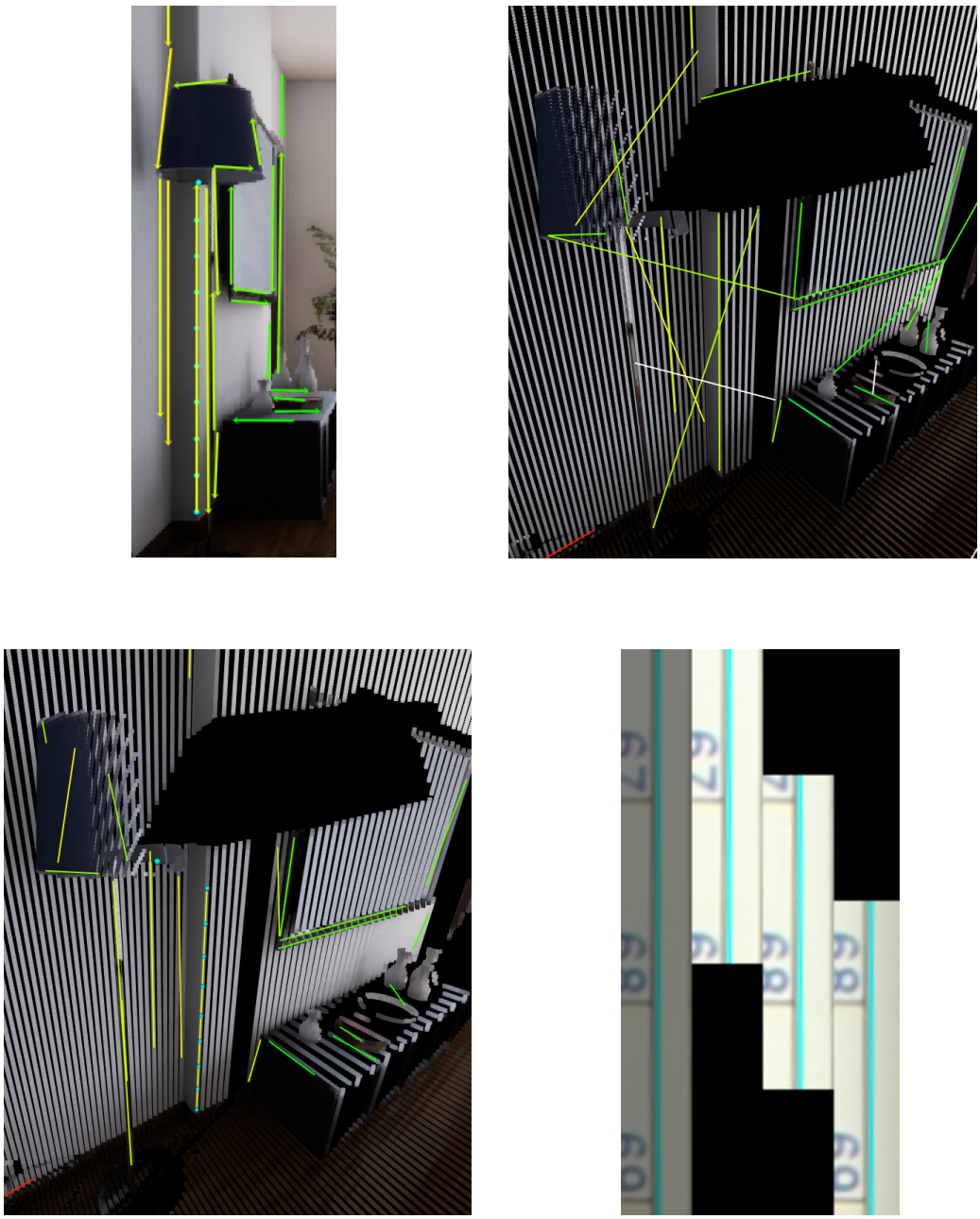


Figure 4.2: The top left image shows the camera view from which it is not possible to see if the lines' depth is correct. The top right and bottom left images show the 3D scene from the side. One can see that the 3D line segments are quite skewed in the top right image. The bottom left image shows the same scene but with the 3D line segments corrected by the RANSAC scheme. The bottom right image shows a visual illustration of a line segment which is split in three parts (cutouts). [LSS19] ©2019 IEEE

We chose those diverse scenes to provide the learning process with a wide variety of different examples. To extract the images and the corresponding 3D-Coordinate-Maps, we defined long splines through the scenes as a path for the camera. Following this path we stepped slowly through the scenes recording data at each step. In total 37689 steps were recorded (12068 *(EZG)*, 17875 *(IB:GL)*, 5328 *(ScHw)*, 2418 *(LvR)*) with about 11.9 Million lines overall.

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

First, we start by detecting 2D line segments with the EDLine detector. Then, we use the extracted 3D-Map to unproject the detected 2D line segments to 3D line segments in the observed scene. Having this 3D information, we project the 3D line segments to 2D line segments into the second stereo image in which they are matched. There, the matching is done in 2D. To be a valid match, two lines have to have a similar angle, have to overlap in their length sideways (in respect to their direction) and have to be close to each other.

Unfortunately, when unprojecting a 2D line segment the resulting 3D line segment is occasionally inaccurate. The problem here is that the 2D segments from the line detection are not always exact. When the segment is an edge line (see Section 4.1), it happens that part of it reaches over the edge. In this case the underlying 3D coordinates are wrong as they come from the background. This results in an inaccurate 3D segment sometimes pointing deep into the scene. To deal with such problems, we employed a RANSAC [FB81] scheme: We sample ten points evenly distributed on the 2D line segment. Of those points, we get the 3D coordinates from the map to obtain 3D points. Then pairs of two are selected to calculate an infinite 3D line model and the other points are used to validate the line model. If enough of the other points are close, then the model is accepted. The distance of the other points is also used as a measure of quality of the model. Using this scheme improves the 3D lines and, thereby, the quality of the line matches. The improvement is depicted in figure 4.2.

4.3 Learning

ResNet has demonstrated a great performance on various benchmarks [HZRS16]. It also solved the problem of deeper networks performing worse than their smaller counterparts by introducing so called *identity shortcut connections*. For these reasons, we chose this architecture for our line matching problem.

During our first experiments with ResNets in different sizes, we quickly discovered that the network does not have to be as deep as 18-layers to solve our problem. The smallest ResNet described by Kaiming He et al. is an 18-layer version. We created a

10-layer version by halving the number of the 3×3 convolutions in the middle of the network which still performed similarly well.

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

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

$$\#\text{cutouts} = \begin{cases} 1 & \text{if } \frac{l}{t} \leq 1.2 \\ 3 & \text{if } \frac{l}{t} \geq 1.4 \\ 2 & \text{else} \end{cases} \tag{4.1}$$

where l is the length of the line segment and t is the desired fixed length (in most of our experiments, we set t to $120\,px$). One has to keep in mind that the matching is appearance based. Thus we made sure, that even with three cutouts the two which are furthest apart from each other still have a slight overlap, as it can be seen in the bottom right image of figure 4.2. The cutouts have labels which assign them to the line segment which they originate from. This information is used to recover which lines are matched to each other.

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

To avoid that the network specializes on one scene, it is important that the examples

Figure 4.3: Visual illustration of a triplet matching sample. [LSS19] ©2019 IEEE

are provided randomly during the training phase.

4.3.1 Triplet loss

We chose to use the triplet loss for our descriptor learning, as this technique yielded good results for *Nearest-Neighbour-Classification* [WS09][SKP15]. The idea using the triplet loss on line segments is as follows: For a given line segment $s_i^a \in \mathbb{S}$ two line segments s_i^+ (positive) and s_i^- (negative) are selected. For their labels $l \in \mathbb{L}$, that hold the information which lines match, applies $l_i^a = l_i^+$ and $l_i^a \neq l_i^-$, which means that one segment is a match and the other is not (see figure 4.3). When calculating the descriptors (here called *embedding*) for each segment, using the function $e(s)$, their descriptor distance is given by the squared euclidean distance. Additionally we enforce a margin δ in the descriptor distance between matching segments and non-matching ones which leads us to:

$$dist\big(e(s_i^a),\ e(s_i^+)\big) + \delta < dist\big(e(s_i^a),\ e(s_i^-)\big) \tag{4.2}$$

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

Many triplets will fulfil equation 4.2 and not encourage a fast convergence. Therefore it is beneficial to select triplets that are difficult, i.e. those which violate equation 4.2. Thus, we use the margin δ to make sure that we get difficult triplets, which are still representative. Therefore the loss function L to be minimized can be written as:

$$\begin{aligned} L = max([&dist\big(e(s_i^a),\ e(s_i^+)\big) \\ &- dist\big(e(s_i^a),\ e(s_i^-)\big) + \delta],\ 0) \end{aligned} \tag{4.3}$$

Triplets can be generated online or offline. Generating them offline, e.g. based on the most recent network checkpoint would have the advantage to select from a greater

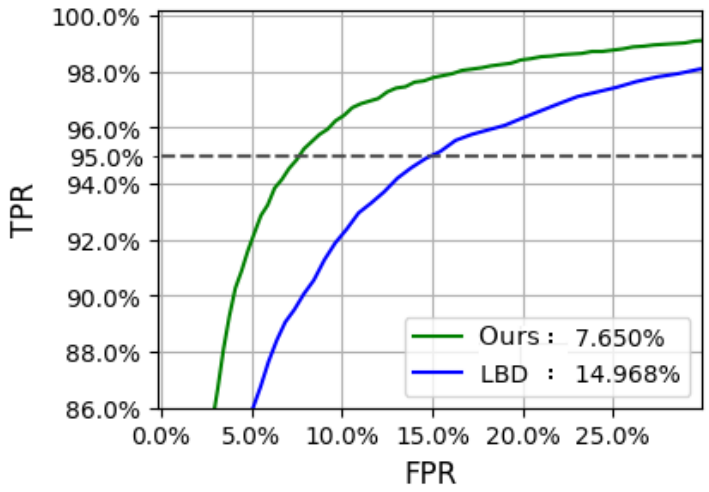


Figure 4.4: This plot shows the receiver operating characteristic curves (ROC-Curve) of our method vs the LBD over all of our eight labelled Middlebury Images [SHK*14]. The legend shows the False Positive Rate (FPR) at the True Positive Rate (TPR) of 95% (dashed line). An advancement of our method is clearly visible. [LSS19] ©2019 IEEE

amount of segments. But it also has the disadvantage that the triplets are not always based on the most recent weights in the network. Also Schroff et al. [SKP15] found it inconclusive whether doing it offline is beneficial. We chose the online generation which yields triplets based on the current mini-batch using the most recent weights.

4.4 Experimental Results

In this section, we evaluate the performance of our descriptor by comparison to the state-of-the-art LBD [ZK13] descriptor. To validate that our descriptor yields good results with real world data, even while trained on synthetic data, we included some real world images from the Middlebury Stereo Dataset [SHK*14] and some self shot images. Our results on the Middlebury images are discussed in the next paragraph and are shown in figure 4.4 and in table 4.1. Furthermore we wanted to evaluate our method with arbitrary camera poses and under difficult conditions. Therefore we took more images, for which the results are shown in figure 4.5.

The EDLine detector [AT11] was used for the line detection with its standard parameter settings. For a fair comparison, the size of the segments was the same for our method as for the LBD. During training we used stochastic gradient descent as optimizer and a δ margin value of five in the triplet loss. The training took approximately one day on a GeForce GTX 1080. During testing, the calculation of one descriptor on the CPU (Core i7 3.4 GHz) takes about one millisecond.

Table 4.1: Performance evaluation on images of the Middlebury Stereo Dataset (2014) [SHK*14]

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

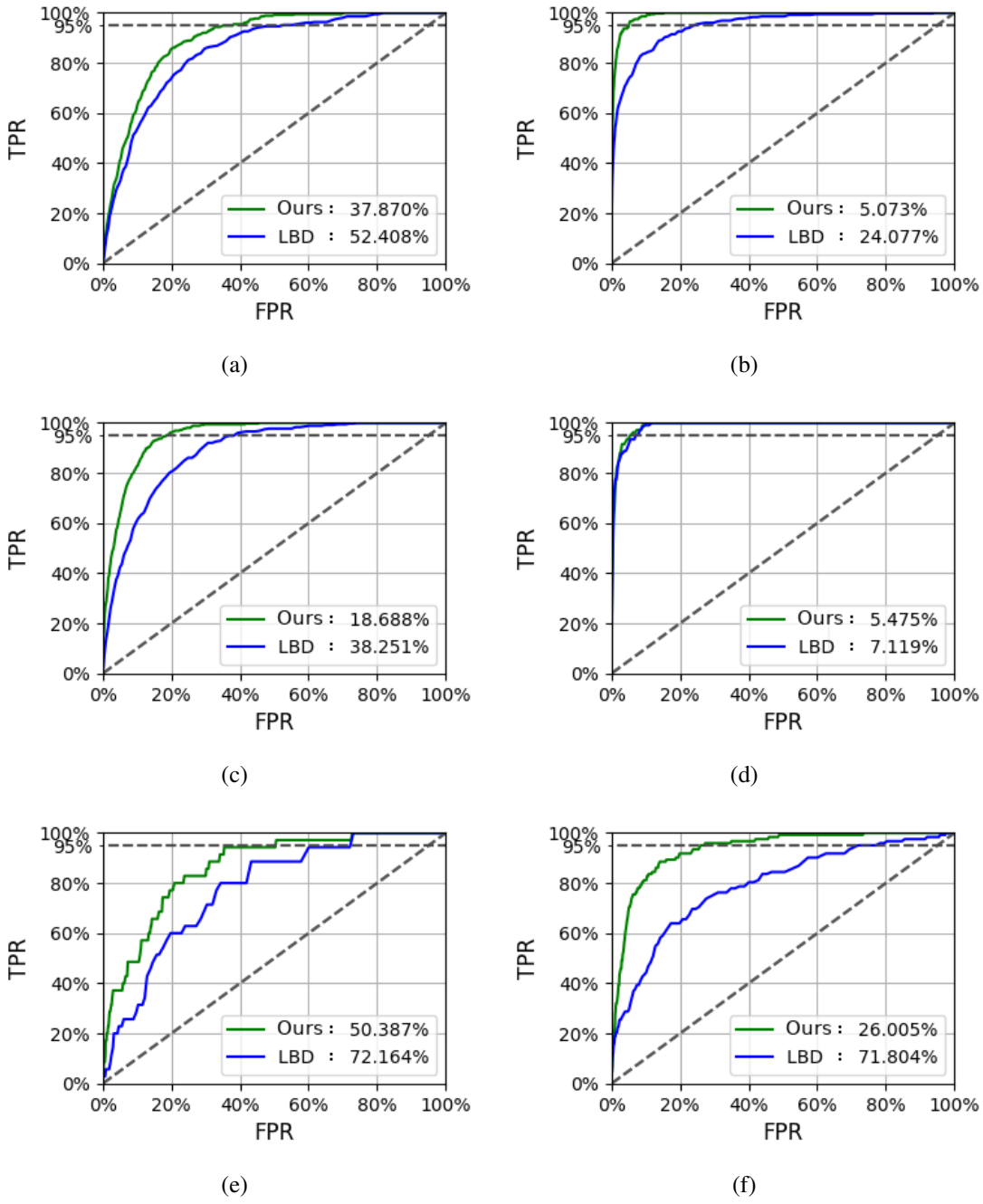


Figure 4.5: The plots (a) to (f) show the ROC-Curves for the following difficult conditions: (a) strong blurring (351 Ground Truth (GT) matches), (b) high jpeg compression (351 GT matches), (c) intense noise (351 GT matches), (d) about 45° rotation (106 GT matches), (e) scale change (35 GT matches) and (f) viewpoint change (122 GT matches). TPR is the true positive rate and FPR is the false positive rate. The greater the area under the curve, the better. [LSS19] ©2019 IEEE

Middlebury evaluation

We took the stereo image pairs from eight scenes of the Middlebury Dataset, ran the line segment detection and labelled the line segments to obtain ground truth matches. The receiver operating characteristic curves (ROC-Curves) including all matches from all eight images are plotted in figure 4.4. We can see that our method is a great advancement in comparison to the LBD. We did not plot the ROC curves to all eight labelled scenes, as they are all similar to the overall ROC curves. But the percentage of false matches at 95% of true matches is listed in table 4.1. This table also lists the number of matches per scene. We can also see here, that our method does better in all scenes.

Difficult conditions scenes

We hand labelled some real world images showing difficult situations for further evaluation of our method compared to the LBD. The results are depicted in multiple ROC curves in figure 4.5. All of the subplots contain a legend which shows the methods false positive rate (FPR) at the true positive rate (TPR) of 95% (less is better). The first plot, figure 4.5(a), demonstrates the performance when strong blur is on one of the two images. We can see that it is difficult for both methods, but ours can handle it better. In figure 4.5(b) a high jpeg compression rate is chosen for one image which is not very difficult for our method, but affects the LBD to a considerable amount. Intense noise is added to the image related to in figure 4.5(c) which impacts both methods negatively, but our method is affected only about half as much as the LBD. We rotated one image about 45° which barely has an effect on both methods, see figure 4.5(d). That was expected, as both methods use a region along the line, making them independent to global rotations. Figure 4.5(e) shows the ROC curves when a strong scale change took place. Both methods are quite affected. We did not augment our training data especially for scale change, still our method is somewhat less affected. The results to a viewpoint change of over a couple of meters and simultaneously more than a 45° change in the viewing direction are plotted in figure 4.5(f). We can see that our approach performs more than twice as well as the LBD. This is an important finding, as wide baseline matching is a difficult yet important task for applications like 3D reconstruction from arbitrary poses or relocalization for a lost robot.

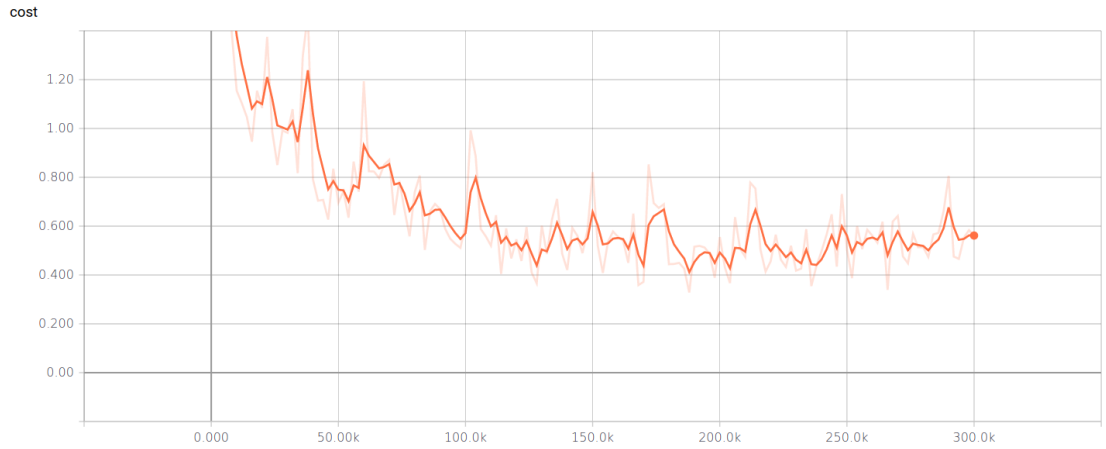


Figure 4.6: This figure shows the error curve of the training error. The x-axis shows the number of training steps and the y-axis shows the training error.

4.5 Summary

This chapter presents a new deep learning based line segment matching approach. We elaborated our ground truth data generation using the Unreal Engine 4 and multiple different scenarios thereof to obtain a wide variety of samples. Using these samples, we trained a ResNet and explained how to use the triplet loss to train line segment descriptors for descriptor matching.

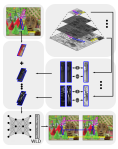
We were able to show our method is an advancement compared to the state-of-the-art LBD descriptor. This was shown on our artificial datasets as well as on images from the Middlebury Dataset and on additional scenes with difficult conditions showing blurring, jpeg compression, noise, rotation, scale change and viewpoint change.

In the next chapter we build upon this descriptor to create an improved version which utilizes image pyramids to take more of the surrounding into account and which uses wavelets as a preprocessing step.

Chapter 5

A Wavelet and Learning based Line Descriptor for Line Feature Matching

This chapter presents our *Wavelet and Learning based Line Descriptor for Line Feature Matching*. The chapter is mostly cited from the paper:



WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching, Vision, Modeling, and Visualization 2020, 10.2312/vmv.20201186

We gave an overview about line related work earlier in section 4.1, for this reason we will put an emphasis on wavelets here.

5.1 Wavelet related work

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

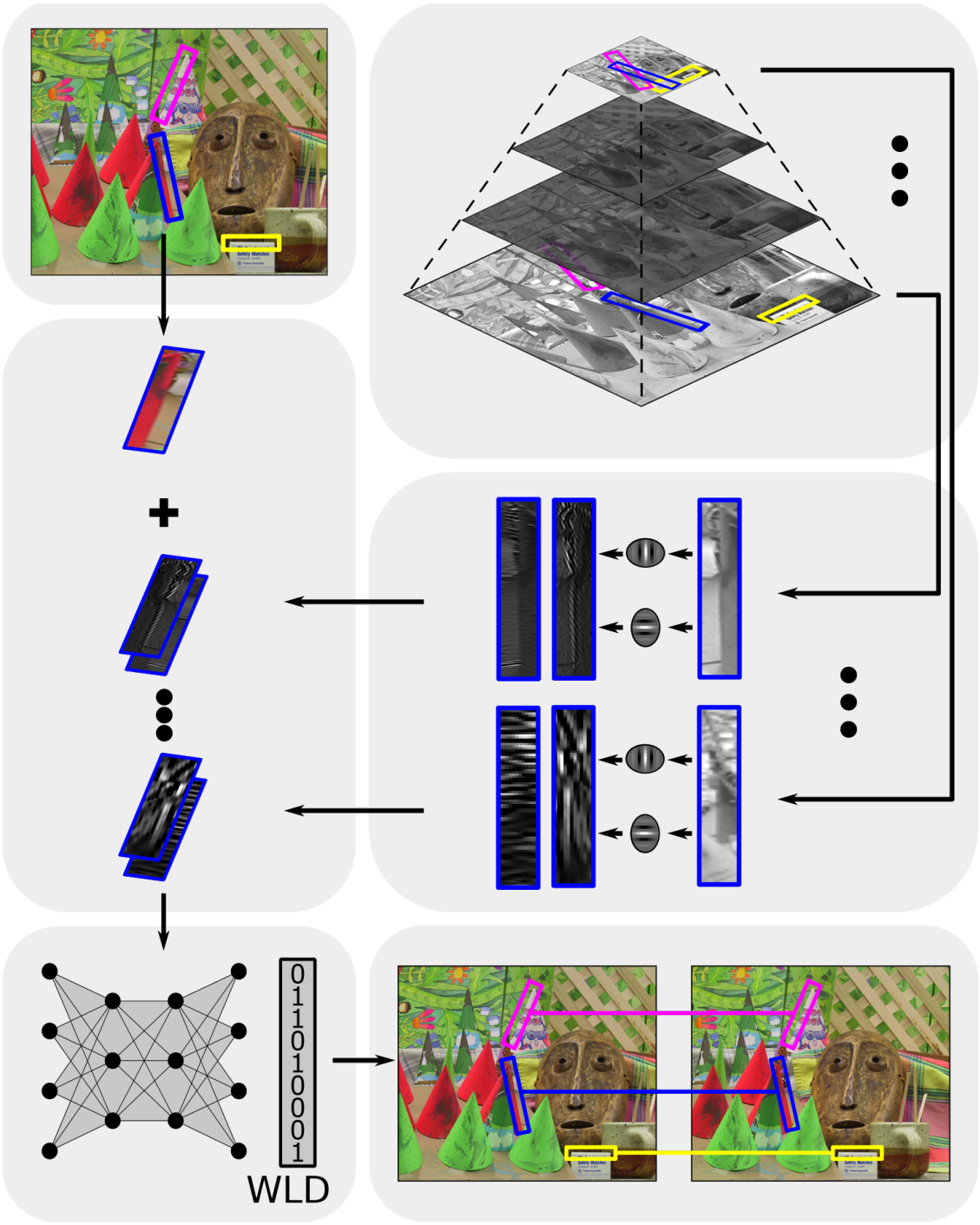


Figure 5.1: Top left shows the original image and top right shows the multi-scale pyramid of it, both with some line segments highlighted for visualization. Middle right depicts the wavelet convolution. The results of all layers are stacked together with the original rgb cutout, as shown in the middle 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).

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 biologically motivated, as mentioned before. An overview of the descriptor creation is given in figure 5.1.
The rest of this chapter is structured as follows: The next Section 5.2 explains our approach in detail. After that, we present the results including comparisons to other methods in Section 5.3. Finally, Section 5.4 concludes this chapter.

5.2 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.

5.2.1 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 [Epi20], 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 labelled 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 colour 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.

5.2.2 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 5.1). In order to increase the receptive field to include a wider area around the line, we use an image pyramid. The pyramid has five levels and the width and height are reduced by a factor of two 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 colour channels for the cutout. We will not consider the colour 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:

$$\psi(x,y;\sigma,\theta,\lambda,\phi,\gamma) = e^{-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}} \cdot e^{i(2\pi\frac{x'}{\lambda}+\phi)}, \tag{5.1}$$

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:

$$\begin{aligned} x' &= x\cos\theta + y\sin\theta, \\ y' &= -x\sin\theta + y\cos\theta. \end{aligned} \tag{5.2}$$

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 colour 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 5.1, 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 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 5.1: Network input - Layers of the cutout stack

	Layer Description	Level	Size	Receptive field
Colour	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

5.2.3 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 shortcut 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

Table 5.2: Layers of the neural network.

layer name	output size	layers
conv1	50×14	RGB $[3 \times 3,\ 64] \mid [3 \times 3,\ 64]$ GW
pool1	50×14	3×3 max pool, stride 2
conv2	25×7	$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}$
conv3	13×4	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}$
conv4	7×2	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}$
conv5	7×2	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}$
	16 Byte	average pool, 1000-d fc, linear

for our problem. An overview of our architecture is given in 5.2. It starts with a 3×3 convolution on the input consisting of the RGB and Gabor wavelet patches. This layer reduces the output size of the tensor. A 3×3 max pooling layer with stride two follows. Then, in the middle of the network, multiple 3×3 convolution layers reduce the output size further, but increase the depth of the tensor. The shortcut connections skipping those convolutional layers while performing an identity mapping are not shown in the table. At the end of the network is an average pooling layer followed by a fully connected layer which computes the 16 Byte descriptor.

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:

$$\mathcal{L} = max\left[\left(dist(e_i^a, e_i^+) - dist(e_i^a, e_i^-) + \delta\right), 0\right], \tag{5.3}$$

where e_i^a is the anchor embedding, e_i^+ is the positive match and e_i^+ the negative match

Table 5.3: 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

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.

5.3 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. 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 figure 5.1. For each line segment the layers from the multi-scale pyramid are convolved with each of the two kernels, as listed in table 5.1 and stacked on the cutout stack. Table 5.3 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.

Setup: We conducted our experiments on a Core i7-4700MQ mobile CPU. The pre-

processing 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 5.2.2. 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.

GLM: In Robust Line Segments Matching via Graph Convolution Networks [MJL20] QuanMeng Ma, Guang Jiang and DianZhi Lai used an end-to-end training approach for line segment matching. This approach does not compute descriptors. Instead it requires both images at once, of which the line segments shall be matched, and returns correspondences. The authors trained a network which consists of three main modules. The first module learns to detect and extract line features and to directly exclude lines of both images, which it expects to be without a match. In the second module discriminative node embedding features are generated using a graph convolution network which features intra- and cross-graph convolution and graph architecture learning. The third module consists of an optimal transport layer which computes an affinity matrix between two line sets followed by an assignment matrix to find matches and reject line segments without a match. They did not compare their method to recent descriptor based approaches. To evaluate this method, we inserted the lines of our ground truth scenes in the input of the second module, skipping their detection module. In our evaluation, their approach is able to outperform the analytical LBD in some of the scenes, but it performs inferior when compared to the learning based methods.

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 DLD, which we published in 2019 [LSS19], crops out a rectangular area

which is centered around the line, like the LBD. 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 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 descriptors for all image pixels at once (while sharing required calculations) 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, when calculating a descriptor for the entire image at once, the directional information of the lines is likely to be lost by this approach. Lines have a distinct direction and also a left and a 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.

5.3.1 Performance on the Middlebury Stereo Dataset

We selected eight images of 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 5.2 shows the receiver operating characteristic curves of our WLD compared to our DLD and to the LBD. These resulting curves show the characteristic of the methods benchmarked on the eight labelled Middlebury Images [SHK*14]. The closer a curve is to the left and to the top, the better the performance of the method. An advancement of

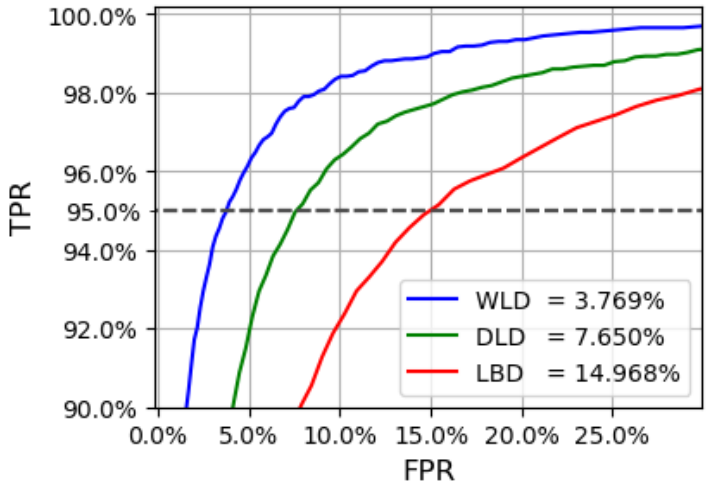


Figure 5.2: This plot shows the receiver operating characteristic curves of our WLD, vs our DLD, vs the LBD over all of the eight labelled Middlebury Scenes [SHK*14]. The legend shows the False Positive Rate (FPR) at the True Positive Rate (TPR) of 95% (dashed line). ©[LRS20] Eurographics Proceedings ©2020 The Eurographics Association

our WLD is clearly visible.
For methods such as 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 5.4. 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 labelled ground truth matches for each scene. The highest number of correct matches for each scene is shown in bold numbers. We can see that the LBD is outperformed in every scene by the WLD, the LLD and the DLD while it can do better than the GLM in all except the *Flowers* scene. The *Flowers* scene is the only scene where the GLM does not score the least number of matches, but it is able to do second best after the WLD. Our 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 WLD 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 method on the *Adirondack* and the *Couch* scenes, we achieve to match over 10% more of the possible matches on each of the other six scenes, relative to the total number of labelled matches. The WLD even matches more than twice as many lines than the LBD on the *Backpack* and the *Flowers* scenes.

Table 5.4: 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	**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
GLM	195	129	238	49	145	94	155	181
Labelled Matches:	318	1216	944	500	296	143	353	517

Table 5.5: 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	**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
GLM	163	163	155	7	42	13	92	32
Labelled Matches:	351	351	351	35	122	106	219	180

5.3.2 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 5.5. The *Blurring*, *Noise* and *Jpeg Compression* scenarios show the same corridor scene which was augmented using Gimp [The20] 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 the WLD, LLD and DLD on all three scenes while being ahead of the GLM in one scene. 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 on all three augmented images. This is probably due to the wavelets which are used in multiple scales. The higher scales will be altered by fewer extent from the augmentation effects as they are 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 beneficial here, but the distance is not as large as it is to the other methods. Our WLD matches all lines correctly in the *Rotation* scenario with our DLD being close. The LBD gets approximately four out of five lines correct. These three approaches make use of the directional information of lines, while at the same time 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 GLM does also not make use of the directional information of lines and is only able to match a few lines correctly. 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. In order to perform well in such scenes, it is necessary to make use of the structural information of the farther surrounding. Here we can see that our WLD approach matches all lines correctly. But the other approaches, besides the LBD and the GLM, 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 such as 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 labelled 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. The GLM performs worst of all approaches in scenes containing repeating patterns. We would have expected it to perform better in these scenes as such similar structures should lead to similar graphs. 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.

5.4 Summary

In this chapter 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 our 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 theoretically could learn this on their own, they did not achieve similarly good results in our tests. In this work Gabor wavelets act like pre-trained filters, which favour convergence and generalization. The next chapter presents an extended evaluation of our training procedure and a practical usage analysis.

Chapter 6

Extended evaluation

6.1 Training evaluation

In this chapter we provide an overview of the most significant trainings that we conducted. We experimented with different settings for the learning rate, the batch normalization, the descriptor size, the pyramid usage and the Gabor wavelets. For each configuration we show the plot of the learning error and of the validation error. But while the learning error is the cost from the triplet loss, the shown validation error is actually the false positive rate at a true positive rate of 95% as shown in ROC curves. Using these values quickly gave us an impression of the current performance. Our validation set consists of the eight scenes of the Middlebury Stereo Dataset (see paragraph 5.3.1) and of the eight scenes of our difficult conditions dataset (see table 5.3.2). Using these scenes from real world scenarios directly yields realistic values. This is preferable to obtaining results calculated on simulated data such as, for example, the training data.

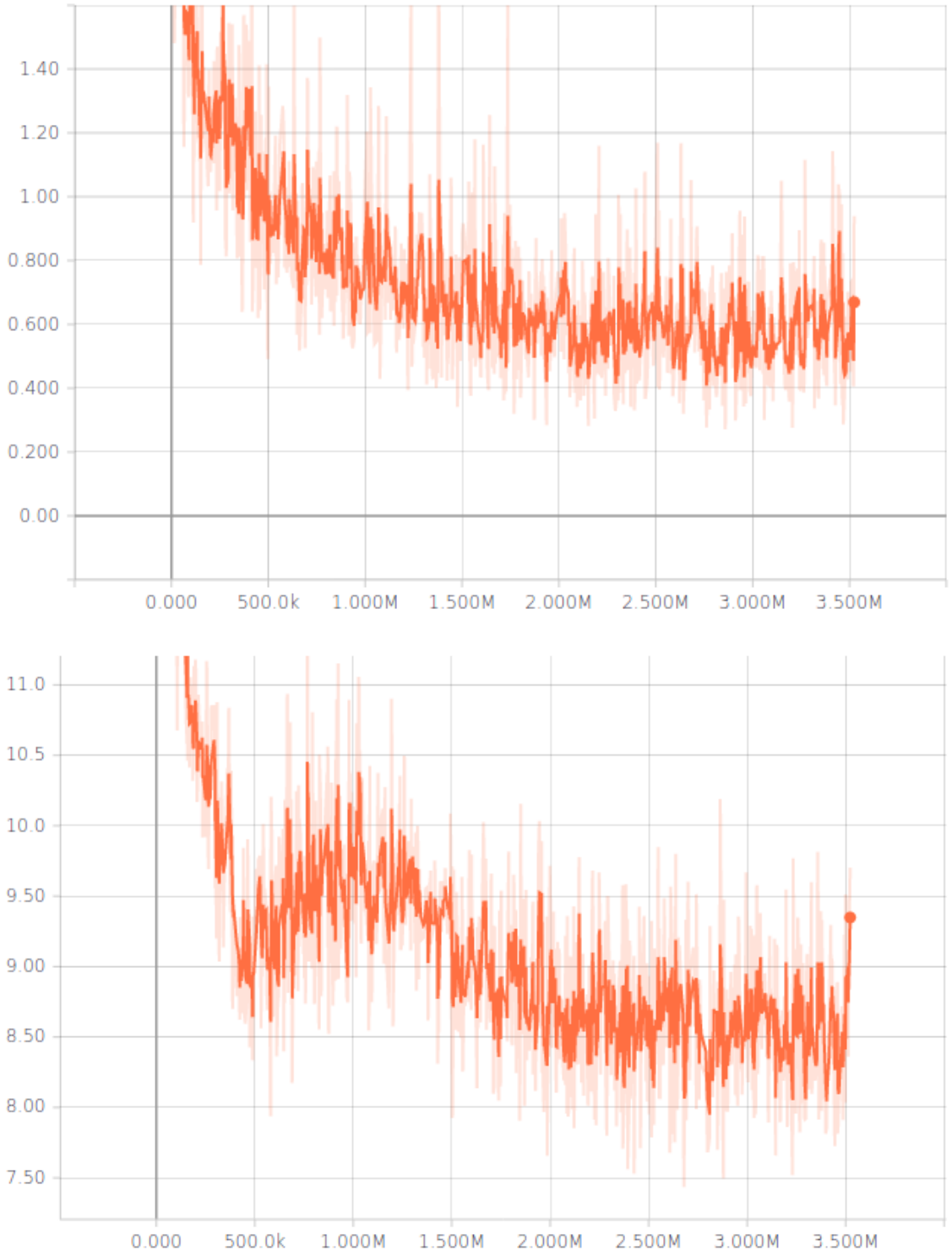


Figure 6.1: These plots visualize the training error curve (top plot) and the validation error curve (bottom plot) with the batch normalization turned on, an eight byte descriptor output and a constant learning rate of $1 \cdot 10^{-5}$. The image pyramid and the Gabor wavelets are utilized. The vertical axis represents the error value and the horizontal axis represents the training step.

6.1.1 Training using an eight byte descriptor, batch normalization, an image pyramid and Gabor wavelets

While it was our goal to beat the LBD descriptor using the same patch size and descriptor size in chapter 3, in this chapter we wanted to find out what is possible when we do not limit our descriptor using the same boundaries as the LBD. We cropped out larger patches and then we combined them with an image pyramid and Gabor wavelets to deal with the increased amount of data. Figure 6.1 shows the training error curve in the top plot and the validation error curve in the bottom plot. The training error curve performs a classical descent that is initially very steep and then flattens out and goes into saturation. The validation curve, on the other hand, starts with a steep descend. But then it ascends again and forms a saddle shape when it descends again. This shape is not noticeable in the training curve. The smoothed validation error saturates in the range between eight and nine percent with spikes of the raw error down to seven and a half percent.

6.1.2 Training using a 16 byte descriptor, batch normalization, an image pyramid and Gabor wavelets

We also wanted to enable a larger descriptor space and therefore doubled the descriptor size to 16 bytes. This barely affected the computational cost of the training process as it only doubled the outgoing connections of the last (fully connected) layer. The increased cost for comparing 16 byte descriptors instead of eight byte descriptors can be neglected. Figure 6.2 shows the training error curve in the top plot and the validation error curve in the bottom plot. The training curve performs a typical descent which is first very steep and then flattens out and goes into saturation. The validation curve, on the other hand, starts with a steep descend up to step 580k, but it then ascends again for over one million steps, forming a long saddle shape as it descends again. This behaviour is not reflected in the training curve. Thus, when the ascent took place for a long time over many steps, we thought the training started to over fit. But we let the training continue as we hoped that the validation error was just executing a saddle shape that we had seen before in our trainings. After the saddle, the validation error went down to about six and a half percent.

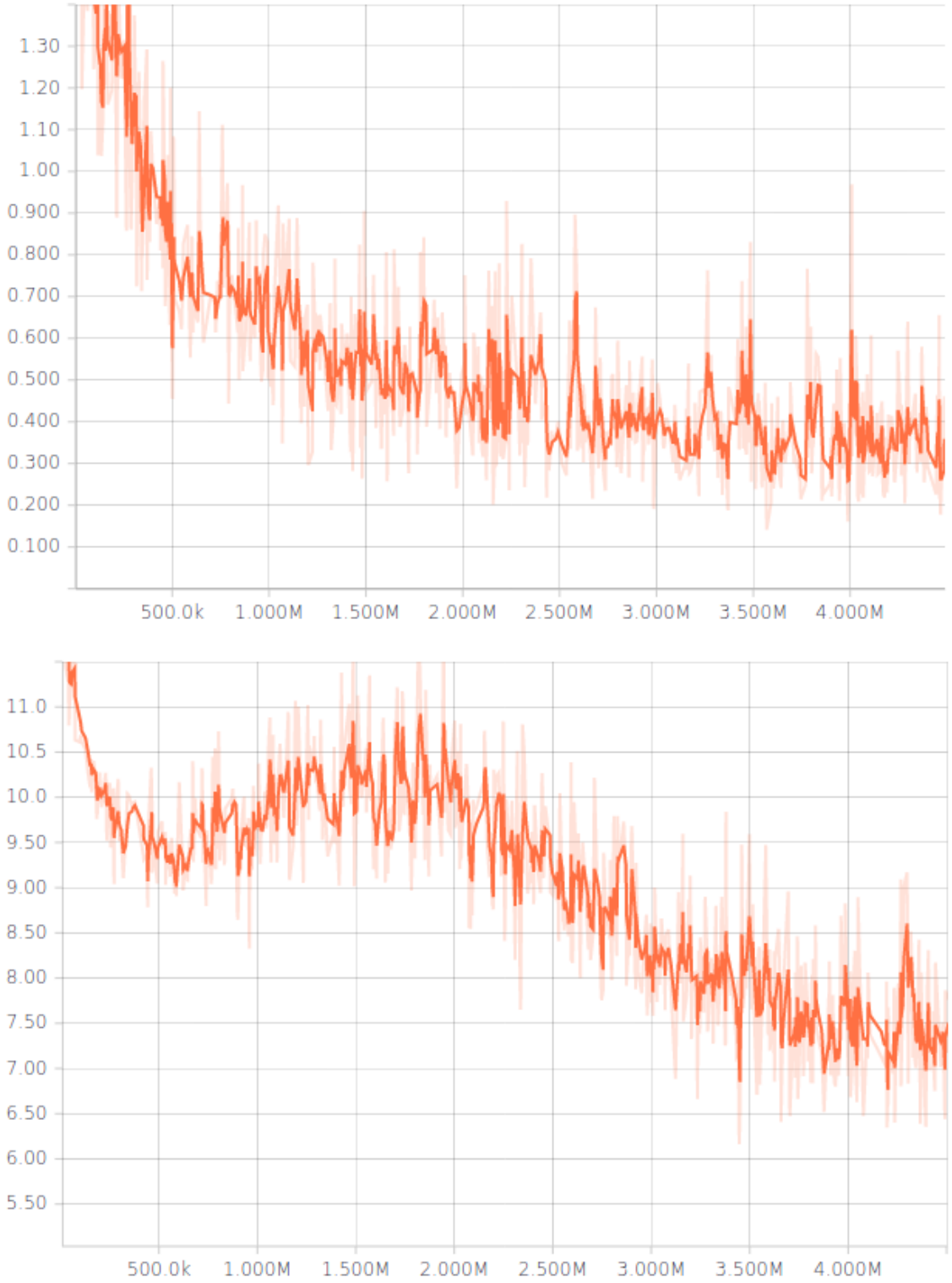


Figure 6.2: These plots visualize the training error curve (top plot) and the validation error curve (bottom plot) with the batch normalization turned on, a 16 byte descriptor output and a constant learning rate of $1 \cdot 10^{-5}$. The image pyramid and the Gabor wavelets are utilized. The vertical axis represents the error value and the horizontal axis represents the training step.

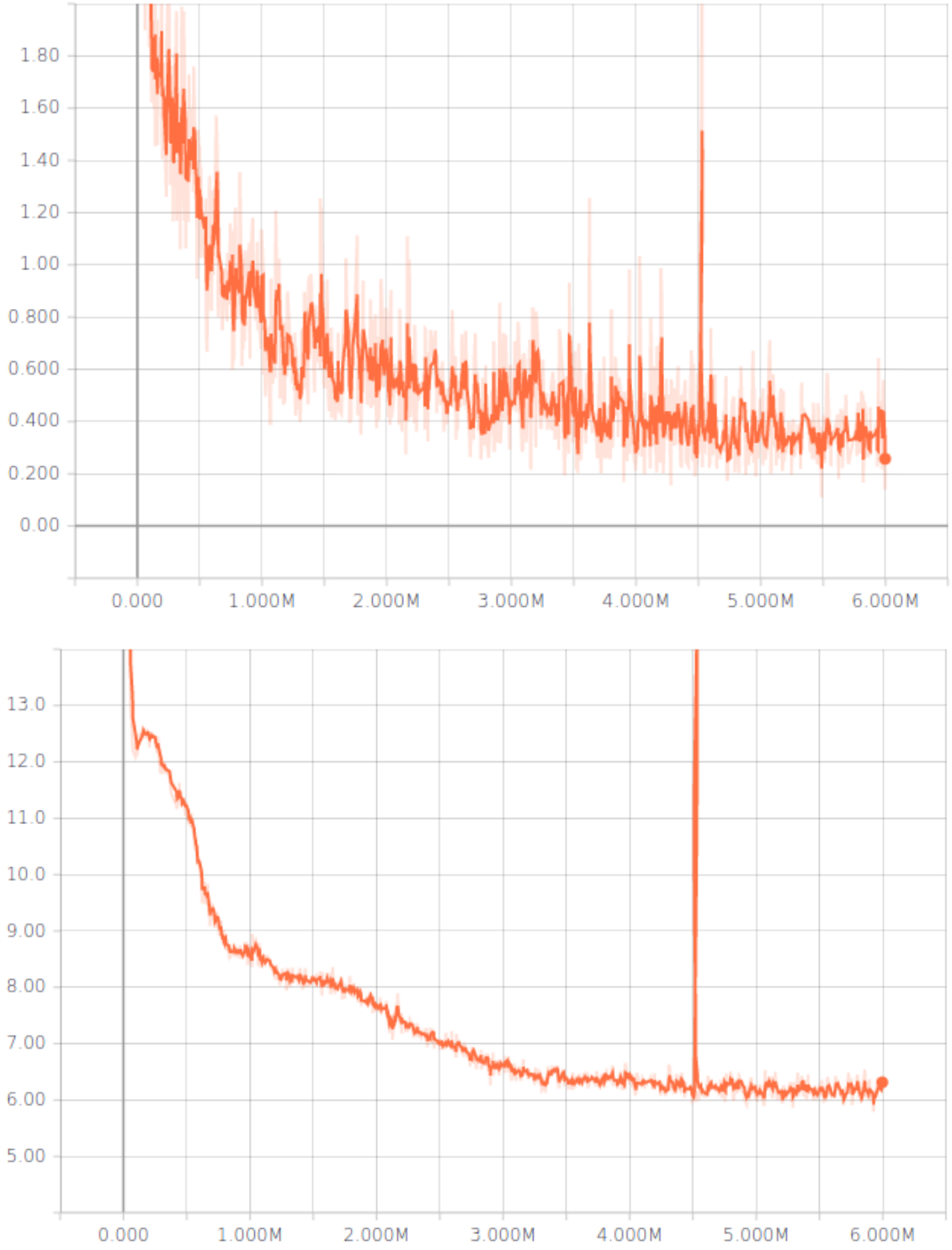


Figure 6.3: These plots visualize the training error curve (top plot) and the validation error curve (bottom plot) with the batch normalization turned off, a 16 byte descriptor output and a constant learning rate of $1 \cdot 10^{-5}$. The image pyramid and the Gabor wavelets are utilized. The vertical axis represents the error value and the horizontal axis represents the training step.

6.1.3 Deactivated batch normalization

Figure 6.3 shows the error curves of our best model. We deactivated the batch normalization to observe its influence on the training process. The variance of the training error, but especially that of the validation error, decreased significantly. We also did not find that convergence deteriorated after deactivating the batch normalisation. The validation error shows a small saddle at about 150k to 300k steps. This is not clearly reflected in the training error. Furthermore, the first approx. 800k steps consist of a steep decline in the training error and the validation error. After that, the training error and the validation error still decline, but much more slowly until about step 3.35M. From then on, there is still a very small decrease in the error, but the saturation sets in. This is also reflected in the training error. There is a strong peak in both error values at about 4.5M steps. Here we were curious to test what happens if, instead of the Gabor-filtered image patches, we insert grey-valued image patches into the pyramid while keeping the colour patches the same. As one can see it resulted in a strong increase of the training error and of the validation error. When switching back to using the Gabor wavelets, the error quickly returned to the previous magnitude. The fact that this short change in the training data did not make the trained weights obsolete is probably due to the small learning rate.

6.1.4 Deactivated Gabor wavelets

We also conducted a training where the Gabor wavelets were deactivated the whole time and the image pyramid was filled with the unfiltered grey-valued patches. The purpose was to evaluate how well the network would perform without the pre-filtered patches and whether it would learn some filter-like convolutions that would lead to a similarly good, or even a better overall performance. The convolutional layers are theoretically capable of learning Gabor filter alike convolutions on their own. It is also possible that they find a filter capable of extracting more useful information from the grey-valued patches than the Gabor wavelets are capable of extracting. We also switched off the batch normalisation based on our previous positive experience without the BN. Figure 6.4 shows the training error curve in the top plot and the validation error curve in the bottom plot. There is a steep decrease in the training and validation error in the first 750k steps. After that, the curve flattens out slowly, until about step four million. From then on, the validation error is saturated at about ten percent error. The smoothed training error is still decreasing, while the validation error is already saturated. This indicates an over-fitting to the training data, but not to the extent that it would lead to an increasing validation error. The minimum unsmoothed values of the training error did not reach new minima even in the phase of a continuous decrease of the smoothed error. With the validation error saturating at about ten percent, the result is clearly worse than the validation error of about six percent when using the Gabor filtered patches. Thus, the ResNet based architecture that we used did not manage to learn a filter which leads to a similarly good performance as the Gabor wavelet filter.

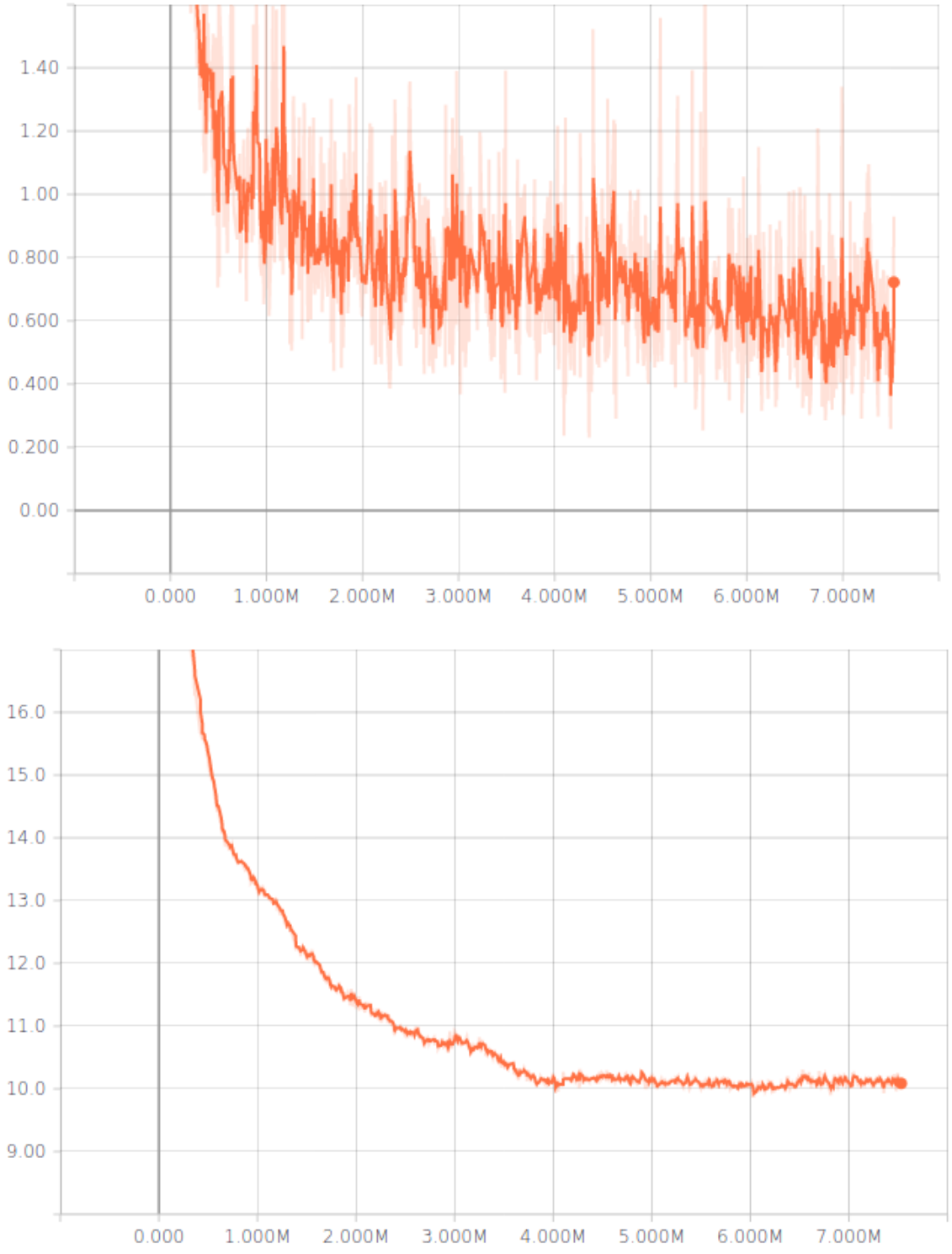


Figure 6.4: These plots visualize the training error curve (top plot) and the validation error curve (bottom plot) with the batch normalization turned off, a 16 byte descriptor output and a constant learning rate of $1 \cdot 10^{-5}$. The image pyramid is used, but the Gabor wavelets are deactivated. Thus, the pyramid is filled with unfiltered greyscale image patches. The vertical axis shows the error value and the horizontal axis the training step.

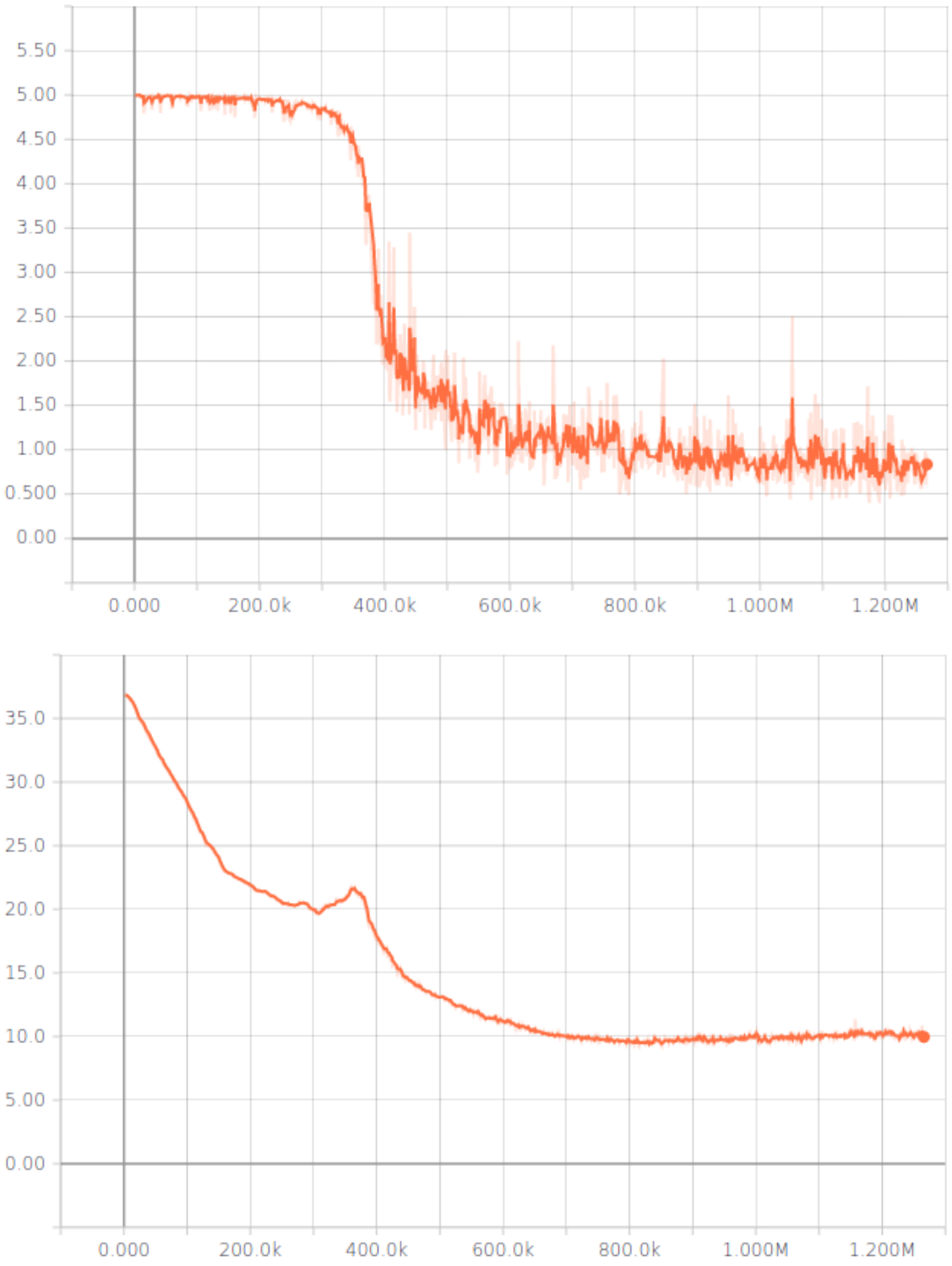


Figure 6.5: These plots visualize the training error curve (top plot) and the validation error curve (bottom plot) with the batch normalization turned off, a 16 byte descriptor output and a constant learning rate of $1 \cdot 10^{-5}$. The image pyramid and the Gabor wavelets are utilized. The colour patches are deactivated. The vertical axis represents the error value and the horizontal axis represents the training step.

6.1.5 Deactivated colour patches

We conducted another training for which we deactivated the colour patches. Here we wanted to find out how the network performs when only the Gabor wavelet filtered patches of the image pyramid are available. We also had the batch normalization turned off due to our previous positive experiences without the BN. Figure 6.5 shows the training error curve in the top plot and the validation error curve in the bottom plot. One can see that the validation error steeply declines during the first 300k steps. Interestingly, the training error hardly decreases until step 300k. It stays near the error value of five, which is the margin we use in the triplet loss. Shortly thereafter, the training error decreases rapidly while, interestingly, the validation error begins to increase forming a small saddle point at about step 360k. If the training was stopped there, one would interpret the increasing validation error in combination with the decreasing training error as overfitting. But, as the validation error joins the training error in a continued decline after the saddle point, the improvements in the training also lead to an improved generalisation. The best performance is achieved at about step 800k with a validation error of about nine percent. Thereafter, the validation error slowly increases while the training error continues to gradually decrease. This indicates a slight overfitting behaviour.

6.1.6 Summary of the training evaluation

When one compares the previous trainings, one can see that we achieved better results when the batch normalization was turned off. We achieved the lowest validation error of less than six percent when we used the image pyramid and the Gabor wavelets (see paragraph 6.1.3). Disabling the Gabor wavelets, but still using the image pyramid with greyscale patches, resulted in a model with a validation error of slightly less than ten percent (see paragraph 6.1.4). When we deactivated the colour patches, but kept the image pyramid and the Gabor wavelets in use, we achieved a validation error of about nine percent (see paragraph 6.1.5). Compared to the other trainings, it reached its saturation quite quickly at step 800k, while the model which was trained without the Gabor wavelets needed 4M training steps to reach its saturation at about ten percent validation error (see figure 6.4). Note that our best model already produced a validation error of less than nine percent after about 750k steps (see figure 6.3). And it outperformed the other models which did not use the colour patches and the image pyramid with Gabor filtered patches at the same time, or which had the batch normalization turned on.

6.2 Practical usage evaluation

In this section we will analyse the applicability of different line segment descriptors. The following evaluations, and especially the analysis in 6.2.5, is of interest for matching problems in general (e.g. matching in medical imaging, point feature matching in robotics for feature graphs, etc.) and is not limited to line matching. When descriptors are used in practical scenarios to solve structure from motion, visual odometry or simultaneous localization and mapping problems, it is usually very important that the matches are correct. This means that the closest match, the one with the smallest descriptor distance, has to be the correct match. It is often not sufficient if the correct match is among the closest three or the closest five matches, but not the first match because in many cases only the closest match is considered. Visualizations such as RoC curves and metrics, that check whether the correct match is among the best n matches, are still useful to compare different descriptor methods as they provide a measure of the quality of a descriptor method. Nevertheless, a descriptor method that performs well on the aforementioned metrics usually computes the first best matches similarly well. There are also global matching optimisation methods, such as the pairwise matcher by Lilian Zhang and Reinhard Koch [ZK13], for which the closest matching distance is not that important. However, these methods are not used as often because they are more computationally expensive.
In the following, we show plots visualizing the first best matches performance of the WLD, the DLD and the LBD. The plots are beneficial to compare the usefulness of the descriptor methods in practical scenarios. They are created by increasing the descriptor distance threshold from zero to the maximum value. During this increase, the lines from the first image are each matched with the first best matching line from the second image. For each additional match added while increasing the descriptor distance threshold, it is checked if the match is a correct match or an incorrect match. A curve is plotted by moving one step in the direction of the y-axis for each correct match and one step in the direction of the x-axis for each incorrect match. As this curve is plotted in steps and the matches are added from the smallest to the largest distance, one would expect the curve to initially rise in the direction of the y-axis and later slope more towards the x-axis when the matches have a larger descriptor distance and are more likely to be false matches. So one can read not only how many true and false matches occur in a practical scenario, but also when they occur. And if, for example, the first half of the matches are predominantly true matches and the false matches occur mainly in the second half of the matches, then one could decide to only regard the better first half of the matches. However, not only can one see how well the method performs compared to other methods and how many of the first best matches are safe to be used, but we can also visualize how different criteria affect the results. Two commonly used criteria are the left-right consistency check and a margin ε that defines how close the second best match is allowed to be. Matches that do not meet the criteria can be rejected. Rejecting matches results in a shorter curve, but the expectation is that more false matches than true matches are removed, resulting in a

steeper curve with a higher proportion of true matches.

Left-right consistency check

The left-right consistency check is a commonly used criterion that originates from stereo matching. However, it is not limited to the stereo matching problem, the principle can be used in various matching tasks. In our case, for each line segment from the first image, we choose the best matching line segment from the second image. Then, for this line segment from the second image, we check whether its first best matching line segment from the first image is the line segment for which we started the check. In short, we check whether they are mutually first best matches. It is not guaranteed that a potential match that passes this check is a correct match. But it is more likely since the case in which the two matching procedures result in the same matching pair is less likely to be incorrect than the case in which one of them alone is incorrect. If the Left-Right consistency check fails, the potential match is rejected.

ε distance margin between the first best and the second best match

Another matching criterion is based on the unambiguity of a match. This can be checked for a line segment of the first image by calculating its descriptor distances to the line segments of the second image. Then the descriptor distances to the best matching line segment and to the second best matching line segment are calculated. The requirement for a potential match to pass the check is the following:

$$dist_{first} <= dist_{second} \cdot \varepsilon, \tag{6.1}$$

where $dist_{first}$ is the distance to the first best match, $dist_{second}$ is the distance to the second best match, and $\varepsilon \in [0,\ 1]$ is the distance ratio factor. If $\varepsilon = 1$, then the check is always passed and the criterion is disabled. In all other cases, the check is only passed if the distance between the best and the second best descriptor distance is large enough. If this distance is small, then it is not very certain that the match is correct. A large distance to the second best match ensures a definiteness of the match. This criterion also does not guarantee that a match is correct, but it increases the chance for a match to be correct.

How to read the following evaluation plots

In the following, we will evaluate the first best matches performance and the influence of the previously mentioned criteria on the descriptor methods. The curves in the plots show the occurrence of true and false matches. The occurrence of the matches is ordered by ascending descriptor distance, which means that the matches with the smallest descriptor distance are the closest to the origin, and the higher the distance, the farther away from the origin on the curve the match is found. Correct matches draw the curve one step upwards in y-axis direction, while false matches result in a step to the right in x-axis

direction. The plot shows several curves, while the difference between the red and the blue curves is that the left-right consistency check is turned off for the red curves and turned on for the blue curves. The difference between the curves of one colour is the ε used. Here, the most rightward sloping curves are those with the highest distance ratio of $\varepsilon = 1$. Each decrement of ε leads to a curve that leans less to the right. Thus, the curves in the plot can be assigned to the labels by counting the legend from the top to the bottom, while counting the curves from the right to the left, for each colour respectively.

How to read the following evaluation tables

The evaluation tables presented in the following paragraphs display numbers about the matches for the descriptor methods. The labels of the columns denote the following:

ε:	distance margin between the first best and the second best match.
L-R CC:	Left-Right Consistency Check.
# matches:	total number of matches.
# true:	number of true matches.
# false:	number of false matches.
% false:	relative number of false matches.
Δ true:	decrease in absolute true matches to the previous row.
Δ false:	decrease in absolute false matches to the previous row.
Δ % true:	decrease in relative true matches to the previous row.
Δ % false:	decrease in relative false matches to the previous row.

6.2.1 First best matches evaluation of the WLD on the Middlebury and the difficult conditions scenes

Figure 6.6 shows the occurrence of true and false matches of the WLD on the Middlebury and the difficult conditions scenes. The number of matches at the endpoints of the curves are compared in table 6.2. We can see that when every line is matched, which is the case when the Left-Right consistency check is off and the distance ratio threshold is inactive at $\varepsilon = 1.0$, there are a total of 6002 matches, of which 5012 are correct and 990, which is about 16.5%, are false. Whilst with the Left-Right consistency check turned on, over one thousand matches are rejected, leaving 4807 matches. Of these, however, only 245 are false, which is 5.1% false matches compared to 16.5% false matches without the check. Also, if we have a look at a result with a similar number of matches, where the Left-Right consistency check is turned off, but the matches are filtered by the distance ratio threshold, for example, the result at $\varepsilon = 0.90$, we have a total of 4919 matches. Of these 4919 matches, 4528 are correct and 391 are false matches. Thus, even though we have more matches here, fewer of them are correct. Even at $\varepsilon = 0.85$ we have fewer matches and fewer true matches than with the Left-Right consistency check turned on

and the distance margin threshold turned off, but we still have more false matches. This behaviour is also reflected in figure 6.6. If one does not require the maximum number of true matches, there is almost always a blue curve (Left-Right consistency check turned on) with a similar number of true matches and a lower number of false matches than on a red curve. This is better visualized in the left plot of figure 6.9, which zooms in on the range of 2572 to 3430 true matches of figure 6.6. Only when the numbers get close to zero false matches, at a distance ratio threshold of $\varepsilon = 0.30$, the Left-Right consistency check is slightly disadvantageous, as two fewer true matches remain while the number of five false matches remains the same. At an ε of 0.19, there are zero false matches and there is no difference whether the Left-Right consistency check is turned on or not. Thus, the more matches are rejected by the distance ratio threshold, the smaller is the remaining influence of the Left-Right consistency check. This is also visible in the plots, where the spacing of the blue and red curves is much smaller at lower ε's than at higher ε's.

Table 6.2: Evaluation table for the WLD applied to the Middlebury and the difficult conditions scenes. (See the explanation of how to read the table in section 6.2.)

ε	L-R CC	# matches	# true	# false	% false	Δ true	Δ false	Δ % true	Δ % false
1.00	ON	4807	4562	245	5.1	-	-	-	-
0.99	ON	4775	4543	232	4.9	19	13	0.4	5.3
0.95	ON	4608	4419	189	4.1	124	43	2.7	18.5
0.90	ON	4442	4295	147	3.3	124	42	2.8	22.2
0.85	ON	4298	4176	122	2.8	119	25	2.8	17.0
0.80	ON	4165	4065	100	2.4	111	22	2.7	18.0
0.70	ON	3897	3833	64	1.6	232	36	5.7	36.0
0.60	ON	3602	3564	38	1.1	269	26	7.0	40.6
0.50	ON	3243	3220	23	0.7	344	15	9.7	39.5
0.30	ON	2434	2429	5	0.2	791	18	24.6	78.3
0.19	ON	1775	1775	0	0.0	654	5	26.9	100.0
1.00	OFF	6002	5012	990	16.5	-	-	-	-
0.99	OFF	5830	4961	869	14.9	51	121	1.0	12.2
0.95	OFF	5311	4726	585	11.0	235	284	4.7	32.7
0.90	OFF	4919	4528	391	7.9	198	194	4.2	33.2
0.85	OFF	4637	4363	274	5.9	165	117	3.6	29.9
0.80	OFF	4412	4214	198	4.5	149	76	3.4	27.7
0.70	OFF	4008	3905	103	2.6	309	95	7.3	48.0
0.60	OFF	3647	3599	48	1.3	306	55	7.8	53.4
0.50	OFF	3259	3233	26	0.8	366	22	10.2	45.8
0.30	OFF	2434	2429	5	0.2	804	21	24.9	80.8
0.19	OFF	1775	1775	0	0.0	654	5	26.9	100.0

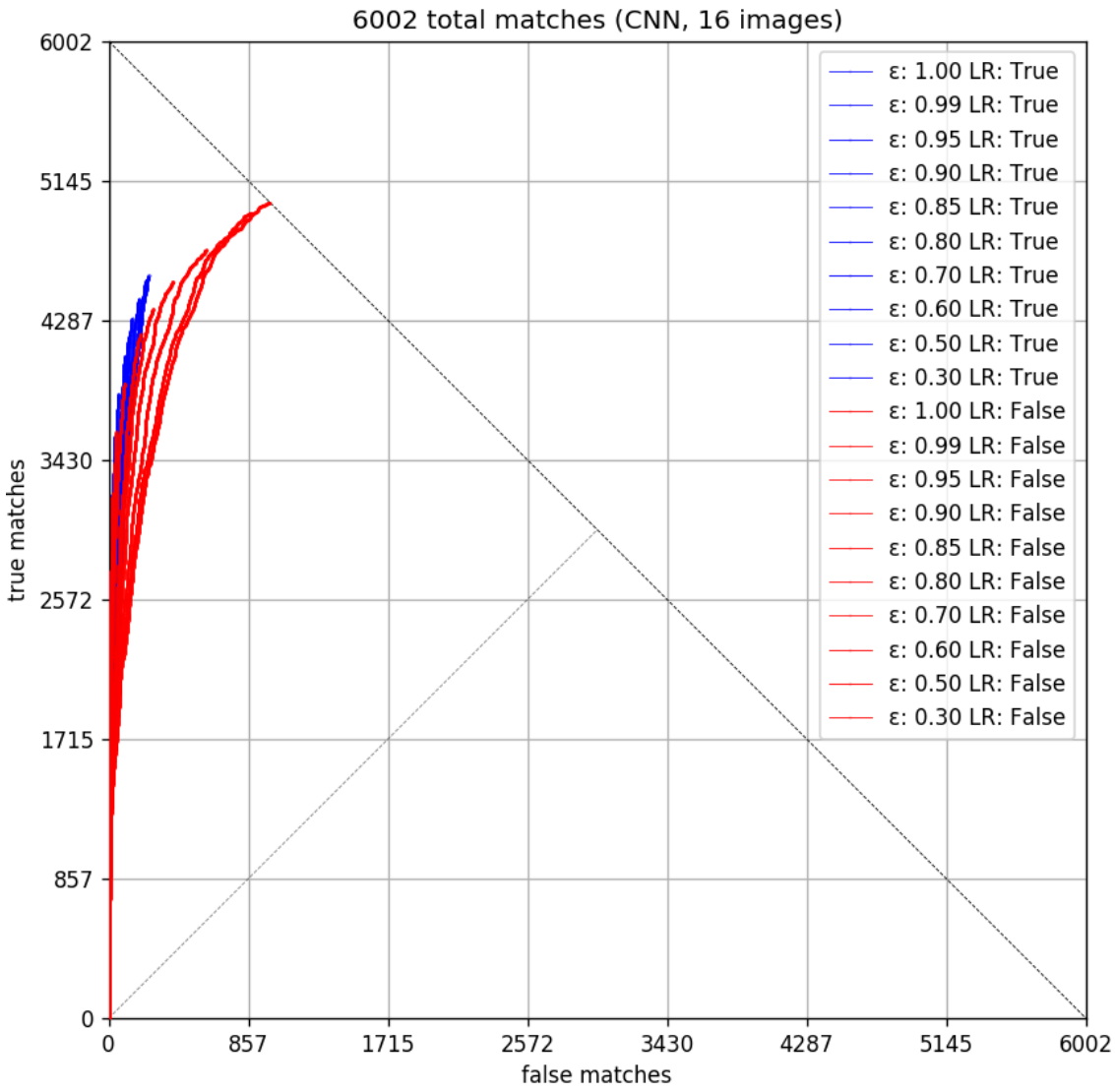


Figure 6.6: This plot shows the occurrence of correct (true) and incorrect (false) matches when for each line from the first image only the first best match from the second image is considered, while the threshold is increased from zero to the maximum. The results were calculated on the Middlebury Stereo Dataset images and on the difficult conditions scenes using our WLD descriptor. The red curves were calculated with the Left-Right check turned off while it was turned on for the blue curves. Each curve corresponds to a distance ratio threshold ε while the rightmost curves (containing the most false matches) correspond to the highest ε.

In the zoomed plots of figure 6.9 one can see that the curves of one colour run side by side and do not cross each other. They would only cross if a decreased ε led to a worsened ratio of rejected true matches to false matches. We can also see that the longer

a curve is, the more it leans to the right, which is what we expected. Each step means a larger descriptor distance and matches with a higher distance are usually more likely to be false matches.

6.2.2 Separated first best matches evaluation of the WLD on the Middlebury and on the difficult conditions scenes

Table 6.3: Evaluation table for the WLD applied to the Middlebury scenes. (See the explanation how to read the table in section 6.2.)

ε	L-R CC	# matches	# true	# false	% false	Δ true	Δ false	Δ % true	Δ % false
1.00	ON	3508	3342	166	4.7	-	-	-	-
0.99	ON	3493	3333	160	4.6	9	6	0.3	3.6
0.95	ON	3387	3252	135	4.0	81	25	2.4	15.6
0.90	ON	3284	3172	112	3.4	80	23	2.5	17.0
0.85	ON	3193	3099	94	2.9	73	18	2.3	16.1
0.80	ON	3099	3019	80	2.6	80	14	2.6	14.9
0.70	ON	2907	2854	53	1.8	165	27	5.5	33.8
0.60	ON	2674	2641	33	1.2	213	20	7.5	37.7
0.50	ON	2370	2351	19	0.8	290	14	11.0	42.4
0.30	ON	1700	1695	5	0.3	656	14	27.9	73.7
0.16	ON	1011	1011	0	0.0	684	5	40.4	100.0
1.00	OFF	4287	3601	686	16.0	-	-	-	-
0.99	OFF	4206	3583	623	14.8	18	63	0.5	9.2
0.95	OFF	3912	3457	455	11.6	126	168	3.5	27.0
0.90	OFF	3652	3332	320	8.8	125	135	3.6	29.7
0.85	OFF	3463	3236	227	6.6	96	93	2.9	29.1
0.80	OFF	3299	3135	164	5.0	101	63	3.1	27.8
0.70	OFF	3001	2912	89	3.0	223	75	7.1	45.7
0.60	OFF	2713	2670	43	1.6	242	46	8.3	51.7
0.50	OFF	2383	2361	22	0.9	309	21	11.6	48.8
0.30	OFF	1700	1695	5	0.3	666	17	28.2	77.3
0.16	OFF	1011	1011	0	0.0	684	5	40.4	100.0

In this paragraph, we want to examine the dependence of the WLD in combination with the Left-Right consistency check and with the distance margin threshold ε on the type of data.
Therefor, we ran the descriptor separately on the Middlebury and on the difficult conditions scenes. Figure 6.7 and figure 6.8 show the occurrence of true and false matches of the WLD on the Middlebury and on the difficult conditions scenes. The corresponding numbers of the matches at the endpoints of the curves are shown in table 6.3 and in table 6.4 respectively. We can see: with the Left-Right consistency check turned on, we get fewer matches, but also fewer false matches in both datasets. The difference is much larger for the Middlebury scenes: with the Left-Right consistency check turned on, and an ε of 0.99, we get 3333 true matches and only 160 false matches (see table 6.3). Whereas with the Left-Right consistency check turned off and an ε of 0.90, we get 3332 true matches, but, with 320 false matches, we have twice as many false matches. This advantage of having fewer false matches at a similar number of true matches by using the Left-Right consistency check continues until its influence diminishes at an ε of 0.30. There, the distance margin threshold already settles all the rejections that the Left-Right consistency check would have done.

For the difficult conditions scenes, the advantage is still there, but it is smaller. For example: with the Left-Right consistency check turned on and an ε of 0.99 we get 1210 true matches and 72 false matches, which is about the same ratio as with the check turned off and an ε of 0.90, resulting in 1196 true matches and 71 false matches (see table 6.4). But the Left-Right consistency check is slightly advantageous at smaller ε's of 0.90 and 0.70. There, we have a similar number of false matches as for ε's of 0.80 and 0.70 when the check is turned off, but a better ratio of true to false matches. Additionally, when comparing the curves of figure 6.7 and figure 6.8, one can also see that the advantage of utilizing the Left-Right consistency check is higher for the Middlebury scenes. Thus, the degree of the advantage of the Left-Right consistency check depends on the data. The same also holds for the question which value of ε is a good choice. For the Middlebury scenes an ε of 0.16 is necessary to achieve zero false matches with fewer than a third of the true matches remaining. On the difficult conditions scenes the ε only has to be set to 0.30 to achieve zero false matches and over half of the true matches remain. The behaviour, that it is dependent on the dataset, which values of ε achieve what ratio of false matches to true matches, is similar for the whole range of values of ε.

Table 6.4: Evaluation table for the WLD applied to the difficult conditions scenes. (See the explanation how to read the table in section 6.2.)

ε	L-R CC	# matches	# true	# false	% false	Δ true	Δ false	Δ % true	Δ % false
1.00	ON	1299	1220	79	6.1	-	-	-	-
0.99	ON	1282	1210	72	5.6	10	7	0.8	8.9
0.95	ON	1221	1167	54	4.4	43	18	3.6	25.0
0.90	ON	1158	1123	35	3.0	44	19	3.8	35.2
0.85	ON	1105	1077	28	2.5	46	7	4.1	20.0
0.80	ON	1066	1046	20	1.9	31	8	2.9	28.6
0.70	ON	990	979	11	1.1	67	9	6.4	45.0
0.60	ON	928	923	5	0.5	56	6	5.7	54.5
0.50	ON	873	869	4	0.5	54	1	5.9	20.0
0.30	ON	734	734	0	0.0	135	4	15.5	100.0
1.00	OFF	1715	1411	304	17.7	-	-	-	-
0.99	OFF	1624	1378	246	15.1	33	58	2.3	19.1
0.95	OFF	1399	1269	130	9.3	109	116	7.9	47.2
0.90	OFF	1267	1196	71	5.6	73	59	5.8	45.4
0.85	OFF	1174	1127	47	4.0	69	24	5.8	33.8
0.80	OFF	1113	1079	34	3.1	48	13	4.3	27.7
0.70	OFF	1007	993	14	1.4	86	20	8.0	58.8
0.60	OFF	934	929	5	0.5	64	9	6.4	64.3
0.50	OFF	876	872	4	0.5	57	1	6.1	20.0
0.30	OFF	734	734	0	0.0	138	4	15.8	100.0

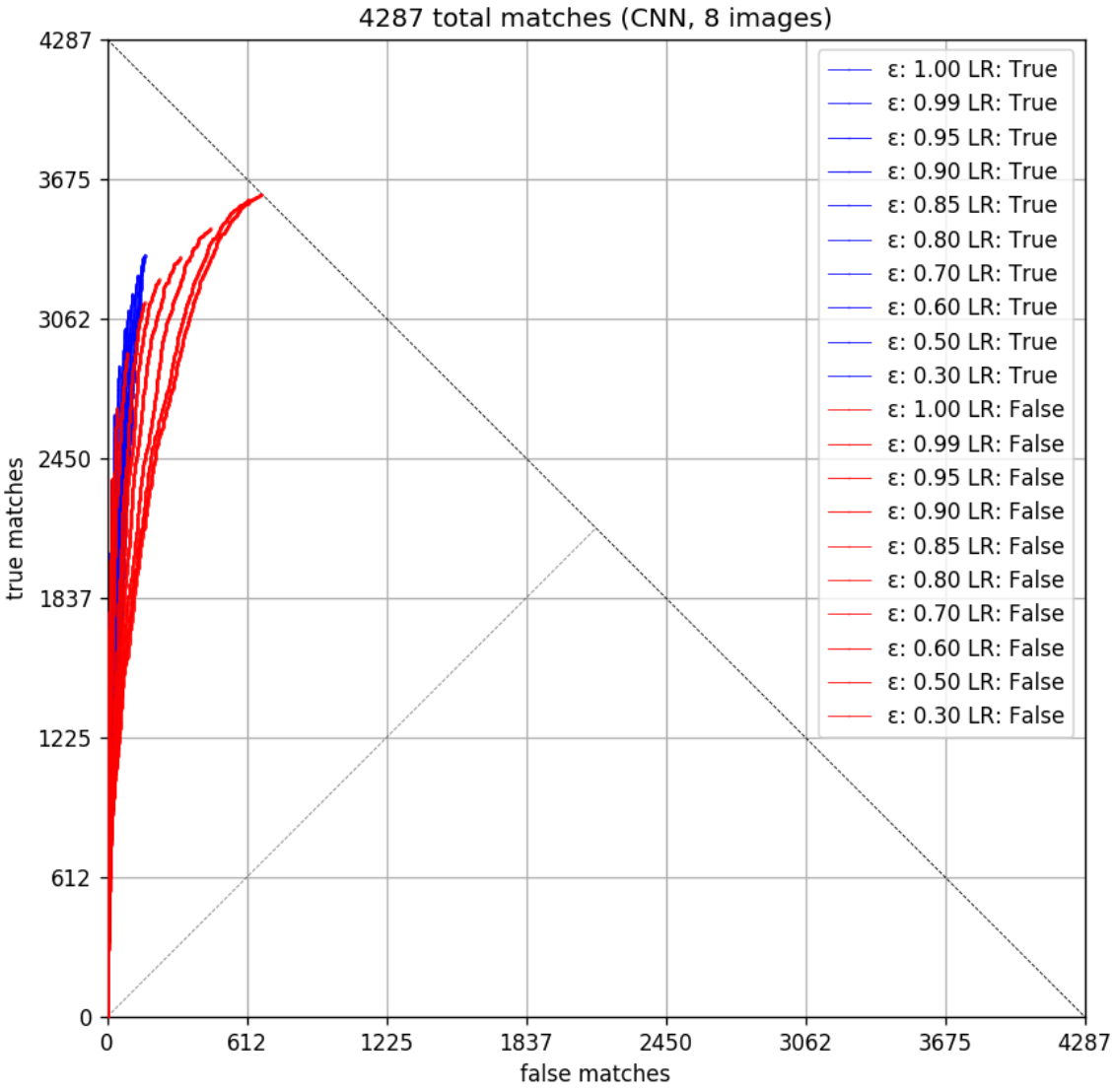


Figure 6.7: This plot shows the occurrence of correct (true) and incorrect (false) matches when for each line from the first image only the first best match from the second image is considered, while the threshold is increased from zero to the maximum. The results were calculated on the Middlebury Stereo Dataset images using our WLD descriptor. The red curves were calculated with the Left-Right check turned off while it was turned on for the blue curves. Each curve corresponds to a distance ratio threshold ε while the rightmost curves (containing the most false matches) correspond to the highest ε.

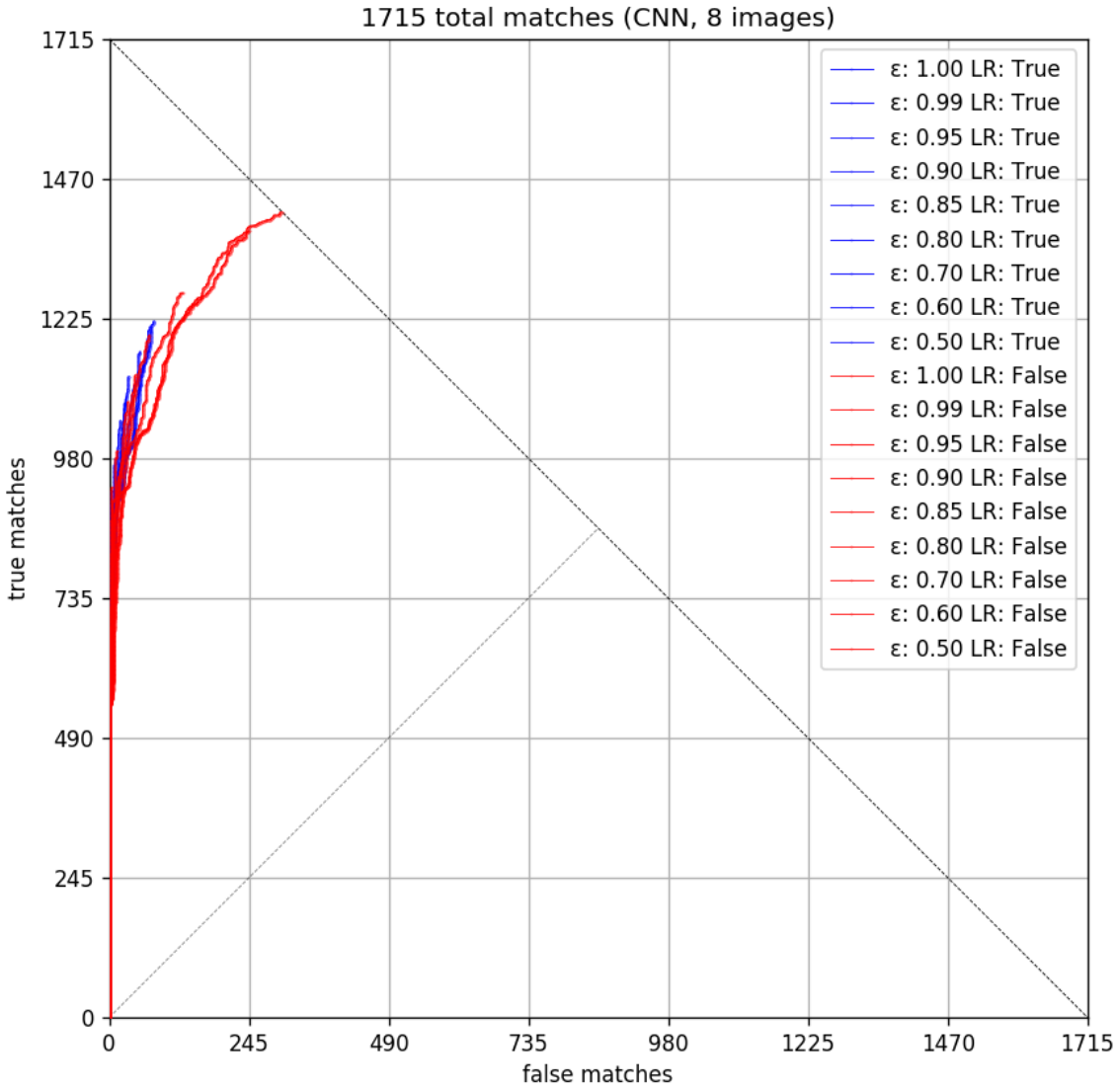


Figure 6.8: This plot shows the occurrence of correct (true) and incorrect (false) matches when for each line from the first image only the first best match from the second image is considered, while the threshold is increased from zero to the maximum. The results were calculated on the difficult conditions scenes using our WLD descriptor. The red curves were calculated with the Left-Right check turned off while it was turned on for the blue curves. Each curve corresponds to a distance ratio threshold ε while the rightmost curves (containing the most false matches) correspond to the highest ε.

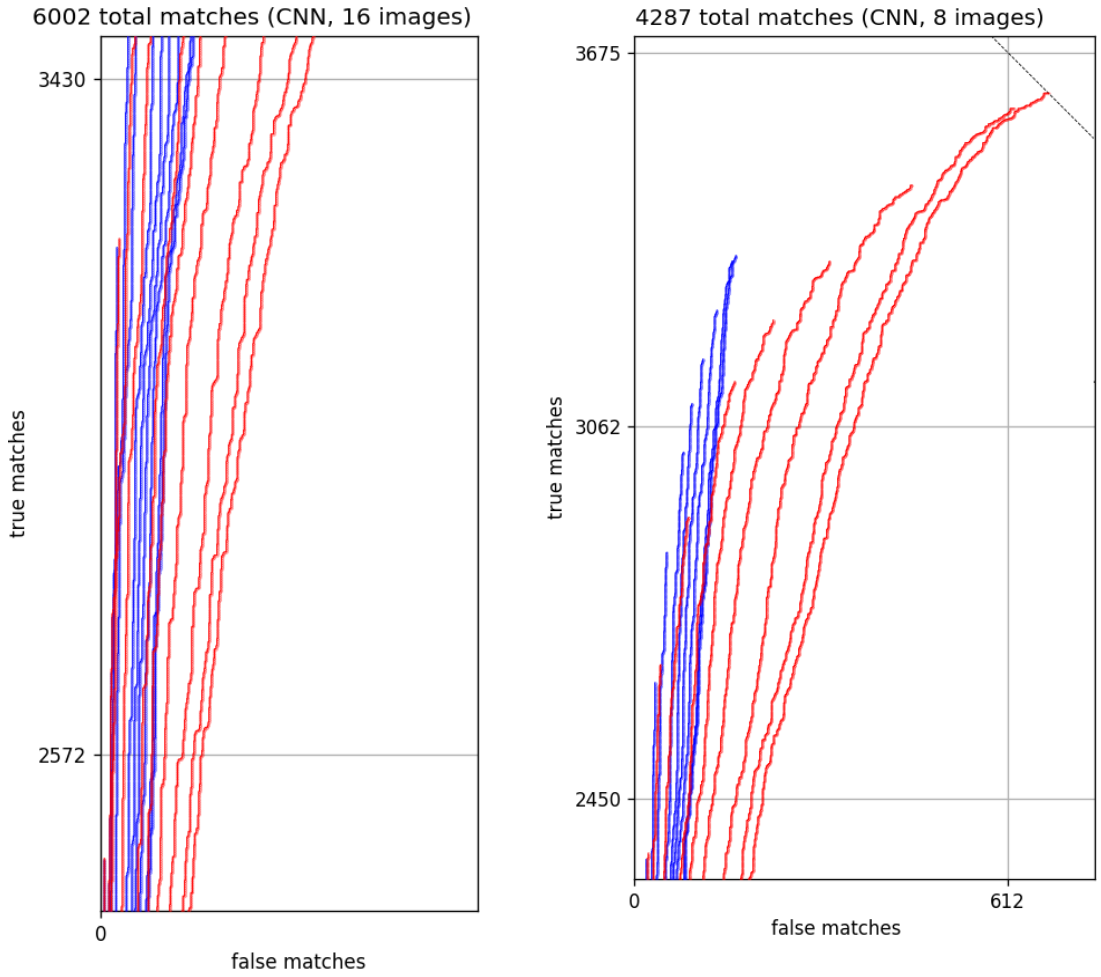


Figure 6.9: The plots show the occurrence of correct and incorrect matches when only the first best match is considered. The left plot is a zoomed in version of figure 6.6. This better visualizes the range of 2572 to 3430 true matches. The plot on the right is a zoomed in version of figure 6.7. This better visualizes the range of 2450 to 3675 true matches.

6.2.3 First best matches evaluation of the DLD on the Middlebury and the difficult conditions scenes

Figure 6.10 shows the occurrence of true and false matches of the DLD on the Middlebury and the difficult conditions scenes. The number of matches at the endpoints of the curves are compared in table 6.5. We get 6002 matches of which 3542 are correct when no filter is enabled. With the Left-Right consistency check turned on, we get a total of 3692 matches of which 3043 are correct. Compared to the WLD (see table 6.2), we have about 1500 fewer true matches in both cases. Additionally, we have a higher rate of false

matches in both cases. Even when the Left-Right consistency check is turned on for the DLD and off for the WLD, the rate of false matches is higher for the DLD at similar numbers of matches. This improvement is also clearly visible when one compares the curves of figure 6.10 to the curves of figure 6.6. The curves of the DLD bend far more to the right than the curves of the WLD, which means that there are more false matches. All this shows the great advancement of the WLD, which also applies in practical usage scenarios.

Table 6.5: Evaluation table for the DLD applied to the Middlebury and the difficult conditions scenes. (See the explanation how to read the table in section 6.2.)

ε	L-R CC	# matches	# true	# false	% false	Δ true	Δ false	Δ % true	Δ % false
1.00	ON	3692	3043	649	17.6	-	-	-	-
0.99	ON	3644	3018	626	17.2	25	23	0.8	3.5
0.95	ON	3465	2924	541	15.6	94	85	3.1	13.6
0.90	ON	3270	2832	441	13.5	92	100	3.1	18.5
0.85	ON	3110	2734	376	12.1	98	65	3.5	14.7
0.80	ON	2954	2641	313	10.6	93	63	3.4	16.8
0.70	ON	2644	2430	214	8.1	211	99	8.0	31.6
0.60	ON	2304	2168	136	5.9	262	78	10.8	36.4
0.50	ON	1958	1891	67	3.4	277	69	12.8	50.7
0.30	ON	1289	1275	14	1.1	616	53	32.6	79.1
0.16	ON	705	705	0	0.0	570	14	44.7	100.0
1.00	OFF	6002	3542	2460	41.0	-	-	-	-
0.99	OFF	5738	3497	2241	39.1	45	219	1.3	8.9
0.95	OFF	4915	3340	1575	32.0	157	666	4.5	29.7
0.90	OFF	4219	3142	1077	25.5	198	498	5.9	31.6
0.85	OFF	3787	2978	809	21.4	164	268	5.2	24.9
0.80	OFF	3417	2822	595	17.4	156	214	5.2	26.5
0.70	OFF	2861	2530	331	11.6	292	264	10.3	44.4
0.60	OFF	2397	2223	174	7.3	307	157	12.1	47.4
0.50	OFF	2004	1922	82	4.1	301	92	13.5	52.9
0.30	OFF	1293	1279	14	1.1	643	68	33.5	82.9
0.16	OFF	705	705	0	0.0	574	14	44.9	100.0

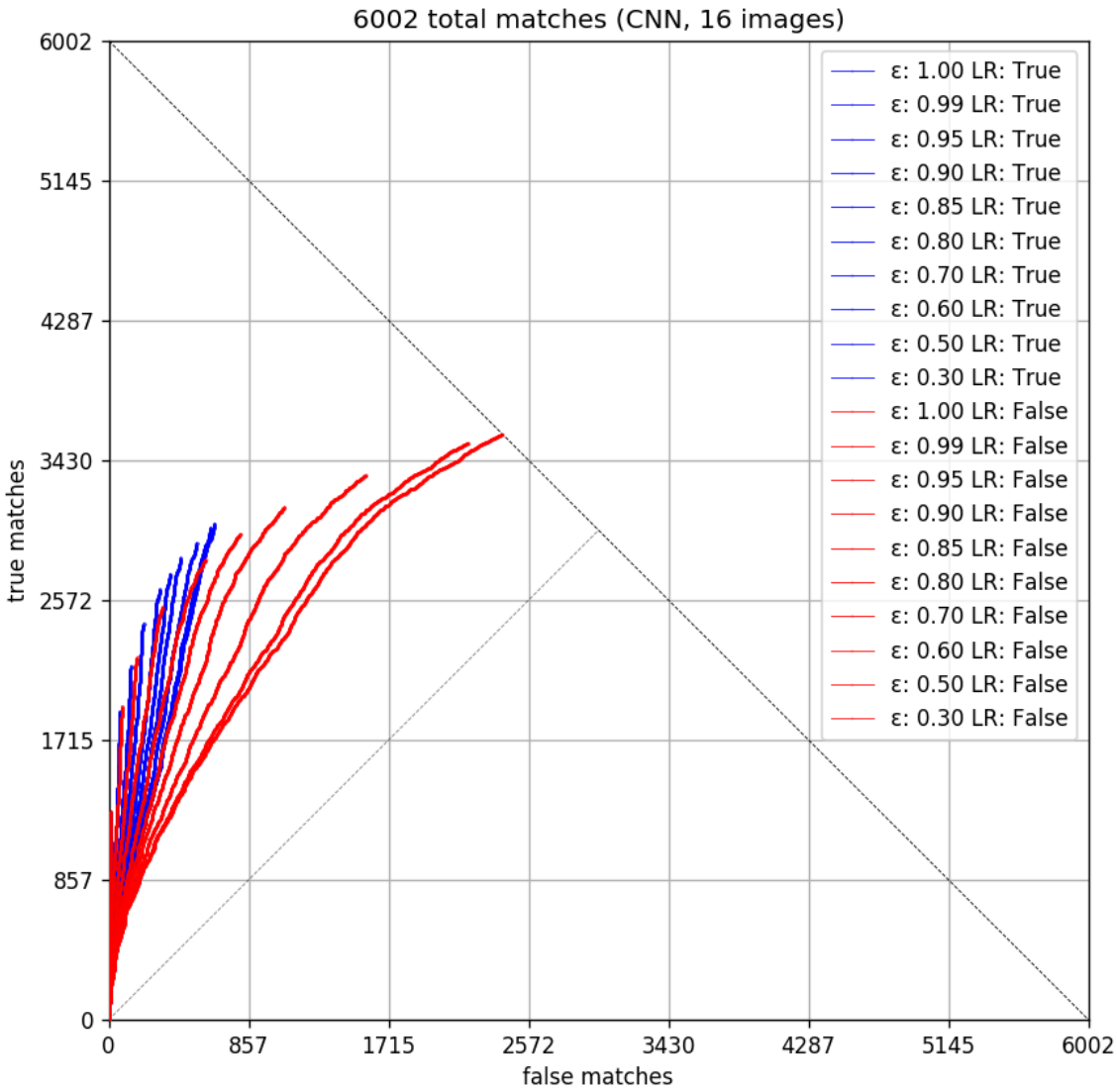


Figure 6.10: This plot shows the occurrence of correct (true) and incorrect (false) matches when for each line from the first image only the first best match from the second image is considered, while the threshold is increased from zero to the maximum. The results were calculated on the Middlebury Stereo Dataset images and on the difficult conditions scenes using our DLD descriptor. The red curves were calculated with the Left-Right check turned off while it was turned on for the blue curves. Each curve corresponds to a distance ratio threshold ε while the rightmost curves (containing most false matches) correspond to the highest ε.

6.2.4 First best matches evaluation of the LBD on the Middlebury and the difficult conditions scenes

Figure 6.11 shows the occurrence of true and false matches of the LBD on the Middlebury and the difficult conditions scenes. The numbers of the matches at the endpoints of the curves are compared in table 6.6. When no filters are active, there are 6002 matches out of which the WLD matches 5012 correctly. The DLD is only able to score 3542 true matches. And the LBD is even farther behind with only 2940 true matches. This means that more than half of the matches of the LBD are false. The high number of false matches of the LBD compared to the DLD and to the WLD continues throughout the value range of ε. When the Left-Right consistency check is utilized, the LBD still yields the highest number of false matches of the three methods. This also becomes clear when the curves of figure 6.11 are compared to the corresponding ones of the DLD or the WLD.

Table 6.6: Evaluation table for the LBD applied to the Middlebury and the difficult conditions scenes. (See the explanation how to read the table in section 6.2.)

ε	L-R CC	# matches	# true	# false	% false	Δ true	Δ false	Δ % true	Δ % false
1.00	ON	3173	2466	707	22.3	-	-	-	-
0.99	ON	3020	2411	609	20.2	55	98	2.2	13.9
0.95	ON	2822	2315	507	18.0	96	102	4.0	16.7
0.90	ON	2505	2165	340	13.6	150	167	6.5	32.9
0.85	ON	2227	2004	223	10.0	161	117	7.4	34.4
0.80	ON	1987	1836	151	7.6	168	72	8.4	32.3
0.70	ON	1562	1497	65	4.2	339	86	18.5	57.0
0.60	ON	1173	1144	29	2.5	353	36	23.6	55.4
0.50	ON	764	757	7	0.9	387	22	33.8	75.9
0.35	ON	317	317	0	0.0	440	7	58.1	100.0
1.00	OFF	6002	2940	3062	51.0	-	-	-	-
0.99	OFF	5138	2825	2313	45.0	115	749	3.9	24.5
0.95	OFF	4052	2605	1447	35.7	220	866	7.8	37.4
0.90	OFF	3065	2335	730	23.8	270	717	10.4	49.6
0.85	OFF	2522	2107	415	16.5	228	315	9.8	43.2
0.80	OFF	2121	1897	224	10.6	210	191	10.0	46.0
0.70	OFF	1589	1512	77	4.8	385	147	20.3	65.6
0.60	OFF	1175	1144	31	2.6	368	46	24.3	59.7
0.50	OFF	765	757	8	1.0	387	23	33.8	74.2
0.35	OFF	317	317	0	0.0	440	8	58.1	100.0

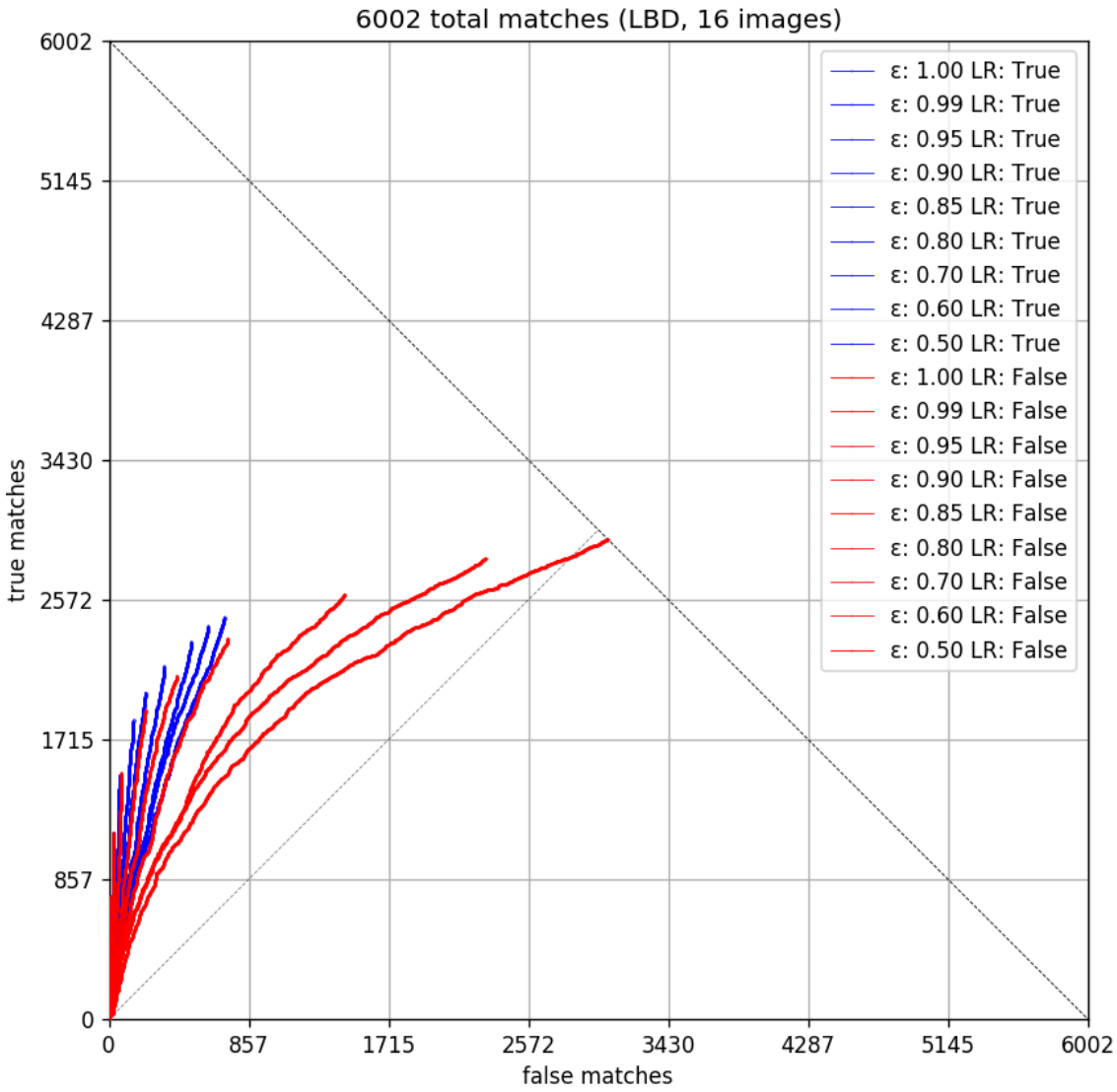


Figure 6.11: This plot shows the occurrence of correct (true) and incorrect (false) matches when for each line from the first image only the first best match from the second image is considered, while the threshold is increased from zero to the maximum. The results were calculated on the Middlebury Stereo Dataset images and on the difficult conditions scenes using the LBD descriptor. The red curves were calculated with the Left-Right check turned off while it was turned on for the blue curves. Each curve corresponds to a distance ratio threshold ε while the rightmost curves (containing the most false matches) correspond to the highest ε.

6.2.5 Best achievable matchings

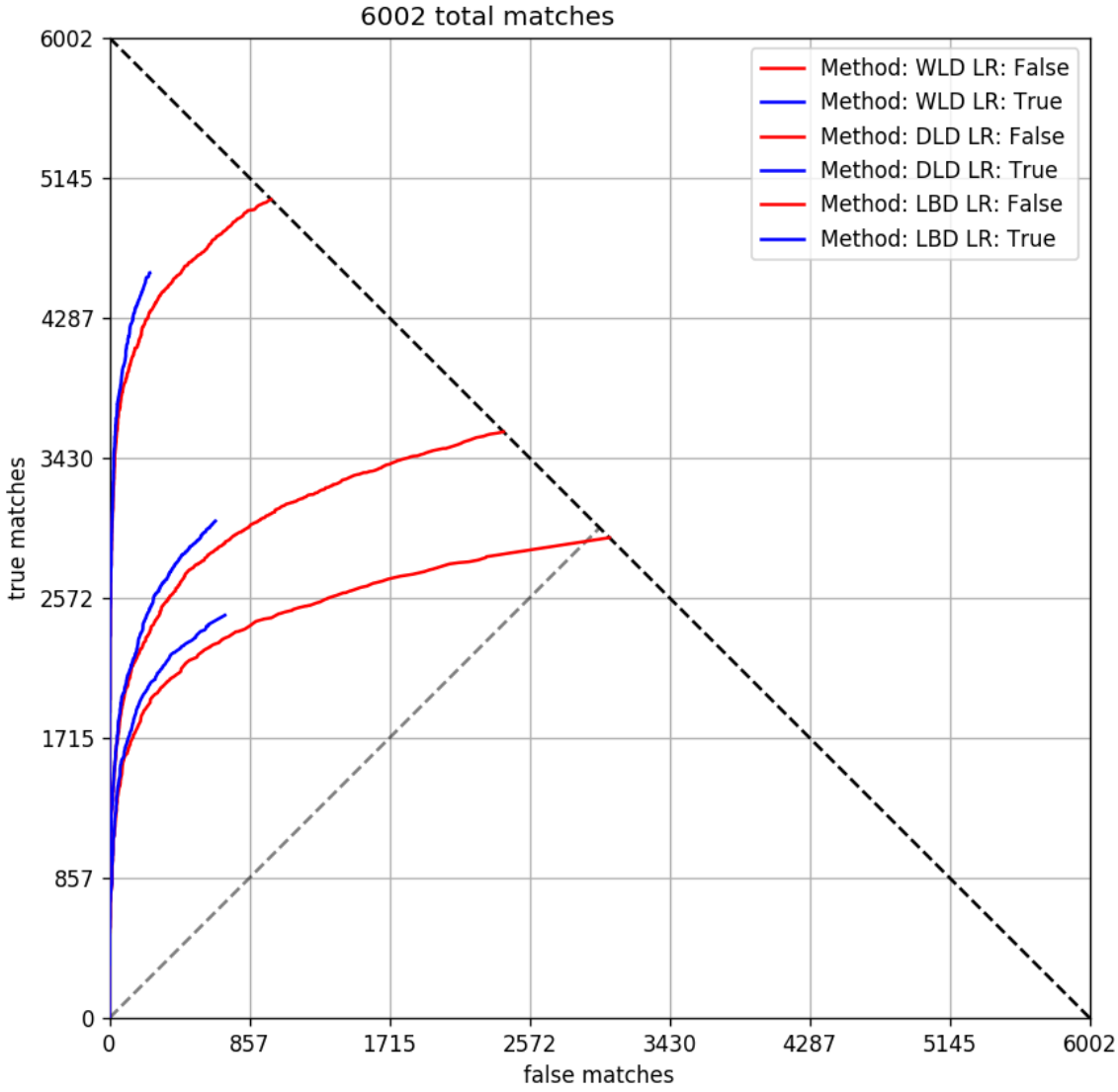


Figure 6.12: The curves of this plot show the highest achievable number of true matches and the corresponding number of false matches for all ε from 0 to 1. The two upper curves represent the WLD, the two curves in the middle represent the DLD and the two lower curves represent the LBD. The Left-Right consistency check is turned on for the blue curves and off for the red curves.

The previous plots showed the match curves when the maximum allowed descriptor distance of the first best match is increased from zero to the maximum, where each curve

had a fixed ε. Comparing the curves showed that in most cases it would be a better choice to improve the ratio of true to false matches by varying the ε and by using the Left-Right consistency check instead of a limit for the maximum descriptor distance. Figure 6.12 shows the highest achievable number of true matches and the corresponding number of false matches for each method when there is no threshold for the descriptor distance but the ε, is used to control the number of true and false matches. There are two curves for each method, one with the Left-Right consistency check turned on and another one with the check turned off. One can see that the curves move along the top peaks of the corresponding previous curves. This shows that it is beneficial to use the distance margin threshold ε to control the number of matches. The curves also provide a comparison between the WLD, the DLD and the LBD. We can see that the WLD yields a much better ratio of true to false matches over the entire matching space. And while the DLD cannot keep up with the WLD, it is still significantly better than the LBD.

6.2.6 Descriptor Distances

Table 6.7: This table shows the ratio and the numbers of true and false matches for the descriptor range of the first best distances for the WLD, the DLD and the LBD (as visualized in figure 6.13).

Method	Type	0%-10%	10%-20%	20%-30%	30%-40%	40%-50%	50%-60%	60%-70%	70%-80%	80%-90%	90%-100%
WLD	true	4266	532	137	40	17	13	4	2	0	1
WLD	false	472	261	143	44	33	19	14	1	2	1
WLD	ratio	9.04	2.04	0.96	0.91	0.52	0.68	0.29	2.00	0.00	1.00
DLD	true	3086	312	96	28	10	4	2	0	1	3
DLD	false	1702	510	154	53	25	8	3	3	1	1
DLD	ratio	1.81	0.61	0.62	0.53	0.40	0.50	0.67	0.00	1.00	3.00
LBD	true	271	898	934	469	208	98	44	12	4	2
LBD	false	76	407	890	816	499	233	99	22	12	8
LBD	ratio	3.57	2.21	1.05	0.57	0.42	0.42	0.44	0.55	0.33	0.25

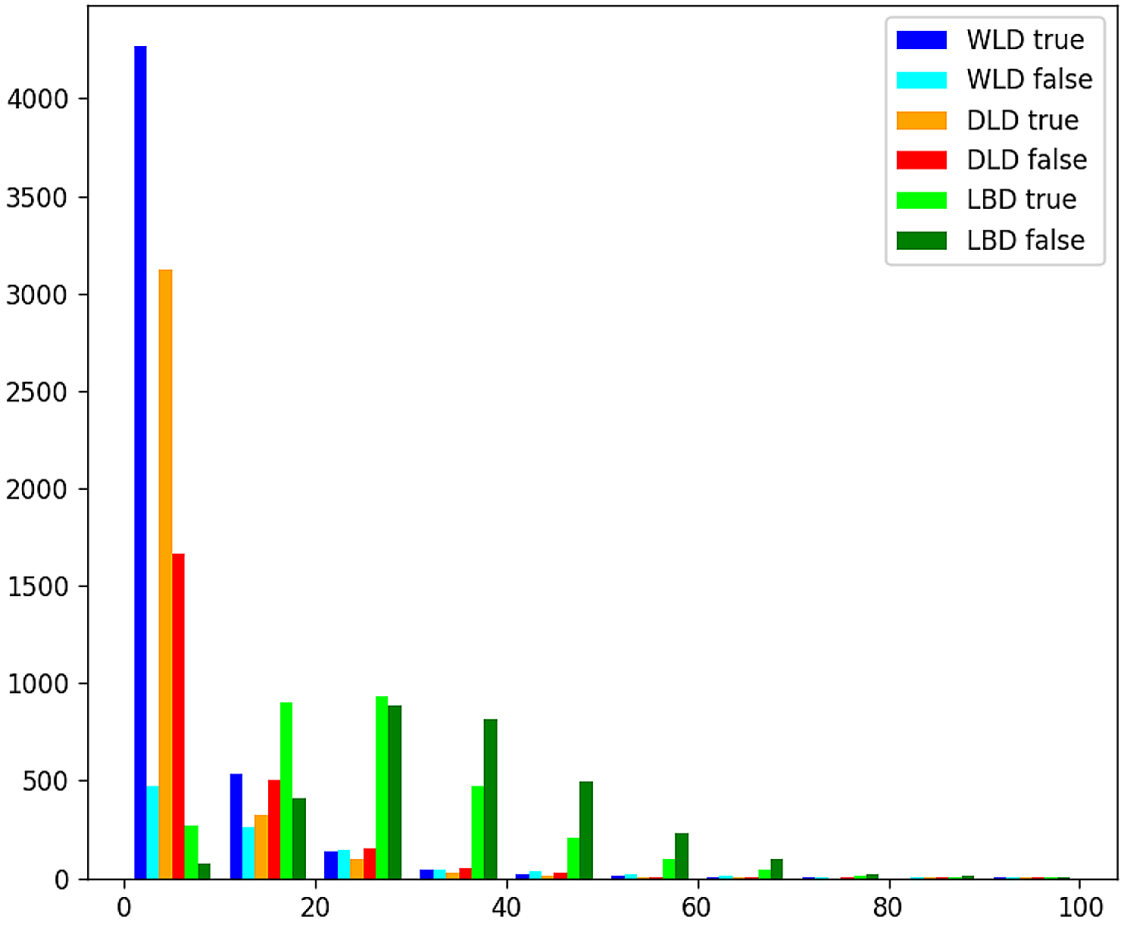


Figure 6.13: The bars of this histogram show how many matches were found, grouped by the descriptor distance. The distance range is normed and true and false matches are represented by individual bars for each of the three methods.

If the matches are filtered by using a maximum descriptor distance, the question arises, which distance is suitable. This depends on the application, the amount of tolerable false matches and on the characteristics of the descriptor. Figure 6.13 shows the occurrence of true and false matches for each descriptor in dependence on the descriptor distance for each ten percent of the range. The range is normalized for each descriptor from zero to the maximum distance among the first best distances of the respective descriptor on the Middlebury and the difficult conditions scenes. Table 6.7 shows the numbers and the ratio of true matches to false matches. One can see that the first ten percent of the WLD

yield the best ratio of over nine true matches to one false match. With 4738 matches almost 80% of the matches occur in the first ten percent. Interestingly, the first percent of the descriptor distance range contains only 12 false matches, but 1105 true matches. This leads to a ratio of 92 true matches to one false match, or roughly one percent false matches, to obtain the first fifth of the true matches.

The higher the descriptor distance, the worse the ratio of true to false matches. This holds for all three descriptors with minor exceptions especially in the second half of the distance, where the number of matches is relatively low compared to the number of matches in the first half. The DLD also has the best ratio at the beginning with the number of true matches and of false matches continuously decreasing thereafter. Among the first percent of the descriptor distance, there are 621 true matches and 101 false matches, resulting in a ratio of 6.15 true matches for each false match.

For the LBD, the number of true and of false matches first increases and then decreases again, forming a hill-shaped curve. The higher peaks of the false matches are shifted towards higher descriptor distances (to the right in the diagram), compared to the higher peaks of the true matches. This is a good thing as it is advantageous when false matches have a higher descriptor distance than true matches. Nevertheless, the LBD has 76 false matches among the 347 matches of the first ten percent, resulting in a ratio of 3.57 true to false matches, which is worse than the ratios of the other descriptors for the first few hundred matches. This is also reflected in figure 6.12.

6.2.7 Summary of the Practical usage evaluation

In this section we evaluated the practical usability of the WLD compared to the DLD and to the LBD. Therefor, we placed an emphasis on the first best matches, as their use is common practice. We filtered the matches using a Left-Right consistency check and a distance margin threshold to see how far one could reduce the amount of false matches by these simple and fast filters. We found that the Left-Right consistency check is almost always beneficial and additionally superior to using a distance margin threshold alone. But when evaluated separately on the Middlebury scenes and the difficult conditions scenes, we also found that the extent of the advantage gained from its use depends on the data it is used on and that its influence diminishes with a stricter distance margin threshold. Similarly, the influence of the distance margin threshold also depends on the data. Therefore, there is not a particular ε for a specific desired rate of true matches or a desired true matches to false matches ratio. We created figure 6.12 which shows curves obtained by calculating the numbers of true and false matches for ε ranging from zero to one. These show the best possible numbers of true matches to false matches for each method with the three checks we evaluated: the Left-Right consistency check, the distance margin threshold ε and the maximum descriptor distance threshold. In addition, we evaluated the distribution of true and false matches of the first best matches in dependence of the descriptor distance. One can see that for all three methods, the rate of true to false matches worsens the higher the descriptor distance of the matches is. The various

evaluations of the different methods also included comparisons which showed that the analytical LBD approach is not nearly as good as the machine learning based WLD or DLD. And we found that the WLD represents a major advancement over to the DLD, both in terms of true matches and for practical usage scenarios.

Chapter 7

Conclusions

7.1 Summary

In this work we presented our contributions to the research on lines. We first explained special properties of lines and gave an overview of common representations for 2D lines as well as for 3D lines in chapter 2. Important details are the number of degrees of freedom a type of line has and the number of parameters a particular representation requires to represent that line type. This varies between the representations and differs depending on whether the line is infinite or a segment and whether the line is a 2D line or a 3D line. We also elaborated various error functions that evaluate the similarity of two lines and created error space plots that visualize the results of each metric. These error metrics and the aforementioned representations are important for the line feature based visual odometry system that we developed, called *Line only stereo Visual Odometry* (LVO), which followed in chapter 3. This method is able to perform visual odometry relying solely on lines. Most of the lines are tracked using a prediction, that is derived from a motion model, and using an error function from chapter 2. We introduced a heuristic matching algorithm that filters matches depending on a prior and on the line features in their neighbourhood. Additionally, we implemented a bundle adjustment for lines that utilizes the Cayley representation for infinite 3D lines to perform the optimization efficiently using only four parameters for the 3D lines. For the evaluation, we applied the proposed method on the scenes of the EuRoC MAV datasets and compared it with other algorithms that use lines, points or both. We were able to show that our method performs best on most scenes of the dataset. The comparison visualizing the number of lines tracked by StVO-L and our own method revealed a reason for our superior performance, as one can see that we were able to track many more lines than the StVO-L approach. This, of course, resulted in a more accurate performance.

One of the things we showed in chapter 3 is how one can efficiently track lines from frame to frame, but matching lines from arbitrary camera positions without prior knowledge was still a difficult task. There was only one well established line descriptor, namely the Line Band Descriptor from Lilian Zhang and Reinhard Koch, which used an analytical approach to create the descriptor. A disadvantage of the LBD is its use of bands along the line direction which are averaged. This averaging smoothes and thereby de-

stroys structural information along the line, resulting in a loss of potentially valuable information. We saw the potential for improving line descriptors using machine learning. Chapter 4 elaborates our approach, called *A Deep Learning Based Line Descriptor for Line Feature Matching* (DLD), in detail. First, we had to find a way to generate large amounts of training data. Therefor, we used the Unreal Engine 4 and several freely available scenes. Our C++ based framework computed the ground truth matching information using the depth maps, which we extracted in addition to the RBG images of the virtual camera, and prepared the data. We used the python based tensorflow framework to train a ResNet. Our goal was to evaluate how much better than the LBD our machine learning based approach could perform, when working on the same input data as the LBD. Therefore, we prepared the image patches, from which the descriptor is calculated, so that they were similar to those used by the LBD. To compare the performance of both methods, we needed ground truth labelled real world data. The Middlebury Stereo Datasets provide high resolution images with ground truth depth information. This depth information enabled the automatic labelling which we implemented to generate the training data. The labels still had to be checked and we added additional ones by hand. We also created a dataset with images for eight scenes representing difficult scenarios for matching like blur, noise, different scales, repeating patterns and more. Comparing both methods on these real world scenarios showed that our learning based approach was able to outperform the analytical LBD in all Middlebury scenes as well as in all of our difficult conditions scenes.

Following these great results, we were curious how much further we could improve the line matching accuracy by not restraining the input data in a similar way as we did with the input data for the LBD. In chapter 5 we improved the aforementioned approach to build a new version, called *A Wavelet and Learning based Line Descriptor for Line Feature Matching* (WLD), which uses an image pyramid and gabor wavelets. We used the image pyramid to provide the network with a wider surrounding of the line. The input has the same height and width as the input for the DLD, but more layers are stacked on top of each other, with each additional level containing more of the surrounding, but at a lower resolution. Additionally, we included gabor wavelets to pre-filter the patches for meaningful information. This approach is inspired by the human visual system, as similar filters are found in the mammalian visual cortex. The basis for the training data is the same as for the DLD, but, of course, it had to be recreated applying the new preprocessing steps. Again, we used our GT labelled Middlebury Data and the difficult conditions scenes for the evaluation. Here, we performed a broader comparison than in chapter 4 as we included two more competing methods, namely *The Learnable Line Segment Descriptor for Visual SLAM* (LLD) from Alexander Vakhitov and Victor Lempitsky and the *Robust Line Segments Matching via Graph Convolution Networks* (GLM) from QuanMeng Ma, Guang Jiang and DianZhi Lai. The results showed that the GLM did not perform very well on most scenes, while the DLD and the LLD competed for the second place in most scenes. Our new WLD achieved the first place with the highest number of correct matches in all except one of the scenes in which our previous DLD

performed better. These results confirmed the usefulness of utilizing an image pyramid in combination with Gabor wavelets.

In chapter 6 we performed an extended evaluation. In the first part, we trained several models with different settings to assess the influence of the descriptor output size, the use of batch normalization during training, the usefulness of the image pyramid and the influence of the Gabor filters. The influence of the Gabor filters was particularly interesting, as the convolutional layers of the ResNet could theoretically learn similar or even better filters on their own. However, we did not know if they would do this automatically. It turned out that we obtained better results when we used the Gabor wavelets than when they were disabled. This means that the convolutional layers did not learn similarly well performing filters on their own and that utilizing the Gabor wavelets is beneficial. The second part of our extended evaluation compared the practical usage of the WLD, the DLD and the LBD and addressed the question what can be done to achieve a low number of false matches. There are three common ways to influence this: setting a maximum descriptor distance threshold, setting a distance margin threshold that defines how much larger the distance of the second best match has to be in comparison to the first best match and activating a left-right consistency check. It turned out that the left-right consistency check is almost always a good measure to reduce the number of false matches. We have also found that in most cases it is more effective to reduce the number of false matches by setting a distance margin threshold for the distance between the first and the second best match than to enforce a maximum descriptor distance threshold for the first best match. The plots created for this part of the evaluation also provided a further comparison between the WLD, the DLD and the LBD. This again showed the superior matching performance of the DLD compared to the LBD. It also showed that the WLD even was a further advancement in terms of matching performance over the DLD; also in the case of practical use where the number of first best matches is quite important.

7.2 Future Work

One difficulty with lines is the problem that they may be split into several parts during the line detection step due to occlusions and properties of the line detector. Using a 3D map when performing visual odometry or SLAM can help, as multiple observations can be assigned to the very same 3D model. Allowing multiple assignments during matching is also possible. A promising approach could be to include more information in a broader understanding. Ya Wang and Andreas Zell, for example, have proposed a method called *Improving Feature-based Visual SLAM by Semantics* [WZ18], that includes semantic information. Higher order knowledge, such as that a line is on a car or even on top of a number plate, could further alleviate the problem of split lines and greatly aid matching. In a more complex system, the lines can be an important part contributing to, and also benefiting from synergy with broader knowledge about the scene.

Another point that could be investigated is the architecture of the line matching network. So far, we achieved the best results using a ResNet architecture. However, recent advancements in the field of machine learning showed that networks which were not designed by hand, but found automatically by a neural architecture search, were often capable to outperform the hand-designed architectures. Thus, this is a promising direction for another improvement of the line matching accuracy.

Finally, one can say, although lines are being researched for a while, there is still a great deal to be investigated.

List of Figures

List of Tables

Abbreviations

AR	Augmented Reality
BA	Bundle Adjustment
BN	Batch Normalization
DLD	A Deep Learning Based Line Descriptor for Line Feature Matching [LSS19]
EuRoC	European Robotics Challenges
FPR	False Positive Rate
GT	Ground Truth
LBD	Line Band Descriptor [ZK13]
LLD	Learnable Line Segment Descriptor [VL19]
LR	Learning Rate
LVO	Line only stereo Visual Odometry [LRS19]
MAV	Micro Aerial Vehicle
RANSAC	Random Sample Consensus
RGB	Red Green Blue Colour Space
ROC	Receiver Operating Characteristic
SfM	Structure From Motion
SLAM	Simultaneous Localization And Mapping
StVO	Stereo Visual Odometry
TPR	True Positive Rate
VO	Visual Odometry
VR	Virtual Reality
WLD	A Wavelet and Learning based Line Descriptor for Line Feature Matching [LRS20]

Bibliography

[ABB95] ALT H., BEHRENDS B., BLÖMER J.: Approximate Matching of Polygonal Shapes. *Annals of Mathematics and Artificial Intelligence 13*, 3-4 (1995), 251–265.

[AMO20] AGARWAL S., MIERLE K., OTHERS: Ceres Solver, accessed: 11.10.2020. URL: `http://ceres-solver.org`.

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

[BNG*16] BURRI M., NIKOLIC J., GOHL P., SCHNEIDER T., REHDER J., OMARI S., ACHTELIK M. W., SIEGWART R.: The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research 35*, 10 (2016), 1157–1163. `doi:10.1177/0278364915620033`.

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

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

[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.

[BT05] BIRSAN T., TIBA D.: One Hundred Years Since the Introduction of the Set Distance by Dimitrie Pompeiu. In *IFIP Conference on System Modeling and Optimization* (2005), Springer, pp. 35–39.

[BTVG06] BAY H., TUYTELAARS T., VAN GOOL L.: SURF: Speeded Up Robust Features. In *European Conference on Computer Vision (ECCV)* (2006), Springer, pp. 404–417.

[CLG03] CHEN J., LEUNG M. K., GAO Y.: Noisy logo recognition using line segment Hausdorff distance. *Pattern Recognition 36*, 4 (2003), 943–955.

[COR*16] CORDTS M., OMRAN M., RAMOS S., REHFELD T., ENZWEILER M., BENENSON R., FRANKE U., ROTH S., SCHIELE B.: The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

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

[DH72] DUDA R. O., HART P. E.: *Use of the Hough Transformation to Detect Lines and Curves in Pictures*. Tech. Rep. 1, New York, NY, USA, Jan. 1972. URL: `https://doi.org/10.1145/361237.361242`, `doi:10.1145/361237.361242`.

[DJ94] DUBUISSON M.-P., JAIN A. K.: A Modified Hausdorff Distance for Object Matching. In *Proceedings of 12th International Conference on Pattern Recognition* (1994), vol. 1, IEEE, pp. 566–568.

[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 (June 2007), 1052–1067. `doi:10.1109/TPAMI.2007.1049`.

[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.

[ED06] EADE E., DRUMMOND T.: Edge Landmarks in Monocular SLAM. In *Proceedings of the British Machine Vision Conference (BMVC)* (2006), BMVA Press, pp. 2.1–2.10. doi:10.5244/C.20.2.

[EKC18] ENGEL J., KOLTUN V., CREMERS D.: Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Mar. 2018).

[Epi20] EPIC GAMES: Unreal Engine. `https://www.unrealengine.com`, accessed: 11.10.2020.

[ESC13] ENGEL J., STURM J., CREMERS D.: Semi-Dense Visual Odometry for a Monocular Camera. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2013), pp. 1449–1456.

[ESC14] ENGEL J., SCHÖPS T., CREMERS D.: LSD-SLAM: Large-Scale Direct Monocular SLAM. In *European Conference on Computer Vision (ECCV)* (September 2014).

[Fab83] FABER R. L.: *Foundations of Euclidean and non-Euclidean geometry*. Dekker New York/Basel, 1983.

[FB81] FISCHLER M. A., BOLLES R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM 24*, 6 (1981), 381–395.

[FB12] FILITCHKIN P., BYL K.: Feature-Based Terrain Classification For LittleDog. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), IEEE, pp. 1387–1392.

[FDB14] FISCHER P., DOSOVITSKIY A., BROX T.: Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT. *arXiv preprint arXiv:1405.5769* (May 2014).

[FLG14] FUHRMANN S., LANGGUTH F., GOESELE M.: MVE - A Multi-View Reconstruction Environment. In *Eurographics Workshop on Graphics and Cultural Heritage* (2014), Klein R., Santos P., (Eds.), CiteSeer, The Eurographics Association, pp. 11–18. doi:10.2312/gch.20141299.

[FPS14] FORSTER C., PIZZOLI M., SCARAMUZZA D.: SVO: Fast Semi-Direct Monocular Visual Odometry. In *IEEE International Conference on Robotics and Automation (ICRA)* (2014).

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

[GLSU13] GEIGER A., LENZ P., STILLER C., URTASUN R.: Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)* (2013).

[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.

[GOBGJ16] GOMEZ-OJEDA R., BRIALES J., GONZALEZ-JIMENEZ J.: PL-SVO: Semi-Direct Monocular Visual Odometry by Combining Points and Line Segments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 4211–4216.

[GOGJ16] GOMEZ-OJEDA R., GONZALEZ-JIMENEZ J.: Robust Stereo Visual Odometry through a Probabilistic Combination of Points and Line Segments. In *IEEE International Conference on Robotics and Automation (ICRA)* (2016).

[GOGJ18] GOMEZ-OJEDA R., GONZALEZ-JIMENEZ J.: Geometric-based Line Segment Tracking for HDR Stereo Sequences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), IEEE, pp. 69–74.

[GOZNM*17] GOMEZ-OJEDA R., ZUÑIGA-NOËL 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).

[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.

[GSC*07] GOESELE M., SNAVELY N., CURLESS B., HOPPE H., SEITZ S. M.: Multi-View Stereo for Community Photo Collections. In *IEEE 11th International Conference on Computer Vision* (2007), IEEE, pp. 1–8. doi:10.1109/ICCV.2007.4408933.

[GZS11] GEIGER A., ZIEGLER J., STILLER C.: StereoScan: Dense 3d Reconstruction in Real-time. In *Intelligent Vehicles Symposium (IV)* (2011), IEEE, pp. 963–968.

[H*64] HUBER P. J., ET AL.: Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics 35*, 1 (1964), 73–101.

[HBL17] HERMANS A., BEYER L., LEIBE B.: In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737* (2017).

[HLJ*15] HAN X., LEUNG T., JIA Y., SUKTHANKAR R., BERG A. C.: MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3279–3286.

[HLVDMW17] HUANG G., LIU Z., VAN DER MAATEN L., WEINBERGER K. Q.: Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 4700–4708.

[HMB14] HOFER M., MAURER M., BISCHOF H.: Improving Sparse 3D Models for Man-Made Environments Using Line-Based 3D Reconstruction. In *Proceedings of the International Conference on 3D Vision* (2014), vol. 1, IEEE, pp. 535–542.

[HS12] HIROSE K., SAITO H.: Fast Line Description for Line-based SLAM. In *Proceedings of the British Machine Vision Conference* (2012), BMVA Press, pp. 83.1–83.11. doi:http://dx.doi.org/10.5244/C.26.83.

[HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[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 (CVPR)* (2016), pp. 770–778.

[IA99] IRANI M., ANANDAN P.: All About Direct Methods. In *International Workshop on Vision Algorithms* (1999), Springer, pp. 267–277.

[IKE21] IKEA: Say Hej to IKEA Place [Promotional Video]. YouTube, 12.09.2017, accessed: 15.02.2021. URL: https://youtu.be/UudV1VdFtuQ?t=51.

[JLW*17] JELLAL R. A., LANGE M., WASSERMANN B., SCHILLING A., ZELL A.: LS-ELAS: Line Segment based Efficient Large Scale Stereo Matching. In *IEEE International Conference on Robotics and Automation (ICRA)* (2017), IEEE, pp. 146–152.

[JŽ17] JELÍNEK A., ŽALUD L.: Line segment similarity criterion for vector images. In *Computer Science Research Notes* (June 2017), pp. 73–80.

[KKKM16] KWON Y. P., KIM H., KONJEVOD G., MCMAINS S.: DUDE (Duality Descriptor): A Robust Descriptor for Disparate Images using Line Segment Duality. In *IEEE International Conference on Image Processing (ICIP)* (2016), IEEE, pp. 310–314.

[KL10] KIM H., LEE S.: Wide-Baseline Image Matching Based on Coplanar Line Intersections. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), IEEE, pp. 1157–1164.

[KM07] KLEIN G., MURRAY D.: Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)* (Nara, Japan, November 2007).

[KS01] KRÜGER V., SOMMER G.: Gabor Wavelet Networks for Object Representation. In *Multi-Image Analysis*. Springer, 2001, pp. 115–128.

[KSm20] KSMRQ: Plücker line coordinate geometry. Wikimedia Commons, 09.07.2006, accessed: 08.12.2020. This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Generic. To view a copy

of this license, visit https://creativecommons.org/licenses/by-sa/2.5/deed.en. URL: https://commons.wikimedia.org/wiki/File:Pl%C3%BCcker_line_coordinate_geometry.png.

[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).

[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.

[LBM19] LANGLOIS P.-A., BOULCH A., MARLET R.: Surface Reconstruction from 3D Line Segments. In *International Conference on 3D Vision (3DV)* (2019), IEEE, pp. 553–563.

[Lee96] LEE T. S.: Image Representation Using 2D Gabor Wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence 18*, 10 (1996), 959–971.

[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 *IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 5550–5557.

[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.

[Low04] LOWE D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[LRL*18] LI S.-J., REN B., LIU Y., CHENG M.-M., FROST D., PRISACARIU V. A.: Direct Line Guidance Odometry. In *IEEE International Conference on Robotics and Automation (ICRA)* (2018).

[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. doi:10.1109/IPIN.2019.8911808.

[LRS20] LANGE M., RAISCH C., SCHILLING A.: WLD: A Wavelet and Learning based Line Descriptor for Line Feature Matching. In *Vision, Modeling, and Visualization* (2020), Krüger J., Niessner M., Stückler J., (Eds.), The Eurographics Association. doi:10.2312/vmv.20201186.

[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. doi:10.1109/IROS40897.2019.8968062.

[LWD10] LIU H.-M., WANG Z.-H., DENG C.: Extend Point Descriptors for Line, Curve and Region Matching. In *Proceedings of the Ninth International Conference on Machine Learning and Cybernetics (ICMLC)* (2010), vol. 1, IEEE, pp. 214–219.

[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.

[LZL*19] LENG C., ZHANG H., LI B., CAI G., PEI Z., HE L.: Local Feature Descriptor for Image Matching: A Survey. *IEEE Access 7* (2019), 6424–6434.

[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.

[MAMT15] MUR-ARTAL R., MONTIEL J. M. M., TARDÓS J. D.: ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics 31*, 5 (2015), 1147–1163. doi:10.1109/TRO.2015.2463671.

[MAT17] MUR-ARTAL R., TARDÓS J. D.: ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics 33*, 5 (2017), 1255–1262. doi:10.1109/TRO.2017.2705103.

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

[MJL20] MA Q., JIANG G., LAI D.: Robust Line Segments Matching via Graph Convolution Networks. *arXiv preprint arXiv:2004.04993* (2020).

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

[Myl22] MYLIUS: Frankfurt on the Main: 3D-reconstruction of the predecessors of the 7th / 8th century of the Königspfalz Frankfurt (Royal Palace Frankfurt), total view as seen from the northwest.

Wikimedia Commons, 19.06.2013, accessed: 29.12.2022. This work is licensed under the Free Art License 1.3. To view a copy of this license, visit https://artlibre.org/licence/lal/en/. URL: https://de.wikipedia.org/wiki/Datei:Frankfurt_Am_Main-Koenigspfalz_Frankfurt-3D-Rekonstruktion-7_8_Jahrhundert-Gesamtansicht_von_Nordwesten.jpg.

[NLD11] NEWCOMBE R. A., LOVEGROVE S. J., DAVISON A. J.: DTAM: Dense Tracking and Mapping in Real-Time. In *IEEE International Conference on Computer Vision (ICCV)* (2011), pp. 2320–2327.

[NNB04] NISTÉR D., NARODITSKY O., BERGEN J.: Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2004).

[NNB06] NISTÉR D., NARODITSKY O., BERGEN J.: Visual Odometry for Ground Vehicle Applications. *Journal of Field Robotics 23*, 1 (2006), 3–20.

[NPVCL08] NEUBERT P., PROTZEL P., VIDAL-CALLEJA T., LACROIX S.: A fast Visual Line Segment Tracker. In *IEEE International Conference on Emerging Technologies and Factory Automation* (2008), IEEE, pp. 353–360.

[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.

[PVA*17] PUMAROLA A., VAKHITOV A. T., AGUDO A., SANFELIU A., MORENO-NOGUER F.: PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines. *IEEE International Conference on Robotics and Automation (ICRA)* (2017), 4503–4508.

[RRKB11] RUBLEE E., RABAUD V., KONOLIGE K., BRADSKI G.: ORB: an efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)* (2011), pp. 2564–2571.

[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.

[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.

[SRD06] SMITH P., REID I., DAVISON A.: Real-Time Monocular SLAM with Straight Lines. In *Proceedings of the British Machine Vision Conference (BMVC)* (Edinburgh, 2006), pp. 17–26.

[SSP*03] SIMARD P. Y., STEINKRAUS D., PLATT J. C., ET AL.: Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *International Conference on Document Analysis and Recognition* (2003), vol. 3, IEEE Computer Society.

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo Tourism: Exploring Photo Collections in 3D. In *ACM Siggraph 2006 Papers*. 2006, pp. 835–846.

[SSTF*15] SIMO-SERRA E., TRULLS E., FERRAZ L., KOKKINOS I., FUA P., MORENO-NOGUER F.: Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 118–126.

[SVZ14] SIMONYAN K., VEDALDI A., ZISSERMAN A.: Learning Local Feature Descriptors Using Convex Optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 36*, 8 (2014), 1573–1585.

[Tas21] TASMANIC EDITIONS: CamToPlan - Maßband messen / Lineal / Zollstock. `https://play.google.com/store/apps/details?id=com.tasmanic.camtoplanfree`, accessed: 15.02.2021.

[TFW17] TIAN Y., FAN B., WU F.: L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 661–669.

[The20] THE GIMP DEVELOPMENT TEAM: Gimp. `https://www.gimp.org`, accessed: 11.10.2020.

[TV98] TRUCCO E., VERRI A.: *Introductory Techniques for 3-D Computer Vision*, vol. 201. Prentice Hall Englewood Cliffs, 1998.

[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.

[VL19] VAKHITOV A., LEMPITSKY V.: Learnable Line Segment Descriptor for Visual SLAM. *IEEE Access 7* (2019), 39923–39934. doi:10.1109/ACCESS.2019.2901584.

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

[WNY09] WANG L., NEUMANN U., YOU S.: Wide-Baseline Image Matching Using Line Signatures. In *IEEE 12th International Conference on Computer Vision* (2009), IEEE, pp. 1311–1318.

[WP16] WIRTZ S., PAULUS D.: Evaluation of Established Line Segment Distance Functions. *Pattern Recognition and Image Analysis 26*, 2 (2016), 354–359. URL: https://doi.org/10.1134/S1054661816020267.

[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.

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

[WZ18] WANG Y., ZELL A.: Improving Feature-based Visual SLAM by Semantics. In *IEEE International Conference on Image Processing, Applications and Systems (IPAS)* (2018), pp. 7–12. doi:10.1109/IPAS.2018.8708875.

[YLG04] YU X., LEUNG M. K., GAO Y.: Hausdorff Distance for Shape Matching. In *Proceedings of the IASTED International Conference on Visualization, Imaging, and Image Processing* (2004), pp. 819–824.

[YS17] YANG S., SCHERER S.: Direct Monocular Odometry Using Points and Lines. In *IEEE International Conference on Robotics and Automation (ICRA)* (May 2017), pp. 3871–3877. doi:10.1109/ICRA.2017.7989446.

[Zha13] ZHANG L.: *Line Primitives and Their Applications in Geometric Computer Vision*. Universitätsbibliothek Kiel, 2013.

[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.

[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.

[ZK15] ZAGORUYKO S., KOMODAKIS N.: Learning to Compare Image Patches via Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 4353–4361.

[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.

[ZL16] ZBONTAR J., LECUN Y.: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *Journal of Machine Learning Research 17*, 1-32 (2016), 2.

[ZZPL16] ZHUANG S., ZOU D., PEI L., LIU D. H. P.: A Binary Robust Line Descriptor. In *International Conference on Indoor Positioning and Indoor Navigation, IPIN, Alcala de Henares, Spain* (October 2016).